University of North Dakota

UND Scholarly Commons

Theses and Dissertations

Theses, Dissertations, and Senior Projects

January 2022

Data Driven Approach To Saltwater Disposal (SWD) Well Location Optimization In North Dakota

Kalyanaraman Venugopal

How does access to this work benefit you? Let us know!

Follow this and additional works at: https://commons.und.edu/theses

Recommended Citation

Venugopal, Kalyanaraman, "Data Driven Approach To Saltwater Disposal (SWD) Well Location Optimization In North Dakota" (2022). *Theses and Dissertations*. 4383.
https://commons.und.edu/theses/4383

This Dissertation is brought to you for free and open access by the Theses, Dissertations, and Senior Projects at UND Scholarly Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of UND Scholarly Commons. For more information, please contact und.commons@library.und.edu.

DATA DRIVEN APPROACH TO SALTWATER DISPOSAL (SWD) WELL LOCATION OPTIMIZATION IN NORTH DAKOTA

by

Kalyanaraman Venugopal

Bachelor of Science & Master of Science, BITS Pilani, 1995

Master of Science, Heriot-Watt University, 2015

Master of Science, University of Houston-Downtown, 2020

A Dissertation

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Grand Forks, North Dakota

August

2022

This dissertation, submitted by Kalyanaraman Venugopal in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Petroleum Engineering from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

Name of Chairperson

Name of Committee Member

Name of Committee Member

Name of Committee Member

Name of Committee Member

This dissertation is being submitted by the appointed advisory committee as having met all of the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

Chris Nelson

Dean of the School of Graduate Studies

Date

PERMISSION

Title	Data driven Approach to Saltwater Disposal (SWD) Well Location Optimization in North Dakota
Department	Petroleum Engineering
Degree	Doctor of Philosophy

In presenting this dissertation in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my dissertation work or, in his absence, by the Chairperson of the department or the dean of the School of Graduate Studies. It is understood that any copying or publication or other use of this dissertation or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my dissertation.

Kalyanaraman Venugopal

Date: 20-Jul-2022

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my advisor, Prof. Minou Rabiei for her guidance and support during my time in the PhD program at the University of North Dakota. I wish to express my heartfelt gratitude to Prof Vamegh Rasouli, who provided me the opportunity to enroll in the PhD program and his constant support and encouragement during my time working on my research. I also acknowledge the North Dakota Industry Commission (NDIC) for their support with the data set and financial support.

To my wife Sudha and my kids Lochana and Uddish

ABSTRACT

The sharp increase in oil and gas production in the Williston Basin of North Dakota since 2006 has resulted in a significant increase in produced water volumes. Primary mechanism for disposal of produced water is by injection into underground Inyan Kara formation through Class-II Saltwater Disposal (SWD) wells. With number of SWD wells anticipated to increase from 900 to over 1400 by 2035, localized pressurization and other potential issues that could affect performance of future oil and SWD wells, there was a need for a reliable model to select locations of future SWD wells for optimum performance. Since it is uncommon to develop traditional geological and simulation models for SWD wells, this research focused on developing data-driven proxy models based on the CRISP-Data Mining pipeline for understanding SWD well performance and optimizing future well locations. NDIC's oil and gas division was identified as the primary data source. Significant efforts went towards identifying other secondary data sources, extracting required data from primary and secondary data sources using web scraping, integrating different data types including spatial data and creating the final data set. Orange visual programming application and Python programming language were used to carry out the required data mining activities. Exploratory Data Analysis and clustering analysis were used to gain a good understanding of the features in the data set and their relationships. Graph Data Science techniques such as Knowledge Graphs and graph-based clustering were used to gain further insights. Machine Learning regression algorithms such as Multi-Linear Regression, k-Nearest Neighbors and Random Forest were used to train machine learning models to predict average monthly barrels of saltwater disposed in a well. Model performance was optimized using the RMSE metric and the Random Forest model was selected as the final model for deployment to predict performance of a planned SWD well. A multi-target regression model was trained using deep neural network to predict water production in oil and gas wells drilled in the McKenzie county of North Dakota.

Contents

List of Figures

List of Tables

Chapter 1

Produced Water Management and Saltwater Disposal

1.1 Introduction

Water is an essential component of oil and gas production. This includes water that is required to support oil and gas drilling and production operations (sourced water) and water that is brought to surface during oil and gas production (produced water). Produced water, a byproduct of oil and gas production is also referred as 'saltwater' due to its typically high salt content (Table 1.1).

Table 1.1 Produced Water Quality

Water Quality	Total Dissolved Solids (TDS, mg/L)
Fresh	<1000
Slightly Saline	1000-3000
Brackish	3000-10000
Saline	10000-35000
Highly Saline	>35000

Source: **(Ground Water Protection Council, 2019)**

Produced water refers to one of the following:

1. Groundwater naturally occurring in deep reservoirs.
2. Water previously injected into the formation for secondary recovery or well treatment purposes.
3. Flowback water that returns to surface after a well is hydraulically fractured.

Managing produced water in an efficient and effective manner is an essential component of oil and gas production. Produced water management involves either disposing produced water as wastewater (saltwater) or consider it as a resource for beneficial reuse such as base fluid for hydraulic fracturing operations, irrigation and livestock watering. Saltwater Disposal (SWD) is a method of reinjecting produced water back into the subsurface for disposal.

1.2 Regulatory Programs

Produced water management in the United States of America (USA) is regulated by federal, state or local agencies. Two major federal permitting programs, the National Pollutant Discharge Elimination System (NPDES) and the Underground Injection Control (UIC) govern produced water management. The NPDES program regulates discharge of any wastewater to the water bodies of the USA such as lakes, rivers and streams. NPDES was created by the USA Environmental Protection Agency (EPA) under the Clean Water Act (CWA). The UIC program, created by EPA under the Safe Drinking Water Act (SDWA) regulates disposal of produced water in injection wells including permitting, construction, operation and closure. According to (UIC), an injection well is used to dispose fluid underground into porous geologic formations. These underground formations may range from deep sandstone or limestone, to a shallow soil layer. Injected fluids may include water, wastewater, brine (saltwater), or water mixed with chemicals.

The (UIC) program consists of six classes of injection wells as given below. Each well class is based on the type and depth of the injection activity and the potential for that injection activity to result in endangerment of an Underground Source of Drinking Water (USDW).

- Class I wells are used to inject hazardous and non-hazardous wastes into deep, isolated rock formations.

- Class II wells are used exclusively to inject fluids associated with oil and natural gas production. Produced water disposal is done through Class II wells.
- Class III wells are used to inject fluids to dissolve and extract minerals.
- Class IV wells are shallow wells used to inject hazardous or radioactive wastes into or above a geologic formation that contains a USDW.
- Class V wells are used to inject non-hazardous fluids underground. Most Class V wells are used to dispose of wastes into or above underground sources of drinking water.
- Class VI wells are wells used for injection of carbon dioxide (CO2) into underground subsurface rock formations for long-term storage, or geologic sequestration.

Figure 1.1 below provides a summary of produced water management options.

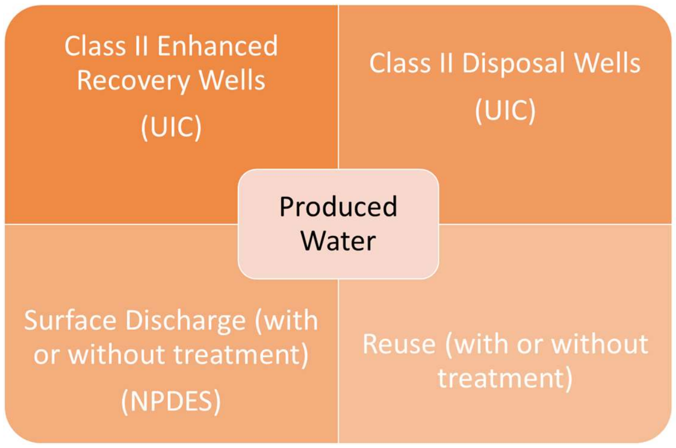

Class II Enhanced Recovery Wells (UIC)
Class II Disposal Wells (UIC)
Produced Water
Surface Discharge (with or without treatment) (NPDES)
Reuse (with or without treatment)

Fig 1.1 Produced water management options

1.3 Produced Water Management in North Dakota

The Clarence Iverson#1 well, drilled in the year 1951and the first successful oil producing well in North Dakota (ND) kicked off the first oil boom in the state (Ziesch & Ritter, 2018). The second oil boom happened in the early 1980s. The third oil boom, also known as the ‘Bakken Boom’ started in 2006. The target oil producing formations are the Bakken and Three Forks formation in the Williston Basin. As a result of horizontal drilling and hydraulic fracturing, there has been a sharp increase in oil and gas production in the Williston Basin of North Dakota since the ‘Bakken Boom’ which has also resulted in a significant increase in produced water volumes. Figure 1.2 below is a plot of well count evolution, annual oil and water production since 2001 in North Dakota’s Bakken shale play, capturing the effect of the ‘Bakken Boom’. A total of 5.45 billion barrels of water was produced in the year 2021 along with 3.69 billion barrels of oil.

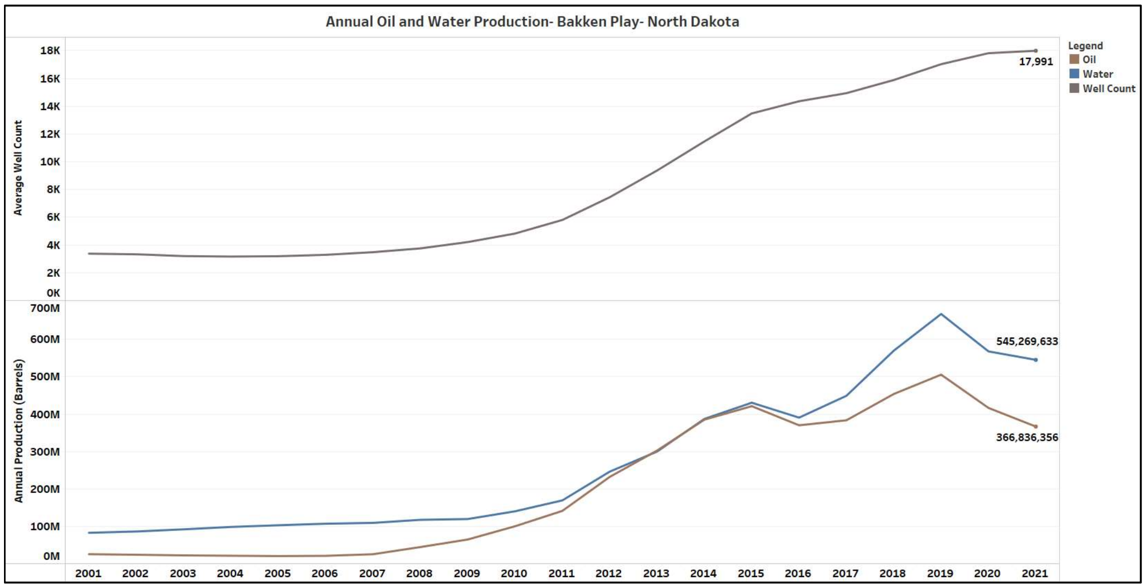


Fig 1.2 Annual oil and water production along with average annual well count, Bakken Play-North Dakota (data courtesy WellDatabase.com, visualization using Tableau software)

Analysis of Total Dissolved Solids (TDS) data for ND’s Bakken shale play from the USGS National Produced Waters Geochemical Database (Blondes, 2018) (year 2006 to year 2012) yields an average TDS of 227,000mg/L. Given the highly saline nature (Table 1.1) of the produced water quality in ND, primary mechanism for disposal of produced water is by injection into underground formations through Class-II SWD wells.

Geology of the area is a major factor in determining viability of underground injection for saltwater disposal. The Lower Cretaceous Dakota geological group in ND’s Williston Basin (Fig 1.3) provides an ideal sequence of geologic units at an optimal depth for saltwater disposal (Bader, 2016).

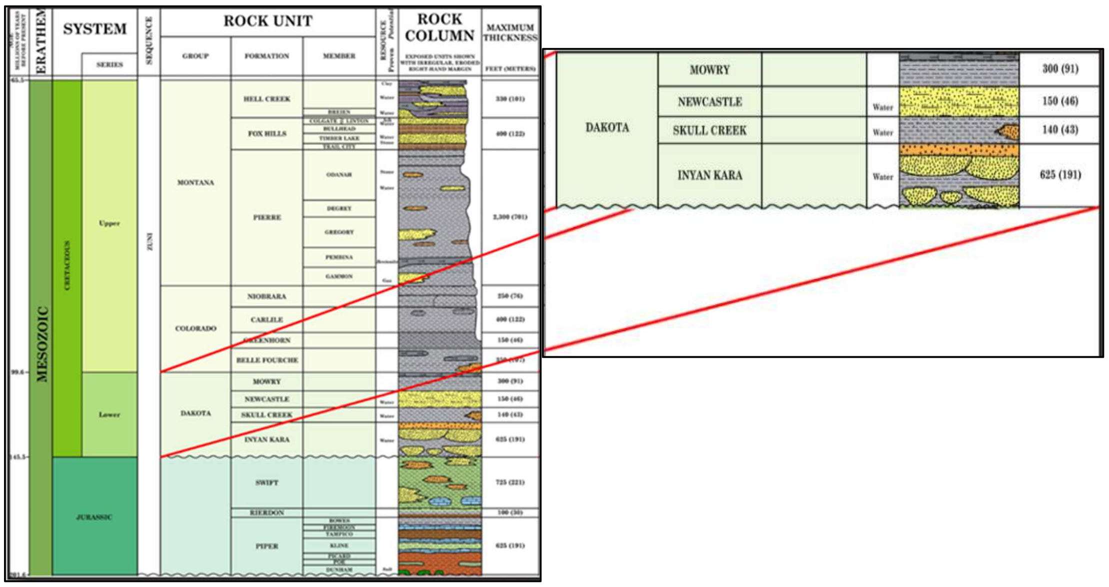


Fig 1.3 ND stratigraphic column showing the Dakota group (Bader, 2016)

The Inyan Kara formation of the Dakota group have good porosity (20-30%) and Darcy level permeability, overlain by the thick Pierre shales and underlain by the Swift formation providing good seal and confinement of the injected saltwater and over 95% of the produced water is disposed through SWD wells in this formation (Bader, 2016). With over 700 SWD active wells in the state already, a 2016 report by Energy & Environmental Research Center (EERC) on Bakken Water Management Practices and Potential Outlook (Kurz et al., 2016) estimates the number of SWD wells to increase to 1485 by the year 2035.

While the highly saline produced water can be treated and reused in the hydraulic fracturing process which needs a lot of water, high treating costs has made disposal the preferred option. It is encouraging to note from a recent article in Journal of Petroleum Technology (Wright, 2022) about the potential use of untreated produced water from the Bakken as an alternative base fluid for use with polymers such as high-viscosity friction reducers that could reduce environmental footprints and lower operating costs. Treating produced water for bacteria, heavy metals and organics may still be needed.

1.4 Objectives

A data-driven approach is when decisions are based on analysis and interpretation of hard data rather than on observation. A data-driven approach ensures that solutions and plans are supported by sets of factual information, and not just hunches, feelings and anecdotal evidence. Application of data mining techniques to use past data to gain deeper insights is now possible due to the availability of robust statistical / machine learning models and computing power.

North Dakota Industrial Commission's (NDIC) Oil and Gas Division which regulates the drilling and production of oil and gas in the state of ND collects data about the performance of the SWD wells along with other data pertaining to the geology of the formations penetrated by the wells.

Objective of the proposed project is to explore the application of data mining techniques to gain insights into SWD well performance in the state of North Dakota and design a data driven proxy model to predict/optimize the performance of planned SWD wells. Depending on the nature of the data available, both unsupervised and supervised machine learning methods will be considered. Graph Data Science (GDS) techniques are gaining a lot of attention now due to their ability to gain insights from connected data. Hence, suitability of GDS techniques to gain deeper insights and improve predictive power of the machine learning models will be examined. As produced water is the reason for SWD, an additional objective is to design data driven proxy models for estimating water production from oil and gas wells.

1.5 Methodology

The proposed research project is based on the Cross Industry Standard Process for Data Mining (CRISP-DM) pipeline shown in Figure 1.4 below, which consists of six steps.

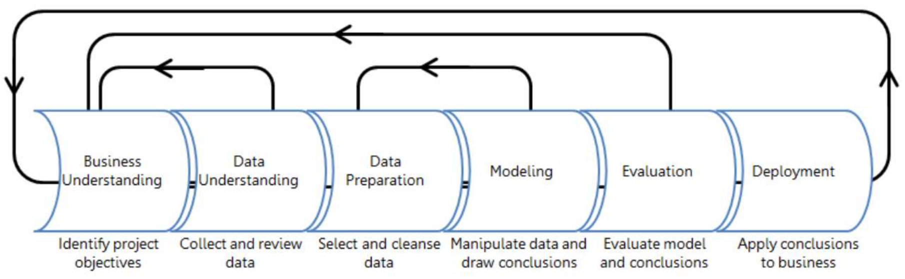


Fig 1.4 Cross Industry Standard Process for Data Mining (CRISP-DM) (Juodyte, 2017)

The following steps will be carried out as part of the project execution:

- Primary and secondary data sources will be identified. The iterative nature of data collection shown in Figure 1.4 will be considered to identify additional data sources during the project execution.
- Technologies for initial data wrangling, visualization and model training / tuning will be identified. Both traditional programming and visual programming tools will be considered with a focus on open source platforms as much as possible.
- Data cleaning, data preparation and data integration will be carried out iteratively including identifying additional data sources and data augmentation. Emphasis will be on identifying data that incorporates surface, subsurface and operational aspects to ensure robustness of the models.
- Exploratory Data Analysis (EDA) and visual analytics will be used to understand data and gain initial insights.
- Analysis will be extended to traditional clustering techniques for gaining additional insights.
- Target variables pertaining to SWD well performance and water production will be identified. Basic and advanced machine learning modeling techniques to predict target variables reliably will be explored. Variable transformation, feature engineering and

cross validation techniques will be incorporated for model tuning and validation purposes.

- The suitability of GDS for creating graph data models, graph analytics and model training and improvement will be explored.
- The final workflow for consuming the trained models and model deployment will be defined.

1.6 Significance

1. While subsurface modeling of oil and gas reservoirs is common, it is not common to see robust subsurface models being developed for saltwater disposal reservoirs. Such models can bring in great benefits to oil and gas companies and regulators in planning future SWD well operations, particularly when produced water (saltwater) volumes are anticipated to increase significantly. Capitalizing on the data available on SWD disposal wells from the state of ND to create a data driven proxy model will enable the state regulator to evaluate permit applications and plan future SWD well locations in an optimum manner. Such models can also be kept alive through regular updates.
2. The use of Orange data mining platform that is based on visual programming approach and provides a no-code environment for developing and deploying data mining models (Demsar, et al., 2013). Such data driven proxy models developed using open source visual programming tools can enable regulators and domain experts understand, utilize and update the models confidently.
3. Data driven models need data. Not all the required data is available in a structured tabular format. Hence, use of Python Beautiful Soup library for web scraping to automatically extract various data relating to SWD well performance from NDIC website and data augmentation considering external data such as proximity to roads and

produced water treatment facilities will enable the development of a performant proxy model

4. Oil and gas data well data is spatial data. Hence, Use of Python GeoPandas library to incorporate spatial data integration and analysis can improve performance of data driven proxy models.
5. Exploring connections in SWD well data through the use of Knowledge Graphs and graph-based clustering techniques with the Neo4j Graph Data Science connected data analytics and machine learning platform can provide additional valuable insights to the regulators at NDIC about performance of current wells and potential to optimize future well locations.

1.7 Dissertation Structure

This dissertation consists of six Chapters.

Chapter 1 is an introduction to the project. A brief overview is provided about produced water management and disposal in the US oil and gas industry, federal regulations and regulations in the state of North Dakota. The objectives, methodology and significance of this project are also presented.

Chapter 2 includes a literature review of the concepts related to traditional subsurface modeling techniques in oil and gas for Field Development Planning (FDP), role of data driven proxy models, the CRISP data mining pipeline and Graph Data Science (GDS).

In Chapter 3 we present a brief overview on the data sets, sources, web scraping and data integration. We also discuss the various technologies used in executing this project such as Orange visual data mining platform, Neo4j graph data science platform, Tableau visual analytics platform and specific Python libraries such as Beautiful Soup and GeoPandas.

In Chapter 4 a discussion of the results of the application of EDA and unsupervised data mining techniques such as clustering including Graph Data Science techniques on the final data is presented.

Chapter 5 is a presentation and discussions of the results obtained from the data driven modeling using supervised regression machine learning algorithms for SWD optimization and produced water prediction carried out in this study.

In Chapter 6 a summary of the findings from this study will be presented along with some recommendations and future studies that can be carried out.

1.8 Summary

In this chapter we presented a brief introduction on produced water (saltwater), a byproduct of oil and gas production operations. Regulatory programs that govern surface and underground disposal of produced water in the USA and the various produced water management options were discussed. Produced water management in the state of North Dakota was reviewed including the evolution of produced water since the 'Bakken Boom' and saltwater disposal being the primary disposal method. The objectives, significance and the methodology related to this project, as well as the structure of this dissertation were also presented.

In the next chapter, a review of the literature related to traditional subsurface modeling techniques in oil and gas for Field Development Planning (FDP), role of data driven proxy models, data mining pipeline and Graph Data Science (GDS) will be presented.

Chapter 2

Literature Review

In this chapter, we will build on the introductory concepts presented in Chapter 1 and present a review of the literature on traditional approaches to Field Development Planning (FDP) using subsurface models, its applicability and limitations for modelling disposal wells, data-driven proxy modelling alternatives and the data mining process. The chapter is divided into three sections related to traditional modelling techniques, data mining process and techniques and Graph Data Science (GDS) techniques.

2.1 Oilfield Subsurface Modeling and Simulation

Sankaran, et al. (2020) describes a model as 'a means to describe or understand a system' that can sometimes be used to predict future outcomes. Figure 2.1 shows the schematic of a

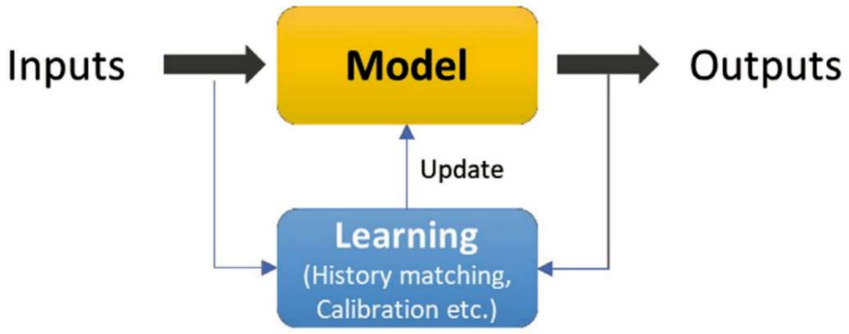


Fig 2.1 Schematic of a Model (Sankaran, et al., 2020)

model based on a mathematical description of the static or dynamic behavior of a system or a process, where a set of equations is used to transform known input parameters to predict outcomes of output parameters. The process of identifying the best model that defines the relationship between the inputs and outputs is known as "learning", also known as history matching, model calibration, or training in the reservoir engineering literature.

As part of oil and gas exploration and production activities, once a new oilfield is discovered and the viability of the reservoir(s) containing oil / gas is established using seismic data from the region and data acquired from the exploration and appraisal wells pertaining to formation properties such as logs, well test etc., to characterize the reservoir, oil and gas companies typically develop a model to describe the reservoir system as part of the Field Development Planning (FDP) process. This is a demanding and involved process, which starts with developing a subsurface geological (static) model using static seismic data, well log data and geo-statistical techniques, creating a dynamic model considering rock and fluid properties and feeding the model to a reservoir simulator.

Edwards (2012) explains that oil and gas simulations, model activities from deep within the reservoir to process plants on the surface and ultimately include economic evaluation and depict the overall FDP process which is shown in Figure 2.2 below. The primary task of a reservoir simulator is to analyze flow through porous media and calculate production profiles as a function of time for the existing and planned wells in the reservoir. These outputs can then be passed on to a surface network simulator for developing well models, then to a process simulator to develop a process plant model and eventually to an economic simulator for understanding the economic performance of the oilfield when it is developed. The reservoir simulators are typically full-physics models, demanding a lot of physical parameters to adequately characterize the system, time and computing power.

Sankaran, et al. (2020) state that when full-physics models are cumbersome to build and calibrate or all the multi-physics aspects are not well understood, alternatives could be reduced-

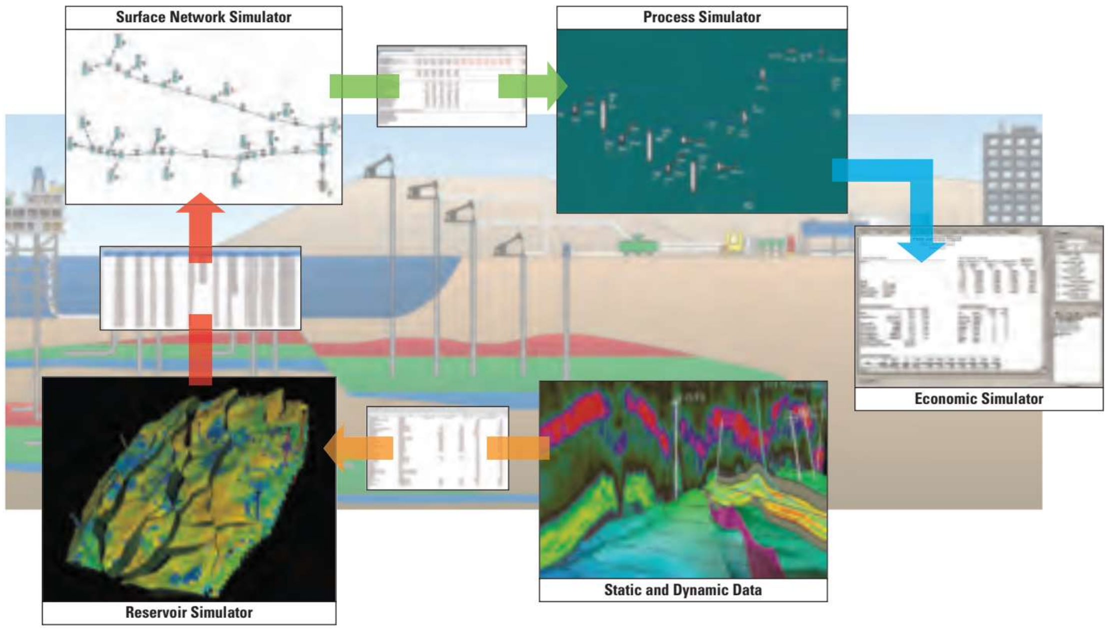


Fig 2.2 Field Development Planning Process (Edwards, 2012)

physics models. These models simplify the physical process(es) through some assumptions or analog models that represent the reservoir system by maintaining accurate relationships between key aspects of the model while absolute values of the original properties may not be preserved. Sankaran, et al., 2020 also mention that for unconventional reservoirs such as shale plays, the complexity of the physical phenomena of flow through porous media, combined with the scale and pace of field development, has pushed the industry to using analog models and data-driven models to answer reservoir management questions.

2.2 Modeling and Simulation of Dakota Sand

As noted in Section 1.3 of Chapter 1, there has been a sharp increase in oil and gas production in the Williston Basin of North Dakota since the ‘Bakken Boom’ which has also resulted in a significant increase in produced water volumes. Normally, produced water is injected into the Dakota Formation, which lies about halfway between the Bakken target and the surface.

S. Basu (2019) identifies in their paper that the Dakota formation pressure has increased in recent years, causing drilling problems for multiple operators across the basin. The Dakota Formation in the Williston Basin does not contain hydrocarbons, resulting in limited core and log data coverage. To address the limitation of a lack of a comprehensive geologic or reservoir model of the Dakota Formation in the Bakken play to guide drilling decisions, S. Basu (2019) developed a geologic model and discussed the results. The paper identifies the challenges to build a reliable geologic model given the limited amount of reservoir properties. The authors used this geological model to build a reservoir simulation model, which was calibrated to injection rates and pressures of saltwater disposal wells in the modelled area.

As of 2016, approximately 94% of produced brines (by volume) was disposed of via deep well injection into the sandstone units of the Inyan Kara Formation, raising questions about the long-term viability of the Inyan Kara as a disposal target. Through the Bakken Production Optimization Program, the Energy & Environmental Research Center developed a reservoir simulation model of a portion of the Inyan Kara Formation to evaluate the effects of current and possible future saltwater disposal (SWD) operations on reservoir pressure and long-term disposal potential. The reservoir simulation model developed for this effort encompassed most of McKenzie County and a portion of northwestern Dunn County. The data inputs for the geologic model included well logs, formation tops, and core sample descriptions and analyses, as well as injected volumes and pressure measurements at individual SWD wells. Injection rate, volume, and pressure data was compiled for the 103 SWD wells in the modeled area and used to history-match the numerical simulation model. The simulation model was used to estimate the current distribution of reservoir pressure and injected salinity plumes as a result of SWD beginning in 1961. Several potential future scenarios were also simulated to evaluate the distribution of pressure and salinity within the Inyan Kara Formation and to evaluate long-term disposal potential out to the year 2050 (Ge Jun et al., 2018).

In addition to the results generated by the simulation model, Ge Jun et al. (2018) also developed an equation-based analog model to calculate the volumetric storage potential of the Inyan Kara within the modeled area under closed boundary conditions. This calculation assumed that all of the available pore space would be accessible via injection wells and that the formation fluid pressure would be at the maximum allowable without exceeding the permitted injection pressure. This simple estimate of maximum storage potential under closed boundary conditions was calculated for comparison to the disposal potential estimated by the reservoir simulation model.

One of the limitations of geological and simulation models is that it takes a lot of time and efforts to update the models. Typical practice in the industry is to update these models once every three to five years and hence the models might not reflect the current status of the field performance, potentially resulting in suboptimal planning scenarios as the model ages. Companies tend to prioritize investing time and efforts in acquiring data for reservoir characterization, developing and updating geological and simulation models for hydrocarbon bearing reservoirs over non-hydrocarbon bearing reservoirs used for water disposal. As a result, geological and simulation models for SWD in the Dakota formation may not be frequently updated.

2.3 Data-Driven Models

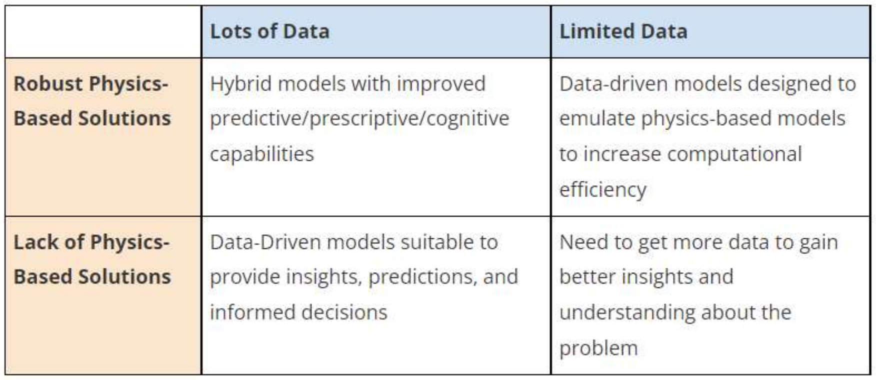
According to Sankaran, et al. (2020), data-driven models are often built using data alone and require an understanding of dependent and independent variables. Some of the advantages of a data-driven models are:

- A precise understanding of the physical processes at play is not required.
- Insights into the processes are derived directly from the data by analyzing patterns.

- Correlation is not identical to causality, but it can often lead to better understanding of causality.
- Recognizing recurring patterns can target an area for investigation and help understand the cause-and-effect process.
- Data-driven analytics are often limited to patterns seen in the historical training data and tend to perform poorly when extrapolated to new operating states. This limitation is usually addressed by retraining models over new patterns.
- The speed of data-driven models under favorable conditions is often very attractive, and they can add substantial business value if they offer a sound and timely decision.

The proliferation of high-resolution datasets and decrease in sensor, storage, and computing costs are significantly extending our ability to grasp new concepts, improve predictions, and perform better field decisions. According to Klie (2021), in a matter of a few years, we have also witnessed the consolidation of data science and machine learning as widespread disciplines that could help generate new technologies derived from data. The abundance of data and the persistence of elusive physical laws to satisfactorily explain the complexity of our assets and operations are promoting a renowned interest for finding ways to extend current model capabilities and decision workflow practices. Moreover, the current stringent economic environment and the increasing interest in pursuing cleaner, safer, and cheaper sources of energy are driving the need for more practical but more robust predictive and prescriptive models. Ultimately, the physics we know needs to rely on data to unmask the physics that we do not yet know.

Whenever robust physics is available, it usually entails computationally intensive processes or workflows. These models may not be suitable for performing real-time actions. The computational demand increases geometrically when these predictive capabilities must be employed to optimize field operations. To make optimization and uncertainty quantification viable approaches, the physics model must be replaced by data-driven surrogate models that are generated from these physics-based models. The interesting fact is that these data-driven models can be trained using both simulation and field data (Klie, 2021). Figure 2.3 below summarizes the contrasting perspective between physics-based and data-driven models.



	Lots of Data	Limited Data
Robust Physics-Based Solutions	Hybrid models with improved predictive/prescriptive/cognitive capabilities	Data-driven models designed to emulate physics-based models to increase computational efficiency
Lack of Physics-Based Solutions	Data-Driven models suitable to provide insights, predictions, and informed decisions	Need to get more data to gain better insights and understanding about the problem

Fig 2.3 Physics-based Vs Data-Driven Models (Klie, 2021)

2.4 Data Mining

Shmueli (2018) defines analytics as the practice and art of bringing quantitative data to bear on decision-making. They refer to data mining as analytics methods that go beyond counts, descriptive techniques, and simple rule-based methods. According to them, data mining stands at the confluence of the fields of statistics and machine learning without the constraints of lack of sufficient data or computing power. Figure 2.4 shows a visual representation of data mining.

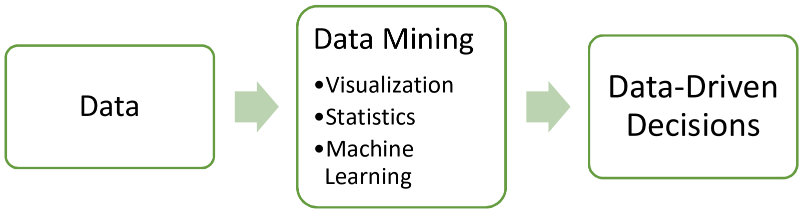


Fig 2.4 Data Mining for Data Driven Decisions

Juodyte (2017) gives an overview on the steps of a data mining project. The usual tasks of a data mining project are grouped together to form the data mining pipeline, which has inputs and a number of processing steps chained together in some way to produce some sort of an output. In the pipeline, each step logically follows the next step providing an outcome. Figure 1.4 represents the CRISP-DM pipeline steps:

- Business Understanding
- Data Understanding
- Data Preparation
- Modeling
- Model Evaluation
- Model Deployment

An overview of the CRISP-DM steps as explained by Juodyte (2017) is provided below:

2.4.1 Business Understanding

This initial phase focuses on understanding the project and requirements from a business perspective, then converting this knowledge into a data mining problem definition, data mining goals, preliminary plan to achieve the objectives and success criteria. Neglecting this step would likely result in a lot of wasted time answering the wrong questions.

2.4.2 Data Understanding

This step starts with initial data collection and proceeds with activities that enable the project team to become familiar with the data. Kabir (2016) explains the concept of data collection as ‘the process of gathering and measuring information on variables of interest in an established and systematic manner’. The author goes on to describe quantitative data as numerical in nature and qualitative data as nominal in nature.

Data mining prefers data that is tabular in nature. Connolly (2015) says that the most common type of data model is a relational data model based on the mathematical concept of a relation that is represented as a table which contains rows (tuples) and columns (attributes). In a relation, each row (tuple) is uniquely identified by an attribute or a set of attributes known as relational keys. A great strength of the relational model is this simple logical structure backed by a sound theoretical foundation.

If the data is available in a website, such data can be collected through web scraping. Karajgikar (2021) defines web scraping as the process of fetching data from third-party website by parsing the Hyper Text Markup Language (HTML) code. The author also identifies the Python BeautifulSoup open source library as a preferred tool for web scraping which is the preferred way to access open data from the web in the absence of Application Programming Interface (API) and explains the importance of understanding the structure of a web page.

Data description using statistical techniques, Exploratory Data Analysis (EDA) and data quality verification are key tasks in this step. EDA helps the project team to gain a good understanding of the data, discover initial insights, identify potential data quality and inconsistency issues and lay the foundation for the data cleaning and preparation activities. Exploratory data analysis can help identify patterns and trends hidden in the data (norm) and deviations from the patterns (exceptions). Using visual analytics to carry out EDA helps to overcome the cognitive limitation faced when dealing with numbers and spreadsheets. One has to realize the importance of creating functional and insightful visualizations by applying Dr. Tufte's visualization design principles (Globus, 1994).

2.4.3 Data Preparation

Data preparation phase covers all activities needed to construct the final dataset that will serve as input to the modeling process, from the initial raw data. Figure 1.4 shows that the iterative nature of data mining and data preparation may be performed multiple times. Attribute selection and sample selection are key components of the data preparation task. Dimensionality reduction, addressing redundancy in attributes, data cleaning and transformation are also key steps included in the data preparation phase.

Loshin (2011) explains data enhancement as the process of increasing the value of the initially available dataset by appending additional value-added knowledge. Enhancement links multiple data sets to pool information from multiple sources, with the intention of identifying actionable knowledge from the combined data. Data enhancement has the potential of increasing the quality of the final dataset and the predictive power of machine learning models trained using the enhanced final dataset.

Combining data from multiple sources to create the final dataset is known as data integration. McKinney (2017) discusses merge and join operations that can be used to combine datasets by linking one or more keys and highlights that these operations are central to relational databases that are tabular in nature.

Oil and gas wells and SWD wells are typically identified using their surface location (latitude, longitude and surface elevation) in addition to a unique well identifier issued by the regulating authority. 4earthintelligence (2021) defines geospatial data as any type of data that contains a geographic component. This includes any data with a location component. As per this definition, oil and gas wells and SWD wells are geospatial data. More specifically, they are considered point data as each point, identified by surface elevation, latitude and longitude, represents a discrete well location (4earthintelligence, 2021). As per 4earthintelligence (2021), line data represents data that is inherently imbued with length (examples being roads and rivers) and polygon data is used to represent an area using perimeters and boundaries (examples being forests, cities and lakes). The concept of data integration using merge and join operations on relational dataset discussed earlier in this chapter can be extended to spatial datasets as well. Spatial join operation joins attributes of one feature to another based on the spatial relationship (ArcGIS Pro). Spatial join is based on proximity of the features calculated using a distance measure. Distance between any two features is calculated as the shortest separation between them (ArcGIS Pro). Figure 2.5 below shows how the distances are calculated between different spatial features.

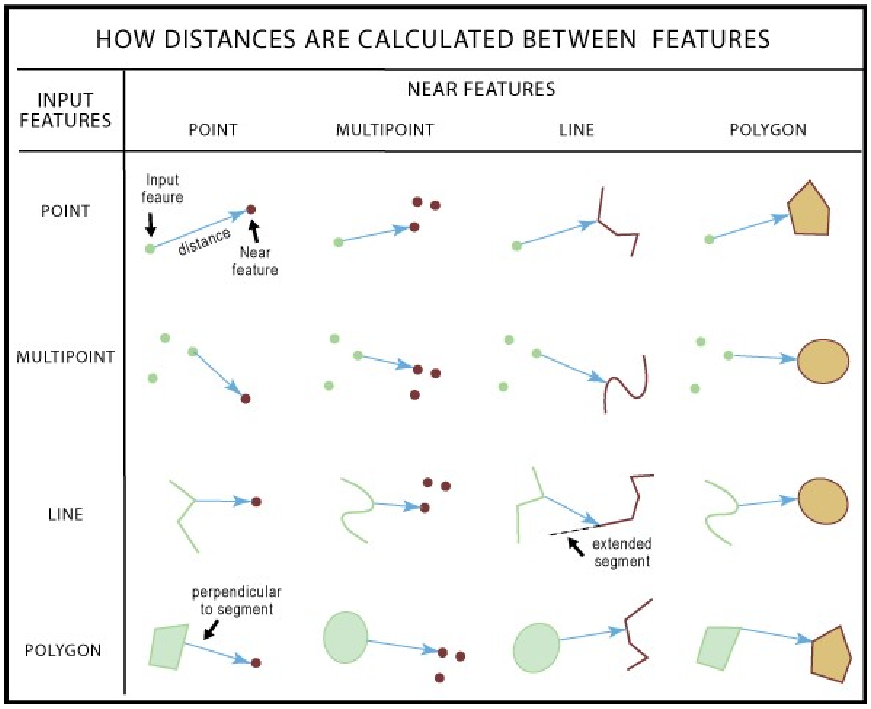


Fig 2.5 Distance Calculation between Spatial Features (ArcGIS Pro)

According to Han (2012), there are several data preparation or preprocessing techniques. Data cleaning can be applied to remove noise and correct inconsistencies in data. Data reduction can reduce data size by, for instance, aggregating, eliminating redundant features, or clustering. Data transformations (e.g., normalization) may be applied, where data are scaled to fall within a smaller range like 0.0 to 1.0. This can improve the accuracy and efficiency of mining algorithms involving distance measurements. These techniques are not mutually exclusive, and they may work together. For example, data cleaning can involve transformations to correct wrong data, such as by transforming all entries for a date field to a common format. Figure 2.6 below provides a summary of the various data preprocessing steps.

Han (2012) identifies dimensionality reduction as one of the data reduction strategies to reduce the number of random variables or attributes under consideration. Principal Component

Analysis (PCA) is a technique of dimensionality reduction that transforms or projects the original data on to a smaller space. Suppose that the data to be reduced consist of tuples or data

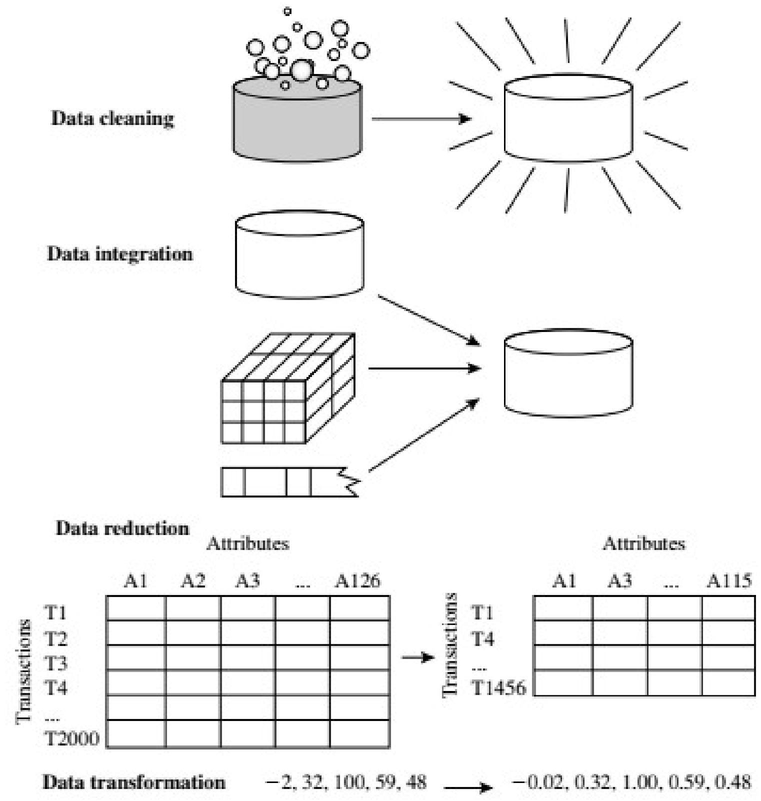


Fig 2.6 Data Preprocessing Steps (Han, 2012)

vectors described by *n* attributes or dimensions. PCA searches for *k* *n*-dimensional orthogonal vectors that can best be used to represent the data, where $k \le n$. The original data are thus projected onto a much smaller space, resulting in dimensionality reduction. PCA combines the essence of attributes by creating an alternative, smaller set of variables. The initial data can then be projected onto this smaller set. PCA often reveals relationships that were not previously suspected and thereby allows interpretations that would not ordinarily result. The basic procedure is as follows:

- The input data are normalized, to ensure that each attribute falls within the same range and that attributes with large domains will not dominate attributes with smaller domains.
- PCA computes *k* orthonormal vectors that provide a basis for the normalized input data, referred to as the principal components. The input data are a linear combination of the principal components.
- The principal components are sorted in order of decreasing "significance" or strength. The principal components essentially serve as a new set of axes for the data, providing important information about variance. That is, the sorted axes are such that the first axis shows the most variance among the data, the second axis shows the next highest variance, and so on. For example, Figure 2.7 shows the first two principal components, *Y1* and *Y2*, for the given set of data originally mapped to the axes *X1* and *X2*. This information helps identify groups or patterns within the data.
- Dimension reduction can be achieved by eliminating the weaker components, that is, those with low variance. Using the strongest principal components, it should be possible to reconstruct a good approximation of the original data.

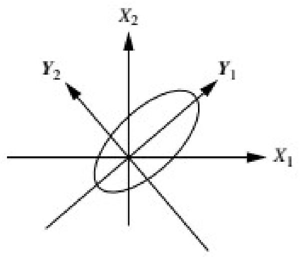


Fig 2.7 Principal Component Analysis.

Attribute subset selection, which retains a subset of the initial attributes, is a method of dimensionality reduction in which irrelevant, weakly relevant, or redundant attributes or dimensions are detected and removed. This usually requires domain knowledge and a good understanding of the nature of the attributes. For *n* attributes, there are 2^n possible subsets. An exhaustive search for the optimal subset of attributes can be prohibitively expensive, especially as *n* increases. Therefore, heuristic methods that explore a reduced search space are commonly used for attribute subset selection. These methods are typically greedy in that, while searching through attribute space, they always make what looks to be the best choice at the time. Their strategy is to make a locally optimal choice in the hope that this will lead to a globally optimal solution. Such greedy methods are effective in practice and may come close to estimating an optimal solution.

2.4.4 Modeling

As per Juodyte (2017), data mining algorithms are applied to the dataset in this step by selecting the appropriate modeling techniques, and their parameters are tuned for optimal model performance. As there might be several techniques for the same data mining problem type that

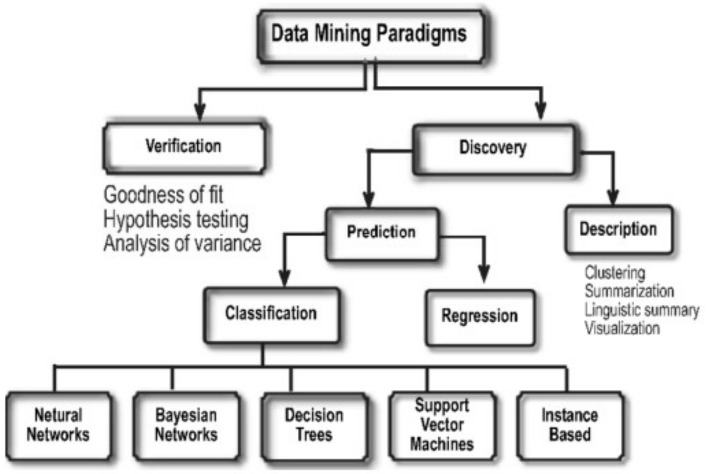


Fig 2.8 Data Mining Paradigms (Juodyte, 2017)

might have specific requirements on data forms, iterating through the data preparation step is often required. Figure 2.8 depicts the data mining paradigms. The two modeling approaches being considered in this research project, clustering and regression are discussed briefly in the following paragraphs.

Clustering:

Clustering is an unsupervised machine learning technique that does not require target labels and belongs to the discovery and description paradigms of data mining (Fig 2.8). Luxborg (2007) states in his technical report that clustering is one of the most widely used techniques for exploratory data analysis, with applications ranging from statistics, computer science, biology to social sciences or psychology. According to the author, clustering enables people to get a first impression on their data by trying to identify groups of "similar behavior" in their data in virtually every scientific field dealing with empirical data. As per Han (2012), clustering techniques consider data tuples as objects and partition the objects into groups, or clusters, so that objects within a cluster are "similar" to one another and "dissimilar" to objects in other clusters. Similarity is commonly defined in terms of how "close" the objects are in space, based on a distance function. Figure 2.9 shows the position of four individuals in a two variable property space composed of variable 1 and variable 2. It is clear from visual inspection that individuals 1 and 2 would be grouped into cluster 1, and individuals 3 and 4 into cluster 2.

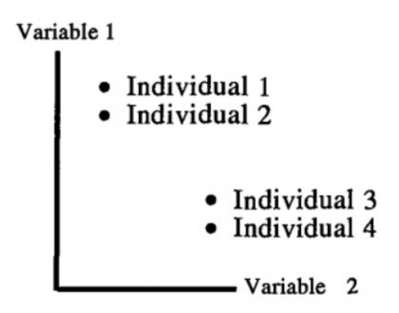


Fig 2.9 Illustration of Distance and Similarity (Juodyte, 2017)

In terms of similarity or dissimilarity, individuals 1 and 2 have high correlations (low distance) between each other, as do individuals 3 and 4. Comparing individuals 2 and 3, however, we see that they are quite dissimilar (exhibiting higher distance between them) on both variable 1 and variable 2, showing that their correlation is low. The formula to calculate the Euclidean distance is given below in Equation 2.1

$$D^2 = \sum_{i=1}^{k} \left(x_{A_i} - x_{B_i}\right)^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2.1)$$

Here *D* is the distance, *K* is the number of variables (dimensions), x_{A_i} signifies the value of variable i for object A, and x_{B_i} signifies the value of variable i for object *B*.

Regression:

Juodyte (2017) discusses regression as an algorithm that attempts to find a function which models the data with the least error that is, for estimating the relationships among data or datasets. It finds the best matching curves to data points, e.g. simple linear regression, multiple linear regression, non-linear regression or logistic regression (for classification problems). An important part of regression is finding a suitable interpolating function. Regression belongs to the discovery and prediction data mining paradigm.

In (simple) linear regression, the data are modeled to fit a straight line. For example, a random variable, *y* (called a response variable), can be modeled as a linear function of another random variable, *x* (called a predictor variable), with the equation:

$$y = wx + b \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2.2)$$

where the variance of *y* is assumed to be constant. In the context of data mining, *x* and *y* are numeric database attributes. The coefficients, *w* and *b* (called regression coefficients), specify the slope of the line and the *y*-intercept, respectively. These coefficients can be solved for by the method of least squares, which minimizes the error between the actual line separating the data and the estimate of the line. Multiple linear regression is an extension of (simple) linear regression, which allows a response variable, *y*, to be modeled as a linear function of two or more predictor variables.

2.4.5 Model Evaluation

The objective of this step is to evaluate and understand the performance of the various machine learning models trained in the previous step. When the outcome is a number as in regression modeling, the most common method for characterizing a model's predictive capabilities is to use the root mean squared error (RMSE). This metric is a function of the model residuals, which are the observed values minus the model predictions. The mean squared error (MSE) is calculated by squaring the residuals and summing them. The RMSE is then calculated by taking the square root of the MSE so that it is in the same units as the original data. The value is usually interpreted as either how far (on average) the residuals are from zero or as the average distance between the observed values and the model predictions (Kuhn & Johnson, 2013). MSE of a model is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.3)$$

Where, y_i is the outcome and $\hat{y}_i$ is the model prediction of that sample's outcome.

Another common metric is the coefficient of determination, commonly written as R^2. This value can be interpreted as the proportion of the information in the data that is explained by the model. Thus, an R^2 value of 0.75 implies that the model can explain three-quarters of the variation in the outcome. The simplest way to calculate R^2 is to find the correlation coefficient between the observed and predicted values and squaring this value. While this is an easily interpretable statistic, the practitioner must remember that R^2 is a measure of correlation, not accuracy (Kuhn & Johnson, 2013).

Using the *K*-fold cross validation technique in model evaluation is essential to ensure the models are not suffering from overfitting and can generalize well for new scenarios when deployed. According to Lantz (2015), the procedure of partitioning the final dataset into training and test datasets randomly is known as the holdout method. The training dataset is used to generate the model, which is then applied to the test dataset to generate predictions for evaluation. The repeated holdout is the basis of a technique known as *k*-fold cross-validation (or *k*-fold CV), which has become the industry standard for estimating model performance. Rather than taking repeated random samples that could potentially use the same record more than once, *k*-fold CV randomly divides the data into *k* completely separate random partitions called folds. Common convention is to use 10-fold cross-validation (10-fold CV). The reason is that the empirical evidence suggests that there is little added benefit in using a greater number. For each of the 10 folds (each comprising 10 percent of the total data), a machine learning model is built on the remaining 90 percent of data. The fold's matching 10 percent sample is then used for model evaluation. After the process of training and evaluating the model has occurred for 10 times (with 10 different training/testing combinations), the average performance across all the folds is reported.

2.4.6 Model Deployment

Creation of the model is generally not the end of the project. Even if the purpose of the model is to increase knowledge of the data, the knowledge gained will need to be organized and presented in a way that the customer can use it. It often involves applying live models within an organizations decision making processes, for example, real-time personalization of Web pages or repeated scoring of marketing databases (Juodyte, 2017).

A traditional learning approach of ML is to acquire enough computer programming skills before diving into ML concepts. Venugopal et al (2021) discusses a code-free ML approach in their paper which alleviates the need for learning programming for building and deploying ML models by embracing a novel approach of building ML models using visual programming approach and trough open source platforms such as Orange. These platforms do not require users to know programming languages such as Python or R, provide built-in visualization tools for rapid data analysis and interactive data exploration, a graphical user interface and enable users to focus on exploratory data analysis rather than on coding. These platforms also enable domain experts and users to deploy ML models in an easy manner to derive maximum benefits in business related decision making.

2.5 Graph Data Science

Networks are a representation; a tool to understand complex systems and the complex connections inherent in today's data (Hodler & Needham, 2021). A graph is a representation of a network, often illustrated with circles to represent entities, also called nodes or vertices, and lines between them. Those lines are known as relationships, links, or edges. Each node represents an entity (a person, place, thing, category or other piece of data), and each relationship represents how two nodes are associated. Figure 2.10 below shows a conceptual representation of a network as a graph.

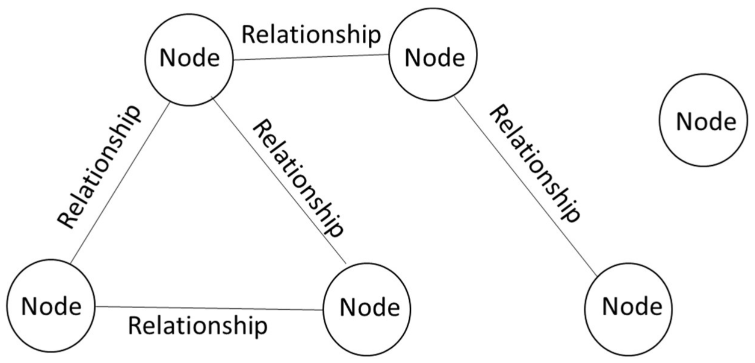


Fig 2.10 Graph Representation of a Network

Nego (2021) states that graphs allow the machine learning system to explore more of the data, faster access, and easier cleaning and enrichment. Traditional learning systems train on a single table prepared by the researcher, whereas a graph-native system can access more than that table. The author also explains that graphs enable data integration through merging of multiple data sources into a single uniform and connected dataset, ready for the training phase of data mining modeling step. This provides a great advantage in data mining by reducing data sparsity, increasing the amount of data available, and simplifying data management.

According to Hunger (2021), while relational databases are powerful tools for the right use case and the right architecture, in today's complex environment where user and business needs are changing rapidly and where real world data is increasing in volume, velocity and variety at a fast pace, data relationships are often more valuable than the data itself. Graph data models are designed to effectively leveraging those connected data relationships.

Hodler & Needham (2021) explain Graph Data Science (GDS) as a science-driven approach to gain knowledge from the relationships and structures in data, typically to power predictions. The authors break GDS down to the three following areas:

- Graph statistics provides basic measures about a graph, such as the number of nodes and distribution of relationships. These insights may influence how you configure and execute more complex analysis as well as interpret results.
- Graph analytics builds on graph statistics by answering specific questions and gaining insights from connections in existing or historical data.
- Graph-enhanced ML is the application of graph data and analytics results to train ML models.

Figure 2.11 below depicts a typical GDS journey. A brief review of the components of the GDS journey, as explained by (Hodler & Needham, 2021) is given below:

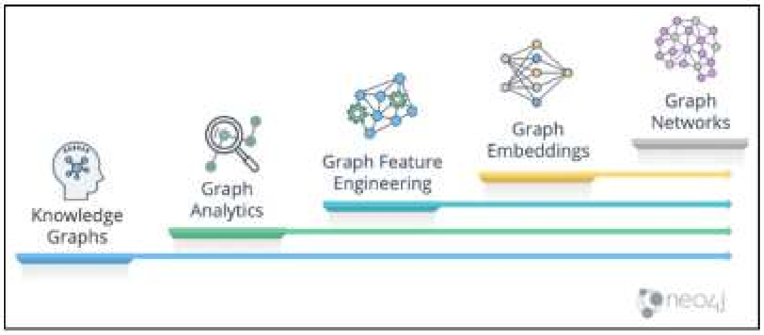


Fig 2.11 GDS Journey (Hodler & Needham, 2021)

Knowledge graphs are the foundation of GDS. At a high level, knowledge graphs are interlinked sets of data points that describe real-world entities, facts, or things and their relationship with each other in a human-understandable form. Unlike a simple knowledge base with flat structures and static content, a knowledge graph acquires and integrates adjacent information by using data relationships to derive new knowledge. According to Nego (2021), knowledge graphs provide a homogeneous data structure for combining not only data sources, but also prediction models, manually provided data, and external sources of knowledge. The resulting data is machine ready and can be used during training, prediction, or visualization.

Graph analytics usually refers to the use of global queries and algorithms that look at entire graphs for offline analysis of historical data and usually involves finding clusters, identifying influential nodes and evaluating different pathways. Graph analytics is employed after implementing the knowledge graph to understand the networks better and answer specific questions based on relationships and topology. We saw earlier in this chapter the role of data visualization and visual analytics in data understanding step of the CRISP-DM pipeline. Nego (2021) presents the following key features of graph-powered data visualization:

- **Data navigation** - Networks are useful for displaying data by highlighting connections between elements. They can be used both as aids to help people navigate the data properly and as powerful investigation tools.
- **Human-brain analysis** - Displaying data in the form of a graph unleashes the power of machine learning by combining it with the power of the human brain, enabling efficient, advanced, and sophisticated data processing and pattern recognition.
- **Improved communication** - Graphs are white-board friendly and conceptually represented on a board as they are stored in the database. This feature reduces the gap between the technicalities of a complex model and the way in which it is communicated to the domain experts or stakeholders. Effective communication improves the quality of the final results because it reduces issues with the comprehension of the domain, the business goals, and the needs and constraints of the project.

Improved communication is particularly important during the business and data understanding phases of the CRISP-DM pipeline, whereas data navigation and human-brain analysis are related mostly to the evaluation phase (Nego, 2021).

Clustering was introduced earlier in this chapter. We present a review of spectral clustering below. Chatterjee (2020) describes spectral clustering as an EDA technique that reduces complex multidimensional datasets into clusters of similar data using the connectivity approach to clustering, wherein communities of nodes (i.e. data points) that are connected or immediately next to each other are identified in a graph. The nodes are then mapped to a low-dimensional space that can be easily segregated to form clusters. Spectral clustering treats the data clustering as a graph partitioning problem without making any assumption on the form of the data clusters. Spectral clustering uses information from the eigenvalues (spectrum) of special matrices (i.e. Affinity Matrix, Degree Matrix and Laplacian Matrix) derived from the graph or the data set. Spectral clustering makes no assumptions about the form of the clusters unlike other popular clustering techniques such as K-Means that assumes the points assigned to a cluster are spherical about the cluster center. The data points should be connected, but may not necessarily have convex boundaries, as opposed to the conventional clustering techniques, where clustering is based on the compactness of data points. Spectral clustering can be computationally expensive for large datasets as eigenvalues and eigenvectors need to be computed.

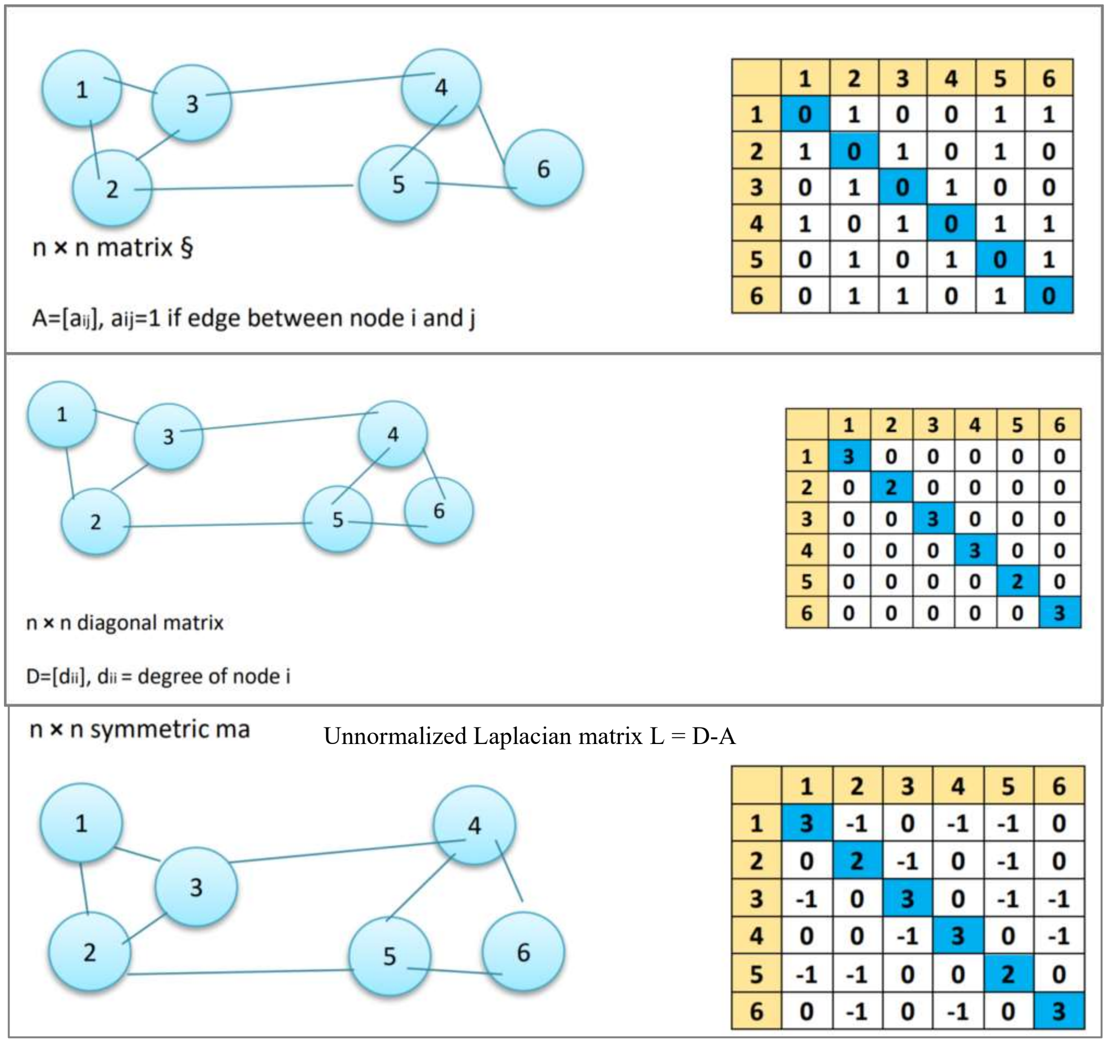
By computing similarities between data points, tabular data can be represented as a similarity graph $G = (V, E)$ such that each vertex v_i in G represents a data point x_i. Two vertices are connected if the similarity s_{ij} between the corresponding data points x_i and x_i is positive or larger than a certain threshold, and the edge is weighted by s_{ij}. G can then be partitioned such that the edges between different clusters (groups) have very low weights (Luxborg, 2007).

Chatterjee, 2020 defines an undirected and unweighted graph containing no loops or multiple edges as a simple graph and a graph $G(V,E)$ with a set V of vertices and a set E of ordered pairs of vertices called directed edges or arrows a directed graph.

While there are a few options to construct the similarity graph, Luxborg (2007) recommends starting with k-nearest neighbor graphs owing to its simplicity. Here the goal is to connect vertex v_i in the graph G with vertex v_j if v_i is among the k-nearest neighbors of v_i. Ignoring the directions of the edges by connecting the vertices with an undirected edge yields the k-nearest neighbor graph. The mutual k-nearest neighbor graph can be generated by connecting vertices v_i and v_j if v_i is among the k-nearest neighbors of v_i and vice versa. Edges are weighted by the similarity of the end points once the appropriate vertices are connected.
Let us review the matrix representations of the similarity graph G, as explained by (Chatterjee, 2020).

Adjacency and Affinity Matrix (A): The graph (or set of data points) can be represented as an Adjacency Matrix, where the row and column indices represent the nodes, and the entries represent the absence or presence of an edge between the nodes (i.e. if the entry in row 0 and column 1 is 1, it would indicate that node 0 is connected to node 1). If the values in the matrix are replaced with the edge weights i.e. affinity, the matrix is known as the Affinity Matrix.
The Degree Matrix (D) is a diagonal matrix, where the degree of a node (i.e. values) of the diagonal is given by the number of edges connected to it. We can also obtain the degree of the nodes by taking the sum of each row in the adjacency matrix.

The Laplacian Matrix (L) is the main tool for spectral clustering and the unnormalized graph Laplacian matrix is obtained by subtracting the Adjacency (Affinity) Matrix from the Degree Matrix (i.e. $L = D - A$). Figure 2.12 illustrates the definitions of A, D and L using a simple example.


n × n matrix §
A=[aij], aij=1 if edge between node i and j
n × n diagonal matrix
D=[dii], dii = degree of node i
n × n symmetric ma
Unnormalized Laplacian matrix L = D-A

	1	2	3	4	5	6
1	0	1	0	0	1	1
2	1	0	1	0	1	0
3	0	1	0	1	0	0
4	1	0	1	0	1	1
5	0	1	0	1	0	1
6	0	1	1	0	1	0

	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	2	0
6	0	0	0	0	0	3

	1	2	3	4	5	6
1	3	-1	0	-1	-1	0
2	0	2	-1	0	-1	0
3	-1	0	3	0	-1	-1
4	0	0	-1	3	0	-1
5	-1	-1	0	0	2	0
6	0	-1	0	-1	0	3

Fig 2.12 Adjacency, Degree and Laplacian Matrices

Unnormalized Spectral Clustering Algorithm (Luxborg, 2007):

Assuming that our data consists of *n* arbitrary points, say $x_1, x_2. . . , x_n$, and their pairwise similarities $s_{ij} = s(x_i, x_j)$ measured by some symmetric and non-negative similarity function, the corresponding similarity matrix can be denoted by $S = (s_{ij}), i,j = 1...n$.

- Inputs: Similarity matrix $S \in Rn \times n$, Number of clusters to construct *k.*
- Construct a k-nearest neighbor or the modified k-nearest neighbor similarity graph.
- Let *A* be its weighted adjacency matrix i.e. affinity matrix.

- Compute the unnormalized Laplacian L as $D\text{-}A$.
- Compute the first k eigenvectors $u_1, \ldots, u_k$ of L.
- Let $U \in Rn\times k$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in R_k$ be the vector corresponding to the i-th row of U.
- Cluster the points (y_i), i=1, ..., n in R_k with the k-means clustering algorithm into clusters $C_1, \ldots, C_k$.
- Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$.

Another graph clustering algorithm is Louvain clustering, an algorithm which can detect communities in large networks. The algorithm first converts the input data into a k-nearest neighbor graph and edges are weighted based on similarity. It then maximizes a modularity score for each community, where the modularity quantifies the quality of an assignment of nodes to communities. This means evaluating how much more densely connected the nodes within a community are, compared to how connected they would be in a random network. The Louvain algorithm is a hierarchical clustering algorithm, that recursively merges communities into a single node and executes the modularity clustering on the condensed graphs (neo4j).

Graph feature engineering is the process of finding, combining, and extracting predictive elements from raw graph data to be used in ML tasks.

Graph embedding simplifies graphs or subsets of graphs into a feature vector, or set of vectors, that are in a lower dimensional form, such as a list of numbers. The goal is to create easily consumable data for tasks like ML that still describe more intricate topology, connectivity, or nodes attributes.

Graph Networks refers to native graph learning that takes a graph as an input, performs learning computations while preserving transient states, and then returns a graph (Luxborg, 2007). This native graph learning process allows the domain expert to review and validate the learning path that leads to more explainable predictions. With this process comes richer and more accurate predictions that use less data and training cycles.

2.6 Data Driven Model for Water Production Prediction

Cross, Sathaye, Darnell, Niederhut, & Crifasi (2020) claim that in unconventional oil fields, water forecasting, and pre-drill water predictions have not received attention commensurate with their economic importance. Operators, regulators, and water disposal companies often rely on simplistic water cut ratios or basin-level extrapolations that ignore the complex interplay of geology, completions, and spacing decisions on water production. Quantifying produced water has importance to a broad set of stakeholders in the oil and gas industry.

The authors mention that forecasting water production is a difficult problem in part because actual water production has been evolving over the life of unconventional fields. Since the Bakken-Three Forks play of the Williston Basin provides an excellent example to study unconventional water production due to the high-quality publicly-available data and long history of development across multiple vintages of completions designs testing both the fringe and the core of the play, the authors have trained a decision-tree based machine learning model using publicly-available data from NDIC to predict water, gas, and oil production at 30-day increments for the first two years of a well's production. Completions related features considered by the authors were per-foot proppant and fluid volumes and stage length. Well spacing related features considered were number of neighbor wells and distance to neighboring wells. Geologic features considered were mean values and P1/10/90/99 per zone of gamma, resistivity, neutron porosity, bulk density, spontaneous potential, and sonic, along with structural tops, isopachs, and mud gas chemistry measurements. The targets for each well were represented as a vector of two years of cumulative production, sampled at 30-day intervals, excluding days where the well was offline or non-producing. The authors used a multitarget averaging ensemble regressor to train the machine learning models. Based on the discussion of the model performance results by the authors, operator choices played the dominant role in influencing water production. While proppant is commonly held to be the largest lever for oil production, the authors' model concludes that completions fluid loading is the most important for water production. However, proppant shows the strongest interaction with geology, with large jobs disproportionately increasing water production in water-prone rocks.

According to Xu (2019), multi-output learning can concurrently predict multiple outputs, In contrast to traditional single-output learning. The outputs can be of various types and structures, and this approach can be used to solve a diverse range of problems. Multi-output learning maps each input (instance) to multiple outputs. Assume $X = R^d$ is a d-dimensional input space, and $Y = R^m$ is an m-dimensional output label space. The aim of multi-output learning is to learn a function $f : X \text{->} Y$ from the training set $D = \{f(x_i; y_i) \mid 1 \leq i \leq n\}$. For each training example $(x_i; y_i)$, $x_i \in X$ is a d-dimensional feature vector, and $y_i \in Y$ is the corresponding output associated with x_i. Multi-output learning can then be defined as: Finding a function $F : X \times Y \text{->} R$ based on the training sample of input-output pairs, where $F(x; y)$ is a compatibility function that evaluates how compatible the input x and the output y are. Then, given an unseen instance x at the test state, the output is predicted to be the one with the largest compatibility score (Xu, 2019).

The author describes 'Multi-target Regression' as the ability to simultaneously predict multiple real-valued output variables for one instance. Here, multiple labels are associated with each instance, represented by a real-valued vector, where the values represent how strongly the instance corresponds to a label. Therefore, we have the constraint of $y_i \in Y = R^m$. Given an unseen instance $x \in X$, the learned multi-target regression function $f(\cdot)$ predicts a real-valued vector $f(x) \in Y$ as the output.

Neural network models support multi-output regression and have the benefit of learning a continuous function that can model a more graceful relationship between changes in input and output. Multi-output regression can be supported directly by neural networks simply by specifying the number of target variables there are in the problem as the number of nodes in the output layer. For example, a task that has three output variables will require a neural network output layer with three nodes in the output layer, each with the linear (default) activation function. Deep learning neural networks are an example of an algorithm that natively supports multi-output regression problems. Neural network models for multi-output regression tasks can be easily defined and evaluated using the Keras deep learning library (Brownlee, 2020).

2.7 Summary

In this chapter, we presented a review of the literature on traditional approaches to Field Development Planning (FDP) using subsurface models, its applicability and limitations for modelling disposal wells. We then presented a detailed literature review of the data mining process based on the CRISP-DM pipeline. We then presented a review of literature on the topic of Graph Data Science (GDS) and its applicability to explore connections in the data using knowledge graphs and graph-based clustering techniques to gain additional insights. We concluded the chapter with a discussion on data driven modelling techniques for estimating water production and Multi-Regression concept.

In the next chapter, we will review the various technologies used in this project and an overview of primary and secondary data sources used to collect the required data for the project. Data collection techniques used and key features in each of the data set will be described.

Chapter 3

Technologies and Data

In this chapter, we present the list of various technologies used in executing this project. We then present all the data sources used to acquire the required data , sourcing methodology, a brief description of the various data sets including its attributes and key data integration aspects.

3.1 Technologies Used

Anaconda Distribution is the world's most popular open-source Python distribution platform that provides access to a variety of open-source software for projects in any field, from data visualization to robotics. Its intuitive and user-friendly platform enables easy search and installation of various packages. Anaconda Navigator is the desktop application to easily manage integrated applications, packages, and environments without using the command line.

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes. It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable; it runs on many Unix variants including Linux and macOS, and on Windows (Python, 2022).

Integrated Development Environment used in the project is Jupyter Notebook that is available through the Anaconda Distribution platform. One of the major components of the

Jupyter project is the notebook, a type of interactive document for code, text (with or without markup), data visualizations, and other output. The Jupyter notebook interacts with kernels, which are implementations of the Jupyter interactive computing protocol in any number of programming languages. Python's Jupyter kernel uses the IPython system for its underlying behavior (McKinney, 2017).

The following open source Python packages were used for executing various tasks during the project:

- **Numpy** - the fundamental package for scientific computing in Python.
- **Pandas** - a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- **GeoPandas** - an open source project to make working with geospatial data in python easier. GeoPandas extends the datatypes used by pandas to allow spatial operations on geometric types.
- **Shapley -** a Python open source package for manipulation and analysis of planar geometric objects.
- **EarthPy -** a Python package to plot and work with spatial raster and vector data using open source tools. Earthpy depends upon geopandas which has a focus on vector data and rasterio with facilitates input and output of raster data files. It also requires the matplotlib library for plotting operations. This library's goal is to make working with spatial data easier for scientists.
- **Requests** - a Python library for making HTTP requests in Python. It abstracts the complexities of making requests behind an Application Programming Interface (API) and allows the user to can focus on interacting with services and consuming the data in their application.

- **BeautifulSoup** - a Python library for pulling data out of HTML and XML files. It creates a parse tree for parsed web pages that can be used to extract data from HTML and is useful for web scraping.
- **RE** - a RegEx, or Regular Expression, is a sequence of characters that forms a search pattern. re is the Python Regular Expression library for creating regular expressions to manipulate string effectively and to create search patterns for data extraction.
- **OS** - is a Python library that contains useful tools and functions to interact with the underlying Operating System being used to execute the running Python code.
- **NLTK** - a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.
- **lasio** - a Python library to read and write Log ASCII Standard files with Python used for borehole data such as geophysical, geological, or petrophysical logs, published by the Canadian Well Logging Society.
- **Keras** - a deep learning API written in Python, running on top of the machine learning platform TensorFlow (an end-to-end, open-source machine learning platform). Keras library was used to carry out multi-output (multi-target) regression analysis to estimate water production in this project.
- **Scikit-learn -** a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed

under the simplified BSD license, encouraging its use in both academic and commercial settings. (Pedregosa, 2011)

Some of the code-free visual programming applications used in this project are listed below:

Orange - An open-source data visualization, machine learning and data mining toolkit. It features a visual programming front-end for explorative rapid qualitative data analysis and interactive data visualization (Demsar, et al., 2013).

Tableau – A commercial visual analytics platform to enable data driven business analytics and decision making. It helps people see and understand data and is transforming the way people use data to solve problems.

Neo4j Graph Data Science - is a connected data analytics and machine learning platform that helps users understand the connections in their data to answer critical questions and improve predictions. It is the only connected data analysis platform that unifies the ML surface and graph database into a single workspace and thus enables data scientists run algorithms and ML models without jumping between tools for Extraction, Transformation and Loading (ETL).

3.2 Data

Backbone of any data driven modeling is data. The CRISP-DM pipeline shown in Fig 1.4 highlights the importance of data understanding and data preparation. The sections below provide a detailed overview of the various data sources (primary and secondary), types of data sets collected, and the technologies used to clean, extract and integrate the required data sets to create the final data set. While significant time and efforts were spent on data collection and preparation, the data understanding and data preparation steps were revisited many times during the project execution, during the exploratory data analysis activities and also as part of model tuning and evaluation activities as highlighted by the iterative nature of the CRISP-DM in Fig 1.4. Domain knowledge and expertise and industry experience of the researcher and the advisor were critical in identifying the various data sources and selecting data sets and features that were relevant for the data driven modeling task.

3.2.1 Primary Data Source

The Oil and Gas Division of North Dakota Industrial Commission's (NDICOG) (https://www.dmr.nd.gov/dmr/oilgas) regulates the drilling and production of oil and gas in North Dakota. Their mission is to encourage and promote the development, production, and utilization of oil and gas in the state in such a manner as will prevent waste, maximize economic recovery, and fully protect the correlative rights of all owners to the end that the landowners, the royalty owners, the producers, and the general public realize the greatest possible good from these vital natural resources.

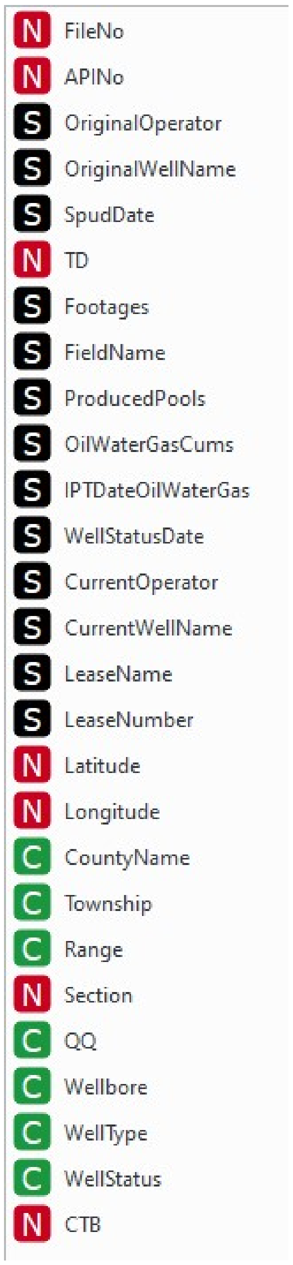
NDICOG was the primary data source for the project. The following data sets were collected from the primary data source, NDICOG using the University of North Dakota (UND) premium subscription. Table 3.1 below lists the various datasets and their types. A detailed description of each of the datasets and extraction techniques are given below.

Table 3.1 Data Sets from Primary Data Source

Data Set	Type
1. Well Index	Flat File (Excel / CSV)
2. Log Tops	Flat File (Excel / CSV)
3. Monthly Disposal Data for SWD Wells	HTML
4. Well Scout Ticket	HTML
5. Log Data	Digital LAS (Log ASCII Standard)

1. **Well Index file**: This file contained a total of 39455 samples at the time of download in Oct 2020. Each sample represented a well drilled in the state of ND for various purposes such as Oil Production, Gas Production, Saltwater disposal etc., The dataset

contained a total of 27 attributes. A snapshot of all the features is given in Figure 3.1 below, extracted using Orange software.



N FileNo
N APINo
S OriginalOperator
S OriginalWellName
S SpudDate
N TD
S Footages
S FieldName
S ProducedPools
S OilWaterGasCums
S IPTDateOilWaterGas
S WellStatusDate
S CurrentOperator
S CurrentWellName
S LeaseName
S LeaseNumber
N Latitude
N Longitude
C CountyName
C Township
C Range
N Section
C QQ
C Wellbore
C WellType
C WellStatus
N CTB

Fig 3.1 Attributes (features) in the NDICOG Well Index dataset

The WellType feature was used to filter the dataset to extract the 905 saltwater disposal (SWD) wells for further analysis.

2. **Log Tops File**: This file downloaded from NDICOG website contained formation tops information for various formations penetrated by each well in the Well Index dataset. This dataset contained 31201 samples and a total of 72 attributes. Each sample represented a well drilled in the state of ND that had log data. Features of interest from this dataset for further consideration is shown in Figure 3.2 below.

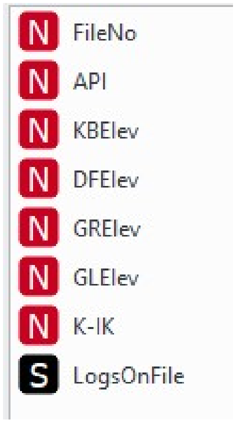
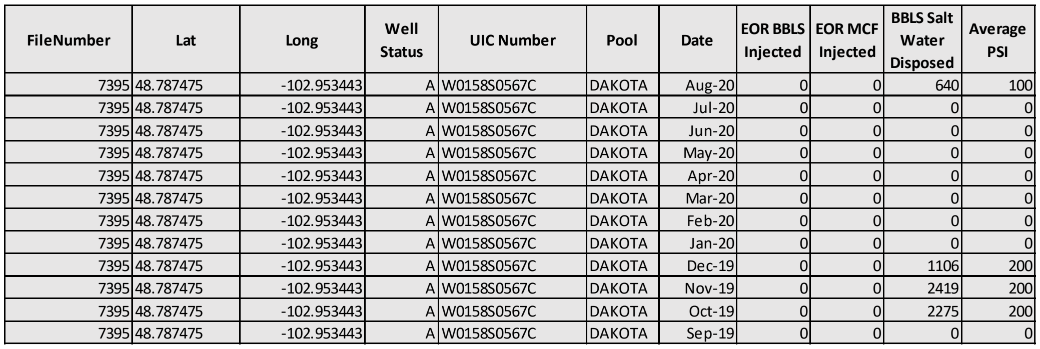


Fig 3.2 Features of interest from the NDICOG Log Tops dataset

The primary keys to combine the above two datasets were API and FileNo, both of which are unique identifiers of each well in the dataset. The K-IK feature provides the formation top of the Inyan Kara formation (subsurface information) where saltwater is disposed in the state of ND as mentioned in Chapter-1. Elevation features when combined with the Latitude and Longitude features from the well index dataset provides the complete surface location of the wells. LogsOnFile attribute was used to examine the wells that had digital logs to potentially extract log data as additional features and evaluate their impact on model performances.

3. **Monthly Disposal Data for SWD Wells**: For each of the 905 SWD wells, the monthly disposal volumes and average injection pressures were available as time series data in HTML format. Web scraping was performed using Python BeautifulSoup library to extract the monthly injection data for each of the well for all the available period. Figure

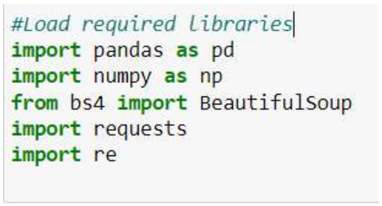
3.3 shows a snapshot of the date for one such SWD well for a period of one year. FileNumber was used as the key to integrate this dataset to the well index and log tops datasets.



FileNumber	Lat	Long	Well Status	UIC Number	Pool	Date	EOR BBLS Injected	EOR MCF Injected	BBLS Salt Water Disposed	Average PSI
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Aug-20	0	0	640	100
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Jul-20	0	0	0	0
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Jun-20	0	0	0	0
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	May-20	0	0	0	0
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Apr-20	0	0	0	0
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Mar-20	0	0	0	0
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Feb-20	0	0	0	0
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Jan-20	0	0	0	0
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Dec-19	0	0	1106	200
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Nov-19	0	0	2419	200
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Oct-19	0	0	2275	200
7395	48.787475	-102.953443	A	W0158S0567C	DAKOTA	Sep-19	0	0	0	0

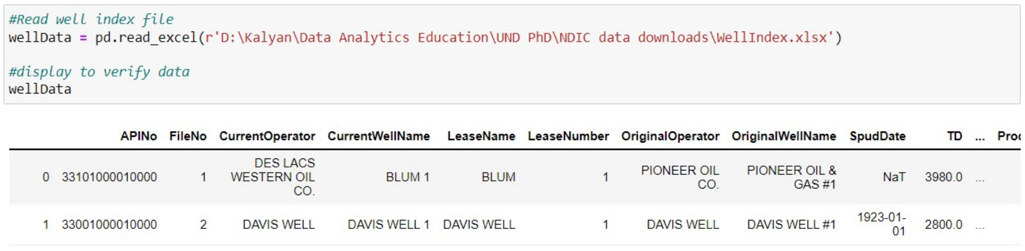
Fig 3.3 Monthly disposal volume and injection pressure for one SWD well

Key Python code snippets to carry out the web scraping are presented below with results where relevant. Jupyter Notebook IDE was used to write, debug and execute the scripts.



```python
#Load required libraries
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
import requests
import re
```

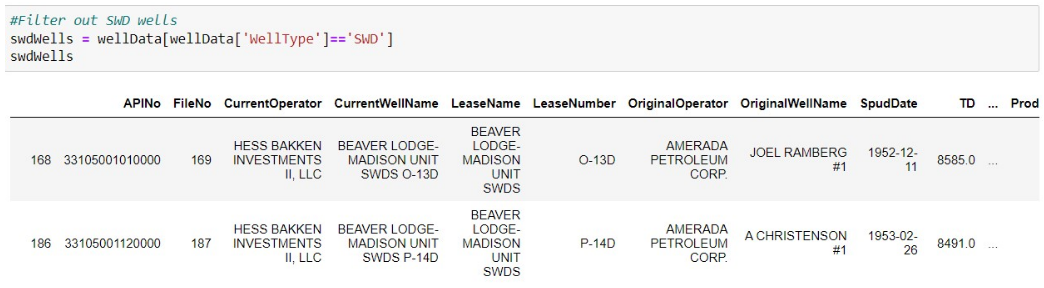
The code snippet below reads the Well Index dataset and displays the top two rows. Python Pandas library was used to carry out this task.



```python
#Read well index file
wellData = pd.read_excel(r'D:\Kalyan\Data Analytics Education\UND PhD\NDIC data downloads\WellIndex.xlsx')

#display to verify data
wellData
```

```
           APINo  FileNo  CurrentOperator  CurrentWellName  LeaseName  LeaseNumber  OriginalOperator  OriginalWellName  SpudDate     TD  ...  Proc
0  33101000010000      1        DES LACS           BLUM 1       BLUM            1       PIONEER OIL     PIONEER OIL &       NaT  3980.0  ...
                             WESTERN OIL                                                       CO.           GAS #1
                                     CO.
1  33001000010000      2      DAVIS WELL     DAVIS WELL 1  DAVIS WELL            1        DAVIS WELL     DAVIS WELL #1  1923-01-  2800.0  ...
                                                                                                                              01
```

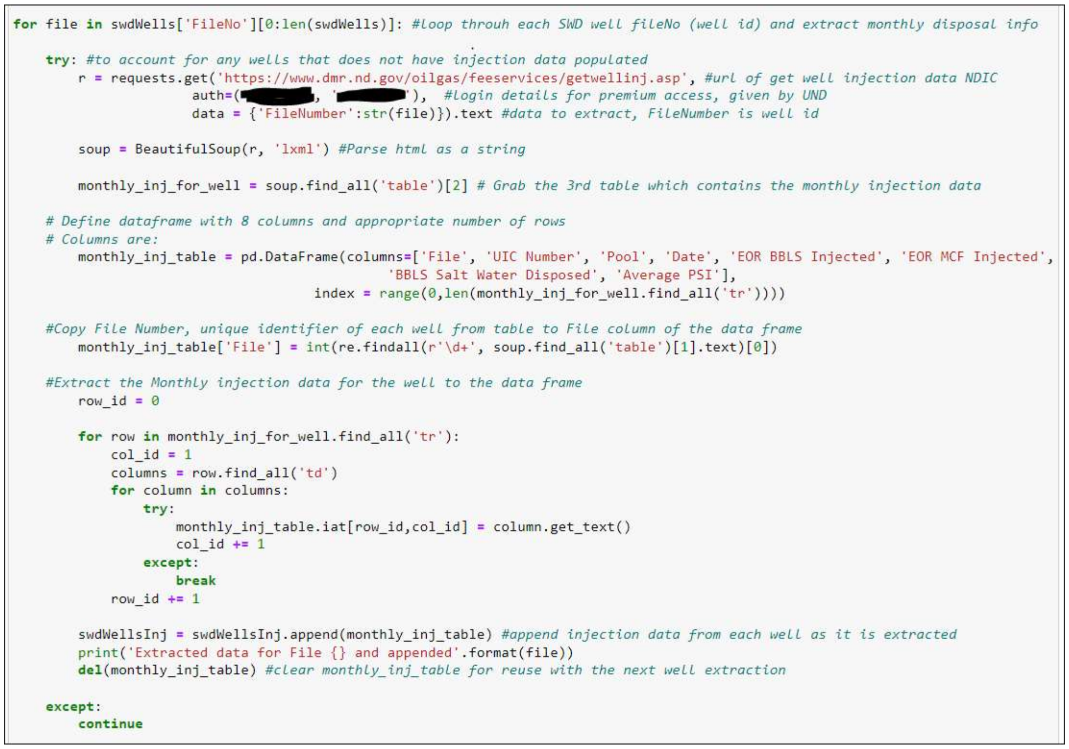
The first code snippet below filters only the SWD wells using the WellType feature set to ‘SWD’. The next code snippet below does the primary task of scarping the NDICOG website that contains the monthly disposal data for all the SWD wells using FileNo as the reference key. Python request library’s get method is used along with the UND



```
#Filter out SWD wells
swdWells = wellData[wellData['WellType']=='SWD']
swdWells
```

	APINo	FileNo	CurrentOperator	CurrentWellName	LeaseName	LeaseNumber	OriginalOperator	OriginalWellName	SpudDate	TD	...	Prod
168	33105001010000	169	HESS BAKKEN INVESTMENTS II, LLC	BEAVER LODGE-MADISON UNIT SWDS O-13D	BEAVER LODGE-MADISON UNIT SWDS	O-13D	AMERADA PETROLEUM CORP.	JOEL RAMBERG #1	1952-12-11	8585.0	...	
186	33105001120000	187	HESS BAKKEN INVESTMENTS II, LLC	BEAVER LODGE-MADISON UNIT SWDS P-14D	BEAVER LODGE-MADISON UNIT SWDS	P-14D	AMERADA PETROLEUM CORP.	A CHRISTENSON #1	1953-02-26	8491.0	...	

premium access authorization details (blacked out) to extract the data for each SWD well. Python BeautifulSoup library's lxml parser is used to parse the HTML page and the soup.find_all method is used to grab the appropriate table with the monthly disposal data as shown in Fig 3.3. Python Regular Expression library re was used to extract the FileNo to a pandas dataframe. Python try / except clause is used where required to handle exceptions (runtime errors).



```
for file in swdWells['FileNo'][0:len(swdWells)]: #Loop throuh each SWD well fileNo (well id) and extract monthly disposal info

    try: #to account for any wells that does not have injection data populated
        r = requests.get('https://www.dmr.nd.gov/oilgas/feeservices/getwellinj.asp', #url of get well injection data NDIC
                     auth=([redacted], '[redacted]'),  #Login details for premium access, given by UND
                     data = {'FileNumber':str(file)}).text #data to extract, FileNumber is well id

        soup = BeautifulSoup(r, 'lxml') #Parse html as a string

        monthly_inj_for_well = soup.find_all('table')[2] # Grab the 3rd table which contains the monthly injection data

    # Define dataframe with 8 columns and appropriate number of rows
    # Columns are:
        monthly_inj_table = pd.DataFrame(columns=['File', 'UIC Number', 'Pool', 'Date', 'EOR BBLS Injected', 'EOR MCF Injected',
                                          'BBLS Salt Water Disposed', 'Average PSI'],
                                 index = range(0,len(monthly_inj_for_well.find_all('tr'))))

    #Copy File Number, unique identifier of each well from table to File column of the data frame
        monthly_inj_table['File'] = int(re.findall(r'\d+', soup.find_all('table')[1].text)[0])

    #Extract the Monthly injection data for the well to the data frame
        row_id = 0

        for row in monthly_inj_for_well.find_all('tr'):
            col_id = 1
            columns = row.find_all('td')
            for column in columns:
                try:
                    monthly_inj_table.iat[row_id,col_id] = column.get_text()
                    col_id += 1
                except:
                    break
            row_id += 1

        swdWellsInj = swdWellsInj.append(monthly_inj_table) #append injection data from each well as it is extracted
        print('Extracted data for File {} and appended'.format(file))
        del(monthly_inj_table) #clear monthly_inj_table for reuse with the next well extraction

    except:
        continue
```

Finally, the Pandas dataframe from the above code snippet that contains the monthly disposal data (Fig 3.2) is integrated with the Well Index data for SWD wells using the built-in pandas join method and using the File Number as the key using the code snippet below and the output is stored as a Microsoft Excel file for further use in the project.

```
#save extracted well injection data to an excel sheet

swdWellsInj.join(swdWells.set_index('FileNo'), on='File').to_excel('SWDwellsInjectionData.xlsx')
```

The SWDwellsInjectionData.xlsx file generated using the above code snipped contains a time series for barrels of SWD disposed and average injection pressure for each of the SWD well in the database, from spud date till the date of data download. For inactive wells, their end time may be earlier than the date of download. An example of the time series for one of the SWD wells is given in Figure 3.4 below.

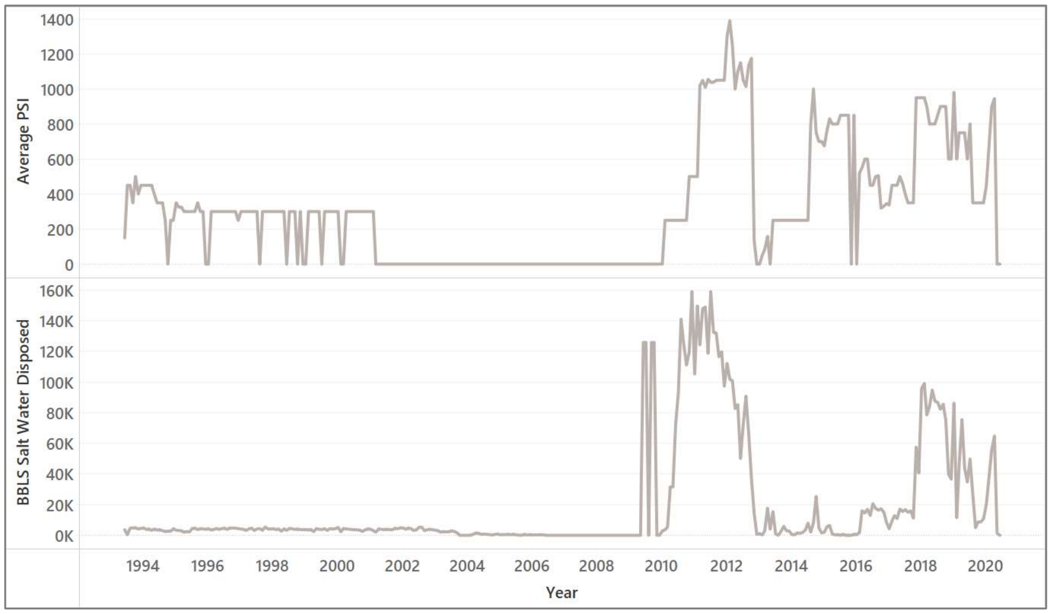
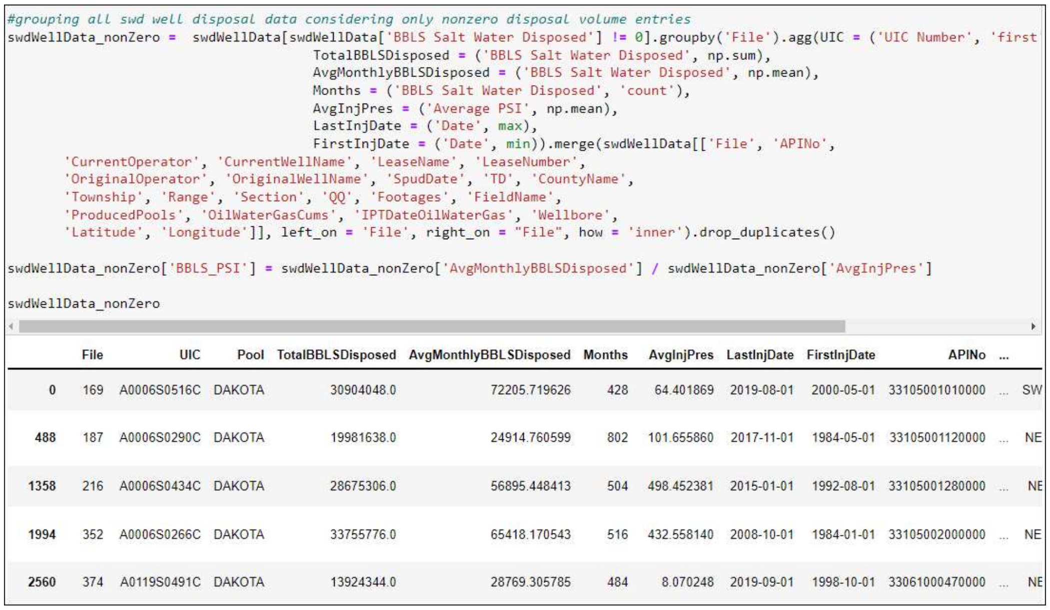


Fig 3.4 Time Series data of SWD barrels disposed and average injection pressure (Example from SWD well 649, plotted using Tableau software)

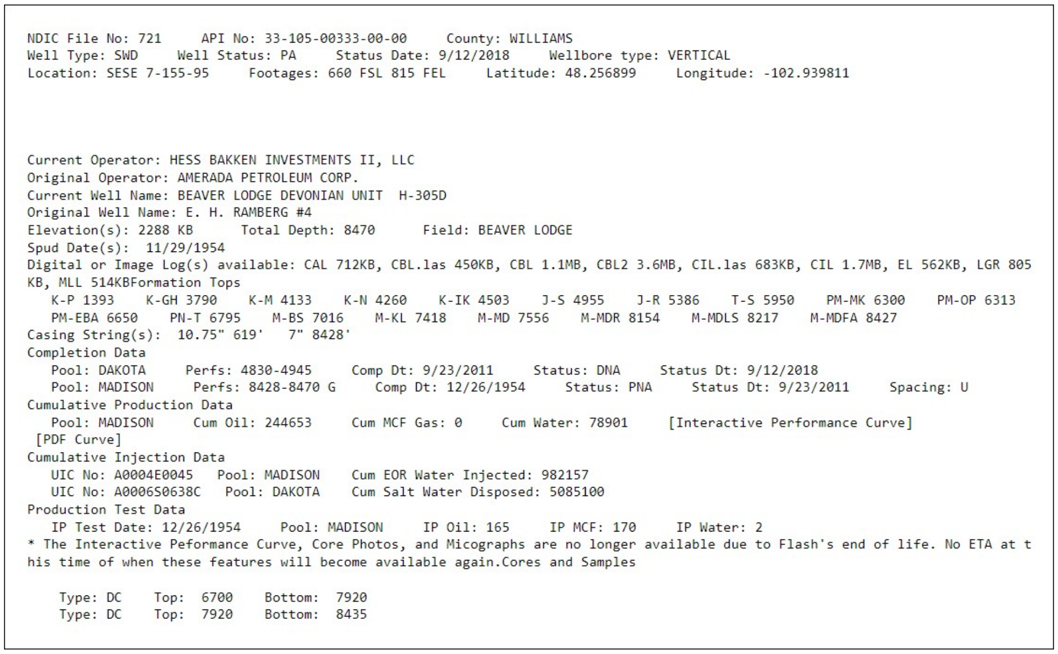
For the requirement of the project analysis, the time series data needs to be aggregated so that there is one value of barrels of SWD disposed per well and one value of average injection pressure. The code snippet below produces aggregated values of barrels of SWD disposed and average of the average monthly injection pressures in pounds per square inch (psi). Total SWD barrels disposed and average monthly barrels disposed were computed along with the total months the well was operating and the average of average monthly injection pressure using the Python Pandas groupby and agg (aggregation) methods. A new feature called BBLS_PSI was also computed as the ratio of average monthly barrels of SWD disposed over the average of average monthly injection pressure. This feature provides a normalized indicator of the SWD well performance.



```
#grouping all swd well disposal data considering only nonzero disposal volume entries
swdWellData_nonZero =  swdWellData[swdWellData['BBLS Salt Water Disposed'] != 0].groupby('File').agg(UIC = ('UIC Number', 'first
                                  TotalBBLSDisposed = ('BBLS Salt Water Disposed', np.sum),
                                  AvgMonthlyBBLSDisposed = ('BBLS Salt Water Disposed', np.mean),
                                  Months = ('BBLS Salt Water Disposed', 'count'),
                                  AvgInjPres = ('Average PSI', np.mean),
                                  LastInjDate = ('Date', max),
                                  FirstInjDate = ('Date', min)).merge(swdWellData[['File', 'APINo',
      'CurrentOperator', 'CurrentWellName', 'LeaseName', 'LeaseNumber',
      'OriginalOperator', 'OriginalWellName', 'SpudDate', 'TD', 'CountyName',
      'Township', 'Range', 'Section', 'QQ', 'Footages', 'FieldName',
      'ProducedPools', 'OilWaterGasCums', 'IPTDateOilWaterGas', 'Wellbore',
      'Latitude', 'Longitude']], left_on = 'File', right_on = "File", how = 'inner').drop_duplicates()

swdWellData_nonZero['BBLS_PSI'] = swdWellData_nonZero['AvgMonthlyBBLSDisposed'] / swdWellData_nonZero['AvgInjPres']

swdWellData_nonZero
```

```
      File        UIC    Pool  TotalBBLSDisposed  AvgMonthlyBBLSDisposed  Months   AvgInjPres  LastInjDate  FirstInjDate           APINo  ...
   0   169  A0006S0516C  DAKOTA        30904048.0            72205.719626     428    64.401869   2019-08-01    2000-05-01  33105001010000  ...  SW
 488   187  A0006S0290C  DAKOTA        19981638.0            24914.760599     802   101.655860   2017-11-01    1984-05-01  33105001120000  ...  NE
1358   216  A0006S0434C  DAKOTA        28675306.0            56895.448413     504   498.452381   2015-01-01    1992-08-01  33105001280000  ...  NE
1994   352  A0006S0266C  DAKOTA        33755776.0            65418.170543     516   432.558140   2008-10-01    1984-01-01  33105002000000  ...  NE
2560   374  A0119S0491C  DAKOTA        13924344.0            28769.305785     484     8.070248   2019-09-01    1998-10-01  33061000470000  ...  NE
```

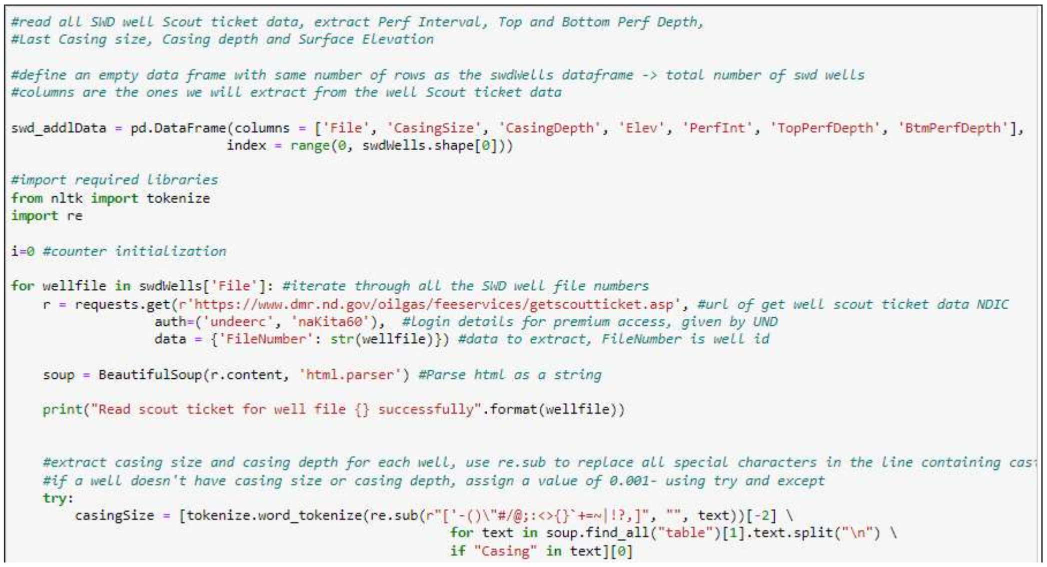
4. **Well Scout Ticket Data Scraping**: NDICOG maintains a well scout ticket data for each of the wells drilled in the state that contains additional essential information about the well from the time it is permitted through drilling, completion and production. Figure 3.5 below shows an example of a well scout ticket data for SWD well 721. File Number for each well (primary key for data integration), Casing Size, Casing Depth, Surface Elevation, 'Perforation Interval, Top and Bottom Perforation depths were some of the additional features extracted from each SWD well's scout ticket through web scraping using the same procedure and technologies described in Section 3 of this Chapter.



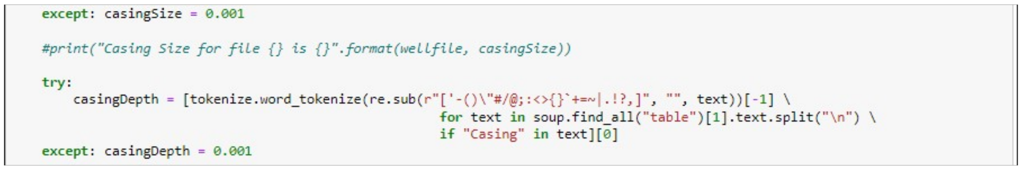
```
NDIC File No: 721     API No: 33-105-00333-00-00     County: WILLIAMS
Well Type: SWD     Well Status: PA     Status Date: 9/12/2018     Wellbore type: VERTICAL
Location: SESE 7-155-95     Footages: 660 FSL 815 FEL     Latitude: 48.256899     Longitude: -102.939811



Current Operator: HESS BAKKEN INVESTMENTS II, LLC
Original Operator: AMERADA PETROLEUM CORP.
Current Well Name: BEAVER LODGE DEVONIAN UNIT  H-305D
Original Well Name: E. H. RAMBERG #4
Elevation(s): 2288 KB      Total Depth: 8470      Field: BEAVER LODGE
Spud Date(s):  11/29/1954
Digital or Image Log(s) available: CAL 712KB, CBL.las 450KB, CBL 1.1MB, CBL2 3.6MB, CIL.las 683KB, CIL 1.7MB, EL 562KB, LGR 805
KB, MLL 514KBFormation Tops
   K-P 1393    K-GH 3790    K-M 4133    K-N 4260    K-IK 4503    J-S 4955    J-R 5386    T-S 5950    PM-MK 6300    PM-OP 6313
   PM-EBA 6650    PN-T 6795    M-BS 7016    M-KL 7418    M-MD 7556    M-MDR 8154    M-MDLS 8217    M-MDFA 8427
Casing String(s):  10.75" 619'   7" 8428'
Completion Data
   Pool: DAKOTA     Perfs: 4830-4945     Comp Dt: 9/23/2011     Status: DNA     Status Dt: 9/12/2018
   Pool: MADISON     Perfs: 8428-8470 G     Comp Dt: 12/26/1954     Status: PNA     Status Dt: 9/23/2011     Spacing: U
Cumulative Production Data
   Pool: MADISON     Cum Oil: 244653     Cum MCF Gas: 0     Cum Water: 78901     [Interactive Performance Curve]
 [PDF Curve]
Cumulative Injection Data
   UIC No: A0004E0045    Pool: MADISON    Cum EOR Water Injected: 982157
   UIC No: A0006S0638C   Pool: DAKOTA    Cum Salt Water Disposed: 5085100
Production Test Data
   IP Test Date: 12/26/1954     Pool: MADISON     IP Oil: 165     IP MCF: 170     IP Water: 2
* The Interactive Peformance Curve, Core Photos, and Micographs are no longer available due to Flash's end of life. No ETA at t
his time of when these features will become available again.Cores and Samples

    Type: DC    Top:  6700    Bottom:  7920
    Type: DC    Top:  7920    Bottom:  8435
```

Fig 3.5 Well Scout Ticket Data example (SWD well 721, from NDICOG)

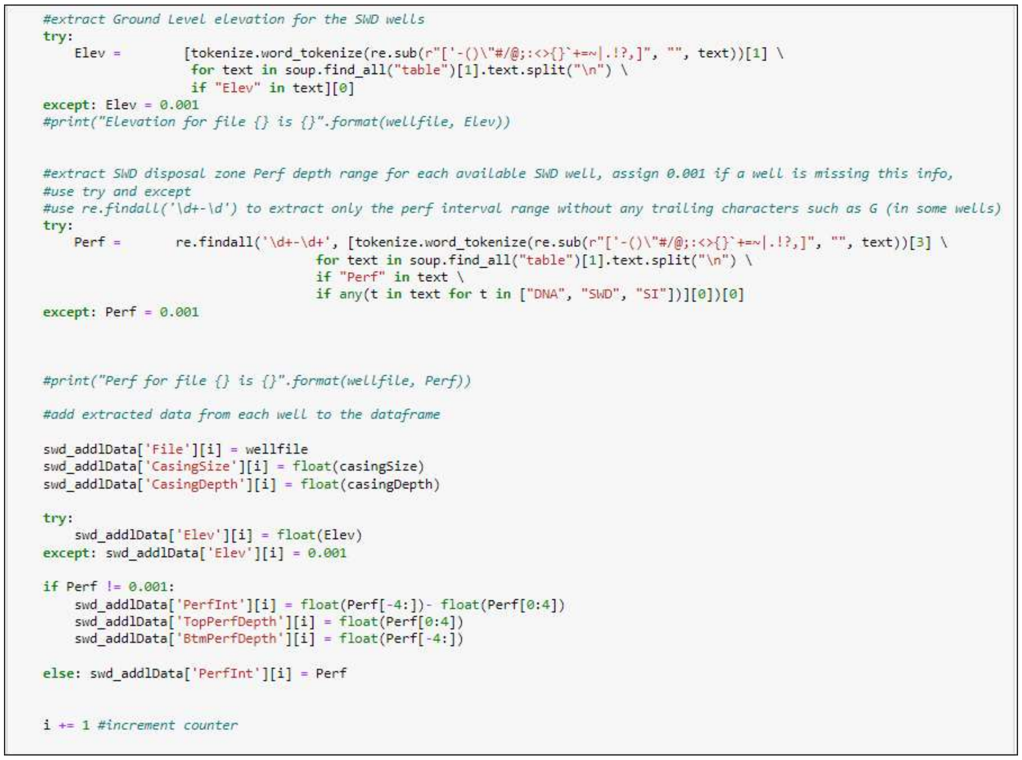
The code snippet shown below completed the task of scarping the well scout ticket for all the SWD wells in the database. In order to deal with string manipulation effectively as part of the scraping process, the tokenize method of the Python NLTK platform was used along with regular expressions.



```python
#read all SWD well Scout ticket data, extract Perf Interval, Top and Bottom Perf Depth,
#Last Casing size, Casing depth and Surface Elevation

#define an empty data frame with same number of rows as the swdWells dataframe -> total number of swd wells
#columns are the ones we will extract from the well Scout ticket data

swd_addlData = pd.DataFrame(columns = ['File', 'CasingSize', 'CasingDepth', 'Elev', 'PerfInt', 'TopPerfDepth', 'BtmPerfDepth'],
                            index = range(0, swdWells.shape[0]))

#import required libraries
from nltk import tokenize
import re

i=0 #counter initialization

for wellfile in swdWells['File']: #iterate through all the SWD well file numbers
    r = requests.get(r'https://www.dmr.nd.gov/oilgas/feeservices/getscoutticket.asp', #url of get well scout ticket data NDIC
                     auth=('undeerc', 'naKita60'),  #login details for premium access, given by UND
                     data = {'FileNumber': str(wellfile)}) #data to extract, FileNumber is well id

    soup = BeautifulSoup(r.content, 'html.parser') #Parse html as a string

    print("Read scout ticket for well file {} successfully".format(wellfile))


    #extract casing size and casing depth for each well, use re.sub to replace all special characters in the line containing cas
    #if a well doesn't have casing size or casing depth, assign a value of 0.001- using try and except
    try:
        casingSize = [tokenize.word_tokenize(re.sub(r"['-()\"#/@;:<>{}`+=~|!?,]", "", text))[-2] \
                                                  for text in soup.find_all("table")[1].text.split("\n") \
                                                  if "Casing" in text][0]
```



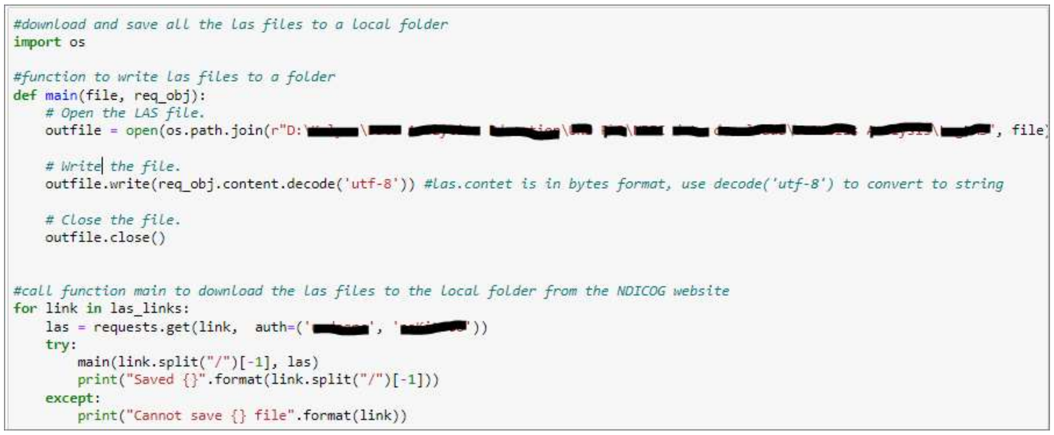
```
    except: casingSize = 0.001

    #print("Casing Size for file {} is {}".format(wellfile, casingSize))

    try:
        casingDepth = [tokenize.word_tokenize(re.sub(r"['-()\"#/@;:<>{}`+=~|.!?,]", "", text))[-1] \
                                                    for text in soup.find_all("table")[1].text.split("\n") \
                                                    if "Casing" in text][0]
    except: casingDepth = 0.001
```



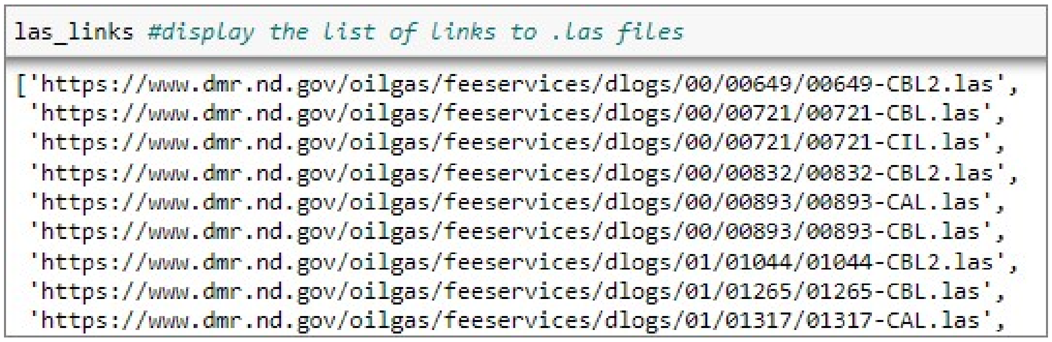
```
    #extract Ground Level elevation for the SWD wells
    try:
        Elev =        [tokenize.word_tokenize(re.sub(r"['-()\"#/@;:<>{}`+=~|.!?,]", "", text))[1] \
                       for text in soup.find_all("table")[1].text.split("\n") \
                       if "Elev" in text][0]
    except: Elev = 0.001
    #print("Elevation for file {} is {}".format(wellfile, Elev))


    #extract SWD disposal zone Perf depth range for each available SWD well, assign 0.001 if a well is missing this info,
    #use try and except
    #use re.findall('\d+-\d') to extract only the perf interval range without any trailing characters such as G (in some wells)
    try:
        Perf =       re.findall('\d+-\d+', [tokenize.word_tokenize(re.sub(r"['-()\"#/@;:<>{}`+=~|.!?,]", "", text))[3] \
                                  for text in soup.find_all("table")[1].text.split("\n") \
                                  if "Perf" in text \
                                  if any(t in text for t in ["DNA", "SWD", "SI"])][0])[0]
    except: Perf = 0.001



    #print("Perf for file {} is {}".format(wellfile, Perf))

    #add extracted data from each well to the dataframe

    swd_addlData['File'][i] = wellfile
    swd_addlData['CasingSize'][i] = float(casingSize)
    swd_addlData['CasingDepth'][i] = float(casingDepth)

    try:
        swd_addlData['Elev'][i] = float(Elev)
    except: swd_addlData['Elev'][i] = 0.001

    if Perf != 0.001:
        swd_addlData['PerfInt'][i] = float(Perf[-4:])- float(Perf[0:4])
        swd_addlData['TopPerfDepth'][i] = float(Perf[0:4])
        swd_addlData['BtmPerfDepth'][i] = float(Perf[-4:])

    else: swd_addlData['PerfInt'][i] = Perf


    i += 1 #increment counter
```

5. **Log Data Scraping**: NDICOG well scout ticket for each SWD well contained the file names of log data files with different log data where they were available (Fig 3.5). Such log data were either in digital format (.las extension, LAS- Log ASCII Standard) or image raster files (.tiff). Our interest was the digital files that contained log data representing subsurface such as Gamma Ray, Resistivity and Porosity. Scraping was carried out using the get method of Python request library and BeautifulSoup html parser to identify each SWD well scout ticket data using the File Number as the key. Only for those wells that contained log data in digital format (.las extension), the .las file hyperlinks were stored in a Python list. Figure 3.6 below shows a snapshot of LAS file links extracted from a few wells. Only 41% of all the SWD wells (369 wells) contained log data in LAS format and a total of 969 files were downloaded directly

using Python script to a local folder using the code snippet shown below. The Python OS library's path.join method was used to point to the local folder path and the LAS file content which was in byte format was decoded using the utf-8 format (UTF-8 is a variable-width character encoding used for electronic communication, defined by the Unicode Standard) prior to download.



```
#download and save all the las files to a local folder
import os

#function to write las files to a folder
def main(file, req_obj):
    # Open the LAS file.
    outfile = open(os.path.join(r"D:[redacted]', file

    # Write the file.
    outfile.write(req_obj.content.decode('utf-8')) #las.contet is in bytes format, use decode('utf-8') to convert to string

    # Close the file.
    outfile.close()


#call function main to download the las files to the local folder from the NDICOG website
for link in las_links:
    las = requests.get(link,  auth=('[redacted]', '[redacted]'))
    try:
        main(link.split("/")[-1], las)
        print("Saved {}".format(link.split("/")[-1]))
    except:
        print("Cannot save {} file".format(link))
```

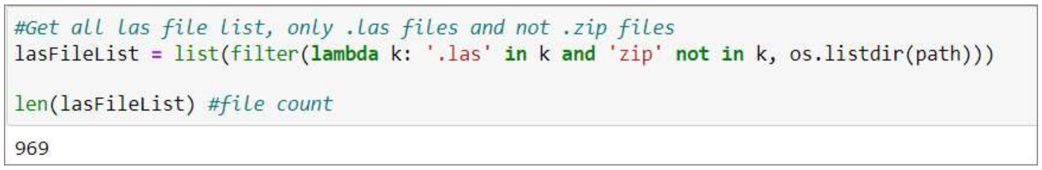


```
las_links #display the list of links to .las files
```

```
['https://www.dmr.nd.gov/oilgas/feeservices/dlogs/00/00649/00649-CBL2.las',
 'https://www.dmr.nd.gov/oilgas/feeservices/dlogs/00/00721/00721-CBL.las',
 'https://www.dmr.nd.gov/oilgas/feeservices/dlogs/00/00721/00721-CIL.las',
 'https://www.dmr.nd.gov/oilgas/feeservices/dlogs/00/00832/00832-CBL2.las',
 'https://www.dmr.nd.gov/oilgas/feeservices/dlogs/00/00893/00893-CAL.las',
 'https://www.dmr.nd.gov/oilgas/feeservices/dlogs/00/00893/00893-CBL.las',
 'https://www.dmr.nd.gov/oilgas/feeservices/dlogs/01/01044/01044-CBL2.las',
 'https://www.dmr.nd.gov/oilgas/feeservices/dlogs/01/01265/01265-CBL.las',
 'https://www.dmr.nd.gov/oilgas/feeservices/dlogs/01/01317/01317-CAL.las',
```

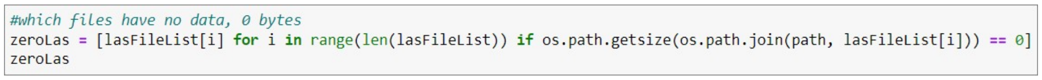
Fig 3.6 LAS file links extracted from well scout ticket data using web scraping

The next step was to read all the LAS files using the Python lasio library using the code snippets below. From the 969 LAS files, only 900 files contained data.

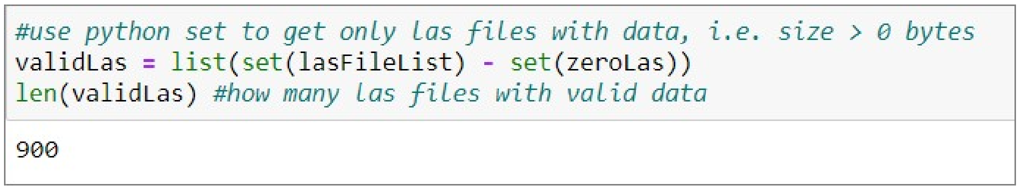


```
#Get all las file list, only .las files and not .zip files
lasFileList = list(filter(lambda k: '.las' in k and 'zip' not in k, os.listdir(path)))

len(lasFileList) #file count
```

```
969
```



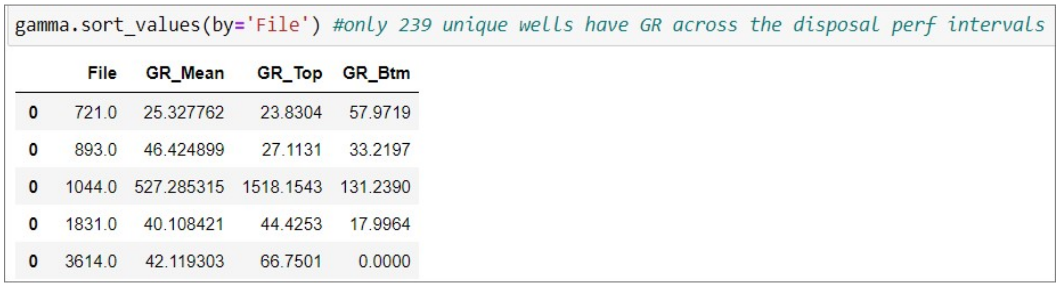
```
#which files have no data, 0 bytes
zeroLas = [lasFileList[i] for i in range(len(lasFileList)) if os.path.getsize(os.path.join(path, lasFileList[i])) == 0]
zeroLas
```



```
#use python set to get only las files with data, i.e. size > 0 bytes
validLas = list(set(lasFileList) - set(zeroLas))
len(validLas) #how many las files with valid data
```

```
900
```

Inspecting the content of the log data from the LAS files revealed that Gamma Ray measurement was the only measurement available in majority of the 369 wells while other measurements such as resistivity and porosity were available in a very small number of wells and would not be useful for data mining purposes. Gamma ray curves were available in different curve nomenclatures such as GR, GAMMA, Gamma etc., and naming inconsistencies were addressed. Since log data is sequential data in nature with the depth as index, the data needs to be aggregated so that there is one representative value of gamma ray per well per aggregation. Such log data were extracted only across the perforated interval which contributes to the SWD operations in a given well, extracted from the given well's scout ticket data as explained in Section 4 of this chapter. Three gamma ray values (Mean, Top and Bottom) were generated for 239 wells out of the 369 wells as shown the code snippet below to be integrated with the other data sources explained earlier in this chapter using well file number as the key.



```
gamma.sort_values(by='File') #only 239 unique wells have GR across the disposal perf intervals
```

	File	GR_Mean	GR_Top	GR_Btm
0	721.0	25.327762	23.8304	57.9719
0	893.0	46.424899	27.1131	33.2197
0	1044.0	527.285315	1518.1543	131.2390
0	1831.0	40.108421	44.4253	17.9964
0	3614.0	42.119303	66.7501	0.0000

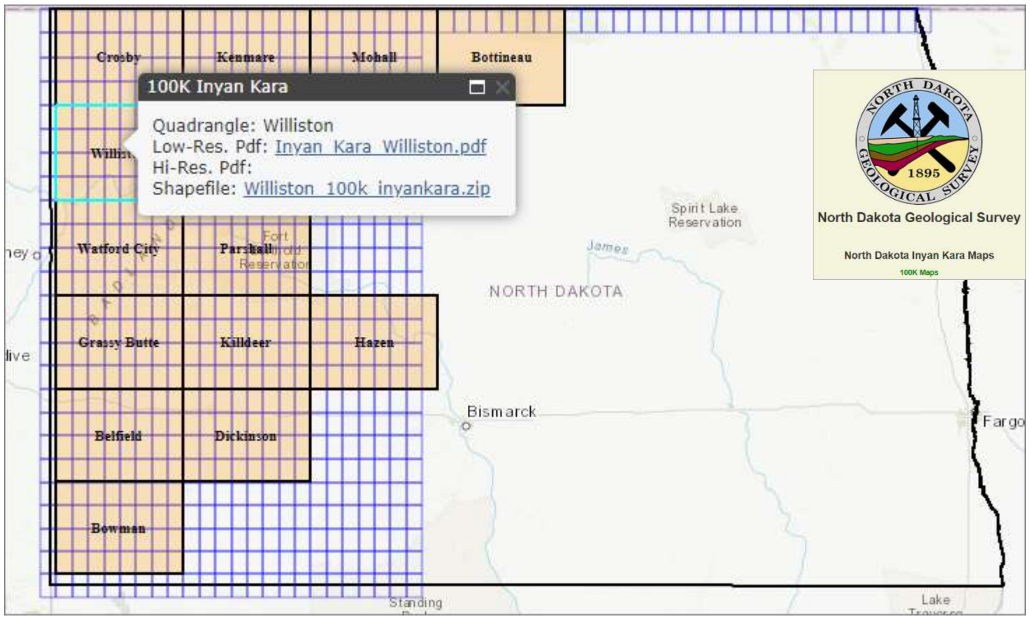
3.2.2 Secondary Data Sources

After spending adequate time to do a thorough review of various data sets available within NDICOG, the primary data source, we examined other secondary data sources listed in Table 3.22 below.

Table 3.2 Data Sets from Secondary Data Sources

	Data Set	Type	Source
a.	North Dakota Inyan Kara Isopach Maps	Shapefile	North Dakota Geological Survey (Inyan Kara Maps)
b.	North Dakota Roads	Shapefile	North Dakota GIS Hub
c.	North Dakota Oilfield Waste Disposal Sites	Shapefile	North Dakota GIS Hub
d.	North Dakota Oil and Gas production, injection and disposal wells – Wells and Production Tables	Flat File (Excel)	Envervus
e.	Bakken Oil and Gas Wells Production Data	Flat File (Excel)	WellDatabase

a. **North Dakota Inyan Kara Isopach Maps :** Since the objective of the research project is to develop a data driven proxy model for SWD wells in ND, considering as many subsurface features as possible will help develop a robust model. From the various data sets extracted from the primary data source, one key subsurface feature- the formation top depth of the Inyan Kara (K-IK) formation was available. Perforation interval was also computed from the top and bottom perforation interval depths extracted from the well scout ticket data for each SWD well along with gamma ray log information. However, gamma ray log data was available for only 239 of the 905 SWD wells. Through additional literature reviews and meetings with technical experts at NDIC and EERC, this important secondary data source was identified that contained valuable thickness information of the Inyan Kara formation across the ND state where oil and gas wells including disposal wells were drilled. Figure 3.7 from the data source of this data set shows the various quadrangles in the state of ND for which Inyan Kara Isposach shapefiles were available. They were all downloaded to a local folder and the Python GeoPandas, Shapley and EarthPy libraries were used to analyze these files, extract the relevant data and integrate them with the SWD data from the data sets in the primary data source.


100K Inyan Kara

Quadrangle: Williston
Low-Res. Pdf: Inyan_Kara_Williston.pdf
Hi-Res. Pdf:
Shapefile: Williston_100k_inyankara.zip

Fig 3.7 Inyan Kara Isopach Map Quadrangles

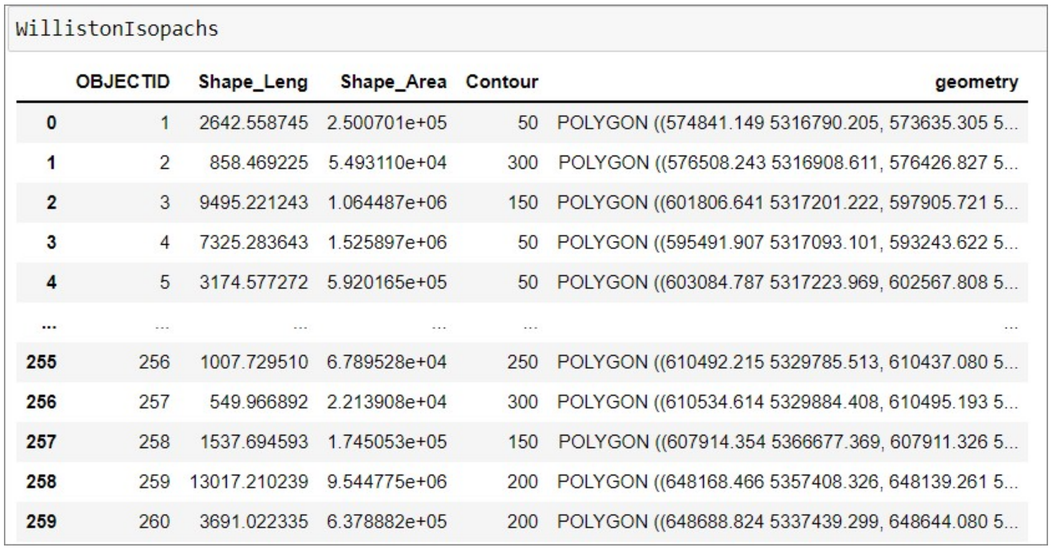
For each quadrangle in Figure 3.7, three shape files were available, one containing all the oil and gas wells in the region, one containing all the SWD and injector wells in the region and one with the Inyan Kara Isopach intervals. A PDF file was also available for each of the quadrangle that identified SWD well performance based on the BBLS_PSI normalized feature. A new categorical feature called 'Well_Quality' was created using this criterion as a reference. The code snippet below was used to accomplish this task.

```
def get_well_quality(bbls_psi):
    if (bbls_psi > 150):
        return ('GOOD')
    elif (bbls_psi > 50):
        return ('AVEGARGE')
    else :
        return ('POOR')
```

```
swdWellsGeo['Well_Quality'] = swdWellsGeo['BBLS_PSI'][swdWellsGeo['BBLS_PSI'] != np.inf].apply (lambda x: get_well_quality(x))
```

The GeoPandas library's read_file method was used to read all the shape files. An example of the Isopach shape file from the Williston quadrangle is shown below in

Figure 3.8. The column 'contour' contains the average thickness of the Inyan Kara formation in feet within the contour. The 'geometry' column provided the boundary coordinates of the contour and the length and area of the contour were also provided.



```
WillistonIsopachs
```

	OBJECTID	Shape_Leng	Shape_Area	Contour	geometry
0	1	2642.558745	2.500701e+05	50	POLYGON ((574841.149 5316790.205, 573635.305 5...
1	2	858.469225	5.493110e+04	300	POLYGON ((576508.243 5316908.611, 576426.827 5...
2	3	9495.221243	1.064487e+06	150	POLYGON ((601806.641 5317201.222, 597905.721 5...
3	4	7325.283643	1.525897e+06	50	POLYGON ((595491.907 5317093.101, 593243.622 5...
4	5	3174.577272	5.920165e+05	50	POLYGON ((603084.787 5317223.969, 602567.808 5...
...	...	...	...	...	...
255	256	1007.729510	6.789528e+04	250	POLYGON ((610492.215 5329785.513, 610437.080 5...
256	257	549.966892	2.213908e+04	300	POLYGON ((610534.614 5329884.408, 610495.193 5...
257	258	1537.694593	1.745053e+05	150	POLYGON ((607914.354 5366677.369, 607911.326 5...
258	259	13017.210239	9.544775e+06	200	POLYGON ((648168.466 5357408.326, 648139.261 5...
259	260	3691.022335	6.378882e+05	200	POLYGON ((648688.824 5337439.299, 648644.080 5...

Fig 3.8 Isopach Shapefile Contents for the Williston Quadrangle

Coordinate Reference System (CRS): Spatial data such as wells are defined using a CRS. According to earthdatascience.org (2020), a coordinate reference system (CRS) is a coordinate-based local, regional or global system used to locate geographical entities. The coordinate reference system is made up of several key components:

- Coordinate System: The X, Y grid upon which the data is overlaid and how where a point is located in space is defined.
- Horizontal and vertical units: The units used to define the grid along the x, y (and z) axis.
- Datum: A modeled version of the shape of the earth which defines the origin used to place the coordinate system in space.
- Projection Information: The mathematical equation used to flatten objects that are on a round surface (e.g. the earth) so you can view them on a flat surface (e.g. your computer screens or a paper map).

Figure 3.9 below shows how a point on the Earth's surface is defined in a 2-dimensional coordinate system.

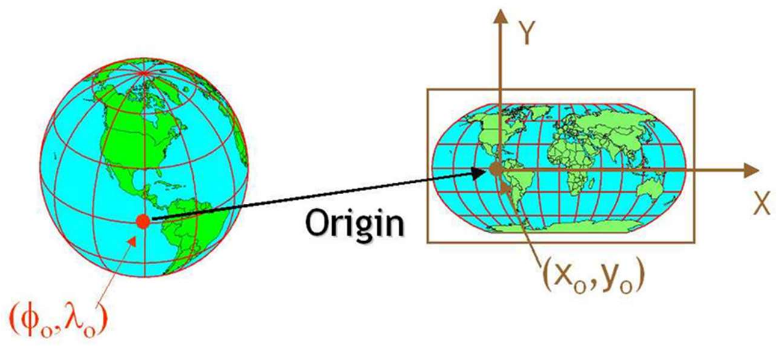
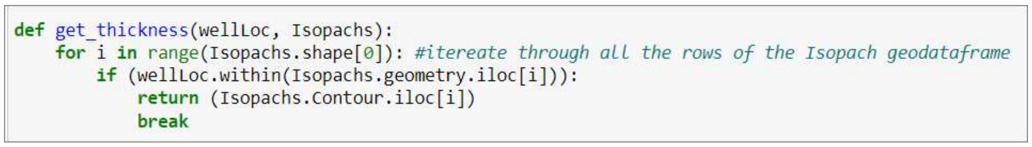


Fig 3.9 Representation of a point on Earth's surface on a 2-d CRS (earthdatascience.org, 2020)

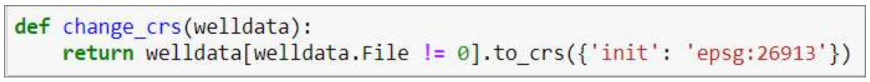
One popular format to document a CRS is the European Petroleum Survey Group (EPSG) format where a 4-5 digit code is used to represent a CRS. As an example, EPSG 4326 represents the 2-dimensional WGS84 geographic CRS.

The Python function below was used to get the Isopach thickness based on the location of each of the SWD well by considering the Inyan Kara Isopach files for all the quadrangles shown in Fig 3.7 and returning the thickness of the contour using the 'within' method of the GeoPandas library.



```python
def get_thickness(wellLoc, Isopachs):
    for i in range(Isopachs.shape[0]): #itereate through all the rows of the Isopach geodataframe
        if (wellLoc.within(Isopachs.geometry.iloc[i])):
            return (Isopachs.Contour.iloc[i])
            break
```

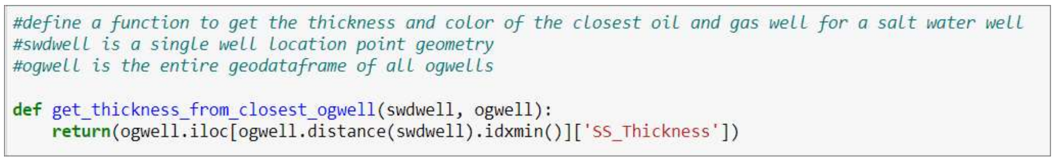
The well location point geometry was converted from Lat-Long to EPSG: 26913 - nad83 / utm zone 13n CRS using the Python function below, prior to calling the above function.



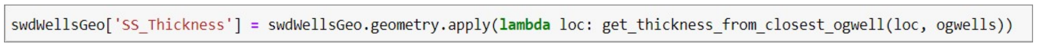
```python
def change_crs(welldata):
    return welldata[welldata.File != 0].to_crs({'init': 'epsg:26913'})
```

The function returned a thickness value for each of the SWD well's Inyan Kara formation from the data base. Upon further analysis, it was noticed that quite a few SWD wells had a null value for the Inyan Kara thickness.

Considering that there were over 35000 oil and gas wells that penetrated the Inyan Kara formation and that the thickness information was available as part of the oil and gas wells shapefile for each of the quadrangles in Fig 3.7, a spatial join operation was carried out to identify the closest oil and gas well to each of the SWD well and the associated thickness of the Inyan-Kara formation of the closest oil and gas well was assigned as the thickness value of the Inyan-Kara formation of the SWD well which resulted in a much more complete thickness column. The code snippets used to carry out this operation are given below:

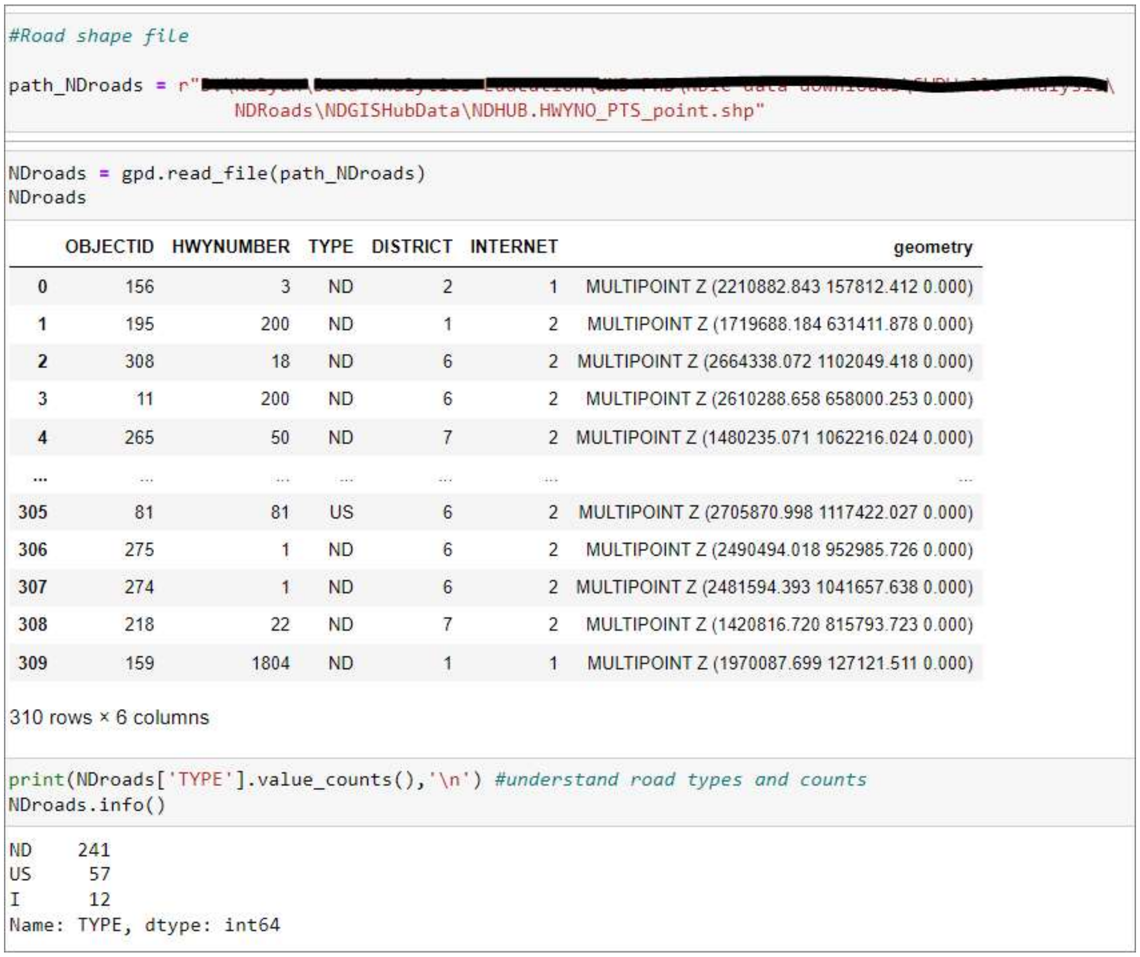


```
#define a function to get the thickness and color of the closest oil and gas well for a salt water well
#swdwell is a single well location point geometry
#ogwell is the entire geodataframe of all ogwells

def get_thickness_from_closest_ogwell(swdwell, ogwell):
    return(ogwell.iloc[ogwell.distance(swdwell).idxmin()]['SS_Thickness'])
```



```
swdWellsGeo['SS_Thickness'] = swdWellsGeo.geometry.apply(lambda loc: get_thickness_from_closest_ogwell(loc, ogwells))
```

b. **North Dakota Roads:** One of the critical factors in finalizing the location of an oil well, gas well, injection well or disposal well in the state of ND or in the entire country (United States of America) is the proximity of the well site to established roads in the state to minimize transportation related hurdles and additional road construction costs. Hence, this secondary data source relating to road infrastructure of all the roads in the state of ND was located from the ND GIS Hub website and downloaded as a shapefile. The code snippet below was used to read the shape file. A total of 310 road entries were identified, 12 of which were Interstate roads, 57 were US roads and the rest were ND roads.



```
#Road shape file

path_NDroads = r"[redacted]\
                 NDRoads\NDGISHubData\NDHUB.HWYNO_PTS_point.shp"
```

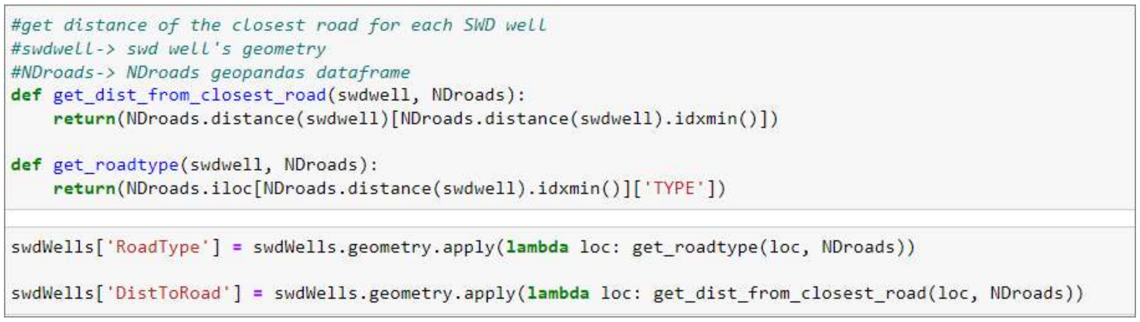
```
NDroads = gpd.read_file(path_NDroads)
NDroads
```

```
     OBJECTID  HWYNUMBER  TYPE  DISTRICT  INTERNET                                         geometry
0         156          3    ND         2         1   MULTIPOINT Z (2210882.843 157812.412 0.000)
1         195        200    ND         1         2   MULTIPOINT Z (1719688.184 631411.878 0.000)
2         308         18    ND         6         2  MULTIPOINT Z (2664338.072 1102049.418 0.000)
3          11        200    ND         6         2   MULTIPOINT Z (2610288.658 658000.253 0.000)
4         265         50    ND         7         2  MULTIPOINT Z (1480235.071 1062216.024 0.000)
...       ...        ...   ...       ...       ...                                              ...
305        81         81    US         6         2  MULTIPOINT Z (2705870.998 1117422.027 0.000)
306       275          1    ND         6         2   MULTIPOINT Z (2490494.018 952985.726 0.000)
307       274          1    ND         6         2  MULTIPOINT Z (2481594.393 1041657.638 0.000)
308       218         22    ND         7         2   MULTIPOINT Z (1420816.720 815793.723 0.000)
309       159       1804    ND         1         1   MULTIPOINT Z (1970087.699 127121.511 0.000)

310 rows × 6 columns
```

```
print(NDroads['TYPE'].value_counts(),'\n') #understand road types and counts
NDroads.info()
```

```
ND    241
US     57
I      12
Name: TYPE, dtype: int64
```

The next step was to use spatial join to identify the closest road to each of the SWD well in the data base, extract the distance of that road in meters along with the road type and integrate them as two additional features for the final data set. This was accomplished using the code snippet below.



```
#get distance of the closest road for each SWD well
#swdwell-> swd well's geometry
#NDroads-> NDroads geopandas dataframe
def get_dist_from_closest_road(swdwell, NDroads):
    return(NDroads.distance(swdwell)[NDroads.distance(swdwell).idxmin()])

def get_roadtype(swdwell, NDroads):
    return(NDroads.iloc[NDroads.distance(swdwell).idxmin()]['TYPE'])
```

```
swdWells['RoadType'] = swdWells.geometry.apply(lambda loc: get_roadtype(loc, NDroads))

swdWells['DistToRoad'] = swdWells.geometry.apply(lambda loc: get_dist_from_closest_road(loc, NDroads))
```

c. **North Dakota Oilfield Waste Disposal Sites:** Figure 3.10 below shows the water lifecycle for unconventional oil and gas production.

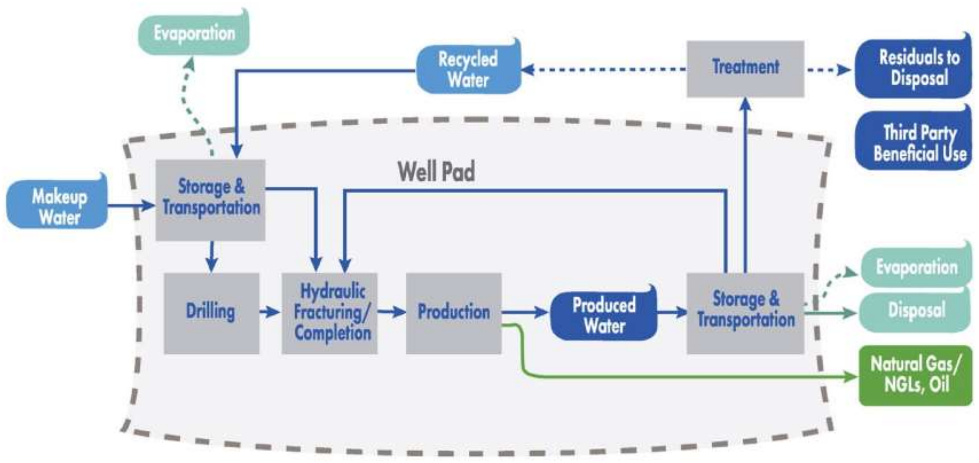
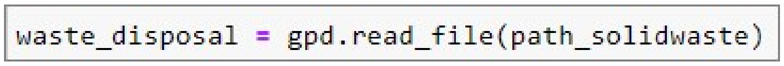
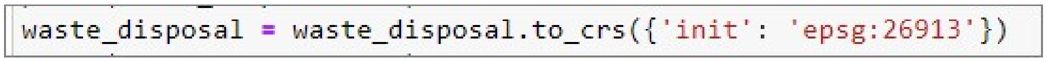


Fig 3.10 Water Lifecycle- Unconventional Oil and Gas Production
(Source: Produced Water Report, GWPC, 2019, page 10)

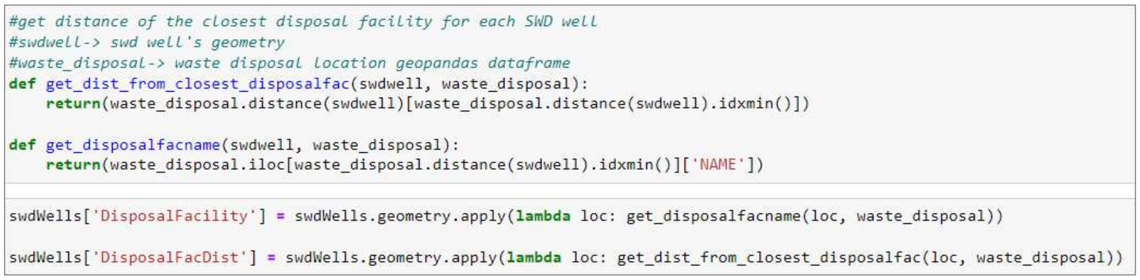
One can see from the above figure the need for treating the produced water (saltwater) prior to recycling or disposal to remove any residual hydrocarbons and other solids. This motivated us to look for data relating to oilfield waste disposal sites that would potentially treat produced water and prepare it ready for disposal. This secondary data source relating to oilfield waste disposal sites in the state of ND was located from the ND GIS Hub website and downloaded as a shapefile. The code snippet below was used to read the shape file using the geopandas read_file method and information related to 10 such sites were extracted as a geopandas dataframe.


```
waste_disposal = gpd.read_file(path_solidwaste)
```

CRS was converted using the code snippet below before carrying out spatial join operations.


```
waste_disposal = waste_disposal.to_crs({'init': 'epsg:26913'})
```

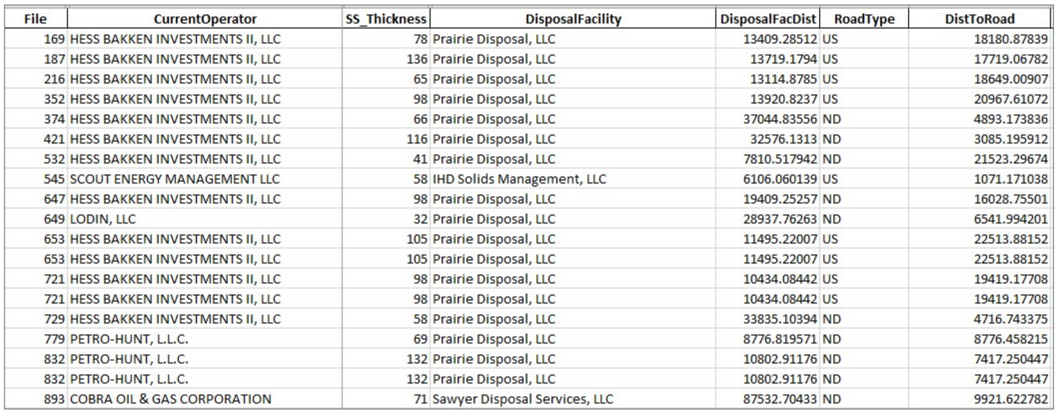
The next step was to use spatial join to identify the closest disposal site to each of the SWD well in the data base, extract the distance of that site in meters, and integrate it as an additional feature for the final data set. This was accomplished using the code snippet below.



```
#get distance of the closest disposal facility for each SWD well
#swdwell-> swd well's geometry
#waste_disposal-> waste disposal location geopandas dataframe
def get_dist_from_closest_disposalfac(swdwell, waste_disposal):
    return(waste_disposal.distance(swdwell)[waste_disposal.distance(swdwell).idxmin()])

def get_disposalfacname(swdwell, waste_disposal):
    return(waste_disposal.iloc[waste_disposal.distance(swdwell).idxmin()]['NAME'])
```

```
swdWells['DisposalFacility'] = swdWells.geometry.apply(lambda loc: get_disposalfacname(loc, waste_disposal))

swdWells['DisposalFacDist'] = swdWells.geometry.apply(lambda loc: get_dist_from_closest_disposalfac(loc, waste_disposal))
```

Figure 3.11 below shows an example collection of a few SWD wells (identified by the file number key) with all the additional features extracted from the secondary data sources integrated together. These features were integrated with the data from the primary data source to obtain the final data set on which exploratory data analysis and data mining activities were carried out.



File	CurrentOperator	SS_Thickness	DisposalFacility	DisposalFacDist	RoadType	DistToRoad
169	HESS BAKKEN INVESTMENTS II, LLC	78	Prairie Disposal, LLC	13409.28512	US	18180.87839
187	HESS BAKKEN INVESTMENTS II, LLC	136	Prairie Disposal, LLC	13719.1794	US	17719.06782
216	HESS BAKKEN INVESTMENTS II, LLC	65	Prairie Disposal, LLC	13114.8785	US	18649.00907
352	HESS BAKKEN INVESTMENTS II, LLC	98	Prairie Disposal, LLC	13920.8237	US	20967.61072
374	HESS BAKKEN INVESTMENTS II, LLC	66	Prairie Disposal, LLC	37044.83556	ND	4893.173836
421	HESS BAKKEN INVESTMENTS II, LLC	116	Prairie Disposal, LLC	32576.1313	ND	3085.195912
532	HESS BAKKEN INVESTMENTS II, LLC	41	Prairie Disposal, LLC	7810.517942	ND	21523.29674
545	SCOUT ENERGY MANAGEMENT LLC	58	IHD Solids Management, LLC	6106.060139	US	1071.171038
647	HESS BAKKEN INVESTMENTS II, LLC	98	Prairie Disposal, LLC	19409.25257	ND	16028.75501
649	LODIN, LLC	32	Prairie Disposal, LLC	28937.76263	ND	6541.994201
653	HESS BAKKEN INVESTMENTS II, LLC	105	Prairie Disposal, LLC	11495.22007	US	22513.88152
653	HESS BAKKEN INVESTMENTS II, LLC	105	Prairie Disposal, LLC	11495.22007	US	22513.88152
721	HESS BAKKEN INVESTMENTS II, LLC	98	Prairie Disposal, LLC	10434.08442	US	19419.17708
721	HESS BAKKEN INVESTMENTS II, LLC	98	Prairie Disposal, LLC	10434.08442	US	19419.17708
729	HESS BAKKEN INVESTMENTS II, LLC	58	Prairie Disposal, LLC	33835.10394	ND	4716.743375
779	PETRO-HUNT, L.L.C.	69	Prairie Disposal, LLC	8776.819571	ND	8776.458215
832	PETRO-HUNT, L.L.C.	132	Prairie Disposal, LLC	10802.91176	ND	7417.250447
832	PETRO-HUNT, L.L.C.	132	Prairie Disposal, LLC	10802.91176	ND	7417.250447
893	COBRA OIL & GAS CORPORATION	71	Sawyer Disposal Services, LLC	87532.70433	ND	9921.622782

Fig 3.11 Sample SWD wells with additional features from the secondary data sources

d. **North Dakota Oil and Gas production, injection and disposal wells – Wells and Production Tables:** This data set was downloaded from Enervus.com, a data, software and insights company that serves the energy industry. From their Exploration and Production services, two flat files (Microsoft Excel) were downloaded much later in the project using the student subscription obtained by the department of petroleum engineering at UND. Their primary use in the project is outlined below:

- Since the wells table contained bottom hole latitude and longitude for all the SWD wells, two additional subsurface features were extracted and integrated to the final data set to see their impact on machine learning models already trained.
- As shown in Fig 3.10, produced water, a byproduct of oil and gas production is the main input for saltwater storage, treatment and disposal. We discussed the importance of estimating produced water in Section 2.5 of Chapter 2. The production table contained cumulative water production after 1-month, 6-months, 12-months and 24-months. These were used with other well related features as explained in Section 3.2.1 of this chapter for all oil wells to train multi-target regression machine learning models to estimate water production.

e. **Bakken Oil and Gas Wells Production Data:** This time series dataset was downloaded from WellDatabase.com during the preparation of the Introduction Chapter of this dissertation document to generate Fig 1.2. WellDatabase is an oil & gas software as a service provider using the latest web and mobile technologies to bring data to the oil & gas world. They take data from public and private sources and use a proprietary process to normalize it into a single searchable database.

Out of the 905 SWD wells identified in Section 3.2.1 of this chapter, only 783 wells were found to contain valid monthly disposal volume and average injection pressure data and hence only these wells were considered for further analysis.

3.3 Summary

In this chapter, we first presented a list of all the key technologies used to execute the project, starting from the Anaconda platform with the Python programming language, the Jupyter

Notebook IDE, all the Python packages used and other technologies such as Orange, Tableau and Neo4j software platforms.

We then introduced NDICOG as the primary data source for this project and all the different data sets from this data source that was extracted from different formats such as flat files and HTML files using web scraping. We presented the techniques used and the code snippets to extract the data using web scraping, data cleaning and data integration using Python. We concluded the chapter with an overview of some important secondary data sources, the need for such data for developing robust data driven models, data formats, extraction and integration techniques.

In the next chapter, we will take a closer look at the descriptive statistics of some of the key data attributes and the results from exploratory data analysis carried out using visual analytics techniques and insights gained. We will conclude this chapter with a presentation of clustering analysis results and GDS results.

Chapter 4

Exploratory Data Analytics and Clustering Analysis

In this chapter, we present the final data set that was generated by integrating the various data sets from the primary and secondary data sources described in Chapter 3. We then present descriptive statistics and univariate analysis of some of the key features and results from Exploratory Data Analysis (EDA). We extend the EDA to include some clustering techniques and present those results later in this chapter.

4.1 Final Data Set

The final data set was created by integrating the following three data sets using Orange software's 'Merge Data' widget in a nested fashion (Figure 4.1 below). The 'Merge Data' widget is used to horizontally merge two datasets, based on the values of selected attributes (columns). The SWD well's File number unique attribute was used as the key. The 'Append columns from Extra Data outputs' merge type was used to merge all rows from the Data, augmented by the columns in the Extra Data.

1. SWD Data- This data set contains features extracted from the primary data source NDICOG, integrated with the Inyan Kara Isopach maps data, ND road data and Oilfield Waste Disposal Sites data.

2. SWD Casing Data- This data set contains casing and perforated interval related features scraped from the NDICOG well scout ticket data.

3. Bottom Hole Well Location (Latitude and Longitude data) – These two subsurface features were extracted from the Enervus data source's wells table only for the SWD wells.

The final data set contained data from 783 SWD wells with over 60 features including 12+ meta features prior to any data cleaning.

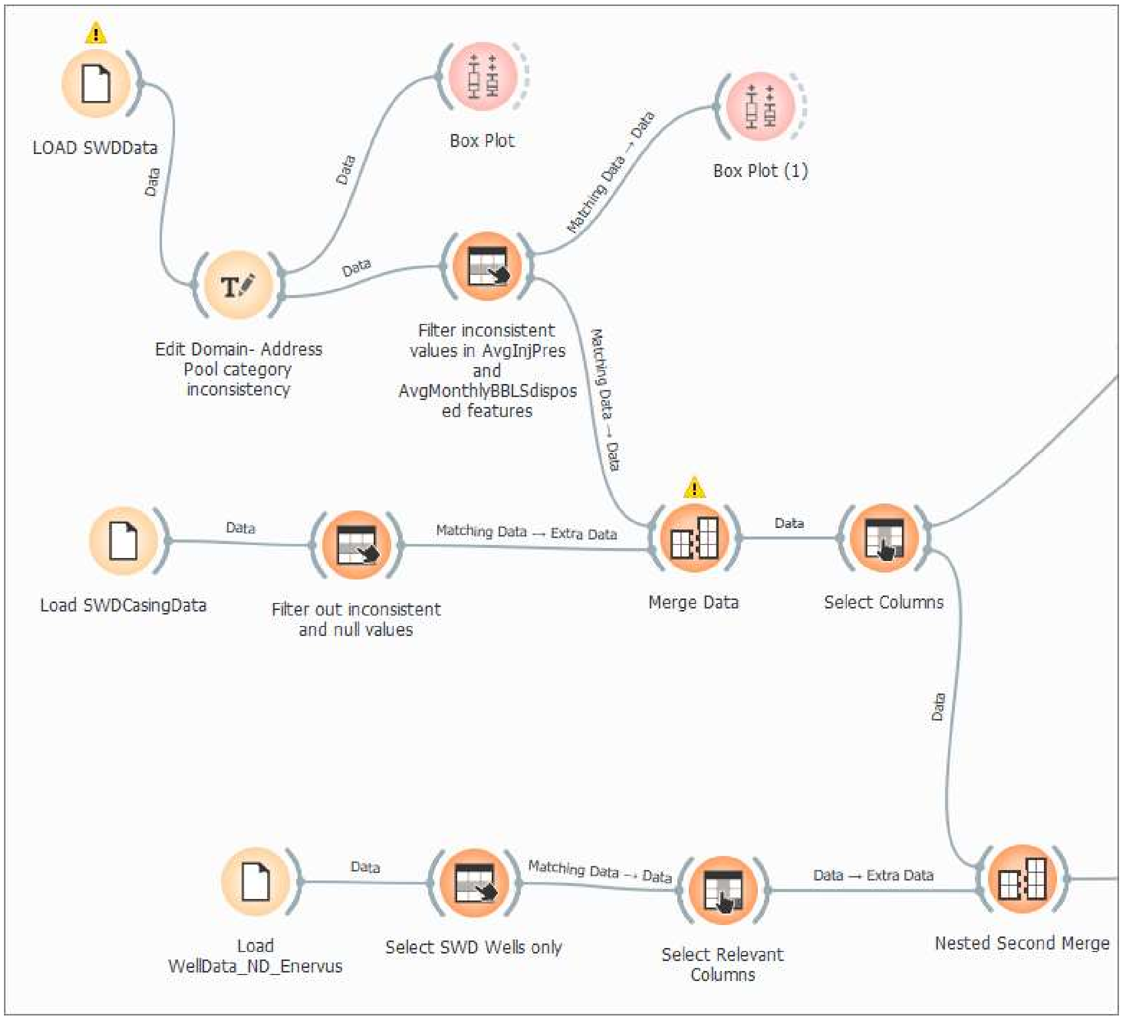
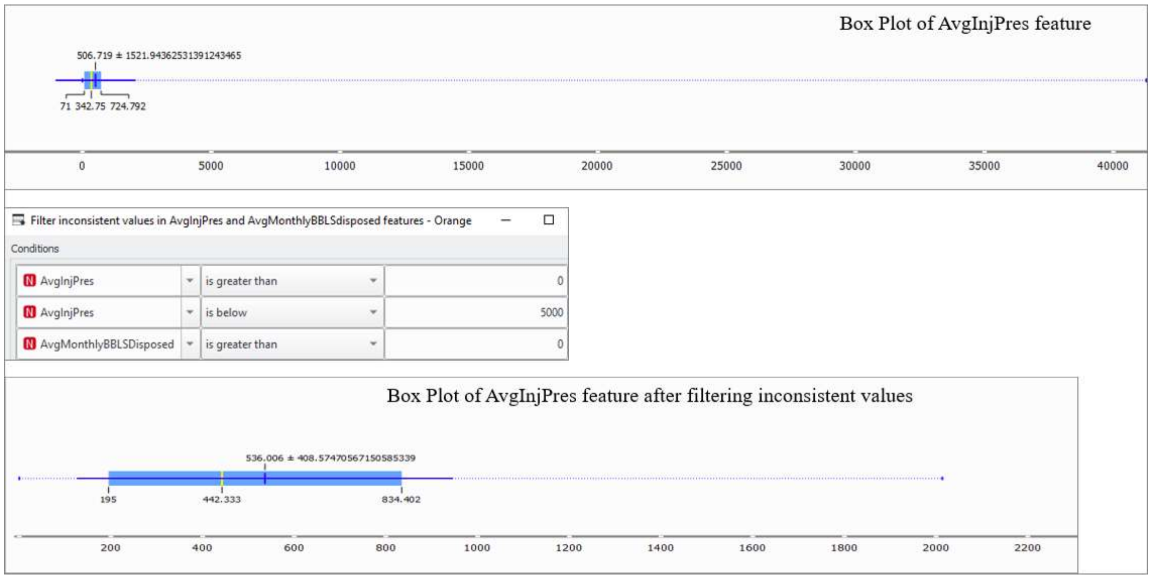


Fig 4.1 Orange workflow for data integration to create the final data set

To address data inconsistency, filtering was applied on selected features. In the SWD Data file described in (1) above, the AvgInjPres and AvgMonthlyBBLSdisposed are two key continuous numerical features. One hundred and twenty-eight SWD wells were filtered out to remove the inconsistent zero value for 'AvgInjPres' which implies that disposal activities happened in a well without any injection pressure, likely due to data reporting error. Twenty-four SWD wells with a zero value 'AvgMonthlyBBLSdisposed' feature implying no disposal activities were carried out, were also filtered out. In addition, an inconsistent value (outlier) exceeding 40,000 psi in 'AvgInjPres' feature was also filtered out. Box plots of 'AvgInjPres' feature before and after filtering, created using the 'Box Plot' widget in Orange software are shown in Figure 4.2 below along with the filtering operation carried out using the 'Select Rows' widget.



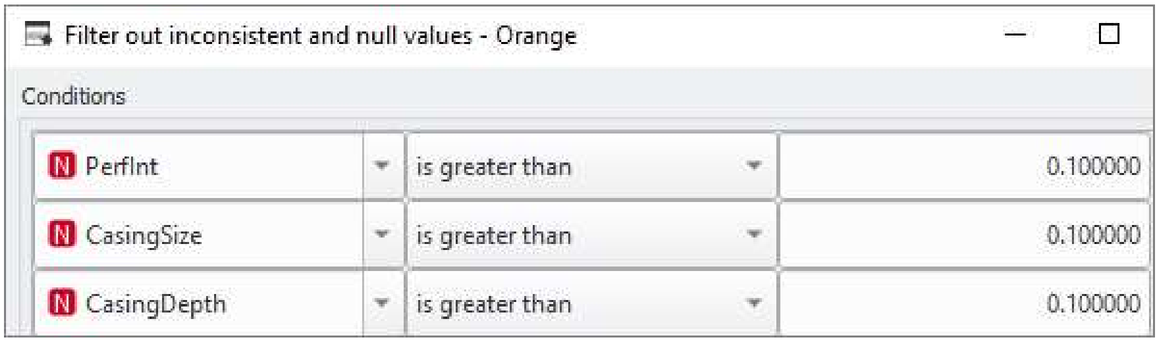
Filter inconsistent values in AvgInjPres and AvgMonthlyBBLSdisposed features - Orange

Conditions

AvgInjPres	is greater than	0
AvgInjPres	is below	5000
AvgMonthlyBBLSDisposed	is greater than	0

Fig 4.2 Univariate analysis of AvgInjPres feature before and after cleaning

The box plot is an effective and popular visualization for univariate analysis of continuous numerical features. In the box plots shown in Figure 4.2 above, the blue highlighted area represents the value between the first (25%) and third (75%) quartiles, the yellow vertical line represents the median, the dark blue vertical line represents the mean value, the thin blue horizontal line represents the standard deviation. The values are also annotated in the box plot. In addition, the following three features (Figure 4.3 below) were filtered to remove SWD wells that did not contain perforating intervals and casing data extracted from NDICOG primary data source using web scraping as described in Section 3.2.1 of Chapter 3, resulting in the elimination of 56 SWD wells from the data set. However, perforating interval feature was available for some of these 56 wells from the Enervus data set which was considered to increase the data set size.



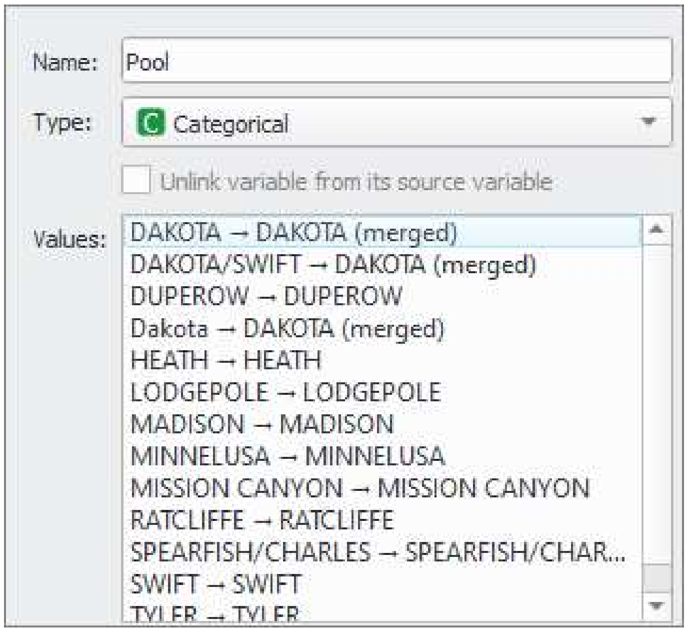
Filter out inconsistent and null values - Orange

Conditions

N PerfInt	is greater than	0.100000
N CasingSize	is greater than	0.100000
N CasingDepth	is greater than	0.100000

Fig 4.3 Data cleaning- filtering operation on some features

In the Pool categorical features that contained the disposal pool name for each SWD well where the target disposal formation was located, inconsistencies were noticed in the Pool name values and they were addressed using the ‘Edit Domain’ widget available in the Orange software, which can be used to edit/change a dataset’s domain - rename features, rename or merge values of categorical features, add a categorical value, and assign labels. Results of this operation are shown below in Figure 4.4 below:



Name: Pool

Type: C Categorical

Unlink variable from its source variable

Values:

```
DAKOTA → DAKOTA (merged)
DAKOTA/SWIFT → DAKOTA (merged)
DUPEROW → DUPEROW
Dakota → DAKOTA (merged)
HEATH → HEATH
LODGEPOLE → LODGEPOLE
MADISON → MADISON
MINNELUSA → MINNELUSA
MISSION CANYON → MISSION CANYON
RATCLIFFE → RATCLIFFE
SPEARFISH/CHARLES → SPEARFISH/CHAR...
SWIFT → SWIFT
TYLER → TYLER
```

Fig 4.4 Addressing data inconsistency in Pool categorical feature

4.2 Exploratory Data Analysis (EDA)

In this section, we will review results of exploratory data analysis of the key features in this data set. We will first review the individual feature, their descriptive statistics and distributions. The Orange workflow for EDA, an extension of Figure 4.1, is shown in Figure 4.5 below.

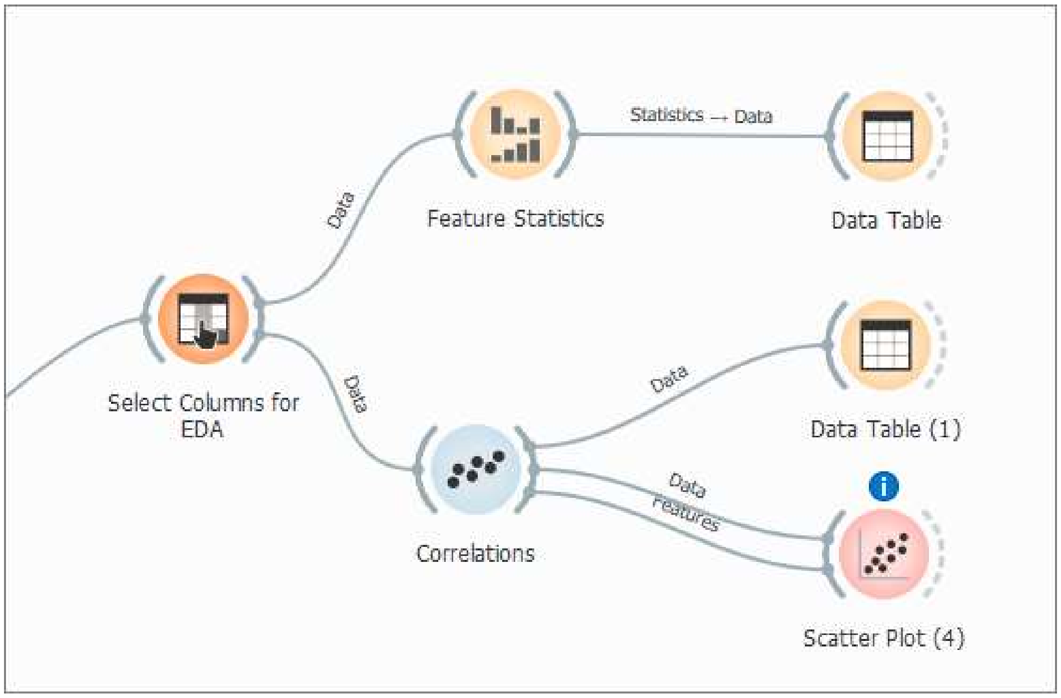


Fig 4.5 Orange EDA workflow

The 'Feature Statistics' widget in Figure 4.5 above provides a quick way to inspect and find interesting features in a given data set. In addition to visually inspecting the features, the widget outputs a statistics table containing descriptive statistics of the selected features. Fig 4.6 below shows a snapshot of some of the categorical (C) and some of the numerical (N) features including the distribution, central tendency and missing value count / percentage. Central tendency for categorical variables such as Pool, RoadType etc., is the mode while for numerical features, both mean and median are presented.

The dispersion value for categorical features is the entropy of the value distribution. Lantz (2015) describes entropy, a concept borrowed from information theory as a measure that quantifies the randomness, or disorder, within a set of class values. Sets with high entropy are very diverse and provide little information about other items that may also belong in the set, as there is no apparent commonality. Entropy is measured in bits, with values ranging from 0 to 1 for a two class variable. For *n* classes, entropy ranges from 0 to $p_i \log_2(p_i)$. In each case, the minimum value indicates that the sample is completely homogenous, while the maximum value indicates that the data are as diverse as possible, and no group has even a small plurality. Mathematical notion of entropy is given in equation 4.1 below:

$$Entropy = \sum_{i=1}^{C} -p_i \log_2(p_i) \text{ ....4.1}$$

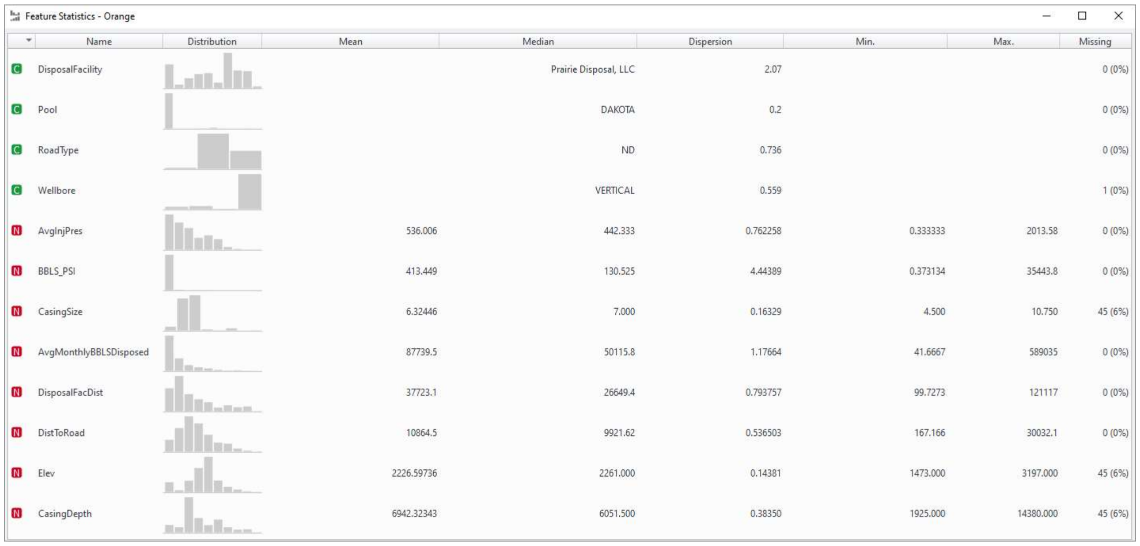
where *c* refers to the number of class levels and p_i refers to the proportion of values falling into class level *i*. DisposalFacility and Pool are multi-class features; WellType is a 2-class feature; RoadType is a 3-class feature and Wellbore is a 4-class feature. Their distributions based on the class frequency are given in Figure 4.6.

For numeric features, the dispersion value is the coefficient of variation which is mathematically written as given below in equation 4.2.

$$C_v = \frac{\sigma}{\mu} \text{ ....……………………….4.2}$$

where σ is the standard deviation and μ is the mean of the numerical feature.

One can observe from the Figure 4.6 and Table 4.1 below the skewed nature of some of the numerical features such as AvgIngPres, BBLS_PSI and reasonably normal nature of features such as CasingSize and Elev(ground elevation). Many of the supervised and unsupervised machine learning algorithms perform optimally when features are normally distributed. Hence, we will explore some data transformation options prior to training machine learning models on this data set. The histogram distributions for categorical variables such as Well_Quality, Wellbore etc., are also part of the Feature Statistics widget output. Dominant Well_Quality category is Good (289 SWD wells) followed by Average (203 SWD wells) and the rest were Poor (159 SWD wells).


Feature Statistics - Orange

	Name	Distribution	Mean	Median	Dispersion	Min.	Max.	Missing
C	DisposalFacility			Prairie Disposal, LLC	2.07			0 (0%)
C	Pool			DAKOTA	0.2			0 (0%)
C	RoadType			ND	0.736			0 (0%)
C	Wellbore			VERTICAL	0.559			1 (0%)
N	AvgInjPres		536.006	442.333	0.762258	0.333333	2013.58	0 (0%)
N	BBLS_PSI		413.449	130.525	4.44389	0.373134	35443.8	0 (0%)
N	CasingSize		6.32446	7.000	0.16329	4.500	10.750	45 (6%)
N	AvgMonthlyBBLSDisposed		87739.5	50115.8	1.17664	41.6667	589035	0 (0%)
N	DisposalFacDist		37723.1	26649.4	0.793757	99.7273	121117	0 (0%)
N	DistToRoad		10864.5	9921.62	0.536503	167.166	30032.1	0 (0%)
N	Elev		2226.59736	2261.000	0.14381	1473.000	3197.000	45 (6%)
N	CasingDepth		6942.32343	6051.500	0.38350	1925.000	14380.000	45 (6%)

Fig 4.6 Feature Statistics Snapshot

Table 4.1 below lists the key numerical features of the data set along with the statistics output from the 'Feature Statistics' widget. % Missing values is based on 651 SWD wells.

Table 4.1 Key numerical features and their statistics

Key Feature	Mean	Median	Dispersion	Min	Max	%Missing Values
AvgInjPres, psi	536.006	442.333	0.762258	0.333333	2013.58	0%
AvgMonthlyBBLSDisposed, bbls	87739.5	50115.8	1.17664	41.6667	589035	0%
BBLS_PSI, bbls/psi	413.449	130.525	4.44389	0.373134	35443.8	0%
DistToRoad, metres	10864.5	9921.62	0.536503	167.166	30032.1	0%
DisposalFacDist, metres	37723.1	26649.4	0.793757	99.7273	121117	0%
CasingSize, in	6.32446	7	0.163291	4.5	10.75	7%
CasingDepth, ft	6942.32	6051.5	0.383496	1925	14380	7%
Elev, ft	2226.6	2261	0.143806	1473	3197	7%
PerfInt, ft	247.554	206	0.923586	10	1652	7%
TD, Total well depth, ft	7988.27	6440	0.431901	2310	20996	7%
SS_Thickness, Inyan Kara thickness, ft	101.582	96	0.439598	8	234	0%
K-IK, Inyan Kara formation top, ft	4801.85	5115	0.19654	1964	5979	5%

Correlation Analysis: According to Lantz (2015), correlation between two numerical variables is a number that indicates how closely their relationship follows a straight line. Pearson's correlation coefficient developed by the 20th century mathematician Karl Pearson is most widely used. The correlation ranges between -1 and +1. The extreme values indicate a perfectly linear relationship, while a correlation close to zero indicates the absence of a linear relationship. Let us look at the mathematical notion of the Pearson's correlation coefficient. Equation 4.3 below defines covariance of two numerical variables *x* and *y*:

$$cov(x, y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{n} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 4.3$$

where $\bar{x}$ and $\bar{y}$ are the mean of the variables x and y and n is the total number of samples. Equation 4.4 below defines the Pearson's correlation coefficient variables *x* and *y* using covariance of *x* and *y* and the standard deviations of *x* and *y*:

$$\rho_{(x,y)} = corr(x, y) = \frac{cov(x,y)}{\sigma_x \sigma_y} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 4.4$$

Lantz (2015) provides one method to interpret correlation coefficient as given in Table 4.2 below.

Table 4.2 Correlation Coefficient Interpretation

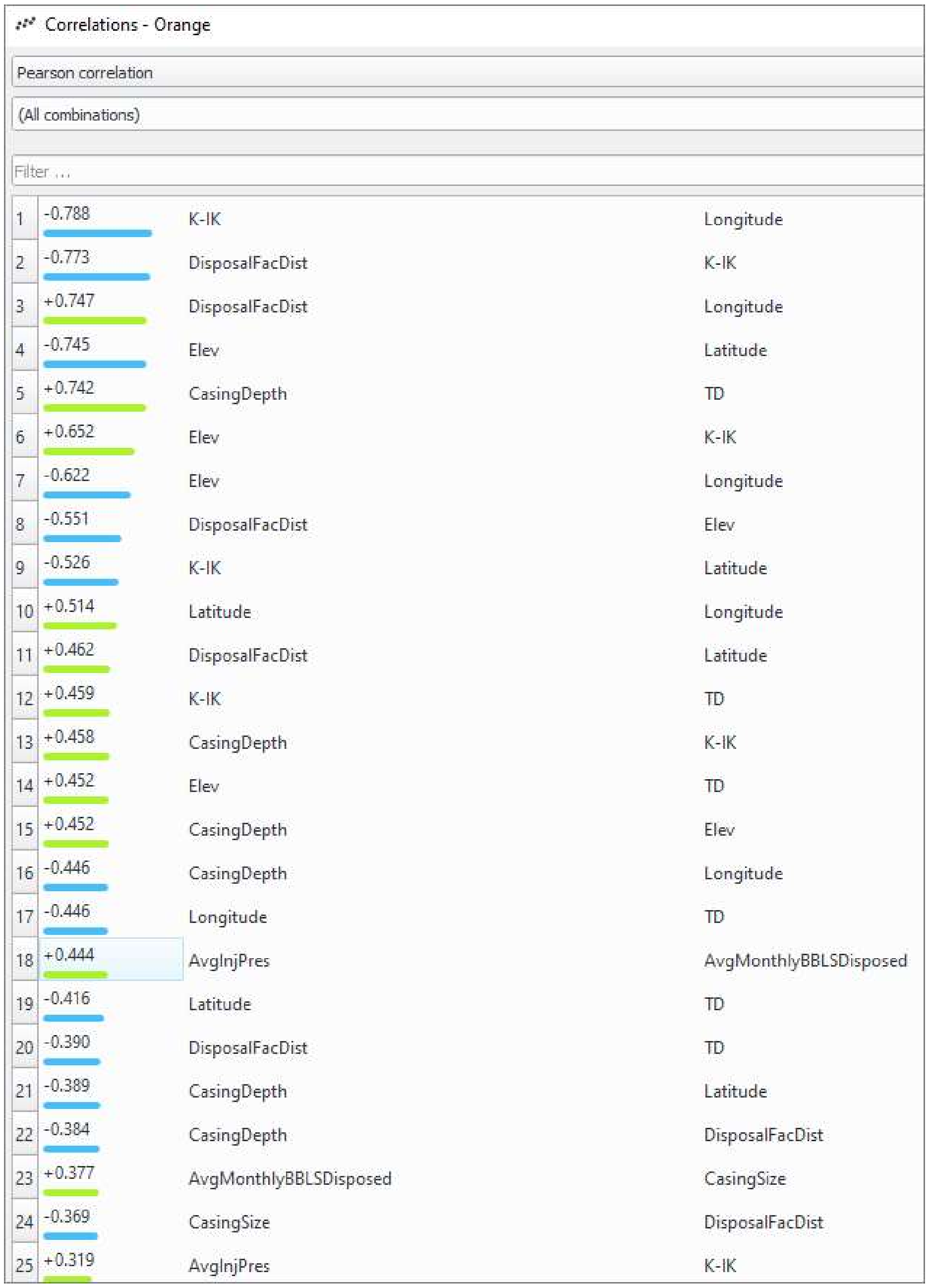
Range	Interpretation
0.1 – 0.3 or -0.3 - -0.1	Weak
0.3 – 0.5 or -0.5 - -0.3	Moderate
>0.5 or < -0.5	Strong

Correlation must however be interpreted in context. For data involving human beings or subsurface information as in this project, a correlation of 0.5 may be considered strong, while for data generated by mechanical processes, a correlation of 0.5 may be weak. Orange software provides a 'Correlation widget' to compute pairwise correlations of all continuous numeric variables (features) in a data set. Figure 4.7 below shows the top 25 Pearson correlation coefficients from the data set. Key observations are:

- DisposalFacility distance has strong positive correlations with Longitude and Latitude (SWD well surface locations) and strong negative correlations with Inyan Kara formation top depth and ground elevation.
- Relationship between average monthly barrels of saltwater volumes disposed and the average wellhead injection pressures is moderate to strong positive.

- Average monthly barrels of saltwater volume disposed has a moderate to strong positive correlation with the casing size.

Relationships between two continuous numerical variables and their correlations can be visualized effectively using a scatter plot. Figure 4.8 below shows examples of two scatter plots created using the 'Scatter Plot' widget in Orange software.



Correlations - Orange

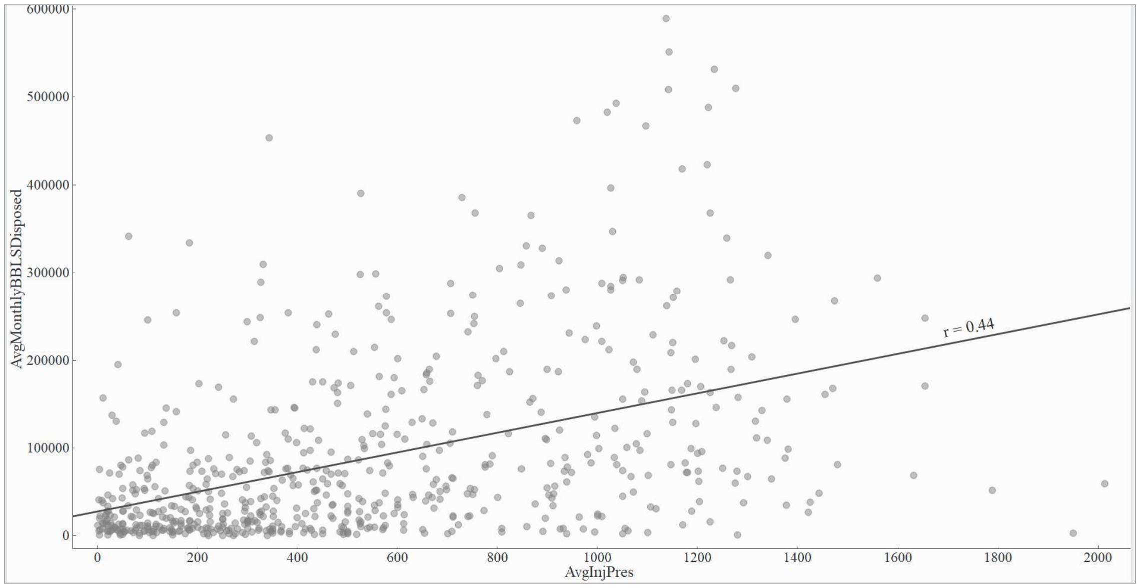
Pearson correlation

(All combinations)

Filter ...

1	-0.788	K-IK	Longitude
2	-0.773	DisposalFacDist	K-IK
3	+0.747	DisposalFacDist	Longitude
4	-0.745	Elev	Latitude
5	+0.742	CasingDepth	TD
6	+0.652	Elev	K-IK
7	-0.622	Elev	Longitude
8	-0.551	DisposalFacDist	Elev
9	-0.526	K-IK	Latitude
10	+0.514	Latitude	Longitude
11	+0.462	DisposalFacDist	Latitude
12	+0.459	K-IK	TD
13	+0.458	CasingDepth	K-IK
14	+0.452	Elev	TD
15	+0.452	CasingDepth	Elev
16	-0.446	CasingDepth	Longitude
17	-0.446	Longitude	TD
18	+0.444	AvgInjPres	AvgMonthlyBBLSDisposed
19	-0.416	Latitude	TD
20	-0.390	DisposalFacDist	TD
21	-0.389	CasingDepth	Latitude
22	-0.384	CasingDepth	DisposalFacDist
23	+0.377	AvgMonthlyBBLSDisposed	CasingSize
24	-0.369	CasingSize	DisposalFacDist
25	+0.319	AvgInjPres	K-IK

Fig 4.7 Top 25 ranked Pearson correlation coefficients



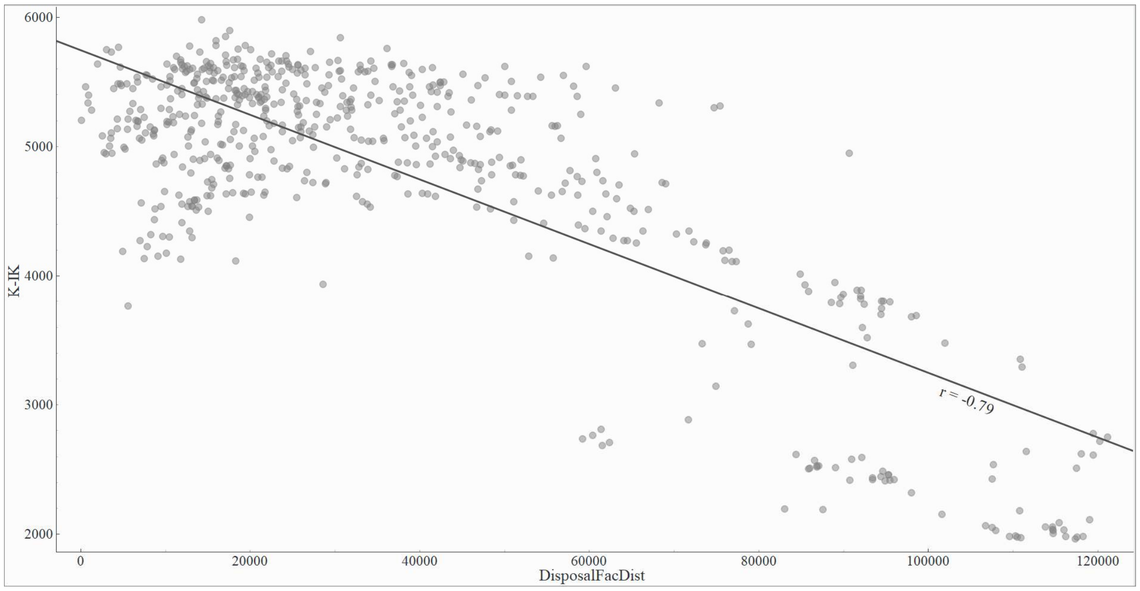


Fig 4.8 Examples of scatter plot visualizations

Three features relating to Gamma Ray log were extracted from 239 wells as discussed in Section 3.2.1 of Chapter 3. Since the data was available for only 37% of the 651 SWD wells considered and the features had a weak correlation with key features discussed earlier in this chapter, it was decided not to include these features for further data mining activities in the project.

The following bar graphs (Figure 4.9) were created using Tableau software to answer questions around the most dominant pool where SWD disposal activities occurred, influence of road types and oilfield waste disposal sites on SWD well locations.

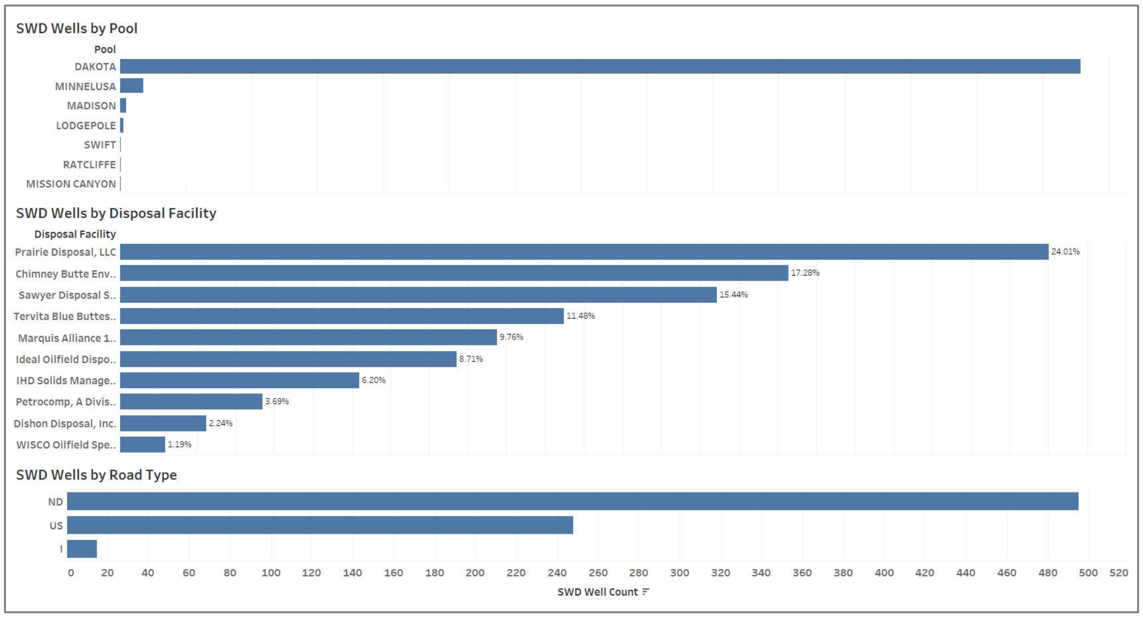
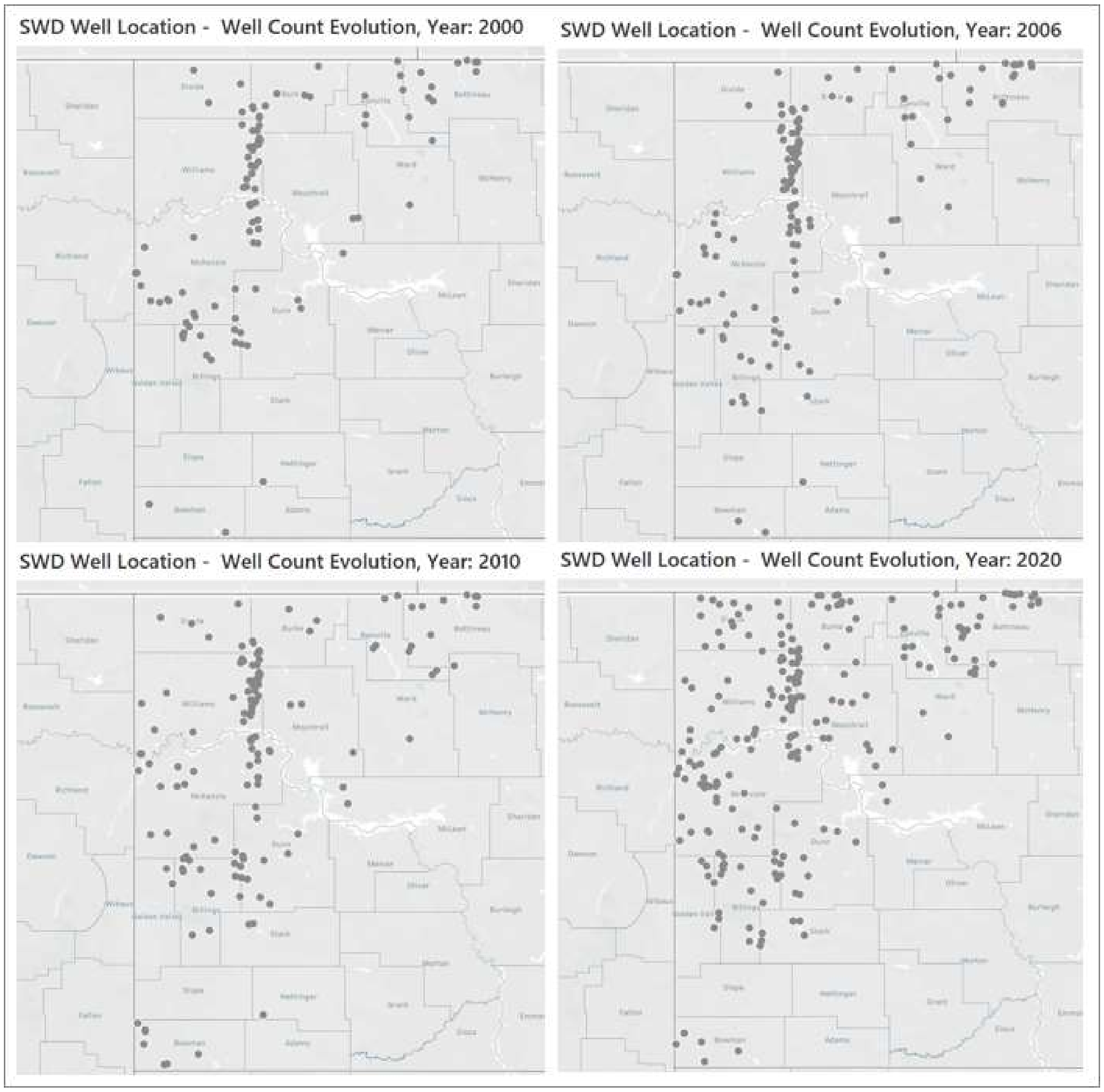


Fig 4.9 SWD Well Count by Pool, Disposal Facility and Road Type

Analysis of the above visualization reveals that the dominant pool is the Dakota pool where most of the SWD activities had occurred, targeting the Inyan Kara formation. Most of the wells were located close to North Dakota state highways while least number of wells were located close to US Interstate highways. The top four disposal sites accounted for more than 65% of the total SWD wells.

Figure 4.10 below captures the evolution of SWD well count in the state of North Dakota over time to understand the impact of the 'Bakken Boom' on SWD well count. Well locations are identified in a map view with county labels and presented during four time periods - year 2000, year 2006 – start of the Bakken Boom, year 2010 and year 2020. The average monthly disposal volumes in barrels are included in the final snapshot to identify wells that dispose higher average volumes of saltwater. One can see the increase in well density over the 20-year period, particularly in the Williams and McKenzie counties.


SWD Well Location - Well Count Evolution, Year: 2000
SWD Well Location - Well Count Evolution, Year: 2006
SWD Well Location - Well Count Evolution, Year: 2010
SWD Well Location - Well Count Evolution, Year: 2020

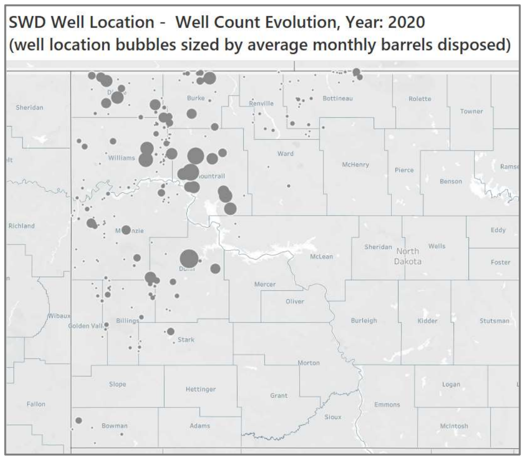

SWD Well Location - Well Count Evolution, Year: 2020
(well location bubbles sized by average monthly barrels disposed)

Fig 4.10 SWD Well Count Evolution over time (snapshots during 5 time periods)

4.3 Clustering Analysis

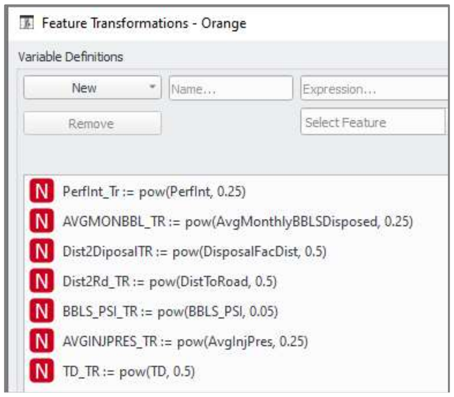
4.3.1 Feature Transformation

We could notice from Figure 4.6 and Table 4.1 that the distributions of some of the numerical features such as 'AvgIngPres' and 'BBLS_PSI' are skewed while some of the other features like 'CasingSize' and 'Ground Elevation' are reasonably normal. Many of the supervised and unsupervised machine learning (clustering) algorithms perform optimally when features are normally distributed. While some supervised machine learning algorithms like linear regression and logistic regression explicitly assume the continuous numeric features have a normal distribution, other nonlinear algorithms like Random Forest and Ada Boost which may not have this assumption, still tend to perform better when variables have a normal distribution. Kuhn & Johnson (2013) mention in their book that another common reason for transformations is to remove distributional skewness. An un-skewed distribution is one that is roughly symmetric. This means that the probability of falling on either side of the distribution's mean is roughly equal which helps in improving model performance. Hence, we will explore some data transformation options prior to training machine learning models on this data set.

According to Sheather (2009), continous numerical features can be trasformed when assumption of linear relationship does not hold between two features. A simple way to observe the nature of the relationship is to plot the two numerical features using a scatter plot similar to the one shown in Figure 4.6. For univariate analysis, a histogram distribution plot similar to the ones shown in Figure 4.4 or Figure 4.1 (Box Plot) can be used to understand if the distribution is normal. Transformation can be applied to convert the skewed distributions to a normal distribution. Power transformation given in equation 4.5 is commonly used.

$$f(x,\lambda) = x^{\lambda} \text{ ....} 4.5$$

where X is the feature of interest and λ is the power. Typical values of λ are -1 (reciprocal), - -0.5 (reciprocal square root), 0 (log), 0.5 (square root), 1 (no transform) and 2 (square) but other values such as 0.25, 0.1 etc., can also be used. The 'Feature Constructor' widget was used along with the 'Distributions' widget in Orange software to iteratively tune the value of λ for the different features that needed transformation and the best value of lambda was selected. In case of supervised machine learning multi-linear regression models, the model performance metrics such as MSE were also used to tune the λ value.



PerfInt_Tr := pow(PerfInt, 0.25)
AVGMONBBL_TR := pow(AvgMonthlyBBLSDisposed, 0.25)
Dist2DiposalTR := pow(DisposalFacDist, 0.5)
Dist2Rd_TR := pow(DistToRoad, 0.5)
BBLS_PSI_TR := pow(BBLS_PSI, 0.05)
AVGINJPRES_TR := pow(AvgInjPres, 0.25)
TD_TR := pow(TD, 0.5)

Fig 4.11 Feature transformation equations

Figure 4.11 shows the different features that were transformed and the final power transformation functions for the 651 SWD wells considered.

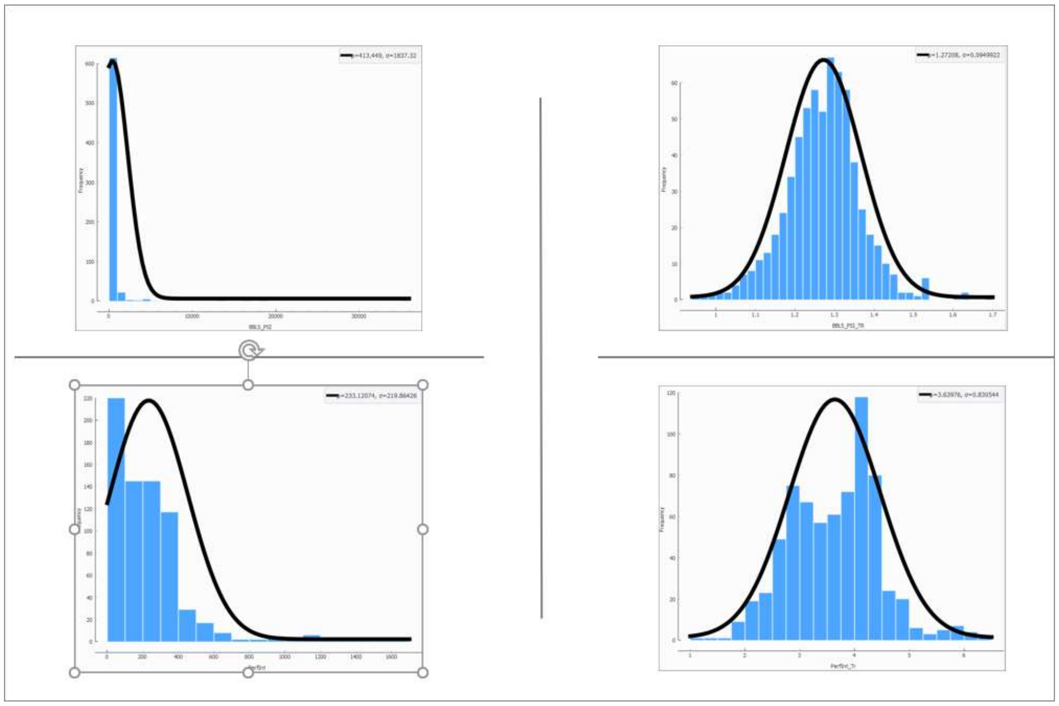


Fig 4.12 Example pre and post transformation distributions for two features (PerfInt and BBLS_PSI)

Figure 4.12 shows the pre- and post-transformation distributions for a couple of the features that were transformed. The distribution visualizations were created using Orange's 'Distribution' widget.

4.3.2 Principal Component Analysis (PCA)

Having transformed the required features to closely represent a normal distribution, the next task carried out was dimensionality reduction using PCA. The concept of PCA was introduced in Section 2.4 of Chapter 2. The 'PCA' widget in Orange software was used for PCA. It is strongly recommended to normalize the features so that all features can have equal importance and to avoid dependence on the choice of the measurement unit. Normalization involves transforming the data within a given feature to fall within a smaller range such as [0.0, 1.0] (Han, 2012). If *minA* and *maxA* are the minimum and maximum values of a feature *A*, then, min-max normalization maps a value v_i of *A* to a value v' in the range [*new- minA, new-maxA],* say [0.0, 1.0], as shown in Equation 4.6 below. Max-min normalization preserves the relationships among the original data values. An 'out of bounds' error will occur if future input cases for normalization falls outside the original range of *A* (Han, 2012).

$$V_i' = \frac{V_i - min\,A}{max\,A - min\,A}(new\_max\,A - new\_min\,A) + new\_min\,A \text{.....4.6}$$

On the other hand, *z*-score normalization (also known as zero-mean normalization) normalizes the values of the feature *A* based on its mean value $\bar{A}$ and standard deviation σ_A as shown in equation 4.7 below. This method of normalization is useful when the actual minimum and maximum of the feature *A* are unknown, or when there are outliers that dominate the min-max normalization (Han, 2012). *Z*-score normalization was selected for normalization.

$$V_i' = \frac{v_i - \bar{A}}{\sigma_A} \text{ ....4.7}$$

A total of 19 numerical features were considered for PCA (Figure 4.13). Figure 4.14 below shows the results of PCA.

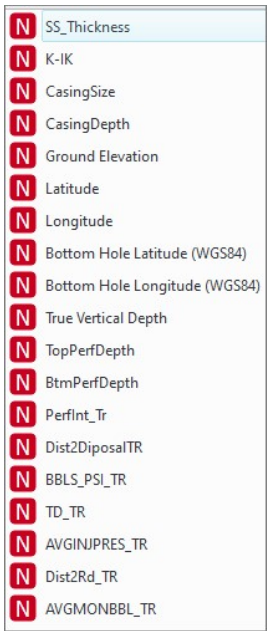

N SS_Thickness
N K-IK
N CasingSize
N CasingDepth
N Ground Elevation
N Latitude
N Longitude
N Bottom Hole Latitude (WGS84)
N Bottom Hole Longitude (WGS84)
N True Vertical Depth
N TopPerfDepth
N BtmPerfDepth
N PerfInt_Tr
N Dist2DiposalTR
N BBLS_PSI_TR
N TD_TR
N AVGINJPRES_TR
N Dist2Rd_TR
N AVGMONBBL_TR

Fig 4.13 Numerical continuous features selected for PCA and Clustering analysis

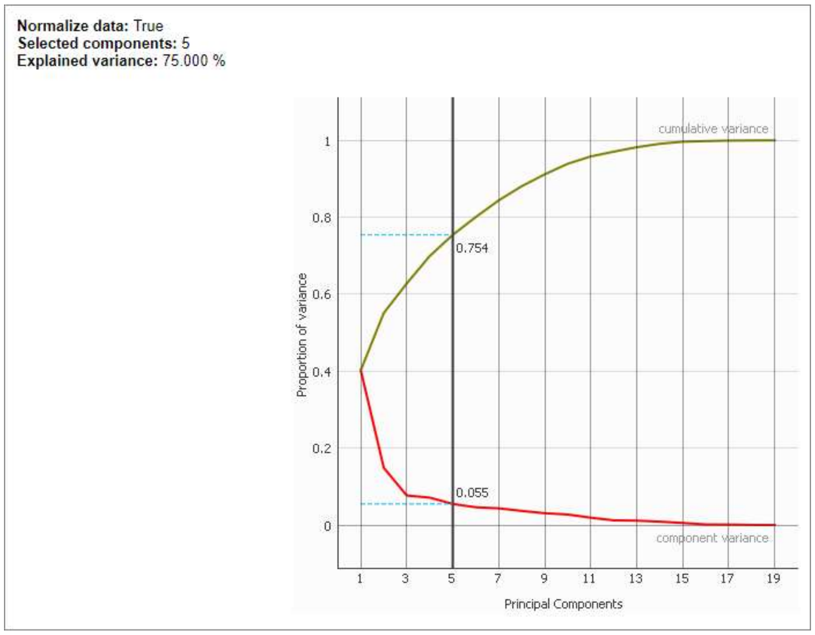

Normalize data: True
Selected components: 5
Explained variance: 75.000 %

Fig 4.14 PCA results

The first five components were selected and explained variance was 75%. Variance explained by the first component is 40%. Table 4-3 below shows the principal component weights for the first and the second principal component. It can be noticed from the highlighted components that PC1 is mainly influenced by geographical positions of the wells and the depths whereas the SWD well performance features such as 'BBLS_PSI_TR' and 'AVGMONBBL_TR' has a good influence on PC2.

Table 4.3 Principal component weights of PC1 and PC2

Components	PC1	PC2
Variance	0.402608605	0.148738179
Ground Elevation	**-0.293857075**	0.038321486
Latitude	**0.262491177**	-0.156195343
Longitude	**0.315304252**	0.056086927
Bottom Hole Latitude (WGS84)	**0.263709807**	-0.154795842
Bottom Hole Longitude (WGS84)	**0.314688994**	0.058505249
True Vertical Depth	**-0.229761478**	0.306414109
TopPerfDepth	**-0.319245927**	-0.070954605
BtmPerfDepth	**-0.318258627**	-0.123479532
PerfInt_Tr	-0.081270356	-0.33085546
Dist2DiposalTR	**0.269673235**	0.167553885
BBLS_PSI_TR	0.006921964	**-0.327713375**
TD_TR	-0.247396793	0.245580167
AVGINJPRES_TR	-0.114617514	-0.157339195
Dist2Rd_TR	-0.023229362	0.206222124
AVGMONBBL_TR	-0.080058335	**-0.461275271**

4.3.3 k-Means Clustering Analysis

Figure 4.15 below shows the Orange workflow for feature transformation, PCA and clustering analysis using the k-Means clustering algorithm, an extension of the Orange workflow in Fig 4.1, from the 'Nested Second Merge' widget.

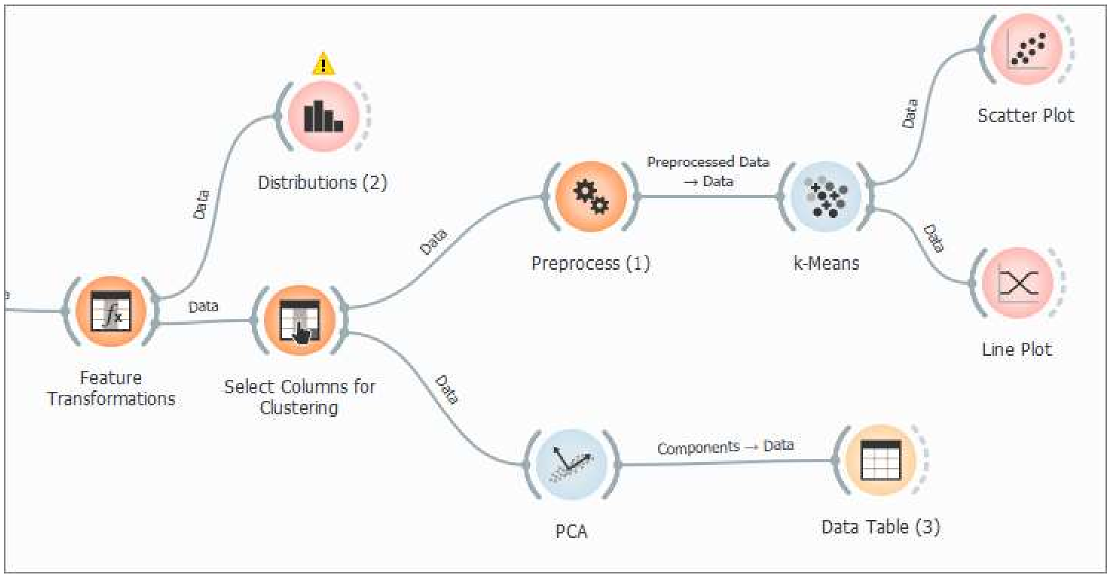
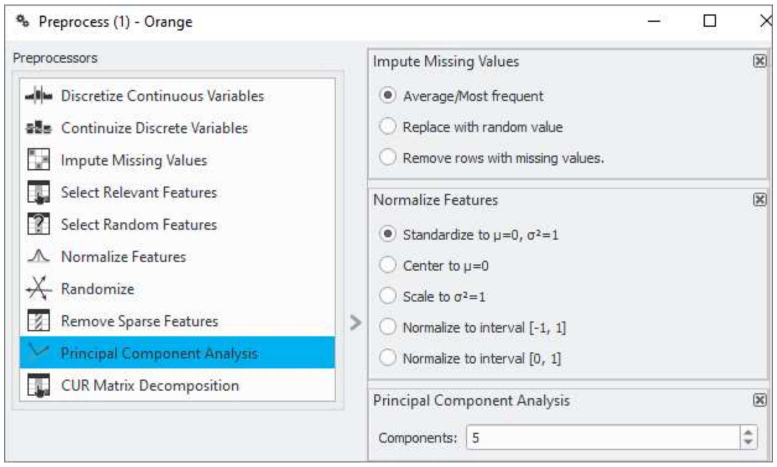


Fig 4.15 Orange workflow for PCA and k-Means clustering analysis

Based on the PCA results in Figure 4.14 above, key preprocessing steps selected using the Orange 'Preprocess' widget were: imputing missing values using average value of the features, z-score normalization and PCA (5-components) as shown in Figure 4.16 below.



Preprocess (1) - Orange

Preprocessors
- Discretize Continuous Variables
- Continuize Discrete Variables
- Impute Missing Values
- Select Relevant Features
- Select Random Features
- Normalize Features
- Randomize
- Remove Sparse Features
- Principal Component Analysis
- CUR Matrix Decomposition

Impute Missing Values
- [✓ Average/Most frequent]
- Replace with random value
- Remove rows with missing values.

Normalize Features
- [✓ Standardize to μ=0, σ²=1]
- Center to μ=0
- Scale to σ²=1
- Normalize to interval [-1, 1]
- Normalize to interval [0, 1]

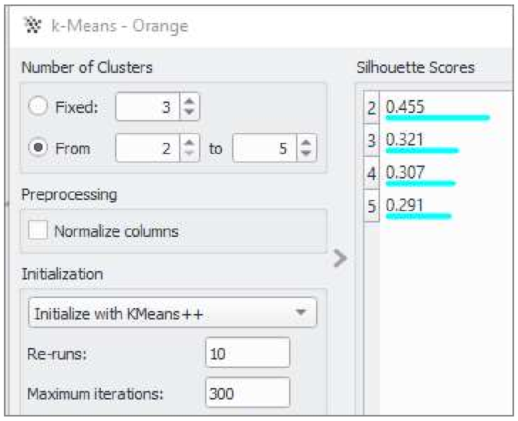
Principal Component Analysis
Components: 5

Fig 4.16 Preprocessing widget prior to clustering using k-Means algorithm

Figure 4.17 below shows the Orange k-Means widget. Value of k (number of clusters) ranged from 2 to 5 and clustering initialization was done using kMeans++ method. Silhouette analysis is used to study the separation distance between the resulting clusters. Silhouette score takes into consideration the intra-cluster distance between the sample and other data points within same cluster (a) and inter-cluster distance between sample and next nearest cluster (b). Silhouette score, S is calculated using equation 4.8 below:

$$s = \frac{(b-a)}{max(a,b)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 4.8$$

Silhouette score has a range of [-1,1]. The silhouette score of 1 means that the clusters are very dense and nicely separated. The score of 0 means that clusters are overlapping. Negative scores might imply that data belonging to clusters may be wrong / incorrect. The silhouette plots can be used to select the most optimal value of the k (number of clusters) in k-means clustering (Pedregosa, 2011). Hence, based on the Silhouette score, 2-clusters were selected.



Number of Clusters

Fixed: 3

[✓ From] 2 to 5

Preprocessing

Normalize columns

Initialization

Initialize with KMeans++

Re-runs: 10

Maximum iterations: 300

Silhouette Scores

2	0.455
3	0.321
4	0.307
5	0.291

Fig 4.17 k-Means Clustering

Cluster interpretation:

Figure 4.18 below presents six snapshots and cluster interpretation is given below for the respective snapshots.

1. C2 is the dominant cluster with 545 (83.7%) of the 651 SWD wells assigned to it.
2. Inyan Kara formation top feature distribution nicely separated by the cluster means.
3. Ground Elevation feature distribution is nicely separated by the cluster means.
4. 'AvgMonthlyBBLSDisposed' feature seem to be well separated based on the cluster centroids with cluster C2 representing SWD wells with an higher average monthly disposal volume. However, there is overlap of the distributions between the two clusters.
5. Scatter plot of average injection pressure and average monthly barrels disposed shows the overlapping nature of the two clusters C1 and C2. C2 generally indicates wells that disposes a higher volume of saltwater in barrels at a higher average injection pressure.
6. This bar graph shows that C2 represents majority of the 'Good' wells based on Well_Quality.

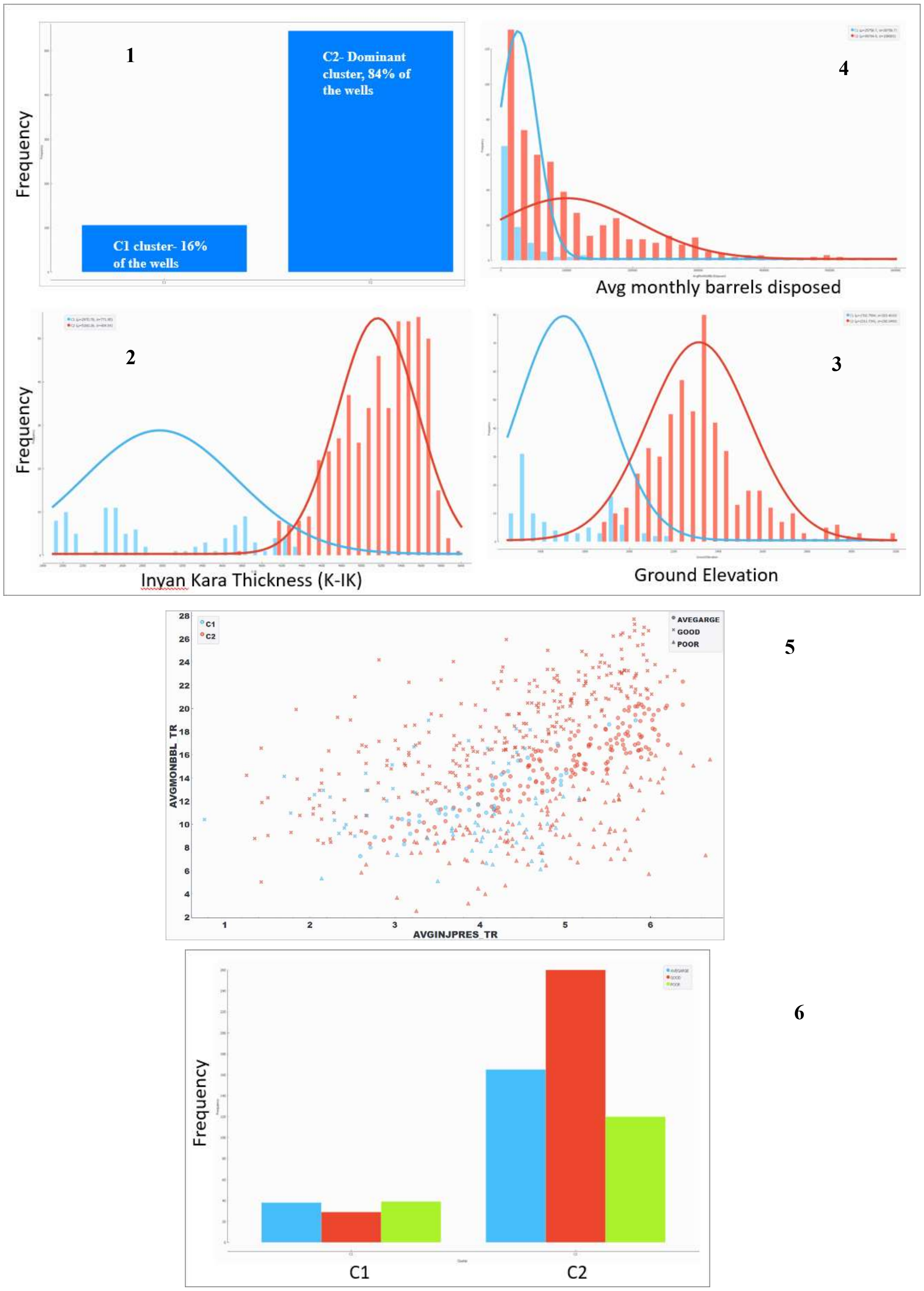


Fig 4.18 Six snapshots to aid k-Means clusters interpretation

4.3.4 Louvain Clustering Analysis

Louvain clustering algorithm was introduced in Section 2.5 of Chapter 2 as a graph-based clustering algorithm to detect communities in large networks. It is a hierarchical clustering algorithm that recursively merges communities into a single node. Figure 4.19 below shows the Orange workflow for Louvain clustering. Features used are the same as k-Means clustering (Figure 4.13).

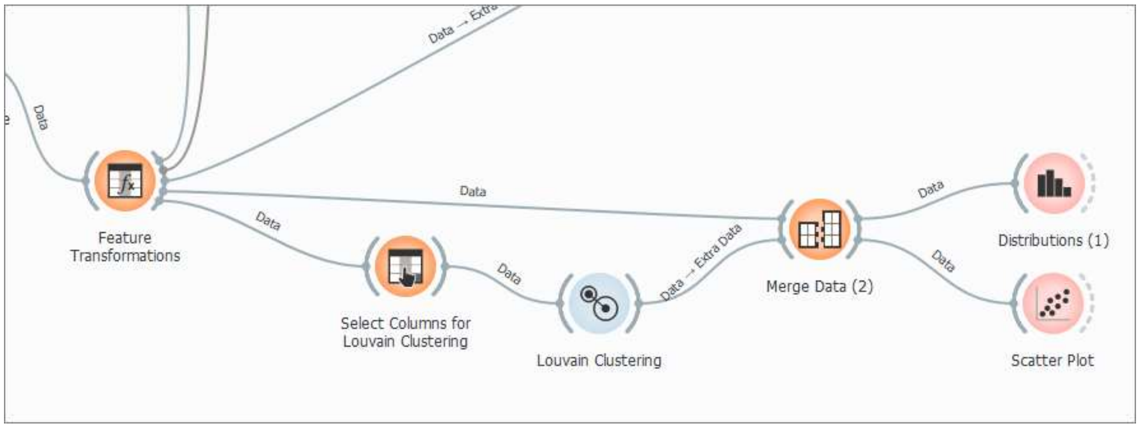
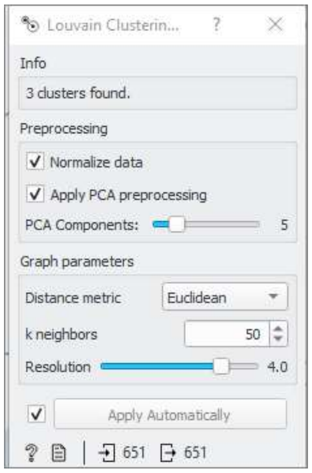


Fig 4.19 Orange workflow for Louvain clustering

Figure 4.20 below shows the Louvain algorithm widget in Orange.


Louvain Clusterin...

Info

3 clusters found.

Preprocessing

✓ Normalize data

✓ Apply PCA preprocessing

PCA Components: 5

Graph parameters

Distance metric: Euclidean

k neighbors: 50

Resolution: 4.0

✓ Apply Automatically

651 651

Fig 4.20 Louvain clustering widget

The algorithm first converts the input data into a *k*-nearest neighbor graph and edges are weighted based on similarity. A modularity optimization algorithm is applied to the graph to retrieve clusters of highly interconnected nodes. For preprocessing, input data is normalized using *z*-score normalization is applied along with PCA (5-components). Graph parameters include a 'Distance metric' (Equation 2.1), 'Number of neighbors (*k*)' and 'Resolution' which affects the size of the recovered clusters. Smaller resolutions recover smaller clusters and therefore a larger number of them, while, conversely, larger values recover clusters containing more data points. Distance metric of Euclidean was used. After iterating over *k* values of [30, 50, 75, 100] and Resolution values of [2,3,4,5] and observing cluster performance, final parameters selected are shown in Figure 4.20 above.

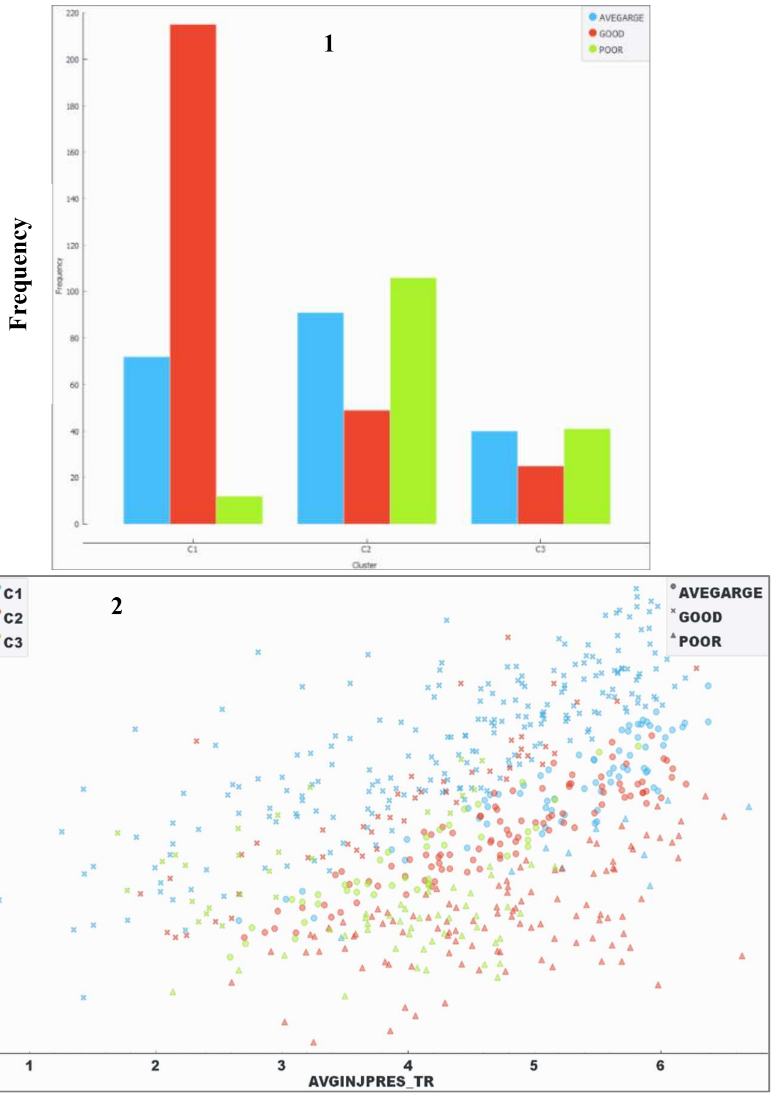


Fig 4.21 Two snapshots- Louvain clustering results

Cluster Interpretation:

Figure 4.21 shows two snapshots for the final 3 clusters.

These snapshots are like snapshots 5 & 6 of Fig 4.18 but one can see the improved clusters visually from the Louvain graph-based algorithm compared to the k-Means algorithm. Snapshot interpretations are given below:

1. C1 is dominated by the 'Good' SWD wells while C2 is dominated more by 'Poor' wells followed by 'Average' wells.
2. Scatter plot shows reasonable alignment Good, Average and Poor wells based on Well_Quality with clusters C1, C3 and C2. Considering that there is still good amount of overlap between C2 and C3, one can also interpret C1 as the cluster of 'Good' wells while clusters C2 and C3 together represents not 'Good' wells.

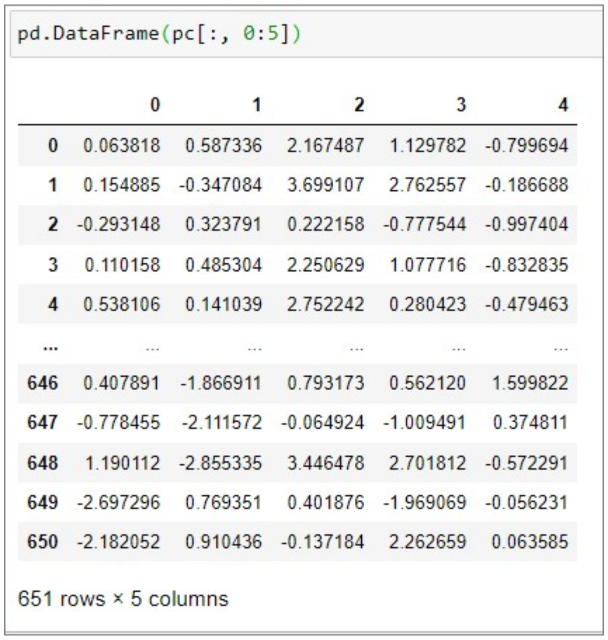
4.3.5 Spectral Clustering Analysis

Spectral clustering analysis was also introduced in Section 2.5 of Chapter 2 as an EDA technique that reduces complex multidimensional datasets into clusters of similar data using the connectivity approach to clustering, wherein communities of nodes (i.e. data points) that are connected or immediately next to each other are identified in a graph. The nodes are then mapped to a low-dimensional space that can be easily segregated to form clusters. Spectral clustering treats the data clustering as a graph partitioning problem without making any assumption on the form of the data clusters.

Since Orange software does not have a widget for Spectral clustering, the data set containing *Z*-score based standardized values of the 19 features identified in Figure 4.13 and used for the k-Means and Louvain clustering analysis was exported as a flat file (.csv). This file was imported using Jupyter notebook IDE and the Python scikit learn library was used to cluster the data set using spectral clustering. PCA analysis was carried out using the code snippet below as part of preprocessing. Principal components 1 to 5 with an explained variance of 75% was used for clustering like the earlier two clustering methods.

```
#PCA - input data is already standardized

from sklearn.decomposition import PCA

pc = PCA().fit_transform(data.iloc[:,:-3]) #PCA on the 19 numerical features
```

The code below displays the first 5 principal components.



```
pd.DataFrame(pc[:, 0:5])
```

```
            0          1          2          3          4
  0   0.063818   0.587336   2.167487   1.129782  -0.799694
  1   0.154885  -0.347084   3.699107   2.762557  -0.186688
  2  -0.293148   0.323791   0.222158  -0.777544  -0.997404
  3   0.110158   0.485304   2.250629   1.077716  -0.832835
  4   0.538106   0.141039   2.752242   0.280423  -0.479463
...        ...        ...        ...        ...        ...
646   0.407891  -1.866911   0.793173   0.562120   1.599822
647  -0.778455  -2.111572  -0.064924  -1.009491   0.374811
648   1.190112  -2.855335   3.446478   2.701812  -0.572291
649  -2.697296   0.769351   0.401876  -1.969069  -0.056231
650  -2.182052   0.910436  -0.137184   2.262659   0.063585

651 rows × 5 columns
```

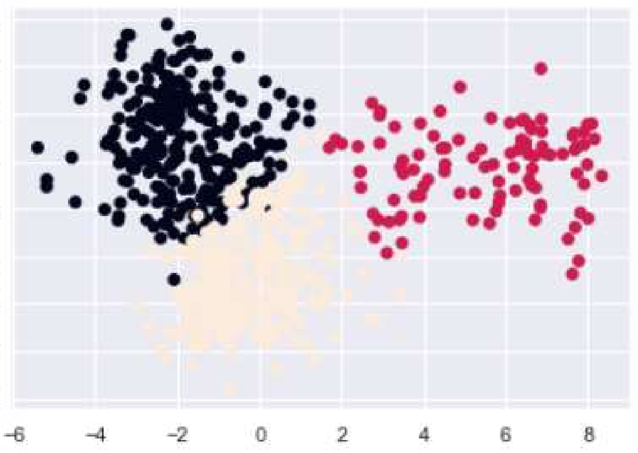
The affinity matrix needed as an input for the spectral clustering algorithm was calculated using the 'nearest neighbors' method which computes a graph of nearest neighbors. The code snippet below was used to create 3 clusters. Random state was set as zero for reproducibility of results. Once the normalized Laplacian matrix was created from the graph network based on nearest neighbors, Cluster labels were generated using discretization approach which is less sensitive to random initialization of clusters (Pedregosa, 2011).

```
#Spectral Clustering on the PCA of original data
sc = SpectralClustering(n_clusters=3, affinity='nearest_neighbors', n_neighbors = 50,
                        assign_labels = 'discretize', random_state=0).fit(pc[:, 0:5])
```

```
labels = sc.labels_ #cluster labels for the 651 SWD wells
```

A scatter plot of principal components PC1 and PC2 was generated and one can see the 3 clusters generated using the spectral clustering algorithm.

```
plt.scatter(pc[:,0], pc[:,1], c=labels)
plt.show()
print(sc)
```

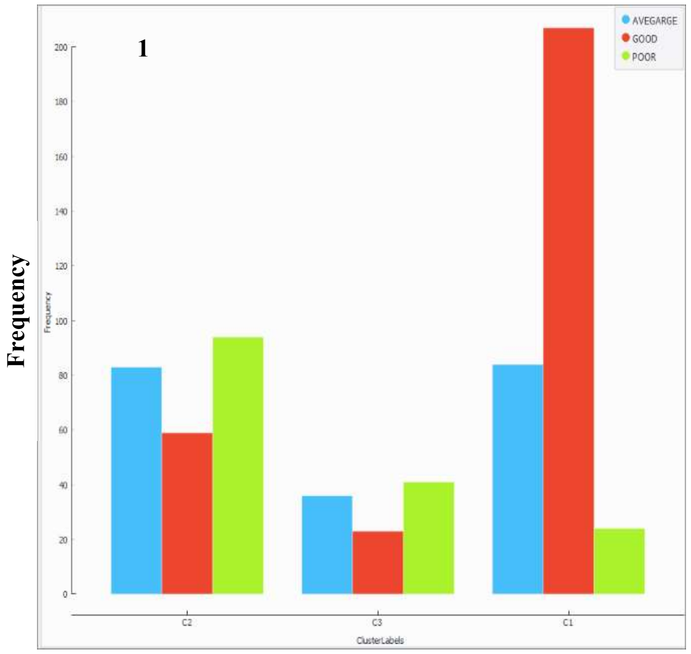


```
SpectralClustering(affinity='nearest_neighbors', assign_labels='discretize',
                   n_clusters=3, n_neighbors=50, random_state=0)
```

The cluster labels were merged with the input dataset and was saved as an excel file. This excel file was read in Orange and the two visualizations, similar to Figure 4.21, were created to understand the clusters (Figure 4.22 below).

The clusters are like the ones generated using Louvain clustering. Snapshot interpretations are given below:

1. C1 is dominated by the 'Good' SWD wells while C2 is dominated more by 'Poor' wells followed by 'Average' wells.
2. Scatter plot shows reasonable alignment Good, Average and Poor wells based on Well_Quality with clusters C1, C3 and C2. Considering that there is still good amount of overlap between C2 and C3, one can also interpret C1 as the cluster of 'Good' wells while clusters C2 and C3 together represents not 'Good' wells.



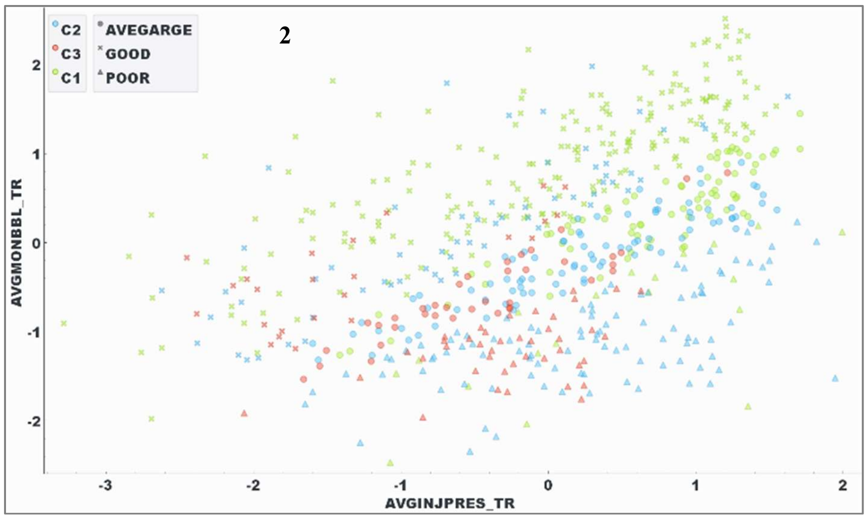


Fig 4.22 Two snapshots- Spectral clustering analysis

4.4 Knowledge Graphs

Knowledge graphs were introduced in Section 2.5 of Chapter 2 as the foundation of GDS. Knowledge graphs are interlinked sets of data points that describe real-world entities, facts, or things and their relationship with each other in a human-understandable form. They acquire and integrate adjacent information by using data relationships to derive new knowledge. Neo4j software platform's AuraDB application which is a fully managed cloud graph database service

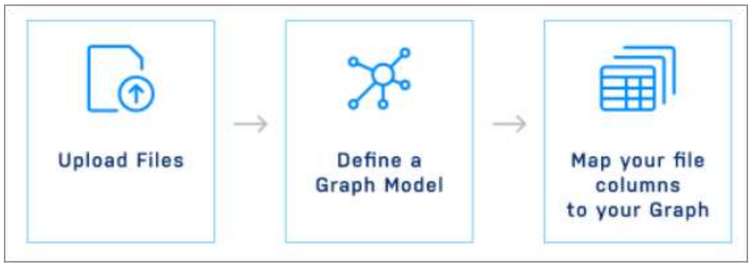
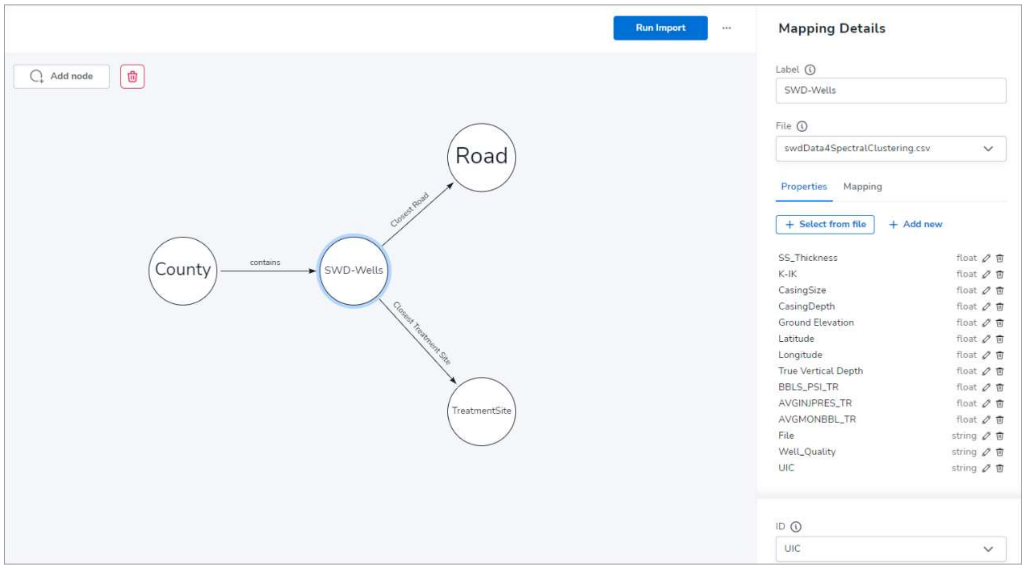


Fig 4.23 Neo4j AuraDB graph model creation workflow using Data Importer

that enables fast querying for real-time analytics and insights was used to create the knowledge graphs leveraging relationships in final data set. Figure 4.23 shows the workflow to define a graph model using the Neo4j's built-in Data Importer which provides a no-code approach to load data into Neo4j. The flat file (.csv) exported from Orange for spectral clustering was used to as the input file and nodes and relationships were identified to define the graph model as shown in Figure 4.24 below. The features in the input flat file were mapped to the graph model and it was imported to AuraDB for querying and analysis.

As seen in Figure 4.24, four nodes, SWD-Wells, County, TreatmentSite and Road were defined along with the associated properties. For example, SWD-Wells node, with UIC as the ID, contains properties such as Inyan Kara thickness, Inyan Kara formation top, ground elevation, true vertical depth, transformed versions of BBLS_PSI, average injection pressure



Mapping Details

Label: SWD-Wells

File: swdData4SpectralClustering.csv

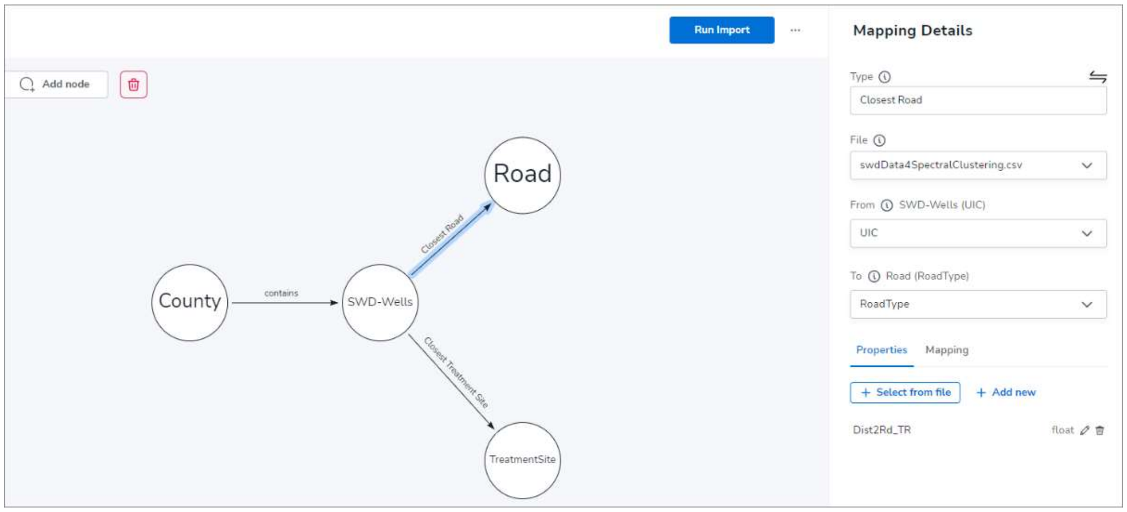
SS_Thickness	float
K-IK	float
CasingSize	float
CasingDepth	float
Ground Elevation	float
Latitude	float
Longitude	float
True Vertical Depth	float
BBLS_PSI_TR	float
AVGINJPRES_TR	float
AVGMONBBL_TR	float
File	string
Well_Quality	string
UIC	string

ID: UIC

Fig 4.24 Graph model for SWD wells

and average monthly barrels disposed as shown in the ‘Mapping Details’ in the right side of Figure 4.24. The other 3 nodes contained just one property – county name, road type and disposal facility name.

Three relationships were defined to complete the graph model. The ‘contains’ relationship connects SWD wells and the county where they are located, ‘Closest Road’ connects each SWD well to its closest road and has an associated property ‘distance to road’ as shown in Figure 4.25 below. ‘Closest Treatment Site’ connects each SWD well to its closest treatment site with ‘distance to disposal facility’ as the associated property.



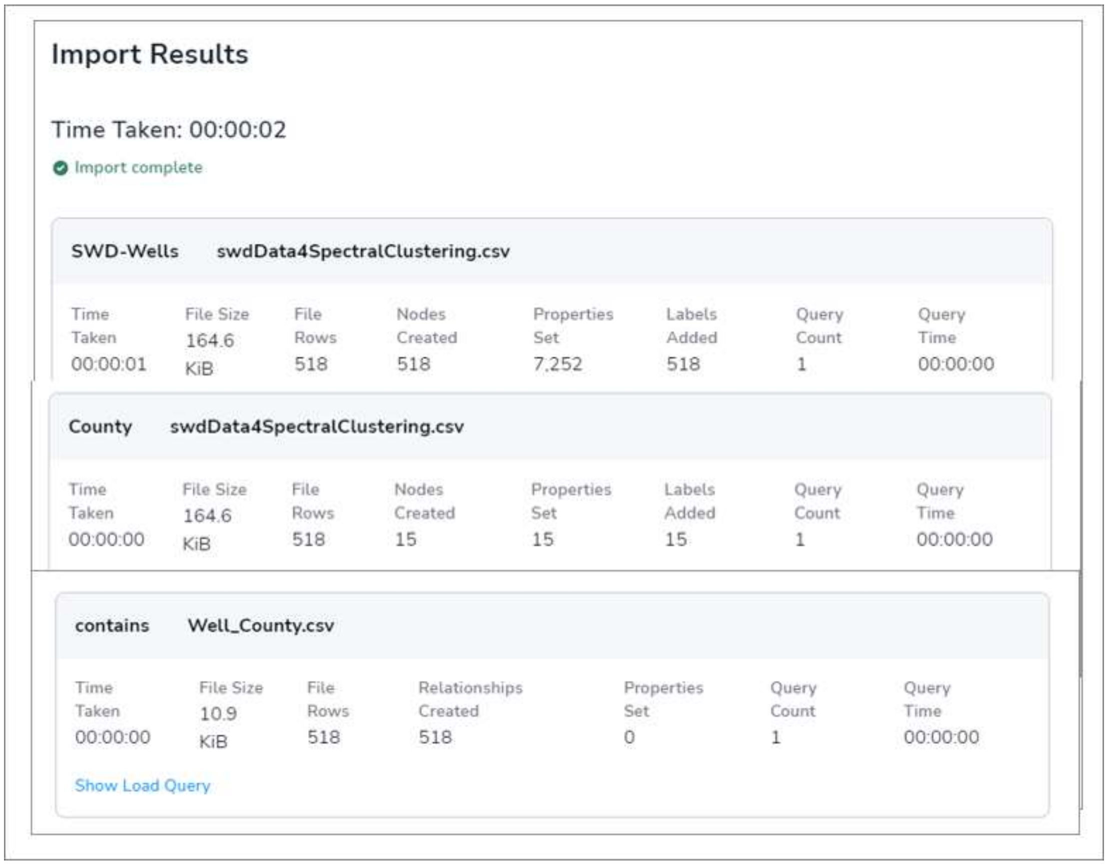
Type: Closest Road
File: swdData4SpectralClustering.csv
From: SWD-Wells (UIC) — UIC
To: Road (RoadType) — RoadType
Dist2Rd_TR float

Fig 4.25 Graph model showing ‘Closest Road’ relation and its mapping

The ‘Run Import’ tool was used to import the graph model and the associated data to the Neo4j AuraDB service. Figure 4.26 shows the import results for the SWD-Wells node, County node and the ‘contains’ relationship such as number of nodes, properties, labels, relationships and query counts.

The Neo4j Bloom no coding data visualization tool was used to explore and interact with the graph data and create knowledge graphs perspectives to gain quick insights. Figure 4.27 provides a quick visual insight that majority of the wells are located close to ‘ND’ state roads while least number of wells are located close to ‘Interstate’ highways. The edges between the

3 road type nodes and the SWD wells nodes are size weighted using the 'distance to road' property to get the insight of how close the wells are to their closest road type. Well legend, based on the Well_Quality feature enables quick identification of wells. Majority of the GOOD SWD wells are located close to ND and US road types.



Import Results

Time Taken: 00:00:02

Import complete

SWD-Wells swdData4SpectralClustering.csv

Time Taken	File Size	File Rows	Nodes Created	Properties Set	Labels Added	Query Count	Query Time
00:00:01	164.6 KiB	518	518	7,252	518	1	00:00:00

County swdData4SpectralClustering.csv

Time Taken	File Size	File Rows	Nodes Created	Properties Set	Labels Added	Query Count	Query Time
00:00:00	164.6 KiB	518	15	15	15	1	00:00:00

contains Well_County.csv

Time Taken	File Size	File Rows	Relationships Created	Properties Set	Query Count	Query Time
00:00:00	10.9 KiB	518	518	0	1	00:00:00

Show Load Query

Fig 4.26 Import results from Neo4j data importer

Figure 4.28 is another knowledge graph visualization that helps us to understand the relationship between SWD wells and counties they are located in. The red nodes represent the 15 counties and the edges connect the SWD wells to the respective counties where they are located. Well legend, based on the Well_Quality feature enables quick identification of wells. McKenzie county has the most of SWD wells followed by Williams county. Mountrail county has the most number of GOOD wells while McKenzie county also has a good proportion of GOOD wells. Billings is one of the counties with lower number of GOOD wells.

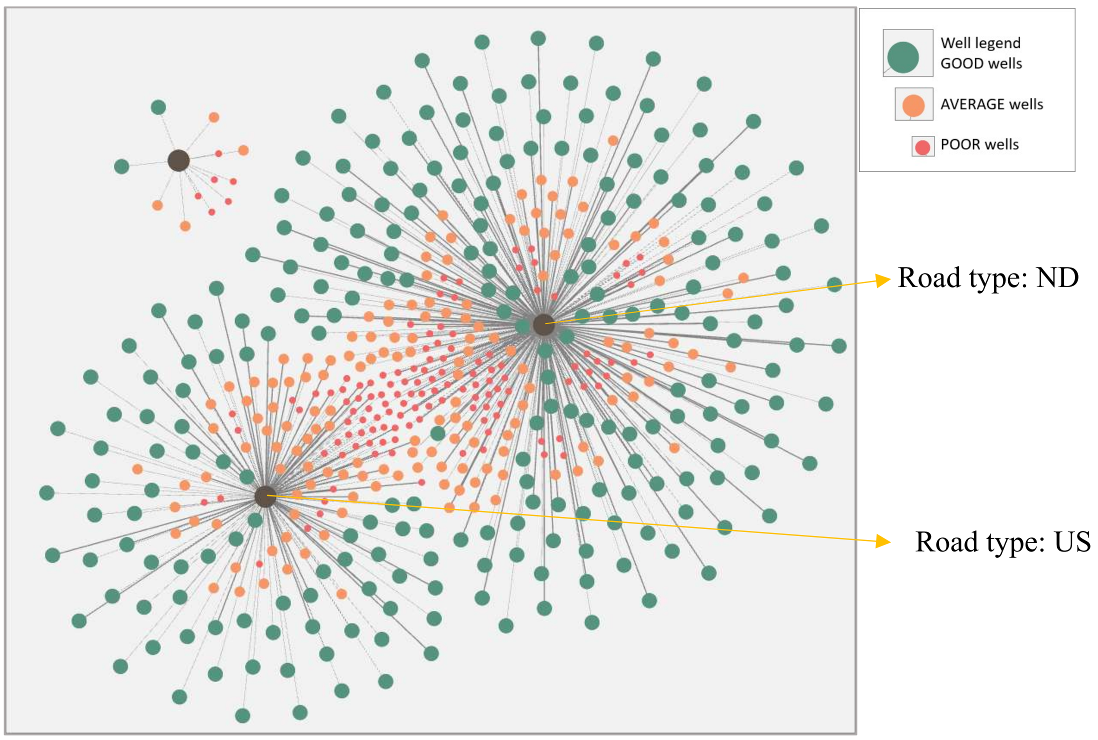


Fig 4.27 Knowledge graph- SWD wells and roads

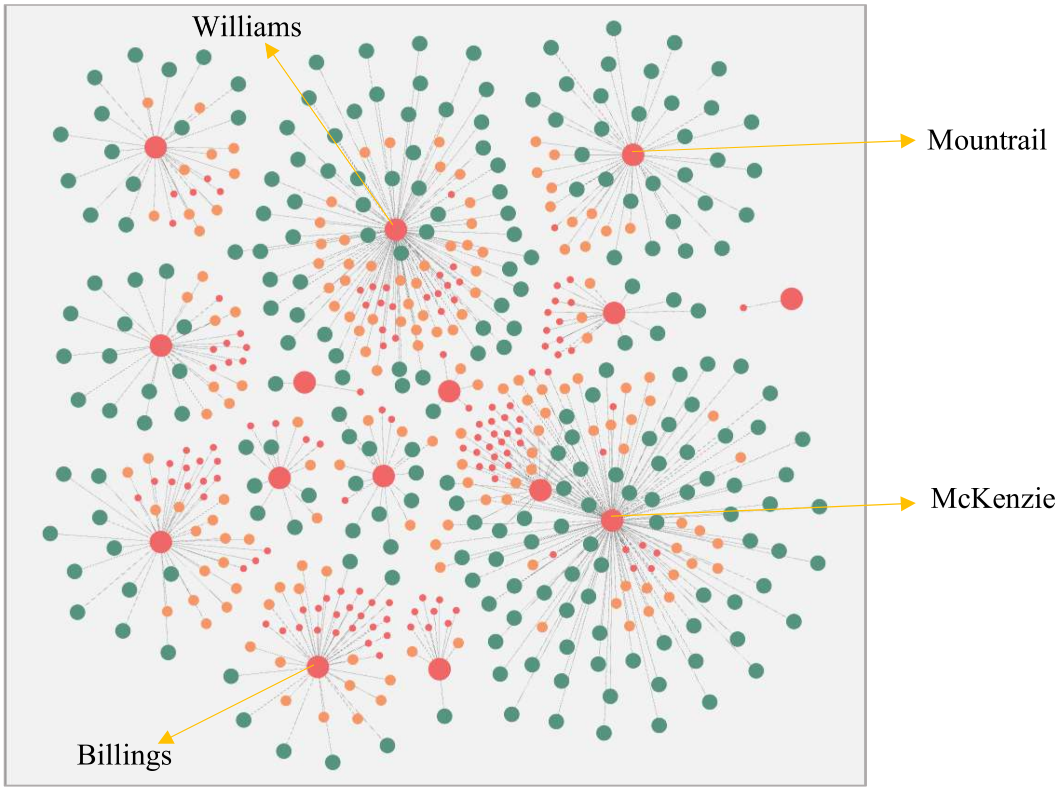


Fig 4.28 Knowledge graph – SWD wells and counties

4.5 Summary

In this chapter, we first presented the workflow to create the final data set using a nested merge operation. We then presented the results of data cleaning to prepare the final data set that contained 651 SWD wells. As part of EDA, we presented univariate analysis of some of the key features to understand SWD well performance using box plots, histograms and statistical summaries. We highlighted the nature of the distributions and the potential need for feature transformation. Results from correlation analysis, scatter plots, bar graphs and map visualizations were presented to gain a deeper understanding of the data set, key features and their relationships to understand SWD well performance.

The section on Clustering analysis started with an overview of feature transformation techniques and how this helped to convert distribution of skewed features to close to normal distributions. The final 19 continuous numeric features were presented along with results from PCA where the first five components with an explained variance of 75% was selected for clustering analysis. K-Means clustering results which identified two clusters was presented first followed by graph-based clustering using Louvain and spectral clustering methods. The clustering results from the graph-based clustering analysis showed an improved cluster representation compared to the k-Means algorithm. Cluster interpretation was presented using appropriate visualizations.

The chapter concluded with an overview of the graph data model created using Neo4j AuraDB and a couple of effective knowledge graph visualizations to gain quick insight of the SWD well performance and their relationships with counties and closest roads.

In the next chapter, we will take a look at the application of supervised machine learning algorithms as part of the data mining activities to model the performance of SWD wells and water production in oil and gas wells in the state of ND. Results from the model tuning and performance of the various models will be presented in this chapter.

Chapter 5

Supervised Machine Learning Models

In this chapter, we present the results of supervised machine learning regression models trained using Orange software considering well location and other relevant features to predict performance of SWD wells. The target variable and the predictors were identified from the final data set leveraging the results from the EDA and clustering analysis described in Chapter 4. We explain the machine learning algorithms identified, the model training and tuning process to arrive at the final models. We also present the results of the neural network multi-target regression model to predict water production in ND oil and gas wells.

5.1 SWD Well Performance Modeling

Three machine learning algorithms were considered for training regression models to predict saltwater performance. Regression is concerned with specifying the relationship between a single numeric dependent variable (the value to be predicted) and one or more numeric independent variables (the predictors) (Lantz, 2015).

Average monthly barrels of saltwater disposed (AvgMonthlyBBLSdisposed) was identified as the target variable from the list of features shown in Figure 4.13. We will briefly discuss the three machine learning algorithms.

5.1.1 Machine Learning Algorithms

k-Nearest Neighbors (kNN): The kNN approach predicts a new sample using the k-closest samples from the training set. kNN is a non-parametric algorithm that does not make any assumptions on the distribution of the underlying data. The construction of this approach is solely based on individual samples in the training data. kNN algorithm is a dual use algorithm that can be used both for classification and regression. To predict a new sample for regression, kNN algorithm identifies the sample's k-nearest neighbors in the predictor space. The predicted response is then the mean of the k-nearest neighbors' responses. The nearest neighbors are identified using distances between the samples. Equation 2.1 which defines the Euclidean distance between samples and is a commonly used metric can be generalized to the Minkowski distance as shown in equation 5.1 below where the distance is same as Euclidean for q=2. For q=1, it is known as the Manhattan (city-block) distance.

$$D^q = \sum_{i=1}^{k} |x_{A_i} - x_{B_i}|^q \ldots\ldots\ldots\ldots\ldots\ldots 5.1$$

Due to its strong dependency on the distance between samples, normalization of the predictors using max-min method (equation 4.6) or z-score method (equation 4.7) is essential to avoid potential bias of predictors with large scales and to enable all the predictors to contribute equally to the distance calculation. Distance between samples cannot be computed if there are missing values in the predictors and hence missing values should be imputed or samples with missing values should be removed. Once the preprocessing steps of normalization and imputing are complete, kNN algorithm can be used to train the models using number of neighbors, k as the tuning parameter and model performance can be evaluated using RMSE (equation 2.3). Distance metric, q, can also be used as a tuning parameter. Orange software has a kNN widget that is used to train kNN regression models.

Linear Regression: Linear regression is a parametric algorithm to describe the relationship between a target (dependent) variable and one or more predictors (independent variables). The algorithm also provides the ability to predict the value of the target variable for a new sample using the trained model. Simple and multi-linear algorithm concepts based on ordinary least squares method were presented in Section 2.4.4 of Chapter 2. Both the kNN and Linear Regression algorithms are considered as base algorithms.

Random Forest: The technique of combining and managing the predictions of multiple models is known as meta-learning; an approach that utilizes the principle of creating a varied team of experts known as ensemble. Ensemble methods are based on the idea of combining multiple weaker learners to create a stronger learner. Figure 5.1 below shows the process diagram for an ensemble model.

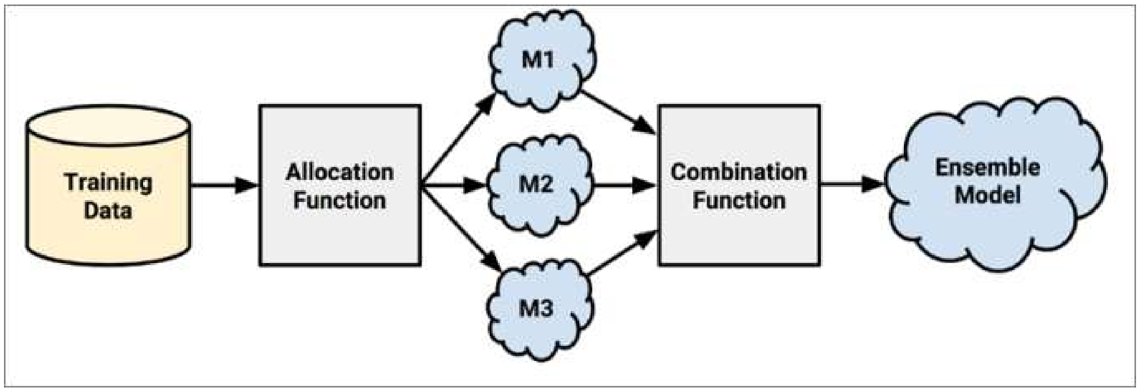


Fig 5.1 Process diagram for an ensemble model

An allocation function dictates how much training data each of the models receive. The allocation function can increase diversity by artificially varying the input data to bias the resulting learners, even if they are the same type. For instance, it might use bootstrap sampling to construct unique training datasets or pass on a different subset of features or examples to each model. After the models are constructed, they can be used to generate a set of predictions. The combination function governs how disagreements among the predictions are reconciled. For example, the ensemble might use a majority vote to determine the final prediction, or it could use a more complex strategy such as weighting each model's votes based on its prior performance. Ensemble methods have better generalizability to future problems by reducing the chance of overfitting the model to the training dataset as several learners are incorporated to arrive at the final prediction. They also tend to perform better on both massive and miniscule datasets (Lantz, 2015).

The Random Forest ensemble-based algorithm, based on ensembles of decision trees, combines the principles of bagging with random feature selection to add diversity to the base decision tree models. Decision trees are powerful dual use algorithms which utilize a tree structure to model relationships among features and the potential outcomes. They are based on a heuristic approach called recursive partitioning based on divide and conquer strategy. While the decision tree algorithm used for classification task uses the concept of entropy introduced in Section 4.2 of Chapter 2 as a measure of homogeneity to decide the feature to split, splitting criterion of Standard Deviation Reduction (SDR) is used as the measure of homogeneity for regression task. As the ensemble uses only a small, random portion of the full feature set, random forests can handle datasets with a large number of features, where the so-called "curse of dimensionality" might cause other models to fail. At the same time, its error rates for most learning tasks are on par with nearly any other method (Lantz, 2015).

5.1.2 Feature Selection and Preparation

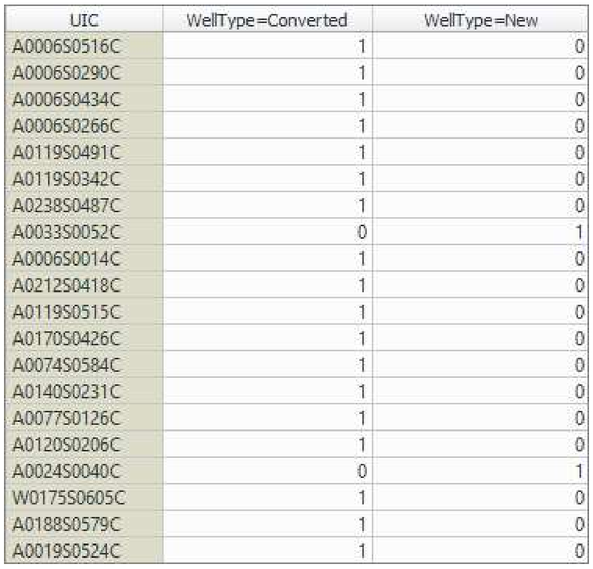
The EDA analysis carried out in Chapter 4 helped in identifying the target variable and the predictors for the regression modeling. In addition to considering all the numerical features identified in Figure 4.13, the following categorical features were also considered.

- WellType- This feature was created as part of feature engineering based on whether an SWD well was drilled as a new well or if an old oil / gas well was converted to an SWD well. 32% of the 651 SWD wells considered were 'Converted' type and the remaining 68% of the SWD wells were 'New' type.
- Wellbore- This feature contained orientation of the SWD wells. Values were 'VERTICAL'- majority of the SWD wells (85%) were of vertical orientation,

‘HORIZONTAL’- 9% of the wells were of horizontal orientation including 1% of horizontal re-entry wells and ‘DIRECTIONAL’- remaining 6% of the SWD wells.

- RoadType- Figure 4.27 shows a visual representation of the 3 road types. ‘I’ – Only 2% of the SWD wells were located close to Interstate highways, ‘US’- 33% of the SWD wells were located close to US roads and ‘ND’- majority of the SWD wells (65%) were located close to North Dakota state roads.

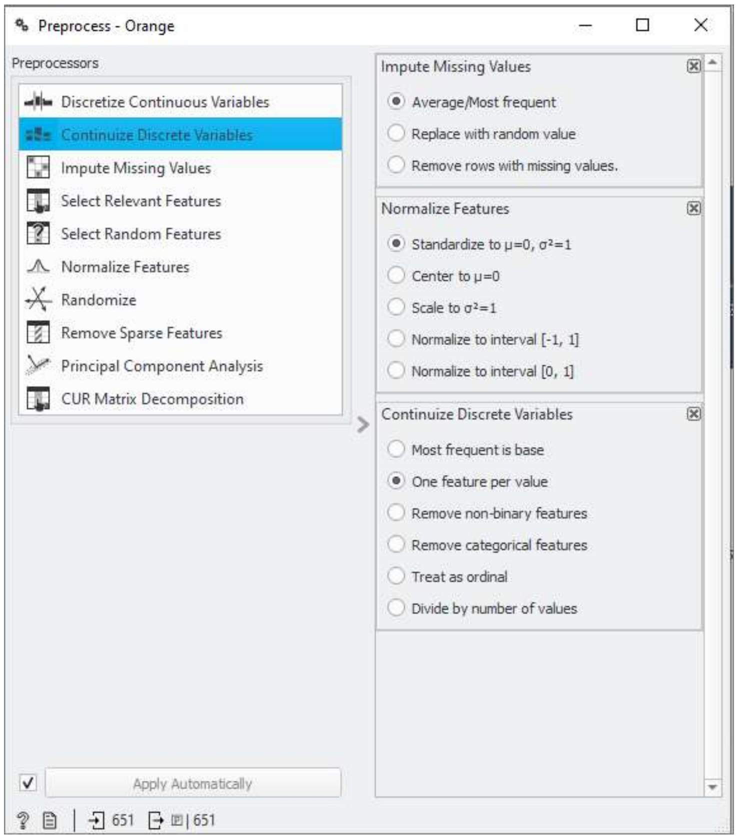
When considering categorical features as predictors, their values should be converted to numerical format using dummy encoding for algorithms such as kNN and linear regression. The Orange ‘Preprocess’ widget introduced in Chapter 4 provides an option for dummy encoding of categorical features. The ‘One Hot Encoding’ approach was used which creates one numerical feature per categorical value of the given categorical feature by placing a value of 1 where a sample has that categorical value and zero otherwise (Demsar, et al., 2013). Figure 5.2 shows an example of ‘One Hot Encoding’ results for the feature ‘WellType’ for twenty SWD wells.



UIC	WellType=Converted	WellType=New
A0006S0516C	1	0
A0006S0290C	1	0
A0006S0434C	1	0
A0006S0266C	1	0
A0119S0491C	1	0
A0119S0342C	1	0
A0238S0487C	1	0
A0033S0052C	0	1
A0006S0014C	1	0
A0212S0418C	1	0
A0119S0515C	1	0
A0170S0426C	1	0
A0074S0584C	1	0
A0140S0231C	1	0
A0077S0126C	1	0
A0120S0206C	1	0
A0024S0040C	0	1
W0175S0605C	1	0
A0188S0579C	1	0
A0019S0524C	1	0

Fig 5.2 One Hot Encoding results for WellType categorical feature

Table 4.1 in Chapter 4, which presented statistical summaries of key numerical features considered, also included percentage of missing values. As part of preprocessing, missing values were imputed using mean values for numerical features. Numerical features were normalized using the z-score normalization, similar to what was done for clustering analysis. Figure 5.3 shows the snapshot of the Orange 'Preprocess' widget and the various preprocessing tasks carried out prior to regression modeling and analysis.



Impute Missing Values
[✓ Average/Most frequent]
Replace with random value
Remove rows with missing values.

Normalize Features
[✓ Standardize to μ=0, σ²=1]
Center to μ=0
Scale to σ²=1
Normalize to interval [-1, 1]
Normalize to interval [0, 1]

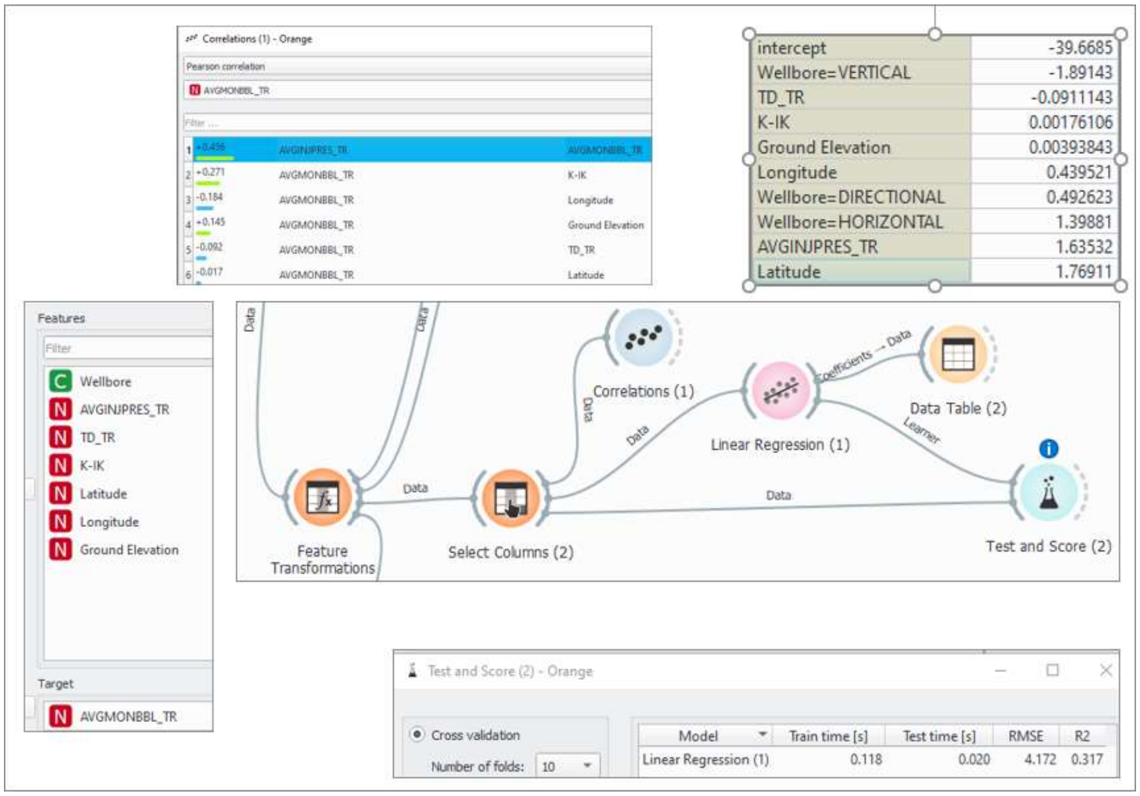
Continuize Discrete Variables
Most frequent is base
[✓ One feature per value]
Remove non-binary features
Remove categorical features
Treat as ordinal
Divide by number of values

Fig 5.3 Orange preprocessing for regression analysis

The concept of feature transformation and its benefits in data preparation were presented in Section 4.3.1 of Chapter 4. A fundamental assumption for linear regression algorithm model validity is that the errors (residuals) computed as difference between the predicted values by the model and the actual values has a zero mean and a constant variance. Feature transformation applied to the target variable helps in addressing this requirement and ensuring model validity. Hence, AVGMONBBL_TR, transformed version of the 'AvgMonthlyBBLSdisposed' was used as the target variable for training regression models.

5.1.3 The Base Model

Once the features from the primary data set were extracted and examined, a base linear regression model was created to understand the relationship between the predictors and the target variable. Figure 5.4 below shows the Orange workflow for the base model. This workflow is an extension of the Orange workflow in Figure 4.1. Six numerical features and one categorical feature were used as predictors. The correlation widget displays the Pearson correlation of the numerical predictors to the target. AVGINJPRES_TR has the highest correlation followed by K-IK (Inyan Kara formation top). The Linear Regression widget uses default preprocessing to carry out the preprocessing steps highlighted in Figure 5.3. The coefficient table shows the parameters of the linear regression model once the training is completed. Latitude and AVGINJPRES_TR are the two predictors with the highest positive impact on predicting the target variable. While an intercept value of -39.6685 does not have any interpretive meaning, it is part of the model parameters and is used to predict the target for future values of the predictors. The 'Test and Score' widget in Orange is used to test the linear regression model that was trained using 10-fold cross validation sampling technique. Model performance measures selected were RMSE and R^2 which were introduced in Section 2.4.5 of Chapter 2. R^2 value of 0.317 can be interpreted as 31.7% of variance in AVGMONBBL_TR target variable can be explained by the base linear regression model and RMSE is 4.172. While this is not a strong model, it does indicate that the relationship between the target and predictors is worth examining to explore the potential for an improved model. As shown in Figure 1.4, data mining and its steps are iterative in nature. We will present the results of linear regression and other models considering the additional features extracted using primary and secondary data sources.



AVGMONBBL_TR

1	+0.456	AVGINJPRES_TR	AVGMONBBL_TR
2	+0.271	AVGMONBBL_TR	K-IK
3	-0.184	AVGMONBBL_TR	Longitude
4	+0.145	AVGMONBBL_TR	Ground Elevation
5	-0.092	AVGMONBBL_TR	TD_TR
6	-0.017	AVGMONBBL_TR	Latitude

intercept	-39.6685
Wellbore=VERTICAL	-1.89143
TD_TR	-0.0911143
K-IK	0.00176106
Ground Elevation	0.00393843
Longitude	0.439521
Wellbore=DIRECTIONAL	0.492623
Wellbore=HORIZONTAL	1.39881
AVGINJPRES_TR	1.63532
Latitude	1.76911

Features: C Wellbore, N AVGINJPRES_TR, N TD_TR, N K-IK, N Latitude, N Longitude, N Ground Elevation

Target: N AVGMONBBL_TR

Model	Train time [s]	Test time [s]	RMSE	R2
Linear Regression (1)	0.118	0.020	4.172	0.317

Fig 5.4 Orange workflow and results for base linear regression model

5.1.4 Advanced Models and Results

Figure 5.5 is the Orange workflow used to train Linear regression, kNN and Random Forest models on the final data set. Figure 5.6 shows the final list of features used to train machine learning models to predict AVGMONBBL_TR, the target variable. Principal Component Analysis (PCA) was carried out on the features using a similar process as presented in Section 4.3.2 of Chapter 4. Explained variance from 10 PC was 86% and 11 components explained variance was 90%. Models were trained using principal components as well but were not considered due to the performance metrics. The Preprocess widget used is the same as Figure 5.3.

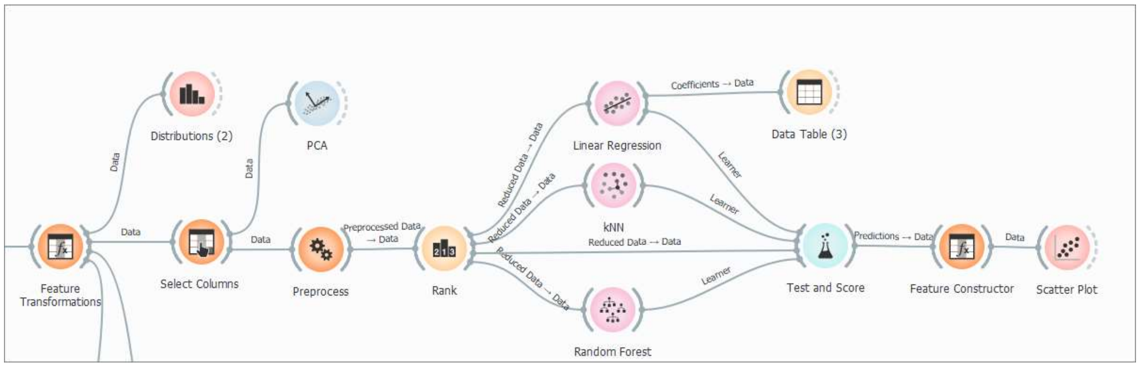
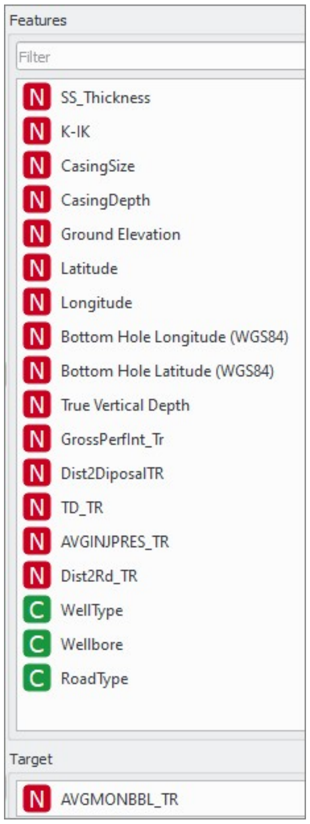


Fig 5.5 Orange workflow for final regression models


Features

Filter

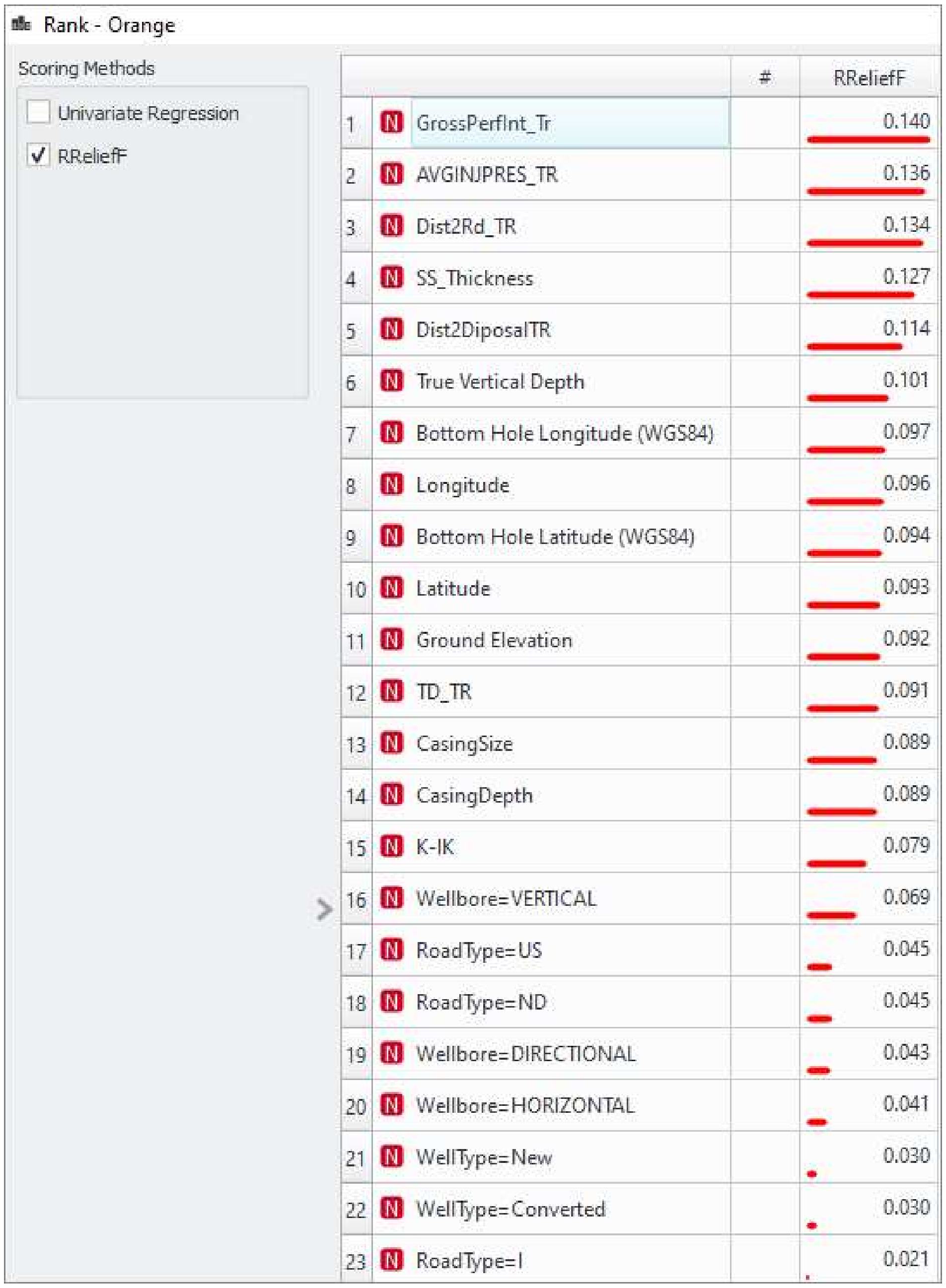
N SS_Thickness
N K-IK
N CasingSize
N CasingDepth
N Ground Elevation
N Latitude
N Longitude
N Bottom Hole Longitude (WGS84)
N Bottom Hole Latitude (WGS84)
N True Vertical Depth
N GrossPerfInt_Tr
N Dist2DiposalTR
N TD_TR
N AVGINJPRES_TR
N Dist2Rd_TR
C WellType
C Wellbore
C RoadType

Target

N AVGMONBBL_TR

Fig 5.6 Final set of features

The ‘Rank’ widget available in Orange software was used to understand the importance of the features in Figure 5.6. For regression, Rank widget has two scoring options- univariate regression based on linear regression for a single variable and RReliefF, which is based on the relative distance between predicted values of two samples or instances. Figure 5.7 below shows the results of the more robust RReliefF method for ranking the selected features.

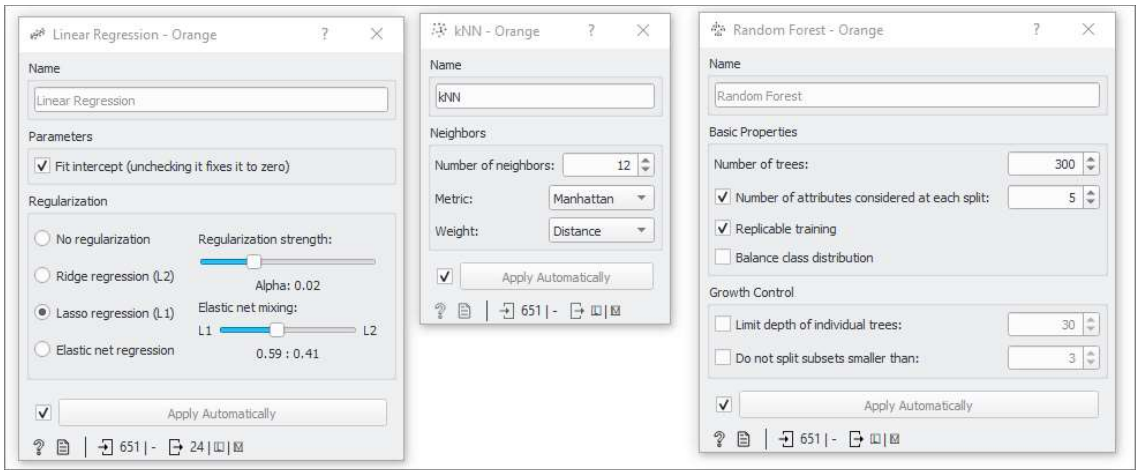


☐ Univariate Regression
✓ RReliefF

		#	RReliefF
1	GrossPerfInt_Tr		0.140
2	AVGINJPRES_TR		0.136
3	Dist2Rd_TR		0.134
4	SS_Thickness		0.127
5	Dist2DiposalTR		0.114
6	True Vertical Depth		0.101
7	Bottom Hole Longitude (WGS84)		0.097
8	Longitude		0.096
9	Bottom Hole Latitude (WGS84)		0.094
10	Latitude		0.093
11	Ground Elevation		0.092
12	TD_TR		0.091
13	CasingSize		0.089
14	CasingDepth		0.089
15	K-IK		0.079
16	Wellbore=VERTICAL		0.069
17	RoadType=US		0.045
18	RoadType=ND		0.045
19	Wellbore=DIRECTIONAL		0.043
20	Wellbore=HORIZONTAL		0.041
21	WellType=New		0.030
22	WellType=Converted		0.030
23	RoadType=I		0.021

Fig 5.7 Feature ranking using RReliefF method

With all the 23 features in Figure 5.7 selected, three machine learning models were trained using Linear Regression, kNN and Random Forest algorithms. Figure 5.8 below shows a snapshot of the 3 model training widgets from Orange with the final set of parameters.


Linear Regression - Orange

Name: Linear Regression

Parameters

[✓] Fit intercept (unchecking it fixes it to zero)

Regularization

○ No regularization
○ Ridge regression (L2)
● Lasso regression (L1)
○ Elastic net regression

Regularization strength: Alpha: 0.02

Elastic net mixing: L1 – L2 0.59 : 0.41

[✓] Apply Automatically

651 | - 24

kNN - Orange

Name: kNN

Neighbors

Number of neighbors: 12

Metric: Manhattan

Weight: Distance

[✓] Apply Automatically

651 | -

Random Forest - Orange

Name: Random Forest

Basic Properties

Number of trees: 300

[✓] Number of attributes considered at each split: 5

[✓] Replicable training

[ ] Balance class distribution

Growth Control

[ ] Limit depth of individual trees: 30

[ ] Do not split subsets smaller than: 3

[✓] Apply Automatically

651 | -

Fig 5.8 Model widgets with the final parameters

Model tuning: The modeling widgets shown in Figure 5.8 were used along with the Test and Score widget to interactively tune each of the models. For linear regression, Lasso (Least Absolute Shrinkage and Selection Operator) model was selected and the tuning parameter- Alpha with a range of [0.0001, 1000] was tuned to find the alpha value (0.02) that yielded the lowest RMSE of 3.862 using 10-fold cross validation sampling method (Figure 5.9). Lasso regression is one method of regularization to train complex models on data sets without severe overfitting or address issues of collinearity by limiting the effective model complexity. Regularization is accomplished by adding a penalty term to the sum of squared error (SSE) term as shown in equation 5.2 below where y_i is the actual value of target, $\hat{y_i}$ is the predicted value and B_j are the predictor coefficient.The L1 penalty term signifies first order penalty and λ(Alpha in Orange) is a tuning parameter. In addition to regularization to improve model performance, Lasso model assists in feature selection by setting some of the coefficients to zero (Kuhn & Johnson, 2013). Figure 5.10 shows the coefficients table for the tuned Lasso model and 7 coefficients are set to zero.

$$SSE_{L_1} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |B_j| \text{...} 5.2$$

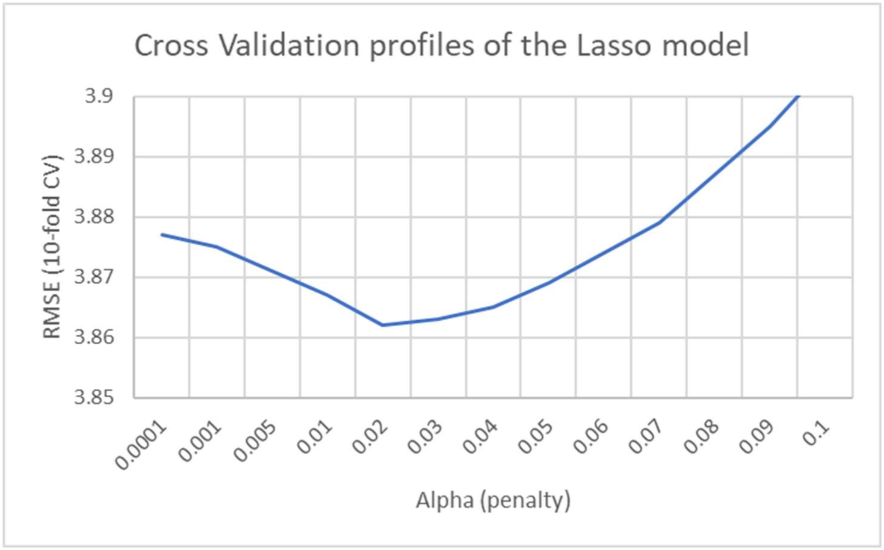
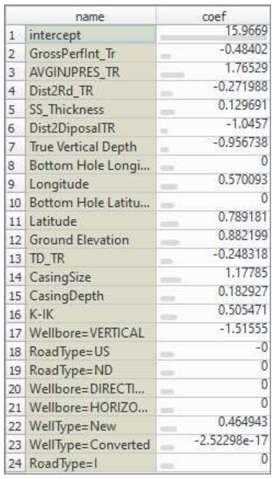


Fig 5.9 Cross validation RMSE profile for the Lasso regression model



	name	coef
1	intercept	15.9669
2	GrossPerfInt_Tr	-0.48402
3	AVGINJPRES_TR	1.76529
4	Dist2Rd_TR	-0.271988
5	SS_Thickness	0.129691
6	Dist2DiposalTR	-1.0457
7	True Vertical Depth	-0.956738
8	Bottom Hole Longi...	0
9	Longitude	0.570093
10	Bottom Hole Latitu...	0
11	Latitude	0.789181
12	Ground Elevation	0.882199
13	TD_TR	-0.248318
14	CasingSize	1.17785
15	CasingDepth	0.182927
16	K-IK	0.505471
17	Wellbore=VERTICAL	-1.51555
18	RoadType=US	-0
19	RoadType=ND	0
20	Wellbore=DIRECTI...	0
21	Wellbore=HORIZO...	0
22	WellType=New	0.464943
23	WellType=Converted	-2.52298e-17
24	RoadType=I	0

Fig 5.10 Final Lasso Regression model coefficients

For kNN, the primary tuning parameter is the number of neighbors(k). A range of [2,30] was used to tune this parameter. In addition, both Euclidean and Manhattan distance metrics were considered along with uniform and distance-based weighting of the neighbors. Twelve neighbors and Manhattan distance metric with distance-based weighting yielded the lowest RMSE of 3.731, a 3.4% improvement over the best Lasso regression model. R^2 also improved by 4% points (Figure 5.11).

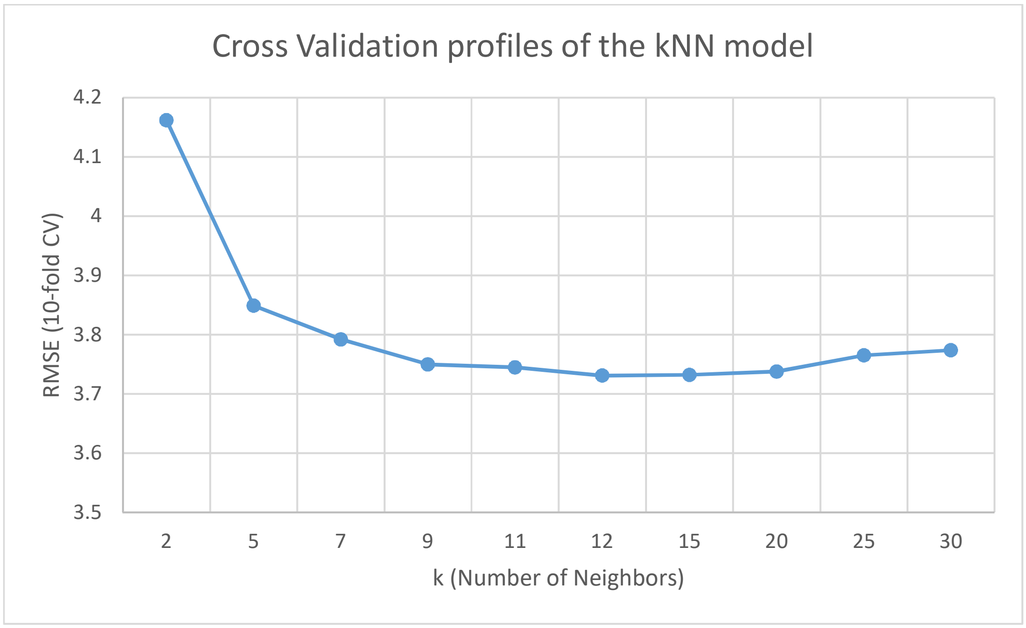


Fig 5.11 Cross Validation profiles of kNN model

For the Random Forest algorithm, two parameters, the number of decision trees and number of attributes considered at each split, were tuned for a combination of values as shown in Figure 5.12 below along with the RMSE from 10-fold cross validation.

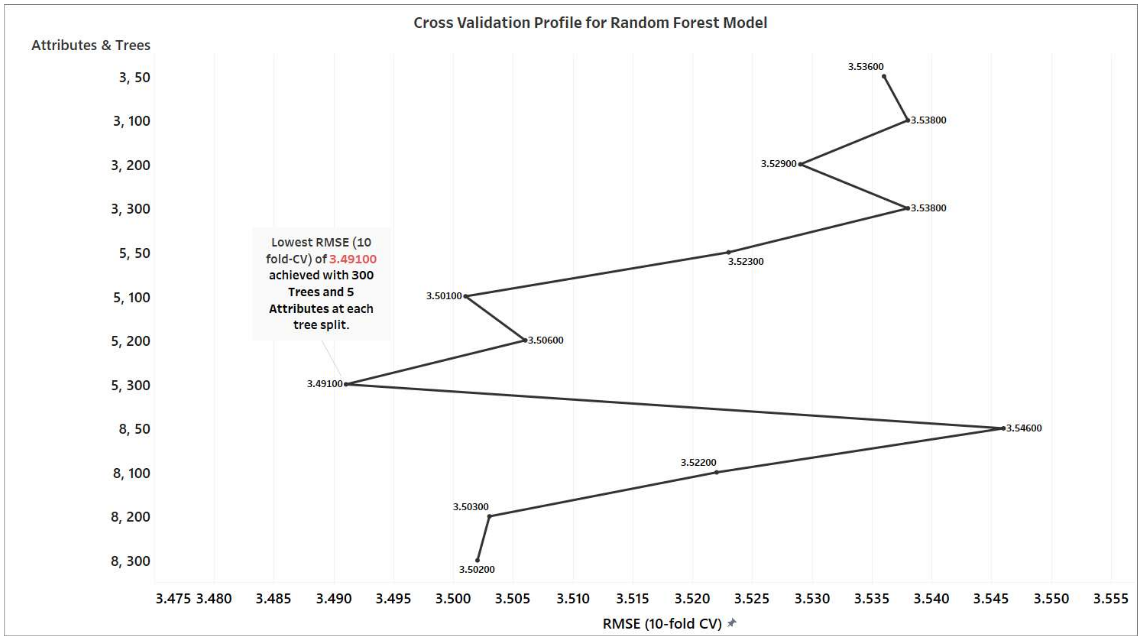
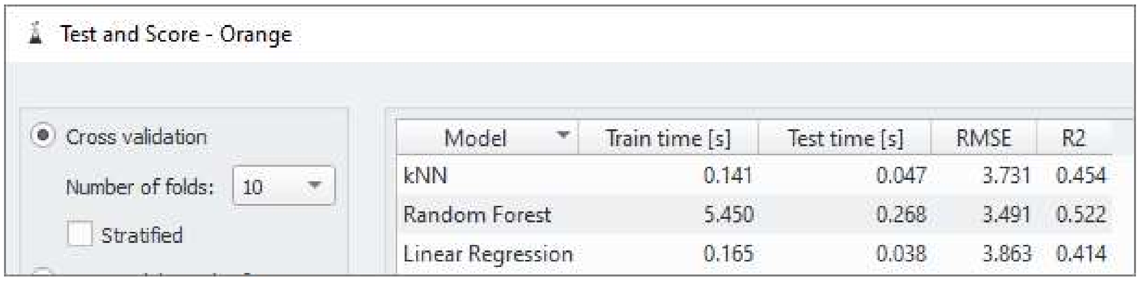


Fig 5.12 Cross Validation profiles of the Random Forest models

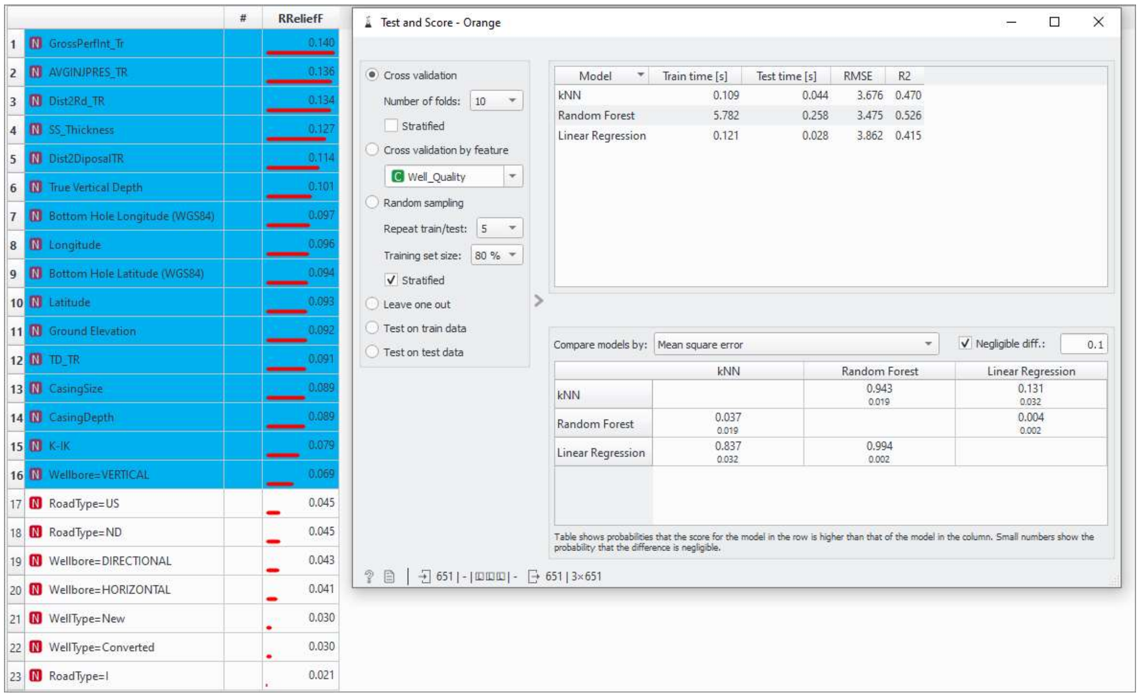
Parameter combination of 300 trees with 5 attributes at each node split yielded the lowest RMSE of 3.491, a 6% improvement over the best kNN model and 10% improvement over the best Lasso regression model. All the Random Forest models outperformed the best kNN and Lasso Regression models. Figure 5.13 is a snapshot of the 'Test and Score' widget with performance of the best models from the 3 algorithms. The Random Forest model's R^2 value showed a 11% improvement over the best Lasso regression model and a 7% improvement over the best kNN model.



Model	Train time [s]	Test time [s]	RMSE	R2
kNN	0.141	0.047	3.731	0.454
Random Forest	5.450	0.268	3.491	0.522
Linear Regression	0.165	0.038	3.863	0.414

Fig 5.13 Comparison of the best kNN, Random Forest and Linear Regression (Lasso) models

Prior to deciding on the parameters for the best Random Forest model, the feature ranking shown in Figure 5.7 was used to eliminate the lowest ranked features and model performance was monitored.



		#	RReliefF
1	GrossPerfInt_Tr		0.140
2	AVGINJPRES_TR		0.136
3	Dist2Rd_TR		0.134
4	SS_Thickness		0.127
5	Dist2DiposalTR		0.114
6	True Vertical Depth		0.101
7	Bottom Hole Longitude (WGS84)		0.097
8	Longitude		0.096
9	Bottom Hole Latitude (WGS84)		0.094
10	Latitude		0.093
11	Ground Elevation		0.092
12	TD_TR		0.091
13	CasingSize		0.089
14	CasingDepth		0.089
15	K-IK		0.079
16	Wellbore=VERTICAL		0.069
17	RoadType=US		0.045
18	RoadType=ND		0.045
19	Wellbore=DIRECTIONAL		0.043
20	Wellbore=HORIZONTAL		0.041
21	WellType=New		0.030
22	WellType=Converted		0.030
23	RoadType=I		0.021

Model	Train time [s]	Test time [s]	RMSE	R2
kNN	0.109	0.044	3.676	0.470
Random Forest	5.782	0.258	3.475	0.526
Linear Regression	0.121	0.028	3.862	0.415

Compare models by: Mean square error ✓ Negligible diff.: 0.1

	kNN	Random Forest	Linear Regression
kNN		0.943 0.019	0.131 0.032
Random Forest	0.037 0.019		0.004 0.002
Linear Regression	0.837 0.032	0.994 0.002	

Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.

Fig 5.14 Final model results with top 16 features

The best performing model was with the top 16 features selected and the model results are shown in Figure 5.14. To summarize, the best performing model to predict AVGMONBBL_TR using various predictors that captured the SWD well location information along with other features was the Random Forest model with the top 16 ranked features, 300 decision trees and 5 attributes considered at each node split. This model's RMSE using 10-fold cross validation showed a marked improvement of 17% compared to the base model (Figure 5.4).

Figure 5.15 shows the model diagnostics plots for the final Random Forest model. The scatter plot of residual was computed using the 'Feature Constructor' widget as the difference between predicted values and the actual values and it appears to be a null plot. The distribution of the residual has a mean close to zero and constant variance confirming the validity of the model. Scatter plot of the predicted versus actual values is also presented to help understand the performance of the model. Regression line shows a correlation of 0.73. An ideal model will have a correlation of 1.

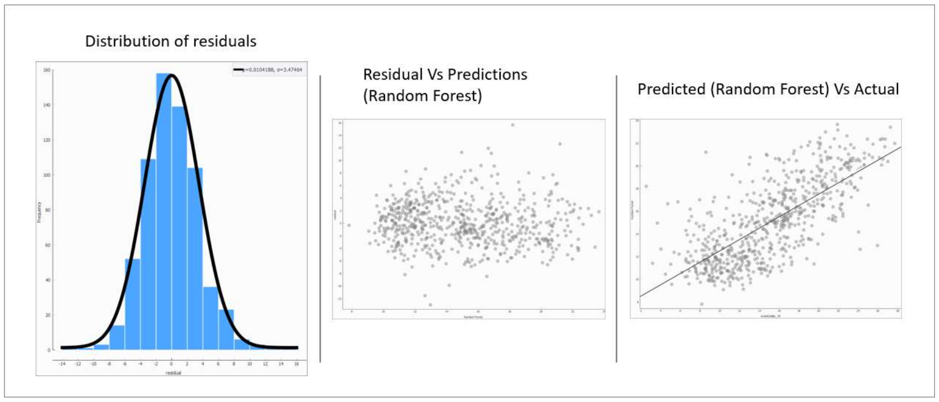
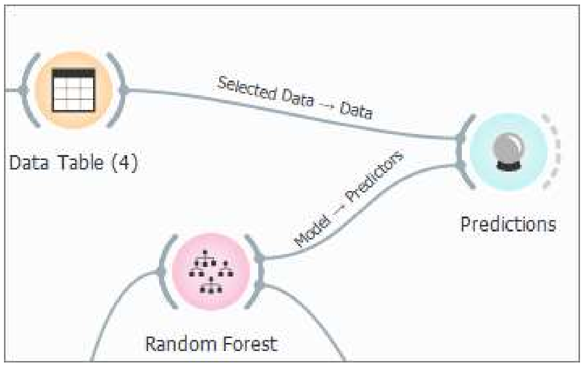


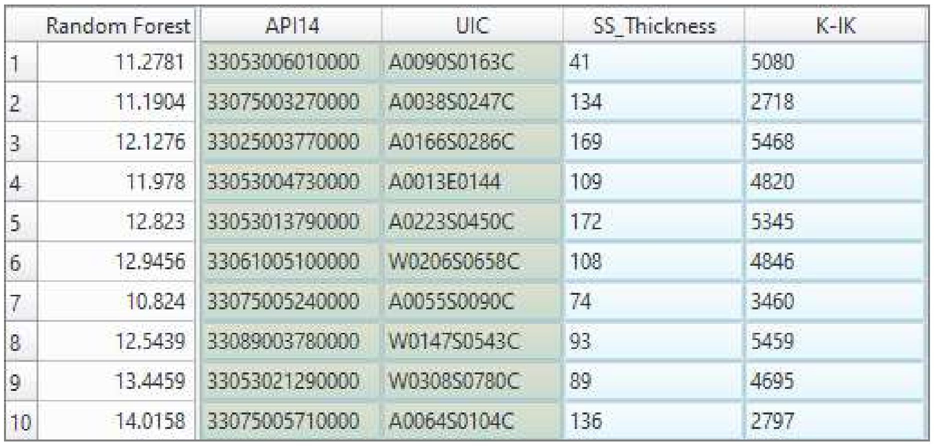
Fig 5.15 Model diagnostics plots for the final Random Forest model

5.1.5 Model Deployment

As discussed in Section 2.4.5 of Chapter 2, model deployment is an essential final step to derive business benefit from the modeling project. We present the procedure to deploy the best performing Random Forest model to predict performance of a newly planned SWD well using

Orange. Figure 5.16 below shows the model deployment of the Random Forest model on a data set not seen by the model and the value of the target variable, AVGMONBBL_TR is unknown. The 'Predictions' widget in Orange software can be used to predict target values by deploying one or more machine learning models. The sample data set for model deployment was selected from the list of SWD wells that were filtered out due to zero values of AvgInjPres (Section 4.1 of Chapter 4). Same transformations were used on the features as Figure 4.11. The preprocessing steps applied during model training such as normalization, imputing etc., are automatically applied by Orange on the features in the new data set on which target values must be predicted. The results of the predictions by the Random Forest model are also shown in Figure 5.16 below. Note that not all the features are displayed. As a final step, the predicted values must be re-transformed by using the inverse transformation on AVGMONBBL_TR (Figure 4.11).





	Random Forest	API14	UIC	SS_Thickness	K-IK
1	11.2781	33053006010000	A0090S0163C	41	5080
2	11.1904	33075003270000	A0038S0247C	134	2718
3	12.1276	33025003770000	A0166S0286C	169	5468
4	11.978	33053004730000	A0013E0144	109	4820
5	12.823	33053013790000	A0223S0450C	172	5345
6	12.9456	33061005100000	W0206S0658C	108	4846
7	10.824	33075005240000	A0055S0090C	74	3460
8	12.5439	33089003780000	W0147S0543C	93	5459
9	13.4459	33053021290000	W0308S0780C	89	4695
10	14.0158	33075005710000	A0064S0104C	136	2797

Fig 5.16 Orange model deployment using the Predictions widget and the prediction results

5.2 Multi-Target Regression Modeling

In this section, we present the results from multi-target regression model to predict water production from horizontal wells in the McKenzie county of ND. Data set (d) from Table 3.2 of Section 3.2.2, Chapter 3 was used as the source data.

5.2.1 Neural Network Algorithm

Artificial Neural Network (ANN) models the relationship between a set of input signals and an output signal using a model derived from our understanding of how a biological brain responds to stimuli from sensory inputs. Just as a brain uses a network of interconnected cells called neurons to create a massive parallel processor, ANN uses a network of artificial neurons or nodes to solve learning problems (Lantz, 2015).

According to Lantz (2015), a neural network can be identified by the following three characteristics:

- An activation function that transforms a neuron's combined input signals into a single output to be broadcasted further in the network. A commonly used activation function is the ReLU function, a graph of which is shown in Figure 5.17 below:

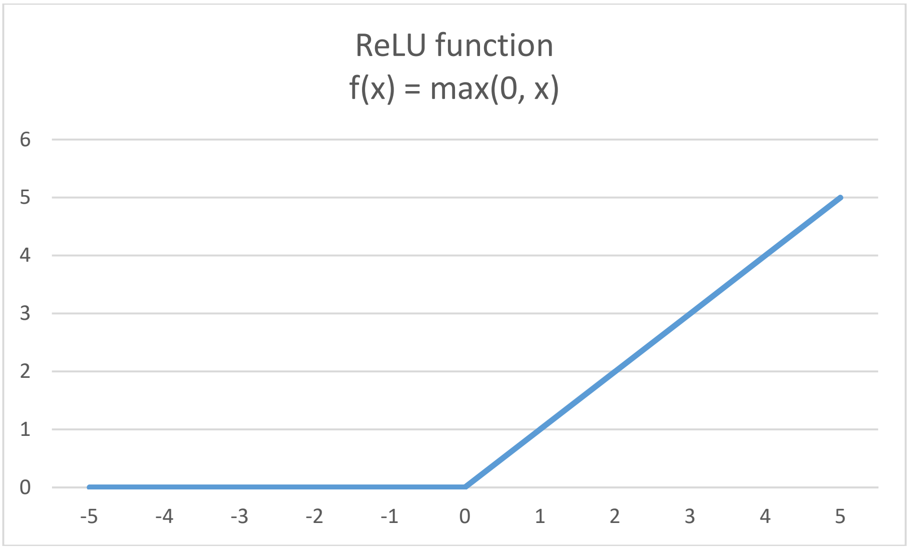


Fig 5.17 ReLU function

- A network topology that determines the complexity of the network can be identified by the following three characteristics:
    - Number of layers
    - Number of nodes within each layer of the network
    - How information travels in the network (forward / backward)
- Training algorithm that specifies how the connection weights are set in order to inhibit or excite neurons in proportion to the input signal.

Figure 5.18 shows the architecture of a typical multi-output feedforward network in which input signal is fed continuously in one direction from connection to connection till it reaches the output layer.

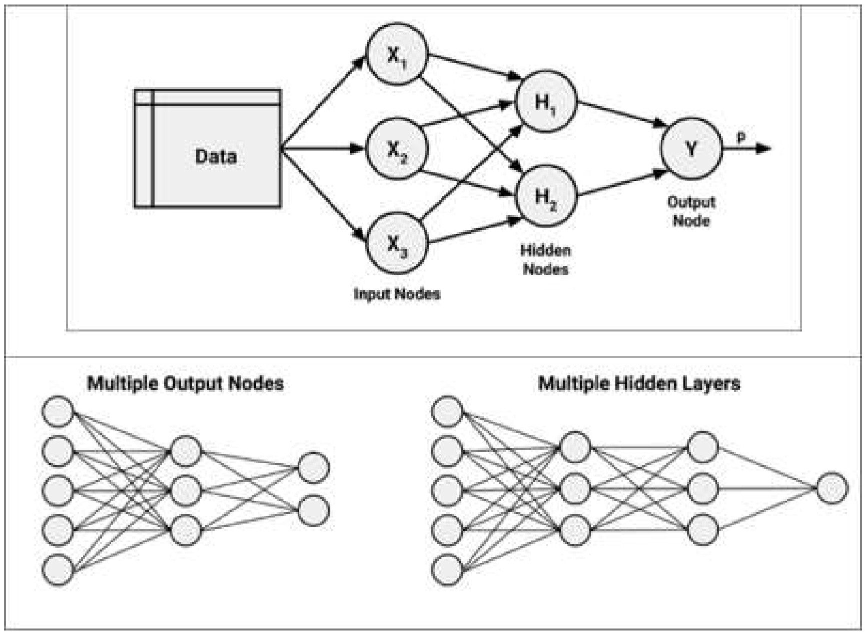
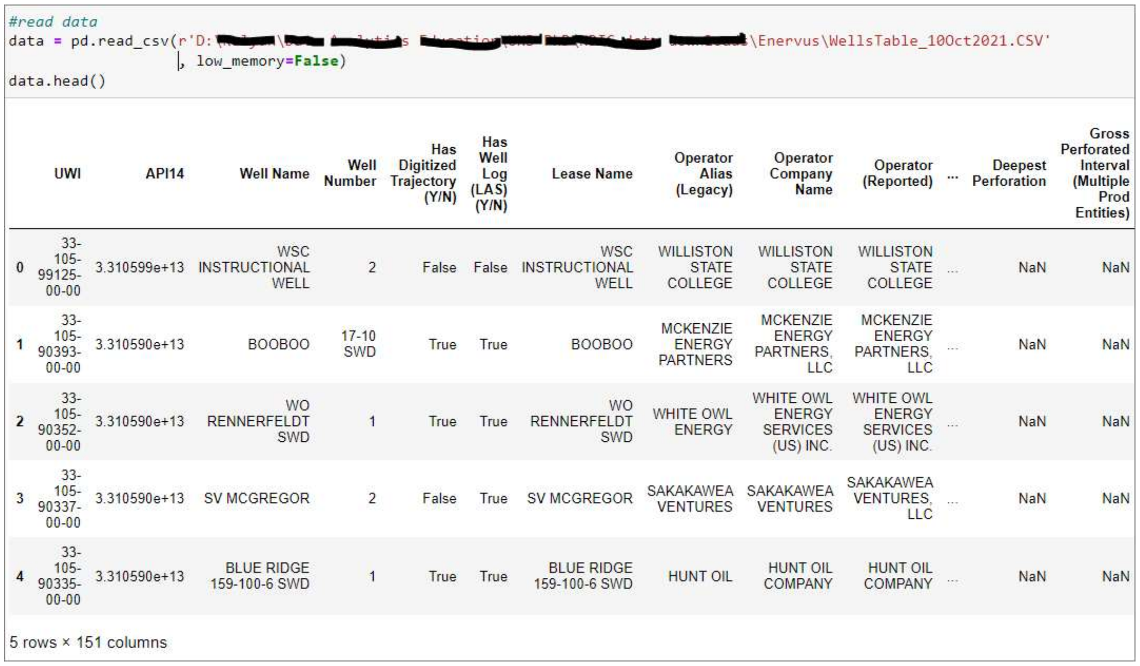


Fig 5.18 Neural network architecture (source: Lantz, 2015)

While the input nodes process the actual data as it is received, the connection between the input nodes to hidden layers and output node have connection weights. Backpropagation is the strategy used to train a neural network by adjusting the connection weights to optimize the model performance. Backpropagation uses the derivative of each neuron's activation function to identify the gradient in the direction of each of the incoming weights. The gradient suggests how steeply the error will be reduced or increased for a change in the weight. The algorithm will attempt to change the weights that result in the greatest reduction in error by an amount known as the learning rate. The Adam optimization algorithm is a commonly used optimization algorithm. Number of layers, number of nodes and learning rate are the model tuning parameters. Including multiple hidden layers in the network for model training is referred to as deep learning (Lantz, 2015).

5.2.2 Model and Results

In order to leverage the multi-target output capabilities of neural network in the Keras deep learning library, the model training was done using Python and code was written and executed using Jupyter Notebook IDE. After loading the required Python libraries such as Pandas, Numpy, Scikit-learn and Keras, the input data set was read using Pandas read_csv method. Data was filtered out to extract 9463 oil and gas wells drilled in the McKenzie county targeting the Bakken play and select the required features. All the target variables (cumulative water production after 1-month, 6-months, 12-months and 24-months) were transformed using a

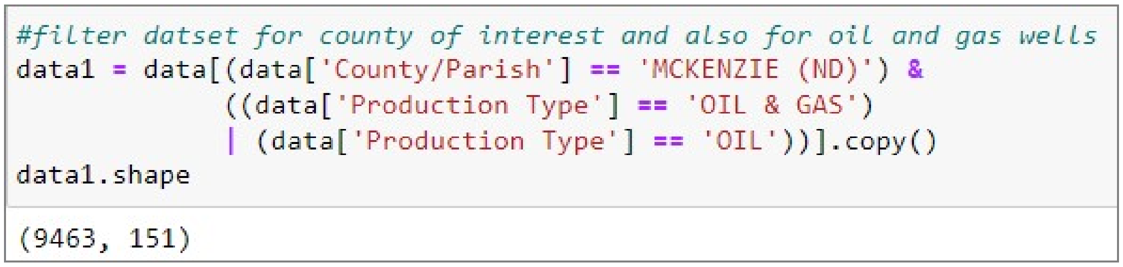


```
#read data
data = pd.read_csv(r'D:[redacted]\Enervus\WellsTable_10Oct2021.CSV'
                  , low_memory=False)
data.head()
```

	UWI	API14	Well Name	Well Number	Has Digitized Trajectory (Y/N)	Has Well Log (LAS) (Y/N)	Lease Name	Operator Alias (Legacy)	Operator Company Name	Operator (Reported)	...	Deepest Perforation	Gross Perforated Interval (Multiple Prod Entities)
0	33-105-99125-00-00	3.310599e+13	WSC INSTRUCTIONAL WELL	2	False	False	WSC INSTRUCTIONAL WELL	WILLISTON STATE COLLEGE	WILLISTON STATE COLLEGE	WILLISTON STATE COLLEGE	...	NaN	NaN
1	33-105-90393-00-00	3.310590e+13	BOOBOO	17-10 SWD	True	True	BOOBOO	MCKENZIE ENERGY PARTNERS	MCKENZIE ENERGY PARTNERS, LLC	MCKENZIE ENERGY PARTNERS, LLC	...	NaN	NaN
2	33-105-90352-00-00	3.310590e+13	WO RENNERFELDT SWD	1	True	True	WO RENNERFELDT SWD	WHITE OWL ENERGY	WHITE OWL ENERGY SERVICES (US) INC.	WHITE OWL ENERGY SERVICES (US) INC.	...	NaN	NaN
3	33-105-90337-00-00	3.310590e+13	SV MCGREGOR	2	False	True	SV MCGREGOR	SAKAKAWEA VENTURES	SAKAKAWEA VENTURES	SAKAKAWEA VENTURES, LLC	...	NaN	NaN
4	33-105-90335-00-00	3.310590e+13	BLUE RIDGE 159-100-6 SWD	1	True	True	BLUE RIDGE 159-100-6 SWD	HUNT OIL	HUNT OIL COMPANY	HUNT OIL COMPANY	...	NaN	NaN

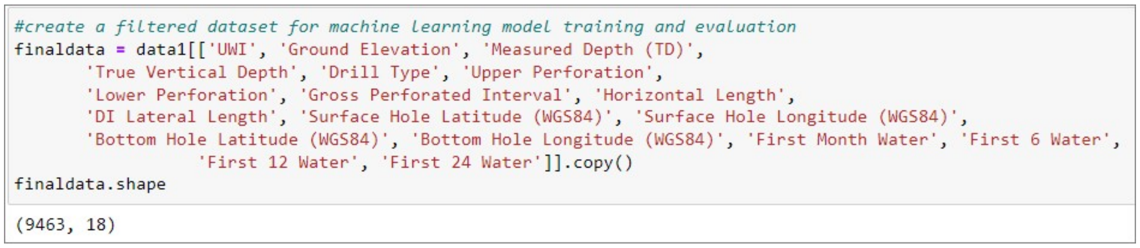
5 rows × 151 columns

power transformation. Null values were removed from the four target variables. Code snippets below were used to achieve the above tasks.



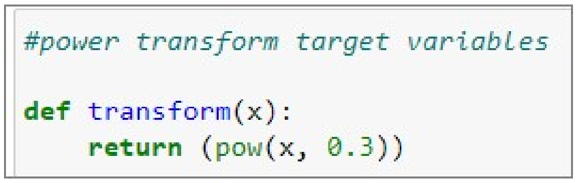
```
#filter datset for county of interest and also for oil and gas wells
data1 = data[(data['County/Parish'] == 'MCKENZIE (ND)') &
              ((data['Production Type'] == 'OIL & GAS')
              | (data['Production Type'] == 'OIL'))].copy()
data1.shape
```

```
(9463, 151)
```

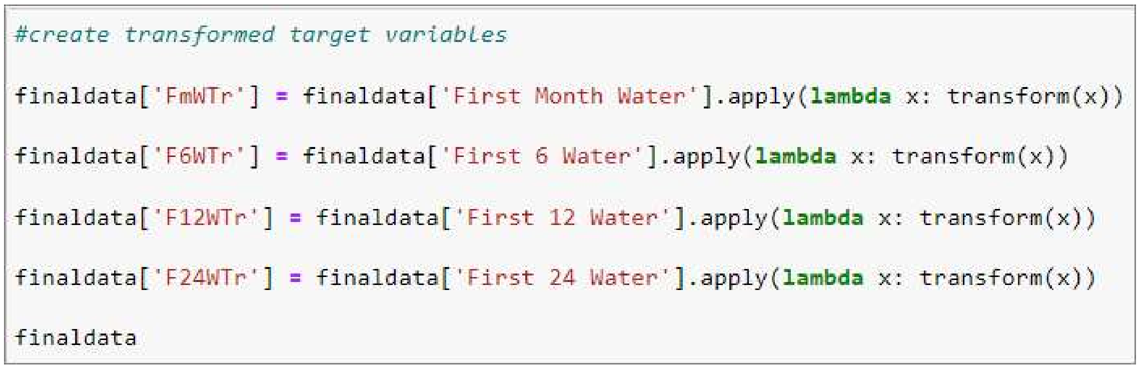


```
#create a filtered dataset for machine learning model training and evaluation
finaldata = data1[['UWI', 'Ground Elevation', 'Measured Depth (TD)',
       'True Vertical Depth', 'Drill Type', 'Upper Perforation',
       'Lower Perforation', 'Gross Perforated Interval', 'Horizontal Length',
       'DI Lateral Length', 'Surface Hole Latitude (WGS84)', 'Surface Hole Longitude (WGS84)',
       'Bottom Hole Latitude (WGS84)', 'Bottom Hole Longitude (WGS84)', 'First Month Water', 'First 6 Water',
               'First 12 Water', 'First 24 Water']].copy()
finaldata.shape
```

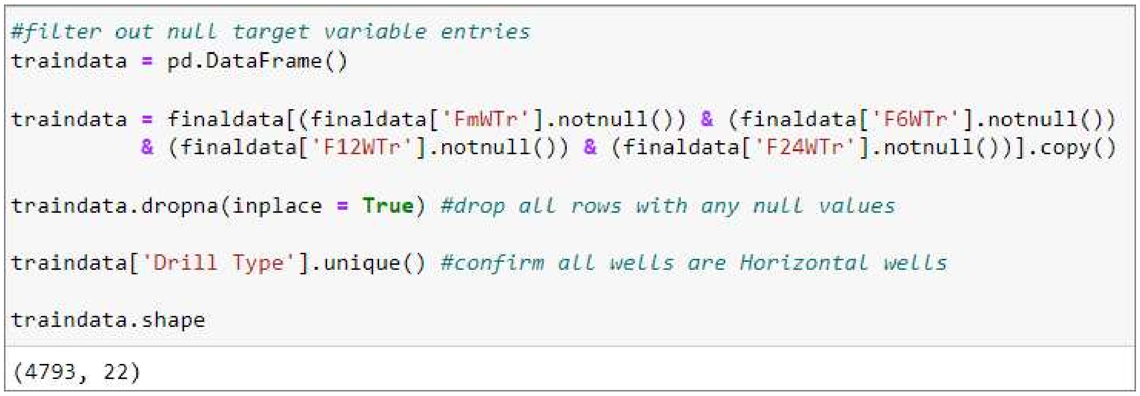
```
(9463, 18)
```



```
#power transform target variables

def transform(x):
    return (pow(x, 0.3))
```



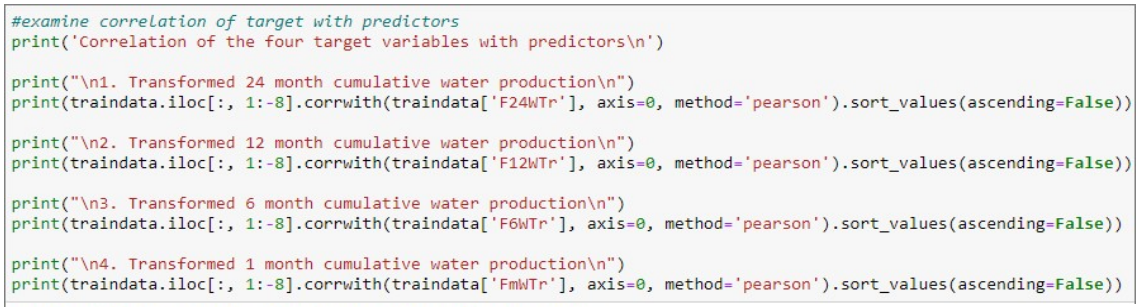
```
#create transformed target variables

finaldata['FmWTr'] = finaldata['First Month Water'].apply(lambda x: transform(x))

finaldata['F6WTr'] = finaldata['First 6 Water'].apply(lambda x: transform(x))

finaldata['F12WTr'] = finaldata['First 12 Water'].apply(lambda x: transform(x))

finaldata['F24WTr'] = finaldata['First 24 Water'].apply(lambda x: transform(x))

finaldata
```



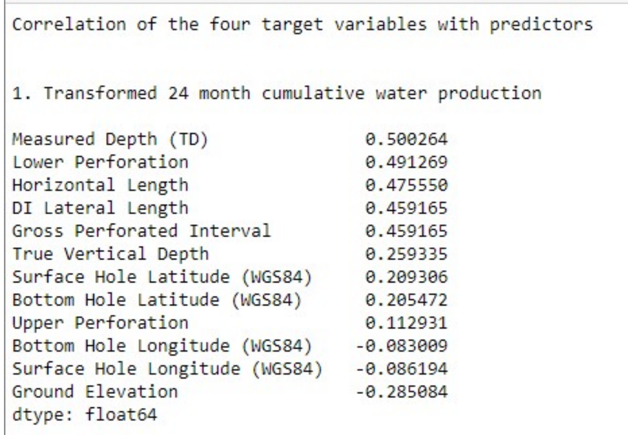
```
#filter out null target variable entries
traindata = pd.DataFrame()

traindata = finaldata[(finaldata['FmWTr'].notnull()) & (finaldata['F6WTr'].notnull())
        & (finaldata['F12WTr'].notnull()) & (finaldata['F24WTr'].notnull())].copy()

traindata.dropna(inplace = True) #drop all rows with any null values

traindata['Drill Type'].unique() #confirm all wells are Horizontal wells

traindata.shape
```

```
(4793, 22)
```

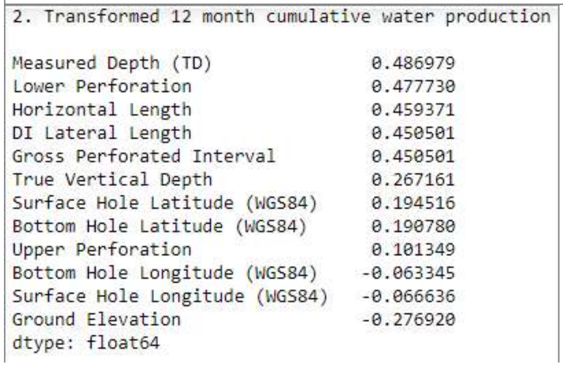
The data set was reduced to 4793 horizontal oil and gas wells in the McKenzie county. Correlation analysis of the four target variables with the thirteen selected predictors were carried out using Pandas 'corrwith' method. Code snippet and results are presented below from which we can identify that features like Measured Depth, Horizontal Length and Gross Perforated Interval have strong positive correlation with the target while Ground Elevation has a moderate negative correlation. Since the weights will be adjusted by the neural network backpropagation algorithm during model training to reduce the effect of collinearity, no further feature reduction was done. The correlation strength is the highest for the 24-months cumulative water production and the lowest for the 1-month cumulative water production.



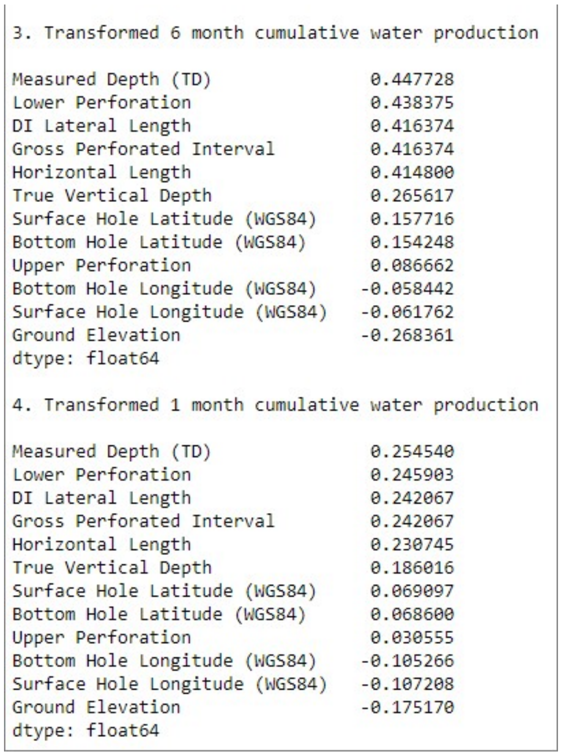
```
#examine correlation of target with predictors
print('Correlation of the four target variables with predictors\n')

print("\n1. Transformed 24 month cumulative water production\n")
print(traindata.iloc[:, 1:-8].corrwith(traindata['F24WTr'], axis=0, method='pearson').sort_values(ascending=False))

print("\n2. Transformed 12 month cumulative water production\n")
print(traindata.iloc[:, 1:-8].corrwith(traindata['F12WTr'], axis=0, method='pearson').sort_values(ascending=False))

print("\n3. Transformed 6 month cumulative water production\n")
print(traindata.iloc[:, 1:-8].corrwith(traindata['F6WTr'], axis=0, method='pearson').sort_values(ascending=False))

print("\n4. Transformed 1 month cumulative water production\n")
print(traindata.iloc[:, 1:-8].corrwith(traindata['FmWTr'], axis=0, method='pearson').sort_values(ascending=False))
```



```
Correlation of the four target variables with predictors


1. Transformed 24 month cumulative water production

Measured Depth (TD)              0.500264
Lower Perforation                0.491269
Horizontal Length                0.475550
DI Lateral Length                0.459165
Gross Perforated Interval        0.459165
True Vertical Depth              0.259335
Surface Hole Latitude (WGS84)    0.209306
Bottom Hole Latitude (WGS84)     0.205472
Upper Perforation                0.112931
Bottom Hole Longitude (WGS84)   -0.083009
Surface Hole Longitude (WGS84)  -0.086194
Ground Elevation                -0.285084
dtype: float64
```

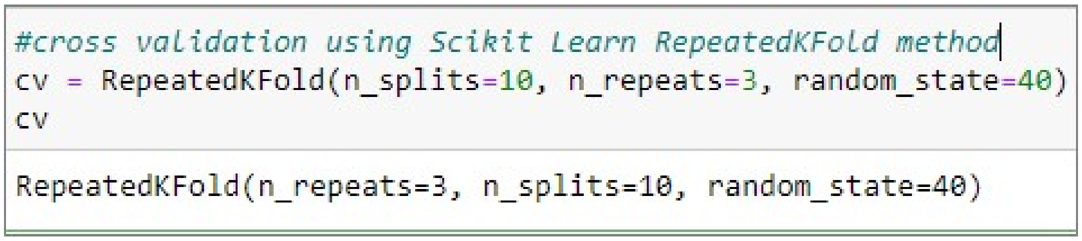


```
2. Transformed 12 month cumulative water production

Measured Depth (TD)              0.486979
Lower Perforation                0.477730
Horizontal Length                0.459371
DI Lateral Length                0.450501
Gross Perforated Interval        0.450501
True Vertical Depth              0.267161
Surface Hole Latitude (WGS84)    0.194516
Bottom Hole Latitude (WGS84)     0.190780
Upper Perforation                0.101349
Bottom Hole Longitude (WGS84)   -0.063345
Surface Hole Longitude (WGS84)  -0.066636
Ground Elevation                -0.276920
dtype: float64
```

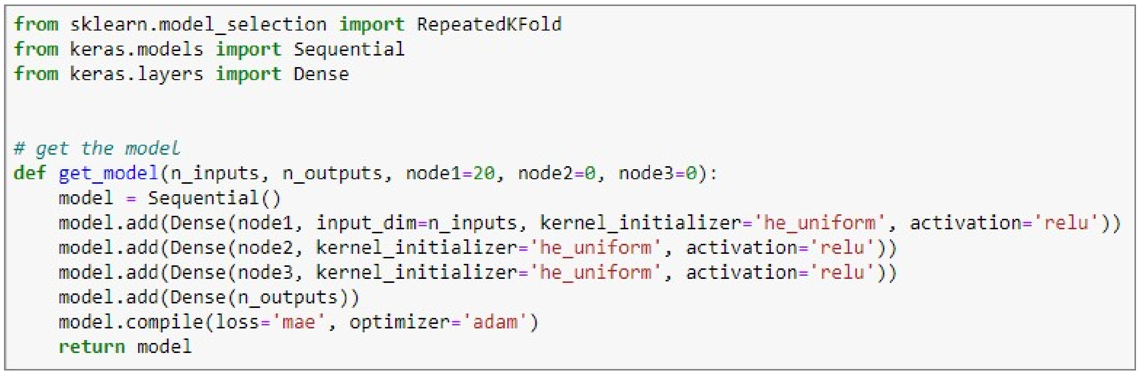


```
3. Transformed 6 month cumulative water production

Measured Depth (TD)              0.447728
Lower Perforation                0.438375
DI Lateral Length                0.416374
Gross Perforated Interval        0.416374
Horizontal Length                0.414800
True Vertical Depth              0.265617
Surface Hole Latitude (WGS84)    0.157716
Bottom Hole Latitude (WGS84)     0.154248
Upper Perforation                0.086662
Bottom Hole Longitude (WGS84)   -0.058442
Surface Hole Longitude (WGS84)  -0.061762
Ground Elevation                -0.268361
dtype: float64

4. Transformed 1 month cumulative water production

Measured Depth (TD)              0.254540
Lower Perforation                0.245903
DI Lateral Length                0.242067
Gross Perforated Interval        0.242067
Horizontal Length                0.230745
True Vertical Depth              0.186016
Surface Hole Latitude (WGS84)    0.069097
Bottom Hole Latitude (WGS84)     0.068600
Upper Perforation                0.030555
Bottom Hole Longitude (WGS84)   -0.105266
Surface Hole Longitude (WGS84)  -0.107208
Ground Elevation                -0.175170
dtype: float64
```

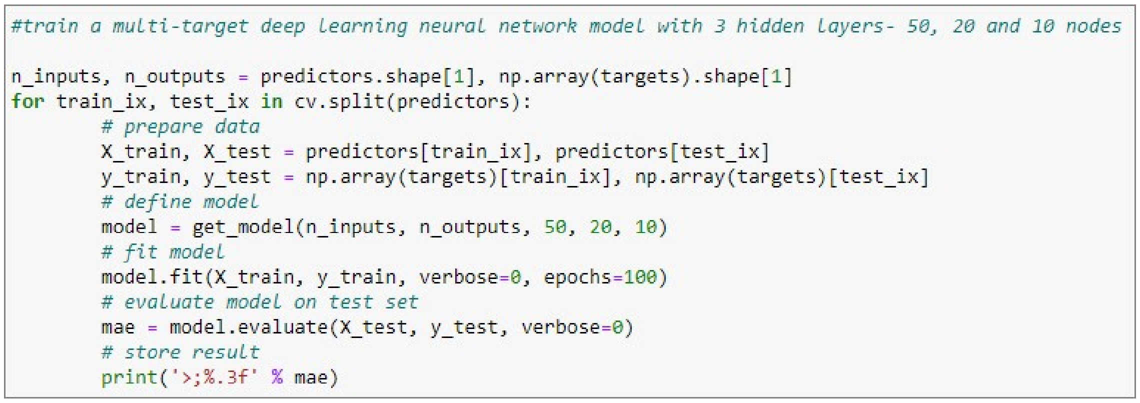
All the predictors were scaled using z-score normalization after which a multi-target neural network model was trained using 10-fold cross validation with 3 repeats using the code shown in the snippets below.



```
#cross validation using Scikit Learn RepeatedKFold method
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=40)
cv

RepeatedKFold(n_repeats=3, n_splits=10, random_state=40)
```



```
from sklearn.model_selection import RepeatedKFold
from keras.models import Sequential
from keras.layers import Dense


# get the model
def get_model(n_inputs, n_outputs, node1=20, node2=0, node3=0):
    model = Sequential()
    model.add(Dense(node1, input_dim=n_inputs, kernel_initializer='he_uniform', activation='relu'))
    model.add(Dense(node2, kernel_initializer='he_uniform', activation='relu'))
    model.add(Dense(node3, kernel_initializer='he_uniform', activation='relu'))
    model.add(Dense(n_outputs))
    model.compile(loss='mae', optimizer='adam')
    return model
```



```
#train a multi-target deep learning neural network model with 3 hidden layers- 50, 20 and 10 nodes

n_inputs, n_outputs = predictors.shape[1], np.array(targets).shape[1]
for train_ix, test_ix in cv.split(predictors):
        # prepare data
        X_train, X_test = predictors[train_ix], predictors[test_ix]
        y_train, y_test = np.array(targets)[train_ix], np.array(targets)[test_ix]
        # define model
        model = get_model(n_inputs, n_outputs, 50, 20, 10)
        # fit model
        model.fit(X_train, y_train, verbose=0, epochs=100)
        # evaluate model on test set
        mae = model.evaluate(X_test, y_test, verbose=0)
        # store result
        print('>;%.3f' % mae)
```

Model was optimized for mean absolute error (mae) using the Adam optimizer. ReLU activation function was used as the activation function for all the layers. Results of the cross validation profile is shown in Figure 5.19. The model.predict() method can be used to predict the four target variables for a new oil and gas well in McKenzie county to get water production estimates. These values must be re-transformed using inverse power transform.

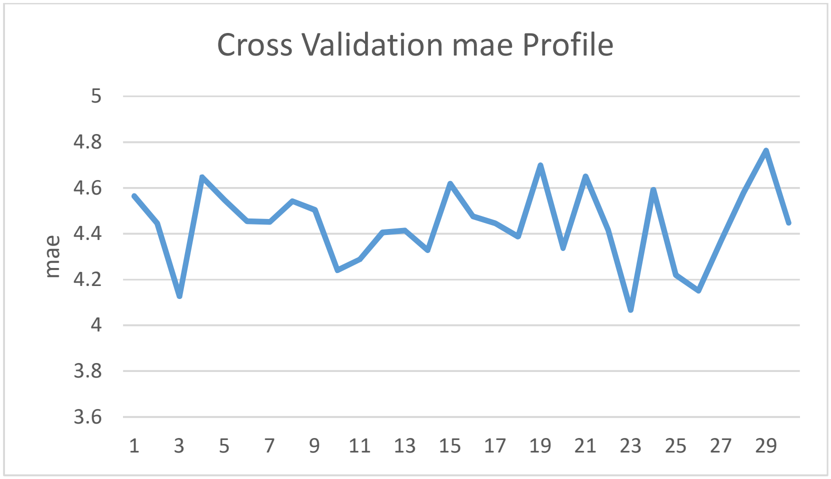


Fig 5.19 Cross validation mae profile for the multi-target neural network model

5.3 Summary

In this chapter, we first presented the concepts behind three machine learning algorithms- kNN, Linear Regression and Ensemble based Random Forest used to train models to predict performance of SWD wells. We then presented the results from feature selection and preparation with specific focus on preprocessing requirements for three machine learning algorithms used.

Results from the base linear regression model was presented which confirmed the moderate relationship between predictors and the target variable with an R^2 value of 32% and encouraged us to explore the potential for an improved model.

Results from the model training and tuning of the 3 algorithms were presented along with the conclusion that the Random Forest algorithm outperformed the other models. The final model showed a 17% improvement over the base model. Orange workflow to use this model for predicting SWD well performance was also presented. Other algorithms such as AdaBoost and Support Vector Machines were also considered but results were not presented as the model performance was at best on par with the Random Forest model.

The chapter concluded with the presentation of the results from deep neural network based multi-target regression model to predict water production from new oil and gas wells in the McKenzie county.

Chapter 6

Conclusions and Recommendations

The first section of this chapter lists the main conclusions made from this work and the second section presents some recommendations for future work to improve the models and expand the scope of the study.

6.1 Conclusions

We can draw the following conclusions from the results of this project:

- The efforts to develop a data-driven proxy model to predict performance of SWD wells and optimize locations of newly planned SWD wells in the state of ND, leveraging data from NDICOG and other secondary sources has yielded an acceptable model. This provides a viable alternative to traditional geological and simulation models.
- While Python is a powerful programming language with many state-of-the-art open source libraries for data analysis, model training and deployment, the Orange visual programming platform enables code-free data preparation, model training and deployment and allows domain experts and end users with little to no programming skills carry out data mining, model training and deployment to derive business benefits.
- Exploratory data analysis was an important step in the data mining process to understand the univariate distributions of the features and the relationship among variables and to gain good insights about the data set, which helped in identifying the most relevant features and preparing the data for unsupervised and supervised machine learning modeling tasks.

- Transformation of both predictors and target variable to remove skewness helped in improving model performance and producing valid regression and cluster models.
- Knowledge graphs, the foundation of Graph Data Science revealed quick insights about the relationships between key features such as SWD well performance, county location and proximity to roads through effective visualizations.
- Graph-based clustering techniques such as Louvain clustering and Spectral clustering performed better than k-Means clustering and revealed three clusters that matched reasonably well with the good, average and poor performing wells based on the SWD well's quality indicator.
- The machine learning model trained using Random Forest ensemble-based algorithm to predict SWD well performance outperformed other models trained using base algorithms such as kNN and Lasso regression.
- Predicting water production from oil and gas wells can help plan downstream operations such as treatment, disposal or reuse of produced water. A multi-target regression model using Keras deep learning framework produced an acceptable regression model to estimate water production from new oil and gas wells in the McKenzie county of North Dakota.

6.2 Recommendations

Data is the most important aspect of data mining to build robust data-driven models.

- Of the 905 SWD wells in the NDICOG, only 651 wells (72%) were available for data mining. Increasing the sample size to include more SWD wells can aid further improvement in model performance. One option to increase sample size is to create synthetic wells by accessing one of the simulation models discussed in Section 2.2 of Chapter 2. Another option is to expand the scope to include other states such as Texas, New Mexico, Colorado and Wyoming with high oil and gas activities where SWD wells are common.

- Increasing the feature space to include additional subsurface data from well logs, geological models and seismic data such as density, porosity, permeability and features pertaining geological structure such as formation dip etc., can also aid in improving model performance. Well log information may be limited from SWD wells and hence such information can potentially be extracted from nearby oil and gas wells.
- Considering feature augmentation through generating graph-based features such as node degree, page rank etc., from the network models presented in Sections 4.3 and 4.4 of Chapter 4 and retraining regression models presented in Section 5.1 of Chapter 5 can result in possible improvement in model performance.
- Training classification models to predict well quality indicator and evaluating model performance could provide an alternative approach to regression models for understanding performance of future SWD wells.
- Since the horizontal oil and gas wells from McKenzie county considered for training neural network based multi-target regression are hydraulically fractured, considering additional features pertaining to hydraulic fracturing such as proppant and fluid volumes, stage count and stage length along with average petrophysical properties such as porosity, water saturation and thickness of the oil and gas reservoir can aid in improving the performance of the multi-target regression models. The scope can be further expanded to include cumulative oil and gas production targets and to develop a full-fledged data-driven proxy model for Bakken play in the McKenzie county.

References

4earthintelligence. (2021, May 10). What is geospatial data? Retrieved from 4EI: https://www.4earthintelligence.com/insights/what-is-geospatial-data/

ArcGIS Pro. (n.d.). Spatial Join. Retrieved Nov 01, 2020, from ArcGIS Pro: https://pro.arcgis.com/en/pro-app/2.8/tool-reference/analysis/spatial-join.htm

Bader, J. W. (2016, January 01). The Dakota Group of the Williston Basin- An Important Geologic Unit for Produced Water from Oil and Gas Development in North Dakota. GEO NEWS, 43(1), pp. 11-15. Retrieved from https://www.dmr.nd.gov/ndgs/newsletter/2016Winter.asp

Blondes, M. G. (2018). U.S. Geological Survey National Produced Waters Geochemical Database (ver. 2.3, January 2018): U.S. Geological Survey data release.

Brownlee, J. (2020, August 28). Deep Learning Models for Multi-Output Regression. Retrieved October 10, 2021, from Machine Learning Mastery: https://machinelearningmastery.com/deep-learning-models-for-multi-output-regression/

Chatterjee, M. (2020, Aug 16). Introduction to Spectral Clustering. Retrieved April 28, 2022, from Great Learning: https://www.mygreatlearning.com/blog/introduction-to-spectral-clustering/

Connolly, T. M. (2015). Database Systems (Sixth ed.). New York: Pearson.

Cross, T., Sathaye, K., Darnell, K., Niederhut, D., & Crifasi, K. (2020). Predicting Water Production in the Williston Basin Using a Machine Learning Model. Unconventional Resources Technology Conference (URTeC). Austin, TX, USA: Unconventional Resources Technology Conference (URTeC).

Demsar, J., Curk, T., Erjavec, A., Gorup, C., Hocevar, T., Milutinovic , M., . . . Zupan , B. (2013). Orange: Data Mining Toolbox in Python. Journal of Machine Learning Research, 14, 2349-2353. Retrieved from http://jmlr.org/papers/v14/demsar13a.html

Dilmegani, C. (2022, 04 18). The Ultimate Guide to Synthetic Data: Uses, Benefits & Tools. Retrieved from AI Multiple: https://research.aimultiple.com/synthetic-data/

earthdatascience.org. (2020, September 11). Intro to Coordinate Reference Systems in Python. Retrieved from Earth Data Analytics Online Certificate: https://www.earthdatascience.org/courses/use-data-open-source-python/intro-vector-data-python/spatial-data-vector-shapefiles/intro-to-coordinate-reference-systems-python/

Edwards, D. A. (2012). Reservoir Simulation: Keeping Pace with Oilfield Complexity. Oilfield Review- Winter 2011/2012, pp. 4-15.

Ge Jun et al. (2018). Modeling and Simulation of the Inyan Kara Formation to Estimate Saltwater Disposal Potential: Final Report. Grand Forks, ND: Energy & Environmental Research Center (EERC).

Globus, A. (1994). Principles of Information Display for Visualization Practitioners. NASA Ames Research Center.

Ground Water Protection Council. (2019). Produced Water Report: Regulations, Current Practices and Research Needs. Oklahoma City: Ground Water Protection Council. Retrieved July 01, 2020, from https://www.gwpc.org/about-us/work-groups/produced-water-work-group/

Han, J. e. (2012). Data Preprocessing. In J. e. Han, Data Mining: Concepts and Techniques (pp. 83-120). Elsevier Incorporated.

Hodler, A., & Needham, M. (2021). Graph Data Science. John Wiley & Sons.

Hunger, M. e. (2021). The Definitive Guide to Graph Databases. neo4j.

Juodyte, M. (2017). Overview: Data Mining Pipeline. Technical University of Munich.

Kabir, S. M. (2016). Basic Guidelines for Research. Chittagong: Book Zone Publication.

Karajgikar, J. (2021, December 17). Scraping Open Data from the Web with BeautifulSoup. Retrieved from PennLibraries: https://www.library.upenn.edu/blogs/rdds/scraping-open-data-web-beautifulsoup

Klie, H. (2021, 05 03). A Tale of Two Approaches: Physics-Based vs. Data-Driven Models. Journal of Petroleum Technology. Retrieved from https://jpt.spe.org/a-tale-of-two-approaches-physics-based-vs-data-driven-models

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. New York: Springer.

Kurz et al. (2016, March). A Review of Bakken Water Mangaement Practices and Potential Outlook. Grand Forks: Energy & Environmental Research Center (EERC).

Lantz, B. (2015). Machine Learning with R. Birmingham, United Kingdom: Packt.

Loshin, D. (2011). Bringing It All Together. In D. Loshin, The Practitioner's Guide to Data Quality Improvement (pp. 351-383). MK Series on Business Intelligence. Retrieved from https://www.sciencedirect.com/topics/computer-science/source-data-set

Luxborg, U. V. (2007). A Tutorial on Spectral Clustering. Max Planck Institute for Biological Cybernetics.

McKinney, W. (2017). Data Wrangling. In W. McKinney, Python for Data Analysis (pp. 221-251). O' Reilly Media Incorporation.

Nego, A. (2021). Graph Powered Machine Learning. Manning Publications. Retrieved Feb 8, 2022 neo4j. (n.d.). Louvain. Retrieved April 27, 2022, from Neo4j Graph Data Science: https://neo4j.com/docs/graph-data-science/current/algorithms/louvain/

Pedregosa, F. e. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Python. (2022, Jul 02). General Python FAQ. Retrieved from Python: https://docs.python.org/3/faq/general.html

S. Basu, T. C. (2019). Salt Water Disposal Modeling of Dakota Sand, Williston basion, to Drive Drilling Decisions. Unconventional Resources Technology Conference (URTeC). Denver: URTeC.

Sankaran, S., Matringe, S., Sidahmed, M., Saputelli, L., Wen, X.-H., Popa, A., & Dursun, S. (2020). Data Analytics in Reservoir Engineering. Richardson, Texas, USA: Society of Petroleum Engineers.

Shmueli, G. e. (2018). Data Mining for Business Analytics: Concepts, Techniques, and Applications in R. John Wiley & Sons.

UIC. (Retrieved 02 02, 2022, from Environmental Protection Agency (EPA): https://www.epa.gov/uic

Venugopal, K., Shastri, D., Radhakrishnan, S., & Krishnamoorthy, R. (2021). An Online Microcredential Certification Program to Upskill Petrotechnical Professionals in Data Analytics and Machine Learning with an Upstream Oil and Gas Industry Focus. 2021 SPE Annual Technical Conference and Exhibition . Dubai: 2021 SPE Annual Technical Conference and Exhibition .

Wright, B. (2022, January 01). The Produced Water Conundrum Grows Acorss Unconventionals. Journal of Petroleum Technology (JPT).

Xu, D. e. (2019). A Survey on Multi-Output Learning. arxiv.org. Retrieved from https://arxiv.org/abs/1901.00248v2

Ziesch, M., & Ritter, A. (2018). Oil Fields of North Dakota. Bismark: North Dakota Geological Survey Educational Series 34.