INTERNSHIP REPORT

PROJECT REPORT

ON

Facies Identification from Well Logs usingMachineLearning

SUBMITTED BY: HITESH NAGAR

COURSE/BRANCH: B.E. (IT)

COLLEGE: MBM UNIVERSITY, JODHPUR

Table of Contents

DECLARATION

I, **HITESH NAGAR** pursuing B.E. IT., from **MBM UNIVERSITY JODHPUR,** hereby declare that the work presented in this project report is the outcome of my own work, is Bonafede and correct to the best of my knowledge.

The work presented in the project does not infringe any patents and has not been submitted to any other university or anywhere else for the award of any degree or any professional diploma.

(STUDENT's SIGNATURE)
HITESH NAGAR

Ms. Rini Maheshwari

Sr. Prog. Officer,

GEOPIC Dehradun, ONGC

ACKNOWLEDGEMENT

I would like to extend my gratitude to Ms. Rini Maheshwari ma'am , Sr. Prog. Officer , GEOPIC Dehradun for being my mentor and guiding me throughout my training tenure. I would sincerely like to thank you for all the knowledge and experience you dawned on me.

A special thanks to Oil and Gas Corporation Limited, Dehradun for

considering my request and providing me with the opportunity to work with the maestros of the industry.

Maam , Without your support and cooperation, this project would not have been completed and the manuscript would not have been in the present form.

HITESH NAGAR
3rd Year, B.E. IT
MBM University Jodhpur

OIL AND NATURAL GAS CORPORATION LIMITED

DATE: 31.07.2023

CERTIFICATE

This is to certify that **HITESH NAGAR**, student of **MBM UNIVERSITY** , B.E. IT, 3rd Year, has successfully completed the project on "Facies Identification from Well Logs using Machine Learning" from 01.06.23 to 31.07.23 under the guidance of Ms Rini Maheshwari - Sr. Prog. Officer, Mr PR Meena - GM (Programming) , Head Software , GEOPIC Dehradun, ONGC.

During this period of his internship program with us, he was found to be diligent and hardworking.

We wish him every success in his life and career

Ms. Rini Maheshwari
Sr. Prog. Officer,
GEOPIC Dehradun, ONGC

Mr P R Meena
GM (Programming)
GEOPIC Dehradun, ONGC

ओएनजीसी
ONGC

OIL AND NATURAL GAS CORPORATION LIMITED

DATE: 31.07.2023

CERTIFICATE

This is to certify that **HITESH NAGAR**, student of **MBM UNIVERSITY** , B.E. IT, 3rd Year, has successfully completed the project on 'Facies Identification from Well Logs using Machine Learning" from 01.06.23 to 31.07.23 under the guidance of Ms Rini Maheshwari - Sr. Prog. Officer, Mr PR Meena - GM (Programming) , Head Software , GEOPIC Dehradun, ONGC.

During this period of his internship program with us, he was found to be diligent and hardworking.

We wish him every success in his life and career

Ms. Rini Maheshwari
Sr. Prog. Officer,
GEOPIC Dehradun, ONGC

31.07.2023

Mr P R Meena
GM (Programming)
GEOPIC Dehradun, ONGC

पी.आर.मीणा / P. R. Meena
महाप्रबंधक (प्रोग्रामिंग) / GM (Prog.)
प्रमुख-सी.एस. सॉफ्टवेयर विभाग
Head - CS (S/W) Division
जियोपिक,ओएनजीसी,देहरादून
GEOPIC, ONGC, Dehradun

Oil And Gas Company's Profile

Introduction to the Industry

The Oil and Natural Gas Corporation (ONGC) is an Indian oil and gas explorer and producer, headquartered in New Delhi. ONGC was founded on 14 August 1956 by the Government of India. It is a public sector undertaking whose operations are overseen by the Ministry of Petroleum and Natural Gas.

It is the largest government-owned-oil and gas exploration and production corporation in the country, and produces around 70% of India's crude oil (equivalent to around 57% of the country's total demand) and around 84% of its natural gas.

In November 2010, the Government of India conferred the Maharatna status to ONGC Oil and Natural Gas Corporation Limited (ONGC) is engaged in the business of exploration and drilling of crude oil and natural gas and is the world's second biggest exploration and production company ONGC owns and operates more than 11000 KM of pipelines in India, including nearly 3200 KM of subsea pipelines. The company contributes more than 78% of India's oil and gas production.

Today, ONGC is the flagship company of India; and making this possible is a dedicated team of nearly 40,000 professionals who toil round the clock. It is this toil which amply reflects in the performance figures and aspirations of ONGC. The company has adapted progressive policies in scientific planning, acquisition, utilization, training and motivation of the team.

Oil And Gas Company's Profile

History Of The Company

The India Petroleum Industry is a case in point for exhibiting the giant leaps India has taken after its independence towards its march to attain a self-reliant economy. During the Independence era of 1947, the India Petroleum Industry was controlled by foreign companies and India's own expertise in this sector was limited.

Now, after 60 years, the India Petroleum Industry has become an important public sector undertaking with numerous skilled personnel and updated technology that is comparable to the best in the world. The vim and the achievement during these years is the growth of productivity in petroleum and petroleum-based products. Even the consumption has multiplied itself nearly 30 times in the post- independence era.

The ONGC originally set up as a Directorate in 1955, was transformed into a Commission in 1956. In 1958, the Indian Refineries Ltd., a government undertaking, came into existence. The Indian Oil Company (IOC), also a government undertaking, was set up in 1959 with the purpose of marketing petroleum-related products.

Indian Oil Corporation Ltd. was formed in 1964 with the merger of the Indian Refineries Ltd. and the Indian Oil Company Ltd. In 2003, ONGC Videsh Limited (OVL), the division of ONGC concerned with its foreign assets, acquired Talisman Energy's 25% stake in the Greater Nile Oil project.

In 2006, a commemorative coin set was issued to mark the 50th anniversary of the founding of ONGC, making it only the second Indian company (State Bank of India being the first) to have such a coin issued in its honor.

In 2011, ONGC applied to purchase 2000 acres of land at Dahanu to process offshore gas. ONGC Videsh, along with Statoil ASA (Norway) and Repsol SA (Spain), has been engaged in deep-water drilling off the northern coast of Cuba in 2012. On 11 August 2012, ONGC announced that it had made a large oil discovery in the D1 oilfield off the west coast of India, which will help it to raise the output of the field from around 12,500 barrels per day (bpd) to a peak output of 60,000 bpd.

In November 2012, OVL agreed to acquire ConocoPhillips' 8.4% stake in the Kashagan oilfield in Kazakhstan for around US$5 billion, in ONGC's largest acquisition to date. The acquisition is subject to the approval of the governments of Kazakhstan and India and also to other partners in the Caspian Sea field waiving their pre-emption rights.

In January 2014, OVL and Oil India completed the acquisition of Videocon Group's ten percent stake in a Mozambican gas field for a total of $2.47 billion.

In June 2015, Oil and Natural Gas Corporation (ONGC) gave a ₹27bn ($427m) offshore contract for the Bassein development project to Larsen & Toubro (L&T).

In February 2016, the board of ONGC approved an investment of ₹5,050 crore in Tripura for drilling of wells and creation of surface facilities to produce 5.1 million standard cubic feet per day gas from the state's fields.

Oil And Gas Company's Profile

Strengths, Opportunities and Competitors

Strengths

-ONGC is India's largest crude oil and natural gas producer Strong brand name of ONGC company High profit making and high revenues has over 30,000 employees in its workforce.

-ONGC produces about 30% of India's crude oil requirement

-Contributes 70%+ of India's crude oil production and 80%+ of India's natural gas production

-Commemorative Coin set was released to mark 50 Years of ONGC 8. -- -Strong advertising and branding of the company along with recognition from several awards

-Owned by the Govt of India, ONGC has got a strong financial backing.

Opportunities

- Increasing fuel/oil prices means higher margins for ONGC

-Increasing natural gas market

-ONGC can increase business by more oil well discoveries

- Expand global export market and have international tie-ups

Competitors

- BharatPetroleum

-IOCL

- RelianceIndustriesLimited

Relevance of this Project In ONGC

The prestigious organization Well Logging Services –O.N.G.C. provides a wide range of service and information, allowing their ASSET to define, reduce and manage their risk operations.

The evaluation of these complex reservoirs via my project involves calibration of mineralogy from core and advanced elemental analysis.

This will add to the companies existing data sets and provide them information about these compounds.

About Well Log

Well log is one of the most fundamental methods for reservoir characterization, in the oil and gas industry, it is an essential method for geoscientists to acquire more knowledge about the condition below the surface by using physical properties of rocks. This method is very useful to detect hydrocarbon bearing zones, calculate the hydrocarbon volume, and many others. Some approaches are needed to characterize reservoirs.

The interpretation of well log data must be done in several steps and it is not recommended for the user to analyze them randomly because the result might be a total error. Figure 1 shows the steps for reservoir characterization by using well log data. Basically, there are two types of properties that will be used in reservoir characterization, they are petrophysics (shale volume, water saturation, permeability, etc.) which are more geology-like and rock physics (elasticity, wave velocity, etc.) which are more geophysics-like.

Every property is related to each other, the relation between each property is shown in figure 2, the author called it the "fish diagram". There are many techniques to find a hydrocarbon bearing zone, the user may use RHOB-NPHI crossover (with some corrections), reflectivity coefficient (just like in seismic interpretation), AI anomaly, etc.

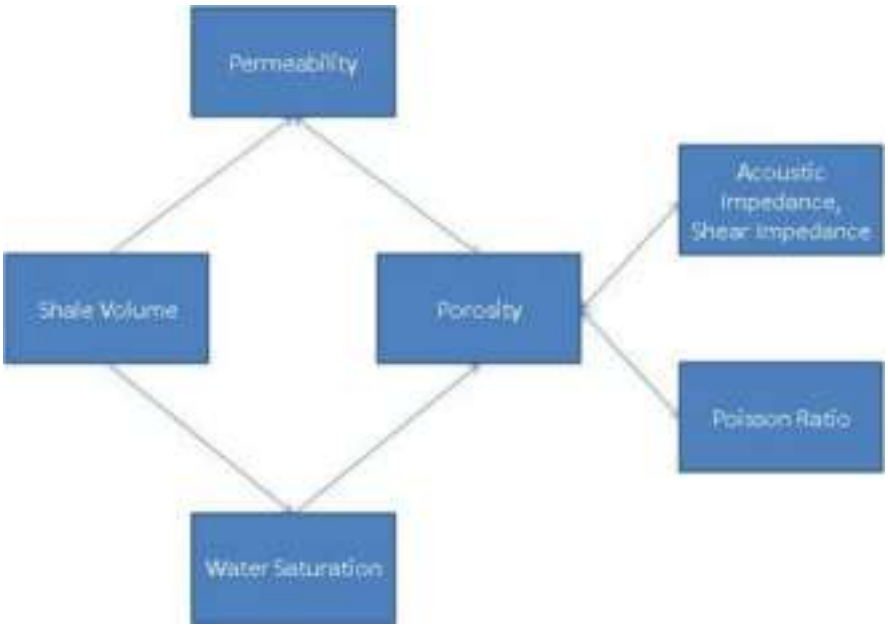
Every method has its own weaknesses, so it is a wise decision to use every method to acquire the right result. There are so many kinds of modern logs, see table 1 for the information about the logs and also their uses.

Table 1 The functions of every log in petrophysical and rock physics properties calculation and analysis.

Name	Uses
Gamma Ray (GR)	Lithology interpretation, shale volume calculation, calculate clay volume, permeability calculation, porosity calculation, wave velocity calculation, etc.
Spontaneous Potential (SP)	Lithology interpretation, Rw and Rwe calculation, detect permeable zone, etc.
Caliper (CALI)	Detect permeable zone, locate a bad hole
Shallow Resistivity (LLS and ILD)	Lithology interpretation, finding hydrocarbon bearing zone, calculate water saturation, etc.
Deep Resistivity (LLD and ILD)	Lithology interpretation, finding hydrocarbon bearing zone, calculate water saturation, etc.

Density (RHOBH)	Lithology interpretation, finding hydrocarbon bearing zone, porosity calculation, rock physics properties (AI, SI, σ, etc.) calculation, etc.
Neutron Porosity (NPHI)	Finding hydrocarbon bearing zone, porosity calculation, etc.
Sonic (DT)	Porosity calculation, wave velocity calculation, rock physics properties (AI, SI, σ, etc.) calculation, etc.
Photoelectric (PEF)	Mineral determination (for lithology interpretation) *not used in this article

.



Well logging technology has developed over many decades. The improvement in technology from one generation of well logs to another has been remarkable. This is both an asset and a liability. It is an asset for new fields where the most modern technology can be applied. It is a liability for old fields where old logs must be combined with new logs in the analysis of a field. Care must be taken when working with logs from different generations of technology to be sure that analytical techniques are appropriate for each well log.

Well logs present a concise, detailed plot of formation parameters versus depth. From these plots, interpreters can identify lithologies, differentiate between porous and nonporous rock and quickly recognize pay zones in subsurface formations. The ability to interpret a log lies in recognizing the significance of each measurement.

Logging tools record the magnitude of a specific formation property, such as resistivity, measured as the tool traverses an interval defined by depth; a well log is a chart that shows the value of that measurement plotted versus depth.

During the early days of commercial well logging, logs were primarily used qualitatively for making formation correlations; within a given field or local geological province, certain formations have distinctive characteristics that appear remarkably similar from one well to the next, providing geologists with a basis for locating the depths of various strata in the subsurface. In 1942, the relationship between resistivity, porosity and water saturation (and thus its inverse: hydro-carbon saturation) was established by G.E. Archie, paving the way for a quantitative evaluation of formation properties using well logs.

Technologies Used

PYTHON:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

JUPYTER NOTEBOOK:

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more

Machine learning algorithms

Logistic Regression

Logistic Regression is a popular statistical method used for binary classification problems, where the goal is to predict one of two possible outcomes (e.g., yes or no, true or false, 0 or 1) based on input features. Despite its name, logistic regression is a classification algorithm, not a regression algorithm.

Here are the key aspects of Logistic Regression:
1. Sigmoid Function: Logistic Regression uses the logistic function (also known as the sigmoid function) to model the relationship between the input features and the probability of the binary outcome. The sigmoid function maps any real-valued number to the range of 0 to 1, making it suitable for representing probabilities.
2. Hypothesis Function: The hypothesis function in logistic regression is defined as the sigmoid function applied to a linear combination of the input features and their corresponding weights. The weights are learned during the training process.
3. Cost Function: The model's performance is measured using a cost function (also known as the loss function), typically the cross-entropy loss, which penalizes the model for incorrect predictions. The goal during training is to minimize the cost function.
4. Training: Logistic Regression is trained using optimization algorithms such as gradient descent. The algorithm adjusts the model's weights iteratively to find the optimal values that minimize the cost function.
5. Decision Boundary: In logistic regression, a decision boundary is the boundary that separates the two classes in the feature space. It is determined by the learned model parameters.
6. Regularization: To prevent overfitting, regularization techniques like L1 (Lasso) and L2 (Ridge) regularization can be applied to penalize large weights.

Logistic Regression is widely used in various applications, including email spam detection, sentiment analysis, medical diagnosis, credit risk assessment, and many other binary classification tasks. It is a simple yet effective algorithm that can serve as a good starting point for classification problems, especially when the relationship between features and the target variable is approximately linear. However, for more complex problems or when dealing with multiple classes, other algorithms like Support Vector Machines, Random Forests, or Neural Networks might be more appropriate.

Random Forest

Random Forest is an ensemble learning method used for both classification and regression tasks in machine learning. It is an extension of decision tree algorithms and is widely known for its high accuracy and robustness. Random Forest works by constructing multiple decision trees during training and then combining their predictions to make more accurate and stable predictions.

Here are the key concepts of Random Forest:
1. Ensemble Learning: Random Forest is an example of ensemble learning, where multiple models (in this case, decision trees) are combined to make a final prediction. The idea is that by combining several weak learners (individual decision trees), the overall performance and generalization of the model can be improved.

2. Bagging (Bootstrap Aggregating): Random Forest uses a technique called bagging to build multiple decision trees from different subsets of the training data. Bagging involves random sampling with replacement, meaning each tree is trained on a different subset of the data, and some samples may appear in multiple subsets while others may not appear at all.
3. Random Feature Selection: In addition to using random subsets of the data, Random Forest also employs random feature selection. Instead of considering all features at each split in a decision tree, only a random subset of features is considered. This further introduces diversity among the trees and reduces the risk of overfitting.
4. Voting or Averaging: Once the individual decision trees are trained, they make predictions independently, and for classification tasks, each tree "votes" for a class, while for regression tasks, the predictions are averaged. The final prediction of the Random Forest is determined by majority voting (classification) or averaging (regression) of the predictions made by individual trees.
5. Advantages: Random Forest offers several advantages, including:
   - High accuracy and robustness against overfitting.
   - Effective for both classification and regression tasks.
   - Can handle a large number of features and high-dimensional data.
   - Provides estimates of feature importance, helping in feature selection.
6. Limitations: While Random Forest is a powerful algorithm, it may not be as interpretable as simple models like decision trees, especially when dealing with a large number of trees. Also, it may require more memory and computational resources compared to single decision tree models.

Random Forest has become a popular choice for various machine learning applications, including image classification, medical diagnosis, finance, and more. It is considered a "black-box" model due to its complexity, but its excellent performance and ability to handle diverse datasets make it a valuable tool in the data scientist's toolkit.

Support Vector Machine

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for both classification and regression tasks. SVM is particularly well-suited for binary classification problems, but it can also be extended to handle multi-class classification tasks. The primary goal of SVM is to find the optimal hyperplane that best separates the data points of different classes.

Here are the key concepts of Support Vector Machine:
1. Hyperplane: In a two-dimensional space, a hyperplane is simply a line that separates the data points of two classes. In higher-dimensional spaces, a hyperplane becomes a flat affine subspace. For binary classification, SVM aims to find the hyperplane that maximizes the margin between the two classes, called the "maximum-margin hyperplane."
2. Support Vectors: The data points that lie closest to the hyperplane are called support vectors. These points are crucial in defining the margin and the optimal hyperplane. Only the support vectors influence the position and orientation of the hyperplane; other data points are irrelevant for the final decision boundary.
3. Margin: The margin is the distance between the support vectors of the two classes and is controlled by the hyperplane. SVM seeks to maximize this margin to improve the classifier's generalization ability, making it less susceptible to overfitting.

4. Soft Margin: In real-world scenarios, data points may not be perfectly separable by a single hyperplane. To handle such cases, SVM introduces a "soft margin" that allows some data points to be misclassified to achieve a better overall separation. The balance between maximizing the margin and allowing misclassifications is controlled by the regularization parameter C.
5. Kernel Trick: SVM can efficiently handle non-linearly separable data by using the kernel trick. The kernel function transforms the original feature space into a higher-dimensional space, where the data points may become linearly separable. Popular kernel functions include the radial basis function (RBF) kernel, polynomial kernel, and sigmoid kernel.
6. Advantages: SVM offers several advantages, including:
- Effective in high-dimensional spaces and with a small number of training samples.
- Robust against overfitting, especially when using a proper regularization parameter C.
- Versatile due to the ability to use different kernel functions for various data types.
- Provides a unique solution since the optimization problem is convex.
7. Limitations: SVM's main limitation lies in its computational complexity, particularly with large datasets. Training a SVM can be time-consuming, especially when using complex kernel functions or when the dataset is imbalanced. Additionally, SVM's interpretability is limited compared to simple models like logistic regression.

Support Vector Machines have proven to be successful in various applications, such as text classification, image recognition, bioinformatics, and more. They are widely used when dealing with both linearly separable and non-linearly separable datasets and are considered a fundamental algorithm in the field of machine learning.

About Artificial Intelligence and Machine Learning

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is
synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce lifelike exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

While artificial emotional intelligence is already a budding industry and an area of interest for leading AI researchers, achieving Theory of mind level of AI will require development in other branches of AI as well. This is because to truly understand human needs, AI machines will have to perceive humans as individuals whose minds can be shaped by multiple factors, essentially “understanding” humans.

This is the final stage of AI development which currently exists only hypothetically. Self aware AI, which, self explanatorily, is an AI that has evolved to be so akin to the human brain that it has developed self-awareness. Creating this type of Ai, which is decades, if not centuries away from materializing, is and will always be the ultimate objective of all AI research.

This type of AI will not only be able to understand and evoke emotions in those it interacts with, but also have emotions, needs, beliefs, and potentially desires of its own. And this is the type of AI that doomsayers of the technology are wary of. Although the development of self awareness can potentially boost our progress as a civilization by leaps and bounds, it can also potentially lead to catastrophe.

This is because once self-aware, the AI would be capable of having ideas like self preservation which may directly or indirectly spell the end for humanity, as such an entity could easily outmaneuver the intellect of any human being and plot elaborate schemes to take over humanity

The alternate system of classification that is more generally used in tech parlance is the classification of the technology into Artificial Narrow Intelligence (ANI), Artificial General Intelligence (AGI), and Artificial Superintelligence (ASI).

Artificial Narrow Intelligence represents all the existing AI, including even the most complicated and capable AI that has ever been created to date. Artificial narrow intelligence refers to AI systems that can only perform a specific task autonomously using human-like capabilities.

These machines can do nothing more than what they are programmed to do, and thus have a very limited or narrow range of competencies. According to the aforementioned system of classification, these systems correspond to all the reactive and limited memory AI. Even the most complex AI that uses machine learning and deep learning to teach itself falls under ANI.

Artificial General Intelligence is the ability of an AI agent to learn, perceive, understand, and function completely like a human being. These systems will be able to independently build multiple competencies and form connections and generalizations across domains, massively cutting down on time needed for training. This will make AI systems just as capable as humans by replicating our multi-functional capabilities.

The development of Artificial Superintelligence will probably mark the pinnacle of AI research, as AGI will become by far the most capable forms of intelligence on earth. ASI, in addition to replicating the multi-faceted intelligence of human beings, will be exceedingly better at everything they do because of overwhelmingly greater memory, faster data processing and analysis, and decision-making capabilities.

The development of AGI and ASI will lead to a scenario most popularly referred to as the singularity. And while the potential of having such powerful machines at our disposal seems appealing, these machines may also threaten our existence or at the very least, our way of life.

At this point, it is hard to picture the state of our world when more advanced types of AI come into being. However, it is clear that there is a long way to get there as the current state of AI development compared to where it is projected to go is still in its rudimentary stage. For those holding a negative outlook for the future of AI, this means that now is a little too soon to be worrying about the singularity, and there's still time to ensure AI safety. And for those who are optimistic about the future of AI, the fact that we've merely scratched the surface of AI development makes the future even more exciting.

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and Predictive maintenance.

Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions. There are four basic approaches:supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. The type of algorithm data scientists choose to use depends on what type of data they want to predict.

Python Libraries Used

PANDAS:

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

Advantages:

-Fast and efficient for manipulating and analyzing data.

-Data from different file objects can be loaded.

-Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data

-Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects

-Data set merging and joining.

-Flexible reshaping and pivoting of data sets

-Provides time-series functionality.

-Powerful group by functionality for performing split-apply-combine operations on data sets

NUMPY:

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It is open-source software.

It contains various features including these important ones:

-A powerful N-dimensional array object

-Sophisticated (broadcasting) functions

-Tools for integrating C/C++ and Fortran code

-Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

MATPLOTLIB:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information.

SKLEARN:

Scikit-learn is an open-source Python library that implements a range of machine learning, pre-processing, cross-validation, and visualization algorithms using a unified interface.

Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.

Advantages:

-Accessible to everybody and reusable in various contexts.

-Built on the top of NumPy, SciPy, and matplotlib.

-Open source, commercially usable – BSD license.

Code and Its Explanation

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.colors as colors
from mpl_toolkits.axes_grid1 import make_axes_locatable
import seaborn as sns
import numpy as np
import time
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler
from sklearn.ensemble import IsolationForest, RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, recall_score, precision_score, classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC, SVC

import warnings
warnings.filterwarnings("ignore")
```

Fig1.

Figure 1 marks the beginning of the code.Importing necessary libraries for data manipulation, visualization, and machine learning.Ignoring warning messages for cleaner output.

1. Load, Visualization and Exploration of Data

1.1. Data Load

```
#Load data
data = pd.read_csv('Facies_vectors.csv')
data
```

```
#Label of Facies
labels = ['SS', 'CSiS', 'FSiS', 'SiSh', 'MS','WS', 'D','PS', 'BS']
data['Facies_Label']=np.select([data['Facies'] == 1,
                                data['Facies'] == 2,
                                data['Facies'] == 3,
                                data['Facies'] == 4,
                                data['Facies'] == 5,
                                data['Facies'] == 6,
                                data['Facies'] == 7,
                                data['Facies'] == 8,
                                data['Facies'] == 9,]
                                ,labels,default='')
```

```
#Unique items in a column that is categorical
data['Well Name'] = data['Well Name'].astype('category')
data['Formation'] = data['Formation'].astype('category')
data['Facies_Label'] = data['Facies_Label'].astype('category')
```

```
data
```


Fig2

In Figure 2 The code starts by loading the data from the CSV file using pandas and assigns it to a DataFrame named 'data'.The code then defines a list of facies labels named 'labels', which contains strings representing different facies types. The 'Facies' column in the DataFrame is numeric, representing different facies types from 1 to 9.Next, the code converts the 'Well Name', 'Formation', and 'Facies_Label' columns to categorical data types.After the data is loaded and processed, it is displayed to provide a glimpse of the DataFrame.

```
In [12]: #Data Info (For dtype and NaN analysis)
         data.info()

In [13]: #Delete Rows with Missing Data
         data = data.dropna()
         data.info()
```

Fig3

In figure 3 The dropna() method is used to remove all rows from the DataFrame that contain any missing values. By default, this method drops any row where at least one element is missing. After removing the rows with missing data, the code displays information about the DataFrame to give insights into the data structure.

2. Data Split (Test/Train)

```
In [18]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1000)
         print('Dimensions of X_train:',X_train.shape)
         print('Dimensions of X_test:',X_test.shape)

         Dimensions of X_train: (1948, 7)
         Dimensions of X_test: (835, 7)

In [19]: X_train.hist()
         plt.show()

In [20]: X_train
```

Fig4

In figure 4 In code snippet, the data is split into training and testing sets using the train_test_split function from scikit-learn. Then, it shows the dimensions of the training and testing sets and visualizes the distribution of features in the training set.

```
In [ ]: # Robust Scaler
        rscaler = RobustScaler() #instantiate
        rscaler.fit(X_train)

In [ ]: X_train_scaled = rscaler.transform(X_train) # transform the train dataset to standardized data

        # Original training dataset
        print("Original median : %s " % rscaler.center_)
        print("Original IQR : %s " % rscaler.scale_)

        #Scaled training dataset
        print("Scaled median : %s " % np.median(X_train_scaled,axis=0))
        print("Scaled IQR : %s " % (np.percentile(X_train_scaled, 75,axis=0)-np.percentile(X_train_scaled, 25,axis=0)))

In [ ]: #Scale the test data using the parameters learnt from the training dataset
        X_test_scaled  = rscaler.transform(X_test)

        print("Median of scaled test data: %s" % np.median(X_test_scaled,axis=0))
        print("IQR of scaled test data: %s " % (np.percentile(X_test_scaled, 75,axis=0)-np.percentile(X_test_scaled, 25,axis=0)))

In [ ]: pd.DataFrame(X_train_scaled,index=X_train.index, columns=X_test.columns).hist()
        plt.show()

In [ ]: X_train=pd.DataFrame(X_train_scaled,index=X_train.index, columns=X_test.columns)
        X_test=pd.DataFrame(X_test_scaled,index=X_test.index, columns=X_test.columns)

In [ ]: X_train
```


Fig 5

In figure 5 the code snippet, data preprocessing is performed using the Robust Scaler, a feature transformation technique, to scale the features of the dataset.
the Robust Scaler is fitted to the training data using the fit method.The training data is then scaled using the transform method of the Robust ScalerThe code then prints out the median and IQR of the original training dataset and the scaled training dataset. This is to show how the scaling process has affected the data.The test data is scaled using the same Robust Scaler, but this time the transform method is applied to the test data Finally, the code creates histograms to visualize the distribution of features in the scaled training dataThe original X_train and X_test DataFrames are then updated with the scaled versions

4.1. Logistic Regression

Tuning Hyperparameters

```
In [ ]: score=[]
        score_train=[]
        C=[]
        for i in range(1,100,1):
            C.append(i/10)
            classifier = LogisticRegression(penalty='l2',C=i/10)
            classifier.fit(X_train, y_train)
            train_pred = classifier.predict(X_train)
            score_train.append(f1_score(y_train, train_pred,average="weighted"))
            y_pred = classifier.predict(X_test)
            score.append(f1_score(y_test, y_pred,average="weighted"))

In [ ]: fig, ax1 = plt.subplots()

        color = 'tab:red'
        ax1.set_xlabel('C')
        ax1.set_ylabel('F1_score')
        ax1.set_ylim([0.5,0.60])
        ax1.plot(C, score, '-o',color=color,label='Test F1_score')

        color = 'tab:blue'
        ax1.plot(C, score_train, '-o',color=color,label='Train F1_score')

        ax1.legend(loc="lower right")

        fig.tight_layout()  # otherwise the right y-label is slightly clipped
        plt.show()
```

Applying Model

```
In [ ]: classifier = LogisticRegression(C=2)
        classifier.fit(X_train, y_train)

        print(classifier.coef_) #a1 and a2
        print(classifier.intercept_) # a0
```

```
In [ ]: prediction = classifier.predict(X_test)
```

```
In [ ]: #Compare against true labels (Accuracy)
        print('Accuracy (generalization)',classifier.score(X_test,y_test)) #Accuracy (generalization)
        print('Accuracy (memorization)',classifier.score(X_train,y_train)) #Accuracy (memorization)
```

```
In [ ]: #Comparing other metrics (f1_score)
        print('F1_score (generalization)',f1_score(y_test,classifier.predict(X_test),average="weighted")) #Accuracy (generalization)
        print('F1_score (memorization)',f1_score(y_train,classifier.predict(X_train),average="weighted")) #Accuracy (memorization)
```

Fig 6

In figure 6 logistic regression is used as a machine learning algorithm for classification. The code involves tuning hyperparameters, applying the logistic regression model, and evaluating its performance

4.3. Support Vector Machine

Tuning Hyperparameters

```
In [ ]: from sklearn.model_selection import GridSearchCV

        #Defining parameter range
        Tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
                             'C': [0.1, 1, 10, 100, 1000]},
                            {'kernel': ['linear'], 'C': [1, 10, 100, 1000]},
                            {'kernel': ['poly'],'C': [1, 10, 100, 1000],'degree': [2,3,4]}]

        svc_mod = GridSearchCV(SVC(), Tuned_parameters, refit = True, verbose = 3)
```

```
In [ ]: svc_mod.fit(X_train, y_train)
        svc_mod.best_estimator_
```

```
In [ ]: print(svc_mod.best_params_)
```

```
In [ ]: #Performance (Accuracy)
        print('Accuracy (generalization)',svc_mod.score(X_test,y_test)) #Accuracy (generalization)
        print('Accuracy (memorization)',svc_mod.score(X_train,y_train)) #Accuracy (memorization)

        #Comparing other metrics (f1_score)
        print('F1_score (generalization)',f1_score(y_test,svc_mod.predict(X_test),average="weighted")) #F1_score (generalization)
        print('F1_score (memorization)',f1_score(y_train,svc_mod.predict(X_train),average="weighted")) #F1_score (memorization)
```


Applying Model

```
In [ ]: srbf=SVC(C=10.0,kernel='rbf',gamma=1)
        srbf.fit(X_train, y_train)
```

```
In [ ]: #Performance (Accuracy)
        print('Accuracy (generalization)',srbf.score(X_test,y_test)) #Accuracy (generalization)
        print('Accuracy (memorization)',srbf.score(X_train,y_train)) #Accuracy (memorization)

        #Comparing other metrics (f1_score)
        print('F1_score (generalization)',f1_score(y_test,srbf.predict(X_test),average="weighted")) #F1_score (generalization)
        print('F1_score (memorization)',f1_score(y_train,srbf.predict(X_train),average="weighted")) #F1_score (memorization)
```

Fig 7

In figure 7 support vector machines (SVM) are used as a machine learning algorithm for classification. The code involves tuning hyperparameters, applying the SVM model, and evaluating its performance

4.4. Random Forest

Tuning Hyperparameters

In [ ]:
```
#Number of trees
score=[]
t=[]
n_trees=[]
for i in range(10,300,10):
    n_trees.append(i)
    start_time = time.time()
    cforest = RandomForestClassifier(criterion='entropy',n_estimators=i,max_depth=10,random_state=1, n_jobs=2) #Creating Instance
    cforest.fit(X_train, y_train) #Learning the decision boundaries
    y_pred = cforest.predict(X_test)
    score.append(f1_score(y_test, y_pred,average="weighted"))
    t.append(time.time() - start_time)
```

In [ ]:
```
fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('Number of trees')
ax1.set_ylabel('F1_score', color=color)
ax1.set_ylim([0.5,0.8])
ax1.plot(n_trees, score, '-o',color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('time (s)', color=color)  # we already handled the x-label with ax1
ax2.plot(n_trees, t, '-o',color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```

In [ ]:
```
#max_depth
score=[]
max_d=[]
score_train=[]
for i in range(1,30):
    max_d.append(i)
    cforest = RandomForestClassifier(criterion='entropy',n_estimators=200,max_depth=i,random_state=1, n_jobs=2) #Creating Instanc
    cforest.fit(X_train, y_train) #Learning the decision boundaries
    train_pred = cforest.predict(X_train)
    score_train.append(f1_score(y_train, train_pred,average="weighted"))
    y_pred = cforest.predict(X_test)
    score.append(f1_score(y_test, y_pred,average="weighted"))
```

In [ ]:
```
fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('Max_Depth')
ax1.set_ylabel('F1_score')
ax1.set_ylim([0,1.01])
ax1.plot(max_d, score, '-o',color=color,label='Test F1_score')

color = 'tab:blue'
ax1.plot(max_d, score_train, '-o',color=color,label='Train F1_score')

ax1.legend(loc="lower right")

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```

In [ ]:
```
#min_samples_split
score=[]
score_train=[]
min_samples=[]
for i in range(1,10,2):
    min_samples.append(i/10)
    cforest = RandomForestClassifier(criterion='entropy',n_estimators=200,max_depth=10,min_samples_split=i/10,random_state=1, n_
    cforest.fit(X_train, y_train) #Learning the decision boundaries
    train_pred = cforest.predict(X_train)
    score_train.append(f1_score(y_train, train_pred,average="weighted"))
    y_pred = cforest.predict(X_test)
    score.append(f1_score(y_test, y_pred,average="weighted"))
```

Applying Model

```
In [ ]: cforest = RandomForestClassifier(criterion='entropy',n_estimators=200,max_depth=7,random_state=1, n_jobs=2) #Creating Instance
        cforest.fit(X_train, y_train) #Learning the decision boundaries
        y_pred = cforest.predict(X_test)
```

```
In [ ]: #Compare against true labels (Accuracy)
        print('Accuracy (generalization)',cforest.score(X_test,y_test)) #Accuracy (generalization)
        print('Accuracy (memorization)',cforest.score(X_train,y_train)) #Accuracy (memorization)
```

```
In [ ]: #Comparing other metrics (f1_score)
        print('F1_score (generalization)',f1_score(y_test,cforest.predict(X_test),average="weighted")) #F1_score (generalization)
        print('F1_score (memorization)',f1_score(y_train,cforest.predict(X_train),average="weighted")) #F1_score (memorization)
```

```
In [ ]: #Feature Importance
        importances = cforest.feature_importances_
        # get sort indices in descending order

        indices = np.argsort(importances)[::-1]

        for f in range(X_train.shape[1]):
            print("%2d) %-*s %f" % (f + 1, 30, X.columns.values[indices[f]], importances[indices[f]]))

        plt.figure()
        plt.title('Feature Importances')
        plt.bar(range(X_train.shape[1]),importances[indices],align='center',alpha=0.5)

        plt.xticks(range(X_train.shape[1]), X.columns.values[indices], rotation=90)
        plt.xlim([-1, X_train.shape[1]])
        plt.tight_layout()
        plt.savefig('fig-forest-feature-importances.png', dpi=300)
        plt.show()
```

Fig8

In figure 8 code demonstrates the process of using Random Forest for classification and the tuning of its hyperparameters to improve its performance.

5. Deployment (Predict targets for new dataset)

```
In [ ]: test_well
```

5.1. Logistic Regression

```
In [ ]: y_pred=classifier.predict(X_test)
        test_well['Prediction'] = y_pred
```

```
In [ ]: compare_facies_plot(test_well, 'Prediction', facies_colors)
```

```
In [ ]: target_names = ['SS', 'CSiS', 'FSiS', 'SiSh','MS', 'WS', 'D','PS','BS']
        print(classification_report(y_test, y_pred, target_names=target_names))
```

```
In [ ]: #Confusion Matrix
        cf_matrix = confusion_matrix(y_test, y_pred)

        sns.heatmap(cf_matrix, annot=True, annot_kws={"size": 12},cmap='Blues',fmt="d",xticklabels=target_names,yticklabels=target_names)

        plt.show()
```

5.3. Support Vector Machine

```
In [ ]: y_pred = srbf.predict(X_test)
        test_well['Prediction'] = y_pred

In [ ]: compare_facies_plot(test_well, 'Prediction', facies_colors)

In [ ]: print(classification_report(y_test, y_pred, target_names=target_names))

In [ ]: #Confusion Matrix
        cf_matrix = confusion_matrix(y_test, y_pred)

        sns.heatmap(cf_matrix, annot=True, annot_kws={"size": 12},cmap='Blues',fmt="d",xticklabels=target_names,yticklabels=target_names)

        plt.show()
```

5.4. Random Forest

```
In [ ]: y_pred = cforest.predict(X_test)
        test_well['Prediction'] = y_pred

In [ ]: compare_facies_plot(test_well, 'Prediction', facies_colors)

In [ ]: target_names = ['SS', 'CSiS', 'FSiS', 'SiSh','MS', 'WS', 'D','PS']
        print(classification_report(y_test, y_pred, target_names=target_names))

In [ ]: #Confusion Matrix
        cf_matrix = confusion_matrix(y_test, y_pred)

        sns.heatmap(cf_matrix, annot=True, annot_kws={"size": 12},cmap='Blues',fmt="d",xticklabels=target_names,yticklabels=target_names)

        plt.show()
```

In figure code demonstrates the deployment of the trained machine learning models (Logistic Regression, Support Vector Machine, and Random Forest) to predict facies labels for a new dataset (test_well)

The whole code and dataset you found on
https://github.com/hiteshnagar2611/ONGC_intenship

Technical Skills I've Learned

-Learned about python libraries.

-Learned about csv data

- Learned how to interpret well log data

- Learned how to train model using supervised machine learning model -Learned about API's

-Learned about Visualization

System Requirements

Operating System : Windows & Linux(most Compatible)

RAM: minimum 4 GB

Storage: 500 MB - 1 GB free space

Processor: i3 and above

Graphic Card for better performance.

Compatible for Server Side & personal use.

Conclusion

In conclusion, this internship project explored the application of Machine Learning techniques for facies identification from well logs. The study successfully built predictive models and demonstrated their potential in accurately classifying different facies based on well log data. The Random Forest model showed the best performance, and while dimensionality reduction did not improve accuracy, the original feature space was found to be effective. The developed models can be valuable tools for reservoir characterization and hydrocarbon exploration.