جامعة ابن طفيل
Ibn Tofaïl University
Faculté des Sciences

Université Ibn Tofail
Faculté des Sciences, Kénitra

Mémoire de Projet de Fin d'Etudes

Master Intelligence Artificielle et Réalité Virtuelle

Prédiction de la demande de carburant auprès des stations Afriquia

Établissement d'accueil : **AKWA Group**

Elaboré par : Mme Gargouri Aicha

Encadré par : Mme Touahni Raja (FSK-UIT)
Mr Ghannam Kamal (AKWA Group)

Soutenu le 19/09/2024, devant le jury composé de :

- Mme Touahni Raja (FSK Université Ibn Tofaïl)
- Mr Messoussi Rochdi (FSK Université Ibn Tofaïl)
- Mme Boukir Khaoula (ENSC Université Ibn Tofaïl)
- Mr Nouri Anass (FSK Université Ibn Tofaïl)

Année universitaire 2023/2024

Dedication

I dedicate this work in loving memory of my father, who passed away. Your strength and values will forever inspire me. May this achievement be a way to honour the love and care you gave me.

To my dear mother, thank you for your endless love and sacrifices. Your unwavering support throughout my upbringing has been the foundation of all I've achieved. I will do my best to remain a source of pride in your eyes.

To my brother, who has always been by my side, offering encouragement and strength. Your belief in me has meant the world and has pushed me to keep going, even in the hardest of times.

To my friends, who have shared in my journey, celebrating the highs and supporting me through the challenges. Your kindness and companionship have been invaluable.

To my teachers, for your guidance and support throughout my academic journey. Your lessons will always be cherished.

Acknowledgements

Before delving into this report, I wish to extend my deepest gratitude to **Pr. Raja Touahni**, the coordinator of my master's program and my project supervisor. It has been an honour to work under your guidance. The unwavering support and meticulous coordination provided have been crucial throughout my academic journey. The kindness and consistent availability have been a source of great inspiration, and I am deeply thankful of the mentorship you have offered.

I also wish to express sincere thanks to **Mr. Kamal Ghannam**, who supervised my internship. The guidance and support received during this period were invaluable, and I am truly grateful for the insights and encouragement provided.

Additionally, I extend my heartfelt appreciation to the academic staff that have supported and guided me over the past two years. The dedication and expertise demonstrated have been instrumental in shaping my path, and I am genuinely thankful for the knowledge and experiences shared.

Lastly, I want to acknowledge and thank everyone who contributed, directly or indirectly, to the success of this project, my family, friends and all those who have offered encouragement and support along the way. Your assistance has been invaluable and deeply appreciated.

Abstract

Artificial Intelligence (AI) has become a crucial tool in various industries, providing advanced solutions to complex challenges. In the context of fuel demand forecasting, AI techniques are vital for improving supply chain efficiency and resource management. Accurate fuel consumption predictions support better planning, inventory management, and decision-making for fuel distribution.

During my internship from May to August, I worked within the Information Systems Department; I focused on predicting fuel demand for Afriquia stations using AI methods. The internship was organized into distinct phases:

The first month was dedicated to thorough research on the subject of fuel demand forecasting. This phase involved studying the existing literature, understanding the problem's context, and identifying the best approaches and techniques for the project.

In the second month, I received the dataset and performed exploratory data analysis (EDA). This step involved examining the data to uncover underlying patterns, trends, and anomalies, which helped in understanding the dataset and preparing it for modeling.

The third month was spent applying various AI models to forecast fuel demand. I used machine learning algorithms such as XGBoost, Random Forest, Decision Tree and Gradient Boosting Machines (GBM) as well as deep learning networks such as RNN, LSTM and GRU; also some time series models like ARIMA and SARIMA to build predictive models and assess their performance in predicting fuel consumption.

In the final month, I focused on optimizing the models to enhance their accuracy and reliability. This involved fine-tuning the models, adjusting parameters, and evaluating their effectiveness to ensure the best possible predictions.

This report provides a detailed account of each phase of the project, the methodologies used, and the results achieved. It highlights the significant role of AI in improving fuel demand forecasting and demonstrates how these techniques can be applied to real-world challenges in the energy sector.

Keywords: Artificial Intelligence (AI), Fuel demand forecasting, Machine learning, XGBoost, Random Forest, Gradient Boosting Machines (GBM), Deep Learning, RNN, LSTM, GRU, Time Series, ARIMA, SARIMA.

Contents

List of Figures

List of Abbreviations and Acronyms

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
XGBoost	eXtreme Gradient Boosting
RF	Random Forest
DT	Decision Tree
GBM	Gradient Boosting Machines
ARIMA	Autogressive Integrated Moving Average
SARIMA	Seasonal ARIMA
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
$\mathbf{R^2}$	R-Squared

Introduction

1. Background and Context:

Fuel demand forecasting is a critical component of the energy sector, essential for optimizing supply chains, managing inventory, and ensuring efficient resource allocation. Accurate predictions of fuel consumption are vital for companies to plan effectively and make informed decisions, which ultimately enhance operational efficiency and reduce costs.

In recent years, artificial intelligence (AI) has emerged as a transformative tool in predictive analytics, offering advanced solutions to the complex challenges associated with forecasting. AI encompasses a range of techniques, including machine learning (ML), deep learning (DL), and time series models, each providing unique capabilities to analyze and predict fuel demand with high precision.

2. Presentation of AKWA Group:

2.1 Overview:

AKWA Group is a leading energy conglomerate based in Casablanca, Morocco, which plays a significant role in the country's dynamic economic landscape. As Morocco advances confidently into the future, AKWA Group remains committed to supporting national development and contributing to the nation's progress.

Figure 1 - AKWA Group logo

2.2 Mission and Expansion:

Originally focused on hydrocarbons, AKWA Group has expanded its operations to include a diverse range of strategic activities.

The company is not only a key player in Morocco's energy sector but has also broadened its scope to meet the evolving needs of the economy and society. This expansion includes a gradual increase in its energy portfolio and storage capacities, solidifying its position as a major energy sector player in Morocco.

2.3 Diversification and Innovation:

Recognizing the importance of adapting to societal changes and urban growth, AKWA Group has diversified into high-value sectors over the past fifteen years. This diversification includes investments in press, hospitality, and real estate.

The aim is to provide varied and innovative solutions to both businesses and individuals, aligning with Morocco's broader development goals.

2.4 Commitment to Sustainability:

AKWA Group is dedicated to sustainable development, including investments in clean energy sources such as LNG (Liquefied Natural Gas). The company embraces economic housing projects and follows Morocco's comprehensive development in social, urban, and energy aspects. This commitment positions AKWA Group as a model of responsible and engaged corporate behaviour in Morocco.

2.5 Core Business Areas:

AKWA Group's services and products are organized into five main areas:

- **Fuels and Lubricants:** Focuses on the supply and management of fuel products.
- **Gas:** Involves the distribution and management of gas resources.
- **Fluids:** Covers various fluid-related services and products.
- **Development:** Engages in projects related to urban and social development.
- **Real Estate:** Involves real estate investments and management.

AKWA Group stands out as a responsible and forward-thinking organization, dedicated to supporting Morocco's development through its diverse business activities and commitment to sustainability.

3. Problem Statement:

Despite its importance, fuel demand forecasting presents several challenges, such as handling vast amounts of data, detecting patterns, and achieving precise predictions. Traditional forecasting methods often fall short in addressing these complexities due to their limited ability to process large datasets and capture intricate patterns.

This project seeks to overcome these limitations by leveraging advanced AI techniques. By integrating ML, DL, and time series models, the aim is to improve the accuracy and reliability of fuel demand predictions for Afriquia stations, addressing the shortcomings of conventional approaches and providing more robust forecasting solutions.

Figure 2 - AFRIQUIA logo

4. Project Scope and Objectives:

The project encompasses a comprehensive approach to fuel demand forecasting using various AI techniques:

- **Machine Learning Models**: Implementing algorithms such as XGBoost, Random Forest, Decision Tree and Gradient Boosting Machines (GBM) to develop predictive models based on historical data.
- **Deep Learning Models**: Utilizing recurrent neural networks (RNN), Long Short-Term Memory networks (LSTM), and Gated Recurrent Units (GRU) to capture complex temporal patterns and relationships in the data.
- **Time Series Analysis**: Applying traditional time series forecasting methods like ARIMA and SARIMA to account for temporal trends and seasonal variations in fuel demand data.

The primary objectives of the project are:

- **Develop Robust Predictive Models**: Create and refine machine learning, deep learning, and time series models to accurately forecast fuel demand.
- **Optimize Model Performance**: Enhance the models' performance through optimization techniques, including hyperparameter tuning and model refinement, to improve predictive accuracy.
- **Evaluate and Compare Models**: Assess the effectiveness of each model type, comparing their performance to identify which approach provides the most reliable and accurate forecasts for fuel consumption.

5. Significance of the Study:

The findings of this project are expected to provide valuable insights for Afriquia stations, improving their inventory management and operational decision-making. Additionally, the study contributes to the broader field of AI applications in fuel demand forecasting, showcasing the potential of advanced techniques in solving real-world challenges.

6. Structure of the Report:

The report is structured into several chapters, starting with this introduction. The following chapters include a detailed methodology, results analysis, and conclusions drawn from the study. Each chapter is designed to provide a comprehensive understanding of the project and its outcomes.

Chapter 1: State Of the Art in Fuel Demand Forecasting

1. Introduction:

Fuel demand forecasting is critical in the energy sector, especially for companies like AKWA Group. It optimizes supply chains, manages inventories, and enhances decision-making processes. Traditional methods such as linear regression and moving averages have been widely used. However, these statistical approaches struggle to handle large, complex datasets and fail to capture the intricate consumption patterns often seen in fuel demand. This limitation has necessitated a shift towards more advanced methodologies, particularly those driven by artificial intelligence (AI).

2. Artificial Intelligence in Forecasting:

2.1 AI Revolution in Forecasting:

AI has transformed demand forecasting by providing robust tools to analyze large datasets, uncover hidden patterns, and improve accuracy. AI techniques are particularly effective at capturing complex, nonlinear relationships in fuel demand data. These advancements enable companies to address the shortcomings of traditional models and make more reliable predictions.

2.2 Types of AI Models:

- **Artificial Intelligence (AI):** AI encompasses a broad range of techniques, including machine learning and deep learning, which can be applied to forecasting tasks.
- **Machine Learning (ML):** ML algorithms can learn from data and make predictions without explicit programming.
- **Time Series Analysis:** Time series models are specifically designed to handle data collected over time, making them suitable for forecasting.

3. Applications of AI in the Energy Sector:

3.1 Case Studies and Benefits:

- **Chevron:** Chevron has utilized AI to optimize oil production and refining processes, leading to improved efficiency and reduced forecast errors.
- **Enel:** Enel has leveraged AI and smart meters to analyze real-time energy consumption data and optimize fuel supply.
- **Duke Energy:** Duke Energy has implemented AI-powered forecasting models to optimize power generation and grid management.
- **Southern Company:** Southern Company has used AI to improve demand forecasting and reduce operational costs.

3.2 Integration with IoT:

- **Smart Meters:** The integration of AI with IoT devices like smart meters has revolutionized demand forecasting. Companies like **Enel** have used smart meters to collect real-time data on energy consumption.

4. Challenges and Limitations:

- **Data Quality:** AI models require high-quality data with minimal missing values and outliers.
- **Model Complexity:** Some AI models can be complex and computationally expensive to train and interpret.
- **Market Volatility:** Sudden changes in fuel demand or market conditions can challenge forecasting accuracy.

5. Future Trends in Forecasting:

5.1 Emerging AI Techniques:

- **Reinforcement Learning:** Companies like **Google DeepMind** are exploring reinforcement learning for dynamic demand forecasting. By allowing AI systems to learn from real-time data and adjust their strategies, reinforcement learning models can provide more adaptive and accurate forecasts.
- **Hybrid Models: Siemens** is developing hybrid forecasting models that combine AI with traditional methods to leverage the strengths of both approaches. These models aim to improve accuracy and robustness in predicting complex demand patterns.

5.2 Big Data Integration:

- **Energy Management Systems:** The integration of AI with **Big Data** and IoT is enhancing energy management systems. For instance, **General Electric (GE)** has implemented such systems to monitor and forecast energy demand in real-time, leading to better grid management and resource optimization.

6. Conclusion:

The landscape of fuel demand forecasting is rapidly evolving with the advent of AI technologies. While traditional models like Time Series remain valuable for certain forecasting tasks, the inclusion of ML, DL, and hybrid AI techniques offers companies like AKWA Group powerful tools to address the complexities of fuel demand.

With continued advancements, AI is expected to play an increasingly crucial role in ensuring the energy sector's operational and economic efficiency.

Chapter 2: Data and Technical Setup

1. Introduction:

Fuel demand forecasting utilizing artificial intelligence (AI) requires careful selection of tools and platforms. This chapter delves into the tools and methodologies employed in the fuel demand forecasting project for Afriquia stations, emphasizing their role in ensuring accurate and reliable predictions.

2. Python as a Programming Tool:

Python was selected as the primary programming tool due to its flexibility and extensive library ecosystem. It supports a wide range of functionalities essential for AI projects, from data handling to model development and visualization.

Figure 3 - Python logo

Key Python libraries used include:

- **NumPy**: For numerical computations and array manipulations.
- **Pandas**: For data manipulation, including reading, cleaning, and processing the dataset.
- **Scikit-learn**: For machine learning tasks such as feature selection, scaling, and model evaluation.
- **XGBoost**: For implementing gradient boosting, a core model in the project.
- **TensorFlow**: For developing deep learning models like RNNs and LSTMs, suitable for sequential data analysis.

Python's comprehensive ecosystem enhances AI model development by integrating tools for data preprocessing, feature engineering, and model evaluation.

3. Kaggle as a Development Platform:

Kaggle, a leading platform for data science, was instrumental in this project. It provides features such as dataset sharing, collaborative notebooks, and community discussions, which facilitated various stages of the project.

Figure 4 - Kaggle logo

Kaggle's notebook environment was utilized for data exploration, preprocessing, and initial model development. The datasets feature allowed access to publicly available data for comparative analysis. The platform's collaborative nature and community feedback were valuable in refining the AI models.

4. Dataset:

4.1 Data Preprocessing:

Preprocessing involved several critical steps to prepare the dataset for model training:

- **Handling Missing Values**: Missing data points were addressed through imputation or removal.
- **Feature Engineering**: Cyclical features were created, such as sine and cosine transformations of the day of the year, month, and week.
- **Data Aggregation**: Data was aggregated by **station** and **day**, reducing complexity and focusing on daily predictions.

4.2 Overview:

The dataset for fuel demand forecasting was provided by AKWA Group and initially contained the following features:

- **DateTransaction**: Date and time of each transaction.
- **Station**: A unique identifier for each of the ten fuel stations.
- **TransactionNumber**: An incremental transaction number for each station, reset after reaching 9999.
- **Pump**: The pump number involved in the transaction.
- **Product**: The fuel type sold during the transaction.
- **TransactionVolume**: Volume of fuel sold (in litres).
- **TransactionPrice**: The monetary value of the transaction (in Dhs).

	DateTransaction	Station	TransactionNumber	Pump	Product	TransactionVolume	TransactionPrice
0	2023-10-16T00:18:18.000Z	1249949532	7982	3	1	0.65	10.00
1	2023-08-16T15:12:15.000Z	1546539305	9914	7	1	0.99	15.00
2	2023-11-23T14:30:14.000Z	1546539305	123	6	2	14.30	200.00
3	2023-09-25T17:31:52.000Z	1546539305	6157	5	2	14.56	200.00
4	2023-01-04T11:13:41.000Z	1249949532	9636	13	2	39.63	550.06

Figure 5 - Initial Dataset Features

The original dataset, comprising **6,778,830 rows**, was aggregated by station and day, resulting in **8,911 rows**. This aggregation was driven by the focus on predicting daily transaction volume by station. Consequently, only three of the initial features were retained for the final model:

- **DateTransaction**: Date and time of each transaction.
- **Station**: A unique identifier for each fuel station.
- **TransactionVolume**: Volume of fuel sold (in litres).

After aggregation, the **DateTransaction** feature was split into separate components, resulting in the following new features:

- **DateOnly**: Date of the transaction, with the time component removed.
- **Month**: Month of the transaction.
- **Day**: Day of the transaction.
- **DayOfWeek**: Day of the week of the transaction.
- **DayOfYear**: Day of the year of the transaction.
- **WeekOfYear**: Week of the year of the transaction.

	Station	DateOnly	TransactionVolume	Month	Day	DayOfWeek	DayOfYear	WeekOfYear
0	1	2022-01-01	18915.48	1	1	5	1	52
1	1	2022-01-02	15040.02	1	2	6	2	52
2	1	2022-01-03	21886.33	1	3	0	3	1
3	1	2022-01-04	18649.30	1	4	1	4	1
4	1	2022-01-05	19775.58	1	5	2	5	1

Figure 6 - Dataset after aggregation

4.3 Feature Engineering:

To capture cyclical patterns in fuel demand, the following new features were engineered:

- **Month_sin** and **Month_cos**: Sine and cosine transformations of the month to capture monthly trends.
- **Day_sin** and **Day_cos**: Sine and cosine transformations of the day to account for daily variations.
- **DayOfWeek_sin** and **DayOfWeek_cos**: Sine and cosine transformations of the day of the week to incorporate weekly variations.
- **DayOfYear_sin** and **DayOfYear_cos**: Sine and cosine transformations of the day of the year to account for seasonal effects.
- **WeekOfYear_sin** and **WeekOfYear_cos**: Sine and cosine transformations of the week of the year to capture yearly cycles.

	Station	DateOnly	TransactionVolume	Month_sin	Month_cos	Day_sin	Day_cos	DayOfWeek_sin	DayOfWeek_cos	DayOfYear_sin	DayOfYear_cos	WeekOfYear_sin	WeekOfYear_cos
0	1	2022-01-01	18915.48	0.5	0.866025	0.201299	0.979530	-0.974928	-0.222521	0.017213	0.999852	0.0	1.0
1	1	2022-01-02	15040.02	0.5	0.866025	0.394356	0.918958	-0.781831	0.623490	0.034422	0.999407	0.0	1.0
2	1	2022-01-03	21886.33	0.5	0.866025	0.571268	0.820763	0.000000	1.000000	0.051620	0.998667	0.120537	0.992709
3	1	2022-01-04	18649.30	0.5	0.866025	0.724793	0.688967	0.781831	0.623490	0.068802	0.997630	0.120537	0.992709
4	1	2022-01-05	19775.58	0.5	0.866025	0.848644	0.528964	0.974928	-0.222521	0.085965	0.996298	0.120537	0.992709

Figure 7 - Dataset after Feature Engineering

These cyclical features were essential for modeling seasonal and cyclical patterns in fuel demand.

5. Data Visualization and Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA) was essential for uncovering trends and anomalies in the dataset. Here's a summary of the key findings:

5.1 Transaction Volume Analysis:

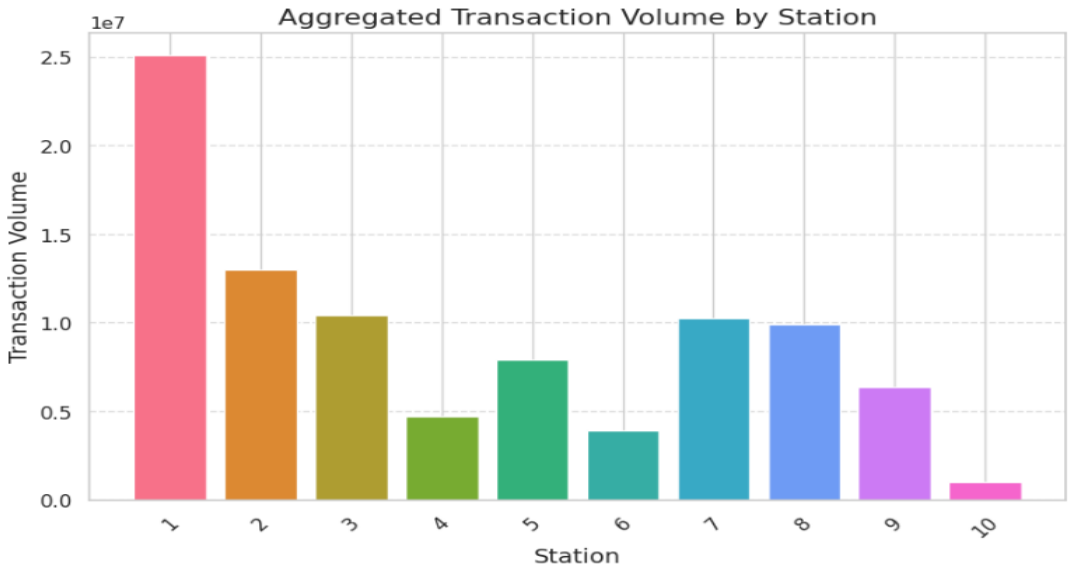

Figure 8 - Aggregated Transaction Volume by Station

- **Broad Trends:**
  - **High-Volume Stations:**
    - Stations such as 1, 2, 3, 7, and 8 are the top performers in terms of fuel sales. These stations consistently report high transaction volumes, suggesting they are major fueling points with steady demand.
  - **Moderate-Volume Stations:**
    - Stations like 4, 5, and 9 show moderate transaction volumes. These stations experience stable fuel demand but at a lower level compared to the high-volume stations.
  - **Low-Volume Stations:**
    - Stations 6 and 10 have the lowest transaction volumes. These stations might be located in less trafficked areas or have other factors affecting their performance.

- **Anomalies:**
  - **Volume Disparities:**
    - There's a clear disparity in transaction volumes among stations. For example, Station 1's volume is significantly higher than Station 10's. This indicates that some stations are much more crucial in terms of fuel demand.
  - **Unexpected Changes:**
    - High-volume stations may show sudden increases in transaction volumes, possibly due to special events or local demand spikes. Conversely, low-volume stations might experience unexpected drops, which could be due to supply issues or regional competition.

- **Correlation Matrix and Heatmap Analysis:**

The correlation matrix helps to identify relationships between variables while the heatmap provides a visual representation of the relationships between features. It helps in identifying which features are related and can guide feature selection for further analysis:

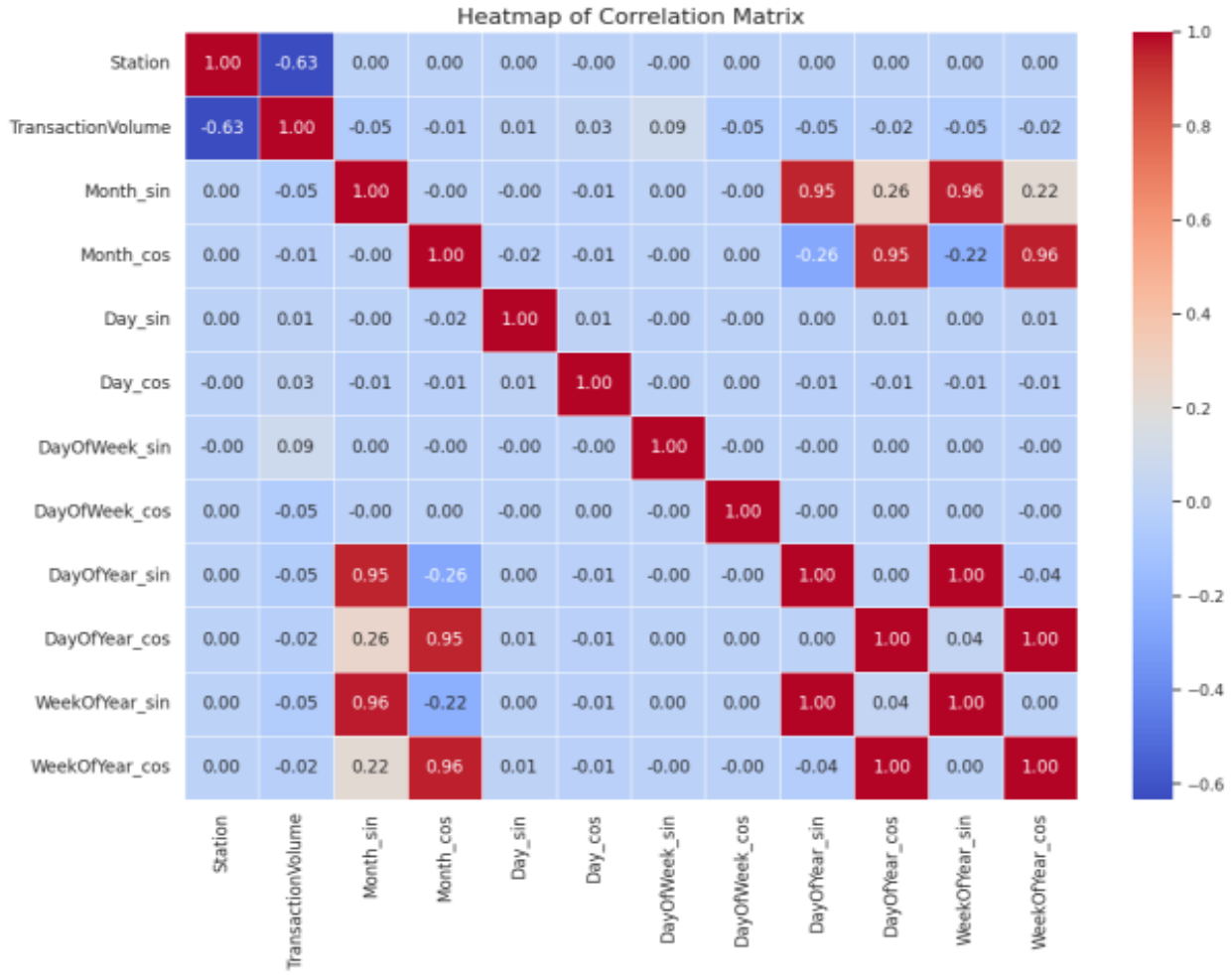

Figure 9 - Heatmap of the Correlation Matrix

- **High Correlation:**
    - **DayOfYear_sin** and **DayOfYear_cos** have a strong correlation with **Month_sin** and **Month_cos**, indicating that seasonal patterns are captured by these features.
- **Low Correlation:**
    - Variables like **Station** and **TransactionVolume** show a moderate negative correlation, suggesting that while the station's identifier does not strongly influence transaction volume directly, there are significant differences in fuel demand across stations.

In summary, the EDA revealed clear patterns in fuel demand across different stations, with notable differences in transaction volumes and some anomalies that could indicate underlying issues or opportunities for further investigation.

6. Data Clustering and Analysis:

To explore the structure of the data, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was applied:

- **Clustering Approach:**
  - DBSCAN is a clustering algorithm that identifies clusters based on the density of data points. It groups together points that are closely packed, and labels points that lie alone in low-density regions as outliers. The algorithm is ideal for datasets with varying densities and can find clusters of arbitrary shapes.
- **DBSCAN Results:**
  - After applying DBSCAN to the scaled data, the algorithm detected only a **single cluster**. This suggests that the data points were too similar or uniformly distributed based on the chosen features, preventing the formation of distinct clusters. The result indicates a **homogeneous dataset** where no significant subgroups or patterns could be differentiated through DBSCAN.
- **Cluster Distribution:**
  - The uniformity in the clustering result highlights that the fuel demand data across different stations and time periods does not exhibit clear separations or localized variations. All data points were grouped together, reflecting a lack of distinct clusters in the dataset.

This finding reinforced the understanding that the dataset may require more complex or different types of analysis to uncover meaningful patterns beyond simple clustering techniques like DBSCAN.

7. Challenges with the Dataset:

Several challenges were encountered while working with the dataset:

- **Initial Uncertainty**: There was initial uncertainty about whether to predict fuel demand by day or by time period (morning, afternoon, evening, night) or by station. This led to multiple trial runs and considerations before settling on daily predictions by station.
- **Missing or Inconsistent Data**: The dataset contained missing values and inconsistencies that required addressing through imputation or removal.
- **Complexity of Transactions**: The original dataset with over 6 million rows was resource-intensive to handle. Aggregating the data by station and day helped manage this complexity, but careful attention was needed to avoid losing relevant information.

Despite these challenges, thorough preprocessing and analysis ensured a high-quality dataset for accurate model training and fuel demand forecasting.

8. Conclusion:

This chapter discussed the tools, platforms, and data that formed the foundation of the fuel demand forecasting project. Python and its associated libraries were pivotal in handling data, training models, and generating insights. Kaggle provided a collaborative platform for development, and careful preprocessing ensured that the data was ready for predictive modeling. The next chapter will focus on the development of machine learning, deep learning and time series models for fuel demand forecasting.

Chapter 3: Modeling and Optimization

1. Introduction:

This chapter presents the various modeling approaches applied to predict fuel demand across Afriquia stations. It covers machine learning models, deep learning architectures, and time series techniques. Each model was carefully tuned and optimized to achieve the best performance, with a focus on capturing both the complexity and temporal patterns in the data.

2. Machine Learning Models:

2.1 Models Descriptions:

- **Decision Tree:**

A decision tree is a machine learning algorithm used to divide data into smaller, more manageable subsets. The process begins with a binary split and continues until no further divisions can be made, resulting in branches of varying lengths.

The objective of a decision tree is to represent the training data with the smallest possible tree. This aligns with the principle that the simplest explanation for a phenomenon is often the best. Smaller trees are also more efficient, offering quicker decisions and being easier to interpret. There are methods to control the depth of the tree, such as pruning, which simplifies the tree when necessary.

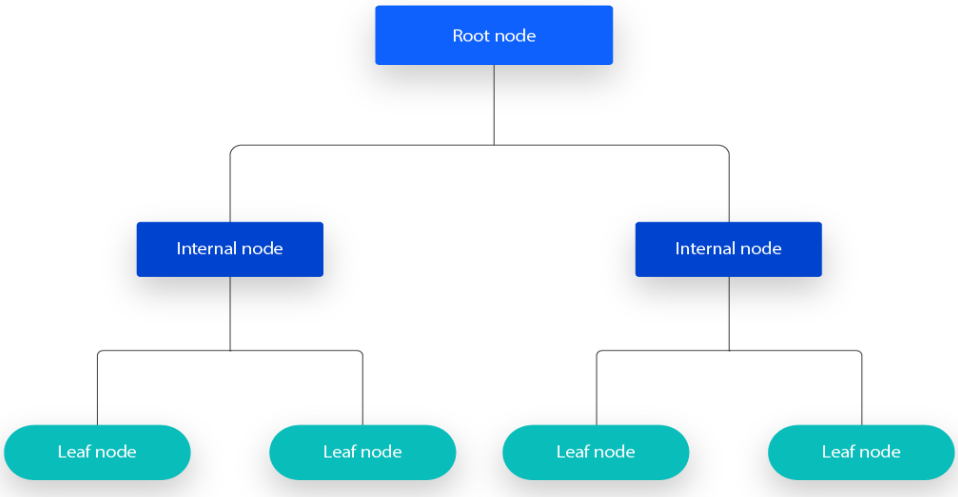

Figure 10 - Decision Tree

Key steps in building a decision tree include:

- Splitting:

At each node of the tree, the dataset is split into two subsets based on a chosen feature and a split criterion. For regression tasks, this criterion could be the sum of squared errors (SSE), and for classification, metrics like Gini impurity or information gain are used.

- **Gini Impurity** for classification:

$$G(p) = \sum_{i=1}^{C} p_i(1 - p_i)$$

Where p_i is the probability of class i in the node, and C is the total number of classes.

- **Sum of Squared Errors (SSE)** for regression:

$$SSE = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Where y_i is the observed value, and $\bar{y}$ is the mean of the observations in the node.

Splitting continues until the node reaches a terminal node or leaf, containing fewer observations or meeting a stopping criterion such as maximum depth.

- Pruning:

To avoid overfitting, which occurs when the model fits the training data too closely, pruning simplifies the tree. Pruning involves converting some branch nodes into leaf nodes by removing small, less significant branches. This can be done by:

$$Cost\ Complexity\ Pruning: R_\alpha(T) = R(T) + \alpha|T|$$

Where $R(T)$ is the error of the tree T, $|T|$ is the number of nodes, and α is a parameter controlling the trade-off between tree complexity and fit.

- Tree Selection:

The best tree is selected based on minimizing cross-validated error. This involves finding the tree T with the lowest cross-validation error $CV(T)$, which generalizes well to unseen data. Typically, the optimal tree balances between high accuracy and minimal depth:

$$CV(T) = \frac{1}{k} \sum_{i=1}^{k} L(T, D_i)$$

Where $L(T, D_i)$ is the loss function for the tree T on the validation fold D_i, and k is the number of cross-validation folds.

In summary by recursively splitting data, pruning over-complex structures, and selecting the smallest effective tree, decision trees offer a balance between simplicity and predictive accuracy. The use of metrics like Gini impurity and SSE helps optimize the splits and create models that generalize well to unseen data.

- **Random Forest:**

The Random Forest algorithm has three primary hyperparameters: node size, the number of trees, and the number of features sampled at each split. It can be used for both regression and classification tasks.

Random Forest is an ensemble of decision trees. Each tree is trained on a bootstrap sample, a subset of the training data drawn randomly with replacement. Roughly one-third of the data is not selected in the bootstrap sample, forming the out-of-bag (OOB) sample, used for validation.

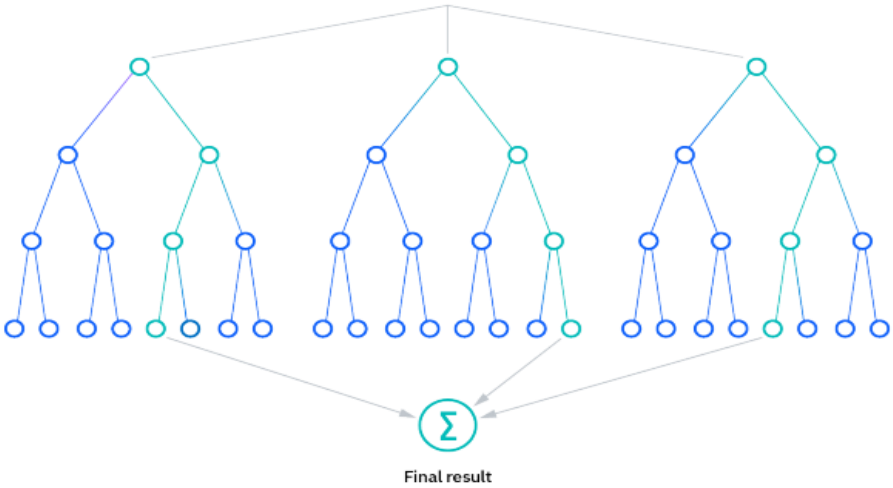

Figure 11 - Random Forest

Another key element of Random Forest is feature bagging: at each node, a random subset of features is chosen from which to select the best split, introducing diversity among trees and reducing correlations between them.

Let's denote the number of trees in the forest by T, and the number of features sampled at each split by m. Each tree is built on a bootstrap sample S_b, and at each node of the tree, a random subset of m features from the full feature set p is selected:

$$m \leq p$$

For a given input x, each tree t produces a prediction $h_t(x)$. The final prediction depends on the task:

- Regression: The prediction is the average of all tree predictions:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

- Classification: The prediction is based on majority voting:

$$\hat{y} = mode\big(h_1(x), h_2(x), \ldots, h_T(x)\big)$$

The **out-of-bag (OOB) error** is computed using the data points not included in the bootstrap sample. For each sample, the OOB prediction $\hat{y}$OOB is calculated by averaging the predic-

tions of the trees where the sample was not used in training. This serves as a form of cross-validation.

In summary, Random Forest reduces overfitting by introducing randomness through both bootstrap sampling and feature bagging. The aggregation of multiple decision trees results in a strong model that generalizes well to unseen data.

- **GBM:**

Gradient boosting is a machine learning ensemble technique that improves prediction accuracy by combining multiple weak learners, often decision trees, in a sequential manner. The main idea is to correct the errors of previous models in each iteration, progressively reducing bias and improving the overall model's performance.

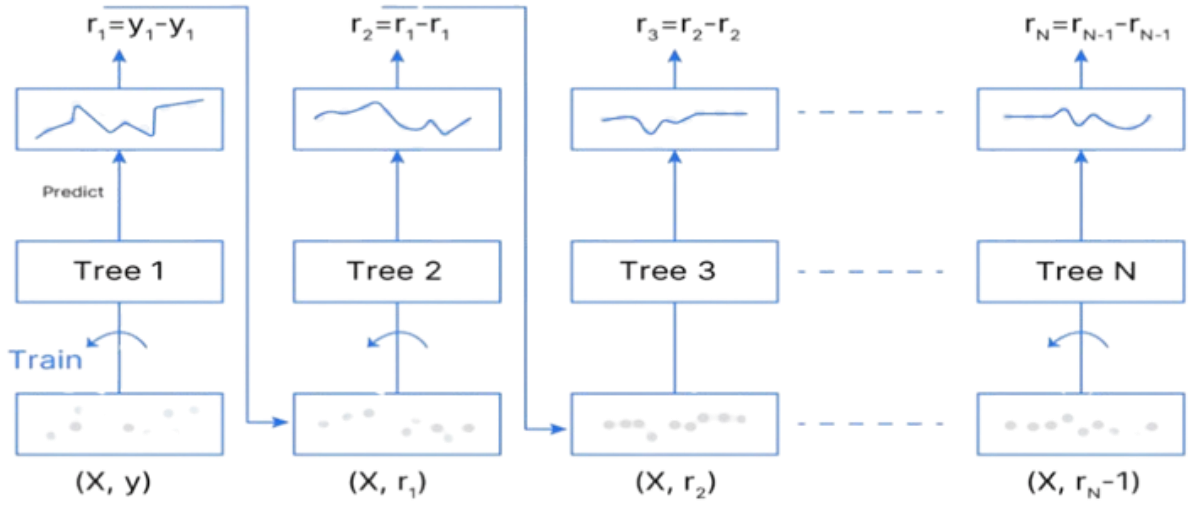

Figure 12 - Gradient Boosting

At each iteration m, a new model $h_m(x)$ is trained to predict the residuals (errors) of the previous model $F_{m-1}(x)$:

$$r_{i,m} = y_i - F_{m-1}(x_i)$$

Where y_i is the true value and $F_{m-1}(x_i)$ is the prediction from the previous model. The new model $h_m(x)$ minimizes the loss function L, often using gradient descent.

- In regression tasks, the loss function is typically Mean Squared Error (MSE):

$$L(y_i, F(x_i)) = \frac{1}{n}\sum_{i=1}^{n}(y_i - F(x_i))^2$$

In each iteration, the update for the model is given by:

$$F_m(x) = F_{m-1}(x) + \eta h_m(x)$$

Where η is the learning rate, controlling the contribution of each new model.

- **In classification tasks**, the loss function is often log-likelihood, and the boosting process focuses on minimizing this loss, with the weak learners focusing on misclassified samples. The general update rule remains the same, but the loss and gradients are tailored to classification.

In summary, gradient boosting sequentially reduces bias by correcting the errors of previous models, with the goal of minimizing a defined loss function (e.g., MSE for regression). The combination of weak learners produces a strong, more accurate model.

- **XGBoost:**

XGBoost (Extreme Gradient Boosting) is a highly efficient machine learning library built on Gradient Boosted Decision Trees (GBDT). It improves weak models by training trees sequentially, where each new tree focuses on the residual errors (difference between actual and predicted values) from the previous trees. This process refines the model step by step, reducing bias and improving accuracy.

- **Gradient Boosting**: In XGBoost, the goal is to minimize a loss function by sequentially adding trees that predict the residuals of the previous ones. Given a dataset with n observations (x_i, y_i), where x_i is the input and y_i is the corresponding output, the model's prediction at iteration t is:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

Where f_t is the newly added decision tree.

- **Loss Function and Optimization**: XGBoost minimizes a custom objective function combining a loss term and a regularization term:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \ell\left(y_i, \hat{y}^{(t-1)} + f_t(x_i)\right) + \sum_{j=1}^{t} \Omega(f_i)$$

Here, ℓ is the loss function (e.g., mean squared error for regression or log loss for classification), and $\Omega(f_i)$ is the regularization term, which penalizes model complexity to avoid overfitting.

- **Regularization**: XGBoost uses both L1 (Lasso) and L2 (Ridge) regularization, making it more robust to overfitting. This differentiates it from traditional GBDT, which typically uses only L2 regularization.

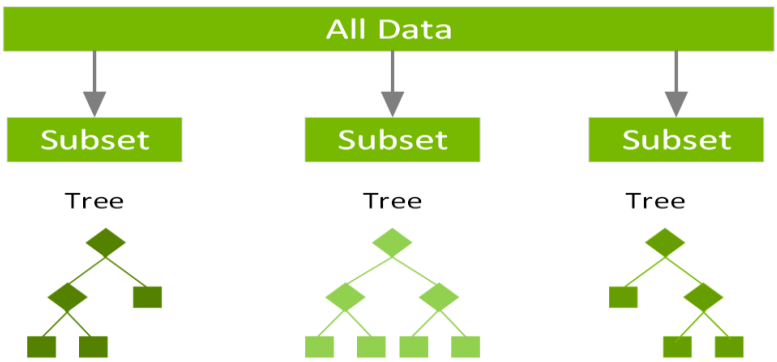

Figure 13 - XGBoost

While both Random Forest and XGBoost use decision trees as their base models, their approach differs:

- **Random Forest**: Grows multiple trees independently using random subsets of data (bagging), with predictions averaged over all trees. This reduces variance and helps avoid overfitting.
- **XGBoost**: Grows trees sequentially (boosting), with each tree focusing on reducing errors of the previous ones. This approach reduces bias and enhances model accuracy over time.

XGBoost enhances the traditional GBDT approach with several optimizations:

- **Parallelized Tree Construction**: Unlike traditional GBDT, XGBoost builds trees in parallel, speeding up training significantly.
- **Level-Wise Tree Growth**: XGBoost builds trees level by level, ensuring that the most significant splits are handled first, which optimizes efficiency and model accuracy.
- **Handling Missing Data**: XGBoost can automatically handle missing values by learning optimal split directions for missing features, unlike traditional decision trees that need explicit handling of missing values.
- **Second-Order Optimization**: XGBoost uses both gradient (first derivative) and Hessian (second derivative) for more precise updates in the boosting process, which helps converge faster and with higher accuracy compared to traditional GBDT methods.

In summary, XGBoost excels in large-scale regression, classification, and ranking tasks by combining speed, accuracy, and advanced regularization. Its improvements over traditional gradient boosting methods make it a widely popular choice in the field of machine learning.

2.2 Feature Selection:

Feature importance was evaluated to determine the top five most impactful features for the XGBoost model. The model was then retrained using only these features to enhance performance and simplify the model.

- **Decision Tree:**

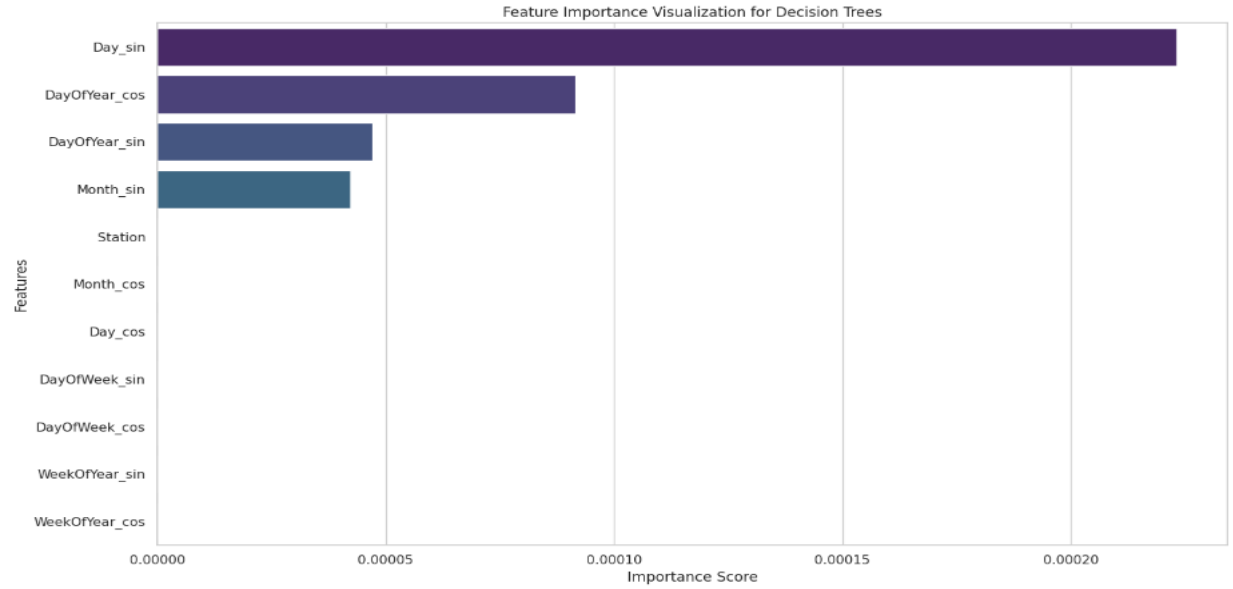

Figure 14 – DT's feature importance

The top five most important features identified by the Decision Tree model were 'Day_sin', 'DayOfYear_cos', 'DayOfYear_sin', 'Month_sin' and 'Station'.

- **Random Forest:**

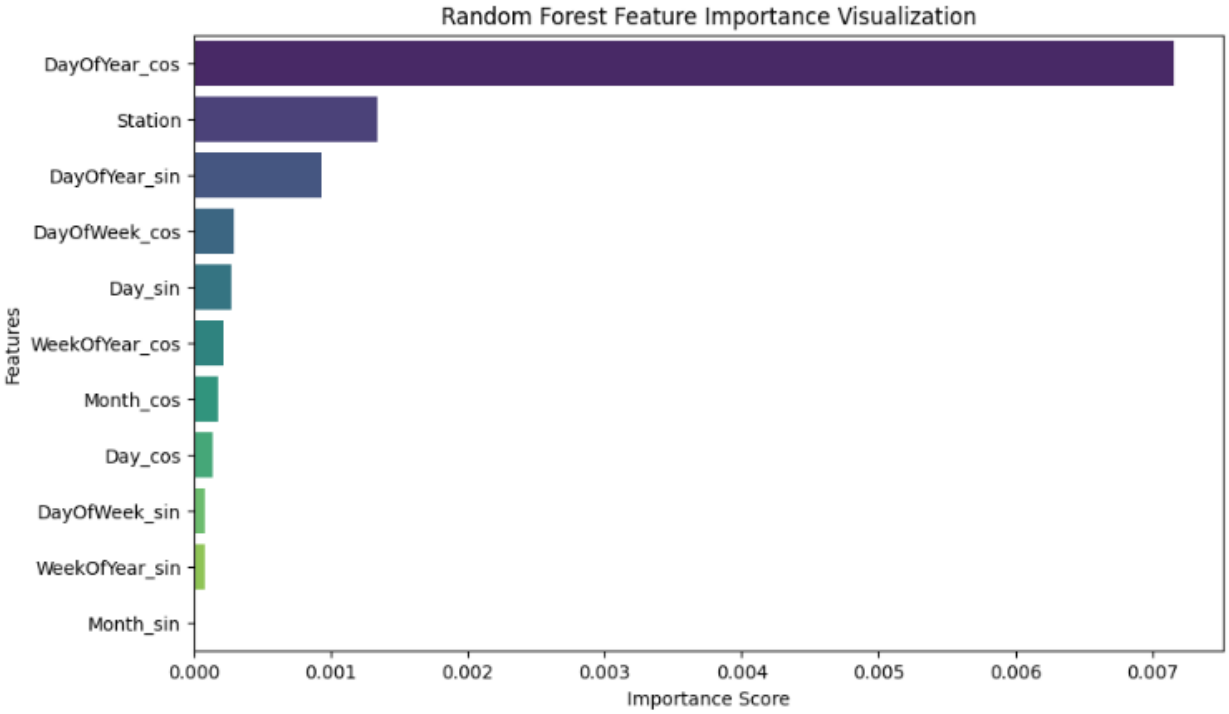

Figure 15 - RF's feature importance

The top five most important features identified by the Random Forest model were 'DayOfYear_cos', 'Station', 'DayOfYear_sin', 'DayOfWeek_cos' and 'Day_sin'.

- **GBM:**

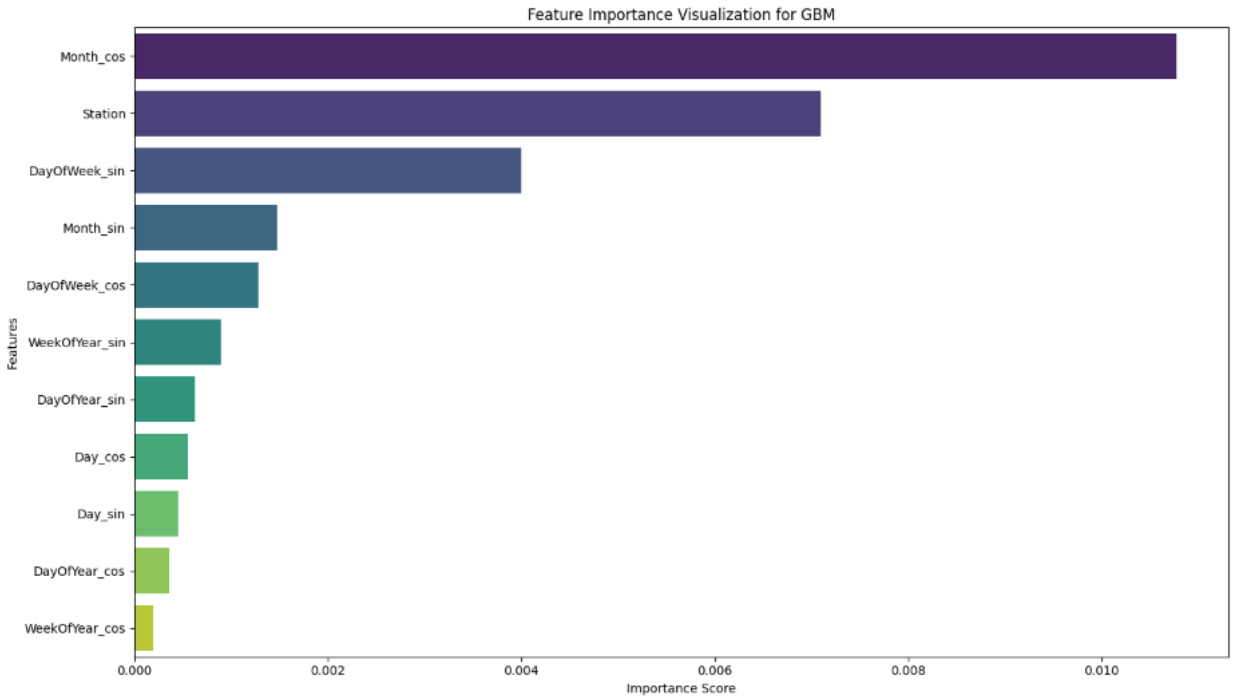

Figure 16 - GBM's feature importance

The top five most important features identified by the GBM model were 'Month_cos', 'Station', 'DayOfWeek_sin', 'Month_sin' and 'DayOfWeek_cos'.

- **XGBoost:**

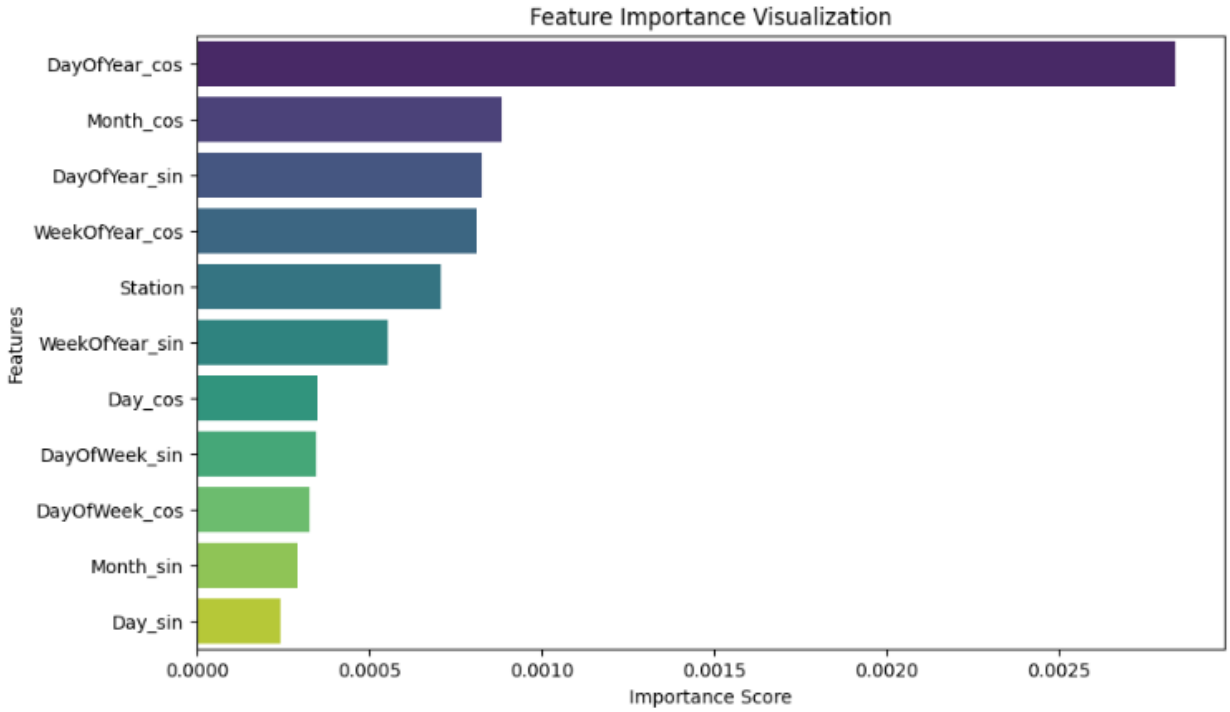

Figure 17 - XBGoost's feature importance

The top five most important features identified by the XGBoost model were 'DayOfYear_cos', 'Month_cos', 'DayOfYear_sin', 'WeekOfYear_cos' and 'Station'.

- **Comparison:**

While most models consistently identified 'DayOfYear_cos', 'DayOfYear_sin', and 'Station' as key features, other features varied across models. For example, the Decision Tree emphasized 'Day_sin' and 'Month_sin', while the Random Forest highlighted 'DayOfWeek_cos'. The GBM and XGBoost models both gave more attention to 'Month_cos', compared to the other models.

Feature importance was used to identify the most influential variables affecting the model's predictions. This approach helps simplify models, improve interpretability, and optimize performance by focusing on the most relevant features as well as the 'TransactionVolume' feature. By reducing the feature set to the top five, we streamline the model while retaining predictive power, leading to more efficient training and better generalization.

2.3 Hyperparameter Tuning:

Figure 18 - Optuna logo

To optimize the performance of my models, I used Optuna, an advanced framework for hyperparameter optimization. Through its efficient search algorithms, Optuna systematically explored a wide range of hyperparameter combinations, and by that fine-tuning the models and achieving optimal predictive accuracy while minimizing overfitting. This approach significantly enhanced the overall performance and robustness of the models.

3. Deep Learning Models:

3.1 Models Descriptions:

- **RNN:**

Recurrent Neural Networks (RNNs) mimic the functioning of the human brain in data science, artificial intelligence, machine learning, and deep learning, enabling computers to recognize patterns in sequential data. They are specifically designed to handle data where previous inputs influence future outputs, making them powerful tools for tasks like time series forecasting, speech recognition, and natural language processing.

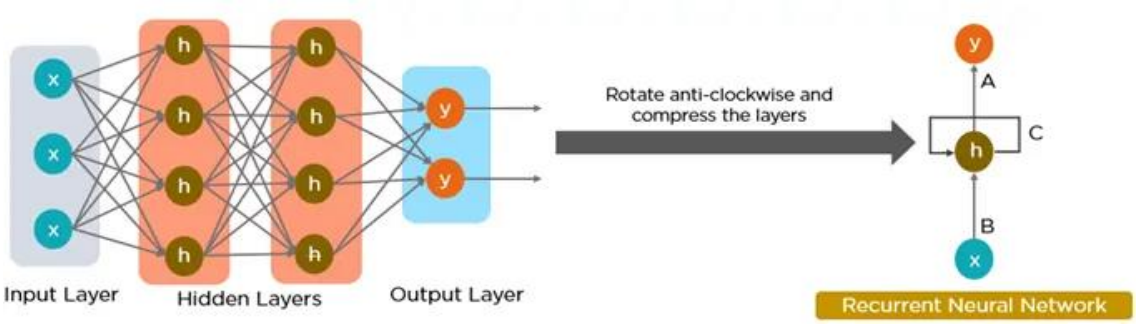

Figure 19 – RNN

The core concept of RNNs lies in their **hidden state**, which retains a memory of previous inputs. This hidden state is updated at each time step, allowing the model to maintain context over time.

Given an input sequence $x = (x_1, x_2, \dots, x_T)$, where T is the length of the sequence, the RNN processes the sequence one time step at a time. At each time step t, the RNN computes:

- Hidden State Update:

$$h_t = \tan h\,(W_h h_{t-1} + W_x x_t + b_h)$$

Where:

- h_t is the hidden state at time step t,
- W_h is the weight matrix applied to the previous hidden state h_{t-1},
- W_x is the weight matrix applied to the current input x_t,
- b_h is the bias term,
- $\tan h$ is the activation function, which squashes values between -1 and 1.

The hidden state h_t encodes information about all previous inputs up to time t, effectively serving as the RNN's memory. The weight matrices W_h and W_x are shared across time steps, meaning the model applies the same parameters repeatedly, allowing it to generalize across different sequence lengths.

- Output Prediction: At each time step t, the RNN can also produce an output y_t, depending on the task. The output is typically a function of the hidden state:

$$y_t = W_y h_t + b_y$$

Where:

- W_y is the weight matrix applied to the hidden state,
- b_y is the bias term for the output.

This equation produces a prediction y_t, at each time step based on the current hidden state h_t.

RNNs are trained through Backpropagation Through Time (BPTT), where the loss is calculated across the entire sequence, and gradients are propagated backward to update the weights.

While RNNs can handle sequential data, they struggle with long-term dependencies due to the vanishing gradient problem. This limitation is addressed by advanced variants like Long Short-Term Memory (LSTM) networks, which retain information over longer periods.

In summary RNNs are effective at learning patterns in sequential data by using hidden states to maintain context across inputs. However, for tasks requiring long-term memory, LSTMs or GRUs are often preferred to overcome the vanishing gradient issue.

- **LSTM:**

Long Short-Term Memory Networks (LSTMs) are a class of sequential neural networks used in deep learning that allow information to persist over time. As a specialized variant of Recurrent Neural Networks (RNNs), LSTMs are designed to tackle the vanishing gradient problem that typically affects traditional RNNs, making them ineffective for long-term dependencies.

In a standard RNN, the output at a particular time step depends not only on the current input but also on the previous hidden state. Mathematically, this is represented as:

$$h_t = \tan h(W_h h_{t-1} + W_x x_t)$$

Where:

- h_t is the hidden state at time t ,
- W_h and W_x are weight matrices for the hidden state and input respectively,
- x_t is the input at time t ,
- $\tan h$ is the activation function.

However, RNNs suffer from the vanishing gradient problem due to the repeated multiplication of small gradients over time during backpropagation through time (BPTT), which leads to the decay of gradient values. This is where LSTMs come into play.

LSTMs use a unique architecture that includes gates to control the flow of information. These gates help LSTMs retain relevant information for longer periods and discard unnecessary information. Each LSTM cell contains three primary gates:

- **Forget Gate**: Determines which information to discard from the cell state.

$$f_t = \sigma\big(W_f . [h_{t-1}, x_t] + b_f\big)$$

Where:

- f_t is the forget gate value,
- σ is the sigmoid activation function,
- W_f and b_f are the weight matrix and bias for the forget gate.

- **Input Gate**: Decides which new information to store in the cell state.

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tan h\,(W_C . [h_{t-1}, x_t] + b_C)$$

Where:

- i_t is the input gate value,
- $\tilde{C}_t$ is the candidate cell state.

- **Output Gate**: Decides what part of the cell state to output.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t.\tan(C_t)$$

Where:

- o_t is the output gate value,
- C_t is the updated cell state.

The core of the LSTM is the cell state C_t , which carries information throughout the sequence. The forget gate updates the cell state by deciding what information to retain:

$$C_t = f_t.C_{t-1} + i_t.\tilde{C}_t$$

This equation ensures that the cell can hold onto useful long-term dependencies and discard irrelevant information when needed.

LSTMs' ability to use these gates to control the flow of information makes them highly effective at capturing long-term dependencies in sequence data. By using mathematical mechanisms such as the forget, input, and output gates, LSTMs manage to overcome the limitations of traditional RNNs, providing superior performance in tasks like time series forecasting, speech recognition, and text generation.

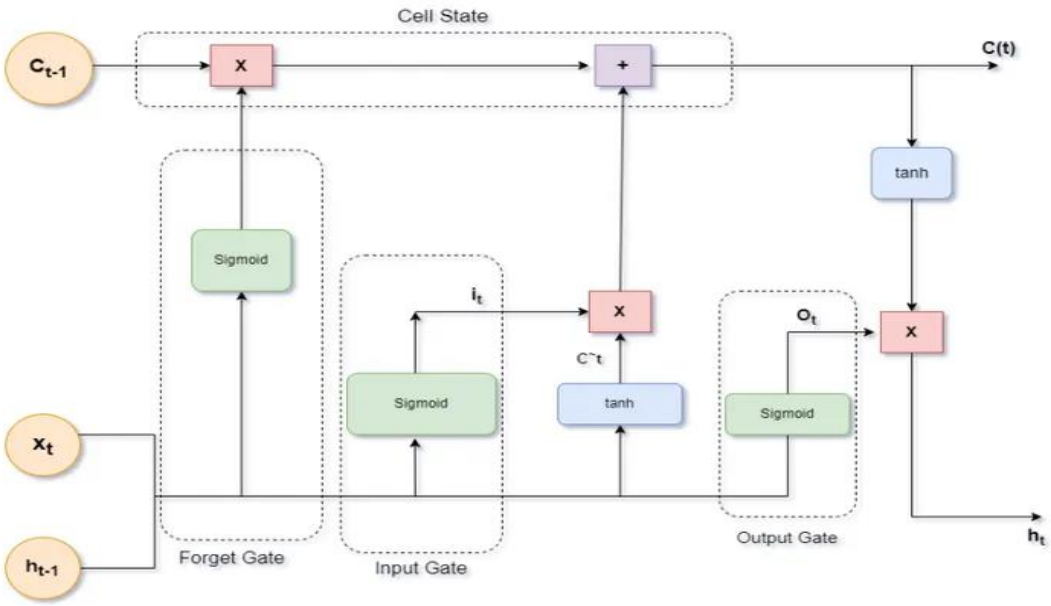

Figure 20 - LSTM

- **GRU:**

In artificial intelligence, Recurrent Neural Networks (RNNs) are widely used for processing sequential data. However, they struggle with learning long-term dependencies due to issues like vanishing gradients. Gated Recurrent Units (GRUs), a type of RNN, address these limita-

tions by using gating mechanisms that control the flow of information, making them more efficient for sequence-based tasks.

GRUs are an improvement over traditional RNNs and Long Short-Term Memory (LSTM) networks, offering a simpler architecture and faster training. Unlike LSTMs, GRUs do not have a separate cell state (C_t); instead, they rely solely on a hidden state (H_t), which reduces complexity.

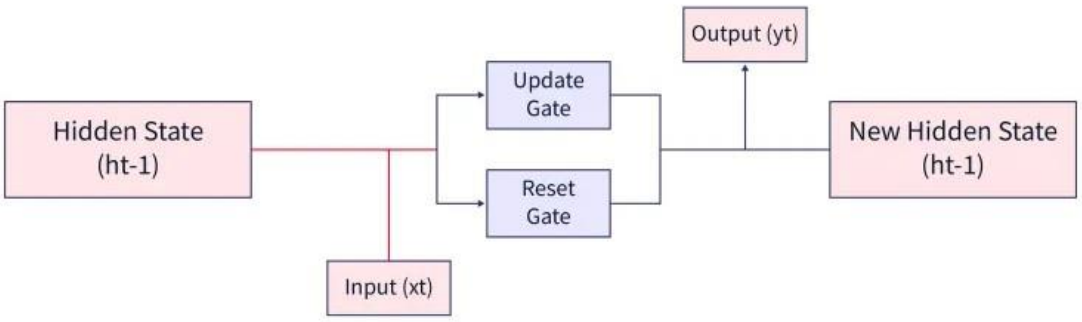

Figure 21 - GRU

Key Mechanisms in GRUs:

- Gated Mechanisms: GRUs utilize two gates:

- **Update gate** z_t controls how much of the previous hidden state is passed to the future.
- **Reset gate** r_t decides how much of the past information should be forgotten.

The hidden state is updated as:

$$H_t = (1 - z_t).H_{t-1} + z_t.\widetilde{H}_t$$

Where $\widetilde{H}_t$ is the candidate activation, computed as:

$$\widetilde{H}_t = \tan h\,(W.(r_t.H_{t-1}) + U.X_t)$$

Where:

- $\tan h$ is the hyperbolic tangent activation function, which introduces non-linearity and outputs values between -1 and 1. It helps in transforming the candidate hidden state to the same range,
- W is the weight matrix for the reset gate's input. It transforms the input X_t combined with the reset gate-modified previous hidden state $r_t.H_{t-1}$,
- r_t is the reset gate at time t. It determines how much of the previous hidden state H_{t-1} should be forgotten. It takes values between 0 and 1,
- H_{t-1} is the hidden state from the previous time step. The reset gate r_t scales this hidden state to control the amount of past information to keep,
- U is the weight matrix for the input data X_t. It transforms the input data into the same space as the hidden state,
- X_t is the input data at time t.

- Mitigating Vanishing Gradients: The gating structure in GRUs helps prevent vanishing gradients by controlling the flow of information, allowing gradients to propagate through longer sequences.

- Faster Training: With fewer parameters than LSTMs, GRUs are faster to train. The simplified architecture leads to more efficient learning, making GRUs suitable for tasks involving long sequences.

In summary, GRUs offer a streamlined approach to sequence modeling, addressing the limitations of standard RNNs and LSTMs by improving memory management and training speed, while still capturing long-term dependencies in data.

3.2 Hyperparameter Tuning:

For hyperparameter tuning of the LSTM, RNN, and GRU models, I employed a systematic approach to optimize their performance. Each model was trained and validated using cross-validation with time series data to ensure robust and reliable performance metrics. Key hyperparameters, such as sequence length, number of epochs, and batch size, were carefully selected to balance training efficiency and model accuracy.

I used the Adam optimizer for all the models, as it is well-suited for handling large datasets and complex architectures by dynamically adjusting the learning rate during training. Early stopping was implemented to prevent overfitting, allowing the training process to halt when validation performance stopped improving, thus preserving the best model weights. Additionally, dropout layers were used in the LSTM model to further enhance generalization by randomly omitting certain neurons during training.

4. Time Series Models:

4.1 Models Descriptions:

- **ARIMA:**

An Autoregressive Integrated Moving Average (ARIMA) model is a statistical tool used to analyze and forecast time series data by combining three key components: autoregression (AR), differencing (I), and moving average (MA).

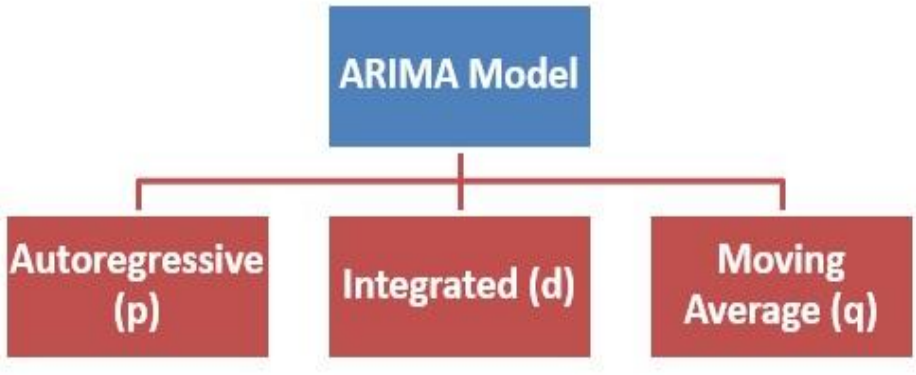

Figure 22 – ARIMA

- **Autoregression (AR)**: Predicts future values based on past observations. The AR part of the model captures the relationship between an observation and its previous values. Mathematically, it can be expressed as:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \epsilon_t$$

Where the ϕ_i are the AR coefficients, p is the number of lag observations, and ϵ_t is the white noise error term.

- **Integrated (I)**: Involves differencing the data to achieve stationarity, which means transforming the series to have constant mean and variance. If d is the number of differences required, the differenced series is:

$$\Delta^d X_t = X_t - X_{t-d}$$

- **Moving Average (MA)**: Models the relationship between an observation and a residual error from a moving average model applied to lagged observations. The MA part can be described by:

$$X_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

Where θ_i are the MA coefficients, q is the size of the moving average window, and μ is the mean of the series.

ARIMA Model Notation**:**

- **p**: Number of lag observations (AR order).
- **d**: Number of differences required to make the series stationary.
- **q**: Size of the moving average window.

An ARIMA model is often denoted as ARIMA(p, d, q), where p, d and q are integer values specifying the model's components. For example, ARIMA(1, 1, 1) would include one lag observation, one difference, and a one-period moving average.

- **SARIMA:**

Seasonal Autoregressive Integrated Moving Average (SARIMA) is an extension of ARIMA designed to handle univariate time series data with a seasonal component. While ARIMA itself cannot manage seasonal data directly, SARIMA incorporates additional parameters to account for seasonality.

$$SARIMA \underbrace{(p,d,q)}_{non-seasonal} \underbrace{(P,D,Q)_m}_{seasonal}$$

Figure 23 - SARIMA

SARIMA augments the ARIMA model with three extra parameters for the seasonal aspects of the time series and one parameter for the seasonality period:

- Seasonal Autoregressive Order (*P*): Specifies the number of lagged observations in the seasonal component.

$$Seasonal\ AR = \phi_1^s X_{t-s} + \phi_2^s X_{t-2s} + \cdots + \phi_P^s X_{t-Ps}$$

Where ϕ_i^s are seasonal autoregressive coefficients and s is the seasonal period.

- Seasonal Difference Order (D): Indicates the number of seasonal differences required to make the series stationary.

$$\Delta_s^D X_t = X_t - X_{t-Ds}$$

Where Δ_s^D denotes seasonal differencing and s is the length of the seasonal cycle.

- Seasonal Moving Average Order (Q): Defines the number of lagged forecast errors in the seasonal component.

$$Seasonal\ MA = \theta_1^s \epsilon_{t-s} + \theta_2^s \epsilon_{t-2s} + \cdots + \theta_Q^s \epsilon_{t-Qs}$$

Where θ_i^s are seasonal moving average coefficients and ϵ_t are the errors.

- Seasonal Period (m): Represents the number of time steps in one seasonal cycle.

To configure SARIMA, one must select hyperparameters for both the trend and seasonal components of the series, ensuring the model can effectively capture both long-term trends and repeating seasonal patterns.

4.2 Hyperparameter tuning:

For hyperparameter tuning of ARIMA and SARIMA models, I utilized the ‘auto_arima’ function from the ‘pmdarima’ library. This approach was chosen due to its ability to automatically determine the optimal parameters for both ARIMA and SARIMA models by evaluating various combinations and selecting the best fit.

For the SARIMA model, I enabled the seasonal component and specified a seasonal period, ensuring the model could handle repeating cycles in the data.

Cross-validation using ‘TimeSeriesSplit’ was employed to assess the model's performance and robustness across different time periods, thereby ensuring reliable and accurate forecasts. This method effectively balances the trade-off between model complexity and performance, leading to well-tuned and robust forecasting models.

5. Conclusion:

In this chapter, we explored a range of modeling techniques, including machine learning, deep learning, and time series methods, to forecast fuel demand. Each model was carefully tuned and optimized to handle the intricacies of the dataset. The insights gained from these models will be crucial for understanding their performance and applicability. The next chapter will delve into the comparative analysis and evaluation of these models, highlighting their strengths and effectiveness in predicting fuel consumption.

Chapter 4: Evaluation and Performance Analysis

1. Introduction:

This chapter focuses on evaluating the performance of the predictive models using both visual predictions and key metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2). We will first present and analyze the predicted values against the actual fuel transaction volumes for each model, followed by definitions of the evaluation metrics and then presenting and comparing the results of these metrics across the models to determine the most effective method for fuel demand forecasting.

2. Predictions vs Actual Values:

To better understand how well each model performs, we plot the predicted fuel transaction volumes against the actual values. These plots offer a visual comparison, allowing us to observe the model's accuracy and any discrepancies between the predicted and actual values. Below are the results for the different models, showing how the predictions align with the actual transaction volumes.

2.1 Machine Learning Models:

- **Decision Tree:**

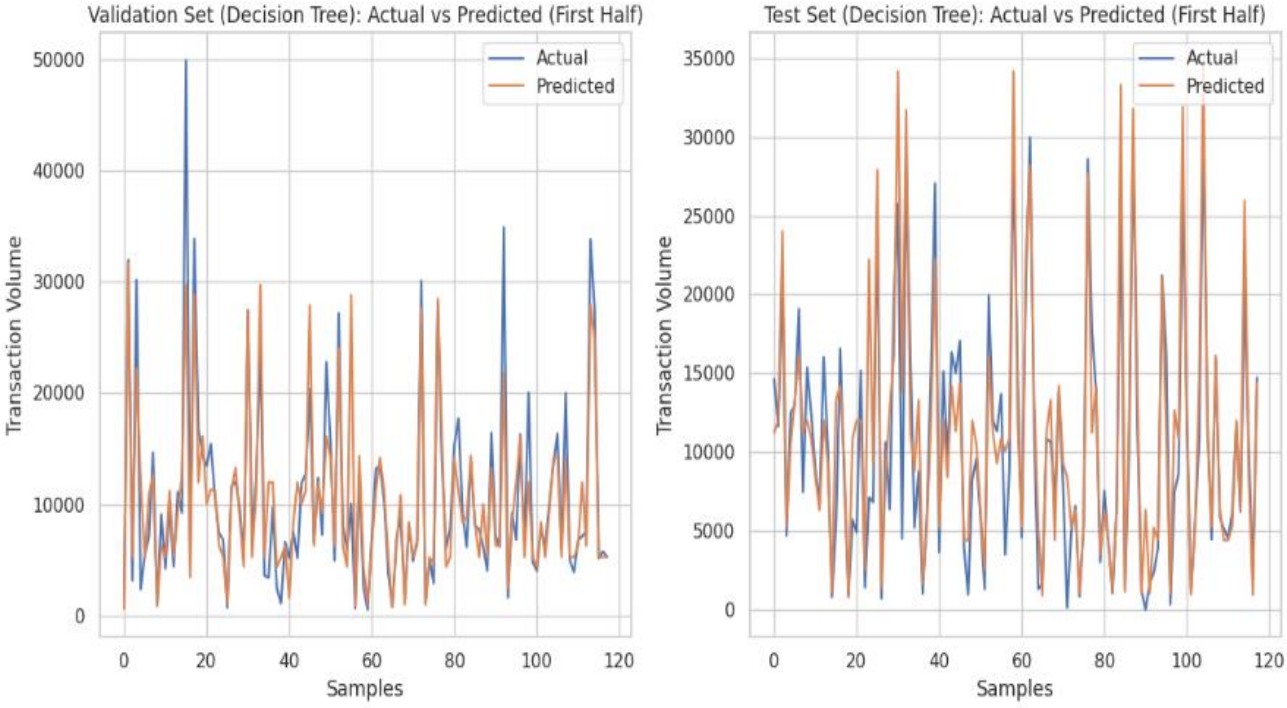

Figure 24 - DT's prediction plot

- **Random Forest:**

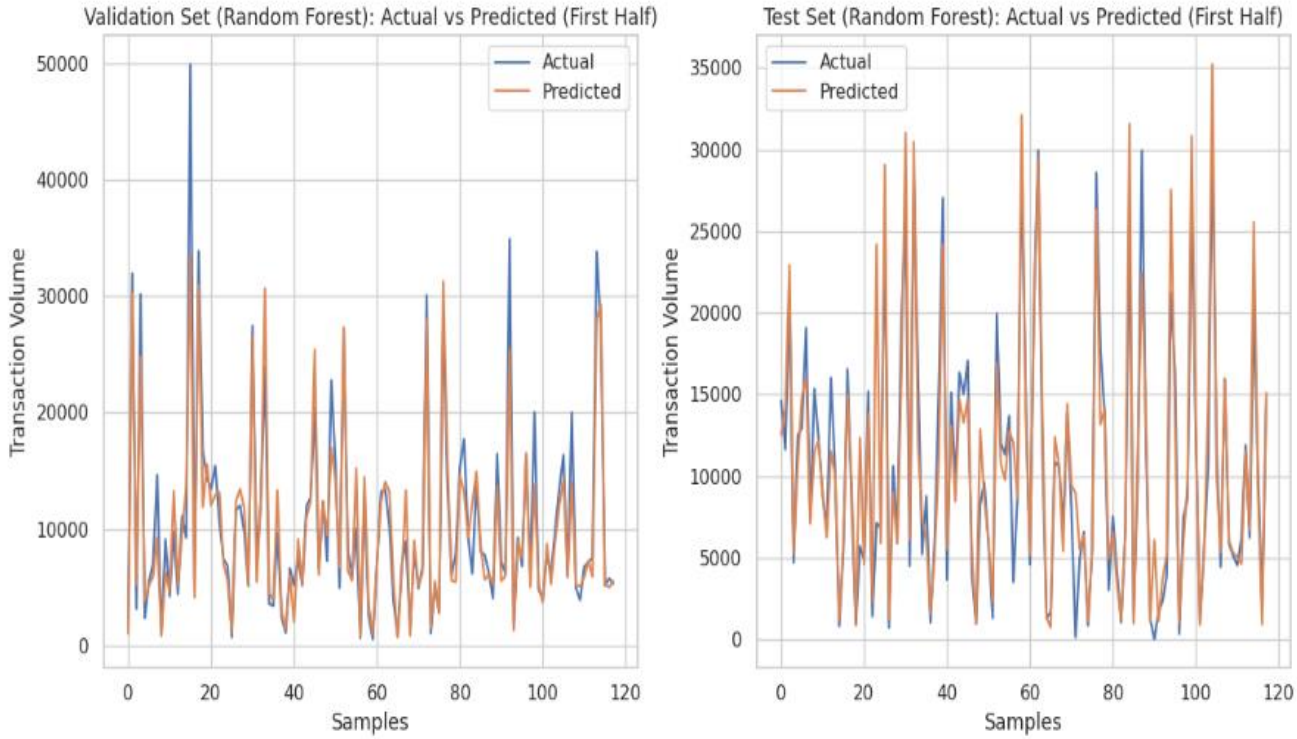

Figure 25 - RF's prediction plot

- **GBM:**

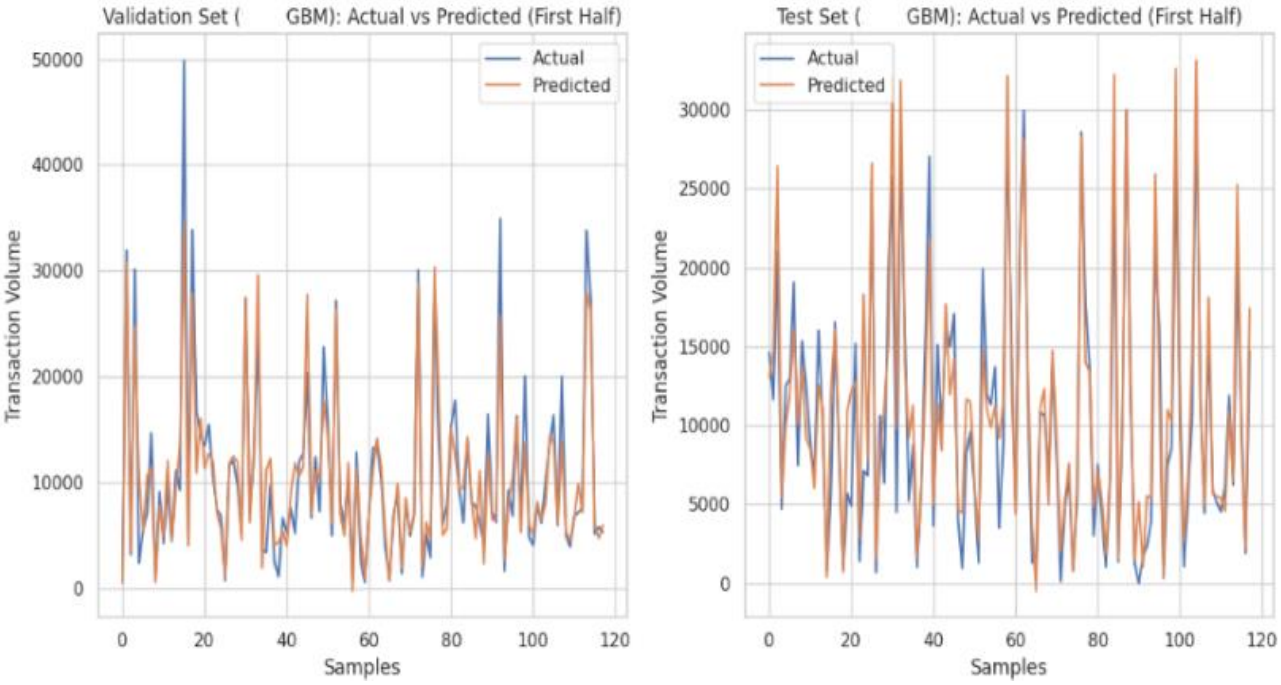

Figure 26 - GBM's prediction plot

- **XGBoost:**

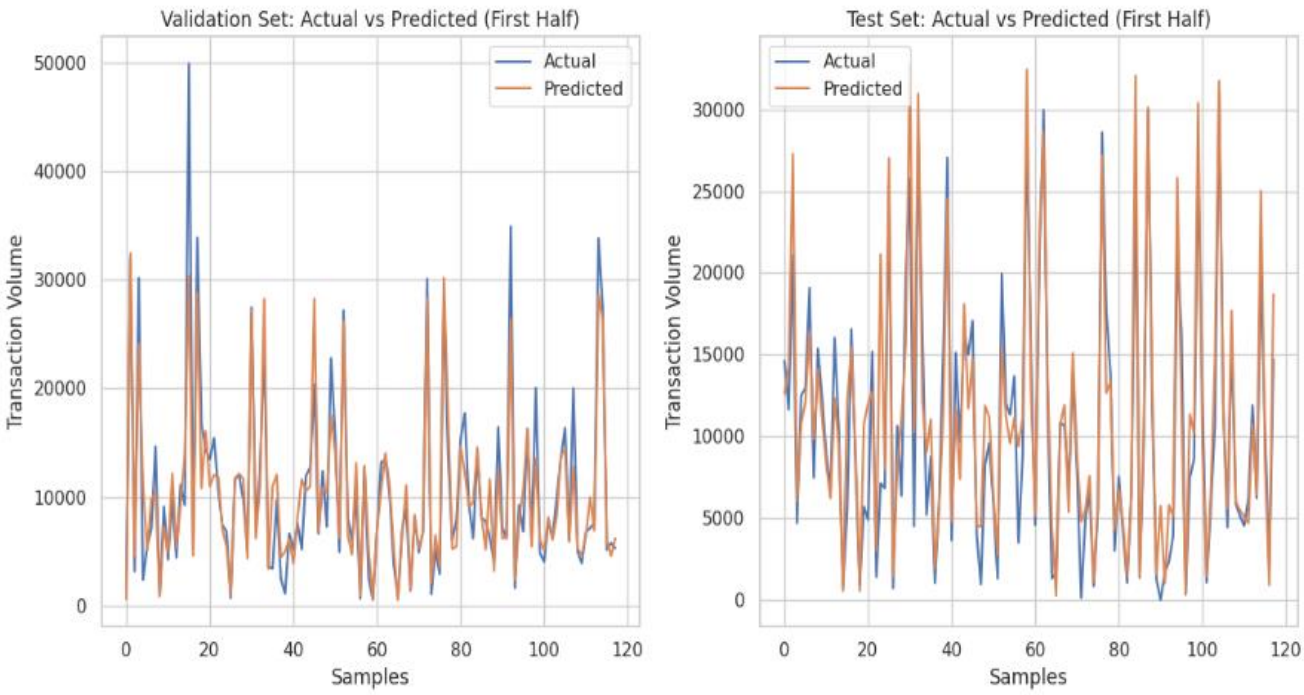

Figure 27 - XGBoost's prediction plot

- **Analysis of results:**

Among the four models, the **GBM** model demonstrates the strongest performance on both the validation and test sets. It accurately captures the peaks and valleys of transaction volume with minimal deviations, particularly on the validation set, where it closely follows the actual data's fluctuations. On the test set, although slight variations in higher and lower transaction volumes are present, the model's predictions still align well with the trends, showcasing its robustness in generalizing to unseen data.

The **Random Forest** model performs well but not as closely as GBM. On the validation set, it follows the data patterns reasonably but shows more deviations in capturing the extremes. However, its predictions on the test set still align well with transaction volume trends, indicating a good fit, though not as precise as GBM.

XGBoost ranks slightly below Random Forest, exhibiting similar behaviour with slightly more pronounced deviations. It captures the general pattern of the data, though its performance on both validation and test sets shows more noise, especially in the peaks, making it less precise than Random Forest and GBM.

The **Decision Tree** model shows the weakest performance. It struggles to capture the fine details in both the validation and test sets, leading to erratic predictions that miss the sharp peaks and valleys of transaction volume. This indicates weaker generalization ability compared to the other models.

2.2 Deep Learning Models:

- **RNN:**

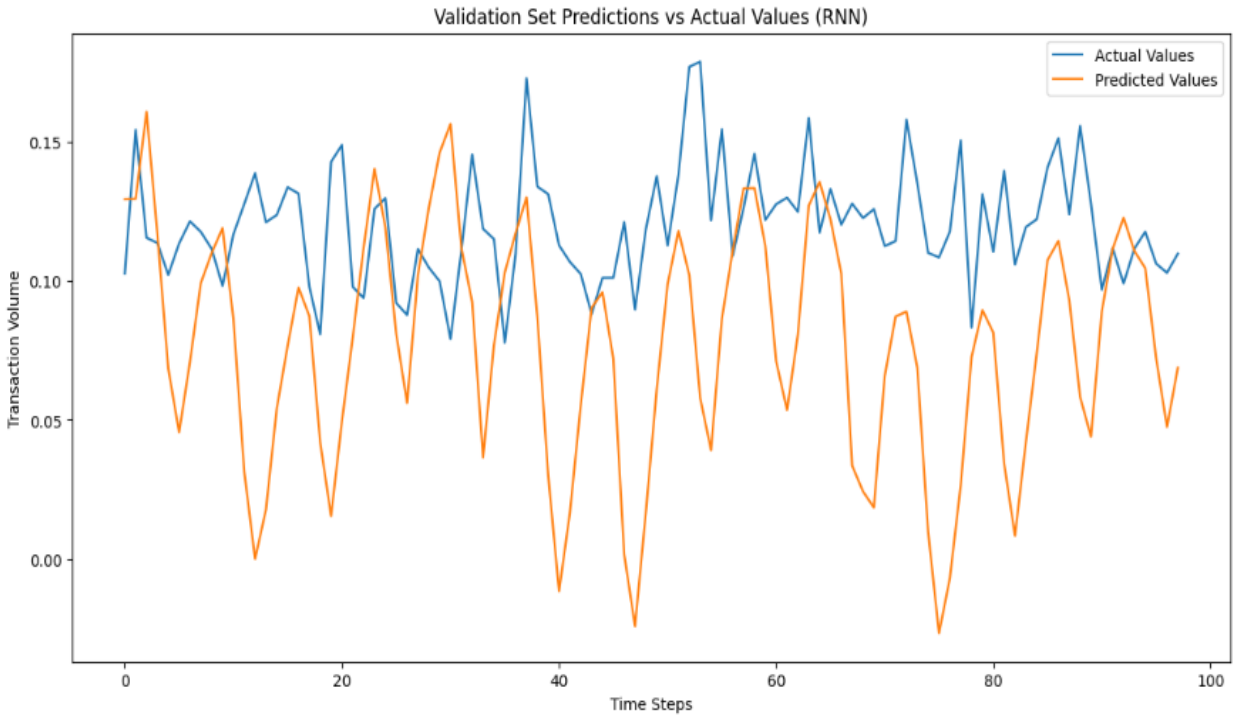

Figure 28 - RNN's prediction plot

- **LSTM:**

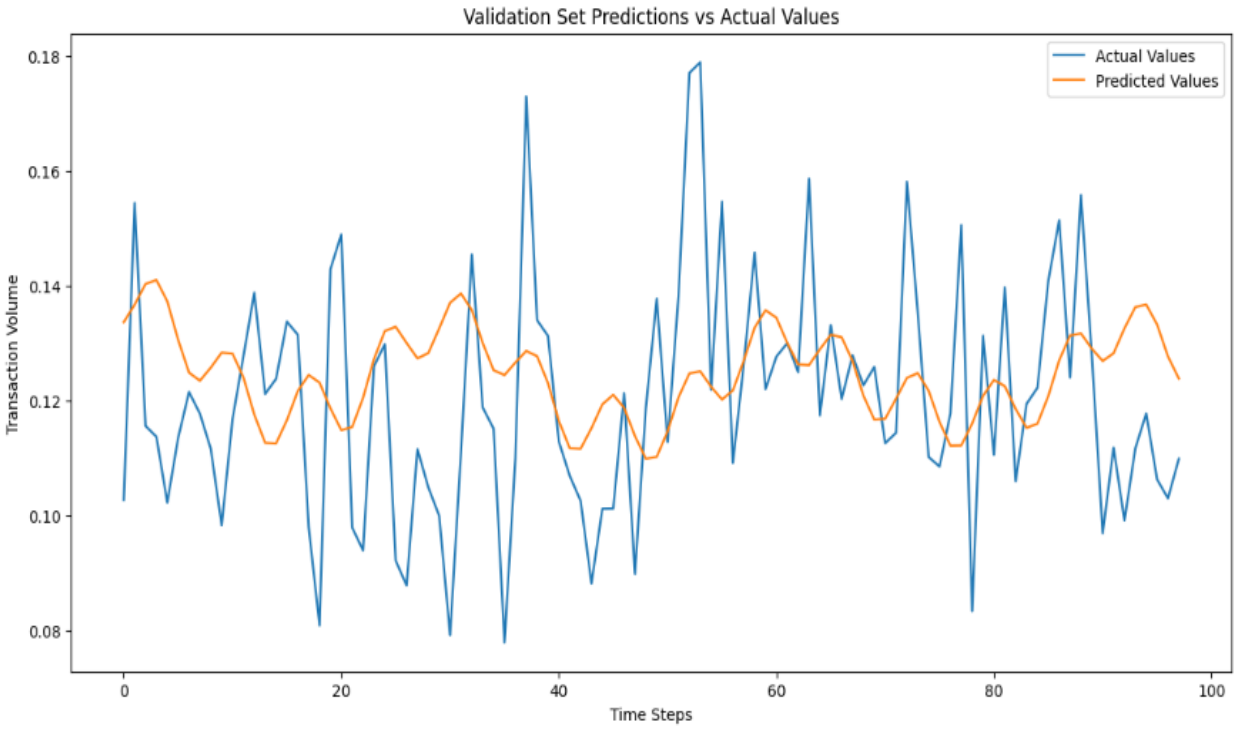

Figure 29 - LSTM's prediction plot

- **GRU:**

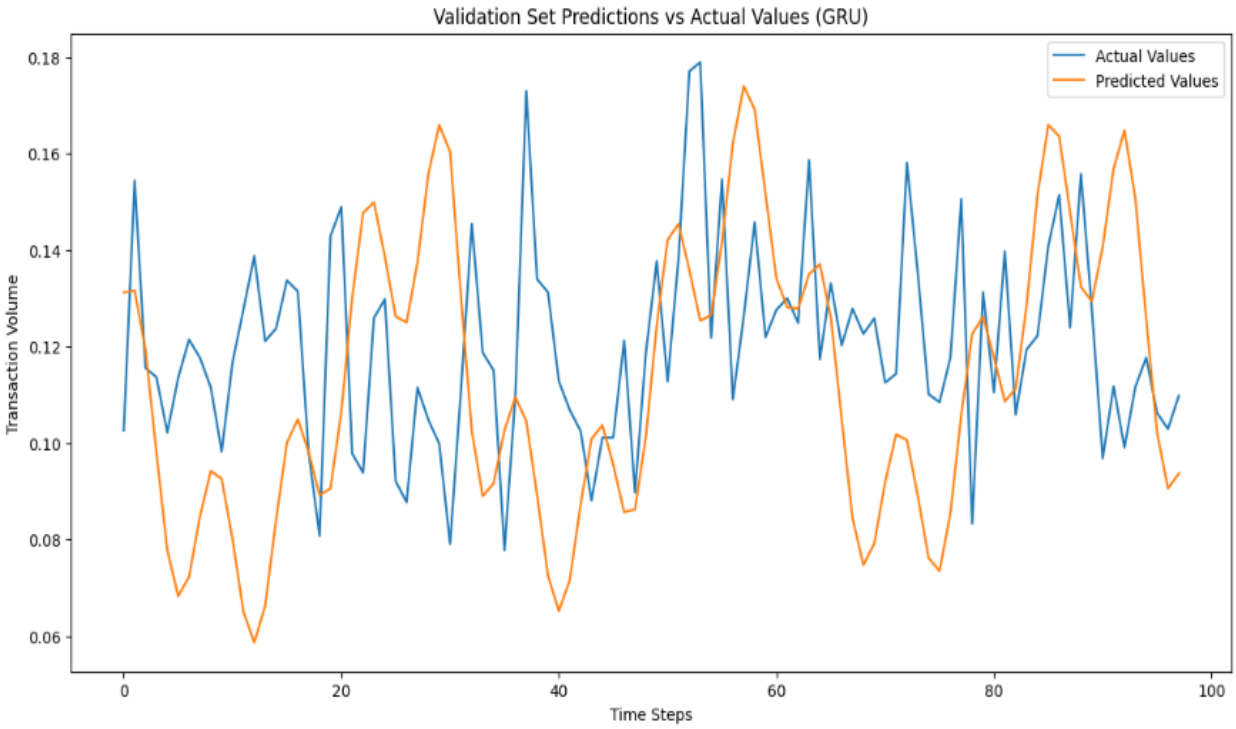

Figure 30 - GRU's prediction plot

- **Analysis of results:**

Based on the analysis of the RNN, LSTM, and GRU models, it is clear that each deep learning model faces challenges in effectively capturing the underlying patterns in the data.

The **RNN** model shows significant deviations in predictions and lacks consistency, indicating underfitting. This suggests that the RNN struggles with the complexity of the temporal dependencies in the data, likely due to its limitations in handling long sequences and potential vanishing gradient issues.

The **LSTM** model, while capable of learning longer dependencies, produces overly smoothed predictions that fail to capture sharp fluctuations in the actual values. This also suggests underfitting, indicating that the LSTM might not be fully leveraging the complexities within the data, even after careful tuning.

The **GRU** model performs relatively better, capturing some of the sharper peaks and valleys. However, there are still notable gaps between the actual and predicted values, implying that the model, like the LSTM, may not be entirely suited to the data's nature.

These observations reveal that, despite thorough preprocessing, feature engineering and hyperparameter tuning, these deep learning models struggle to capture the data's patterns, suggesting they are not well-suited for this problem. The persistent underfitting indicates that the issue lies in the nature of the data itself. Therefore, traditional machine learning models, like Random Forests or Gradient Boosting Machines, may be better suited to capture the relationships within the data and provide more accurate results.

2.3 Time Series Models:

- **ARIMA:**

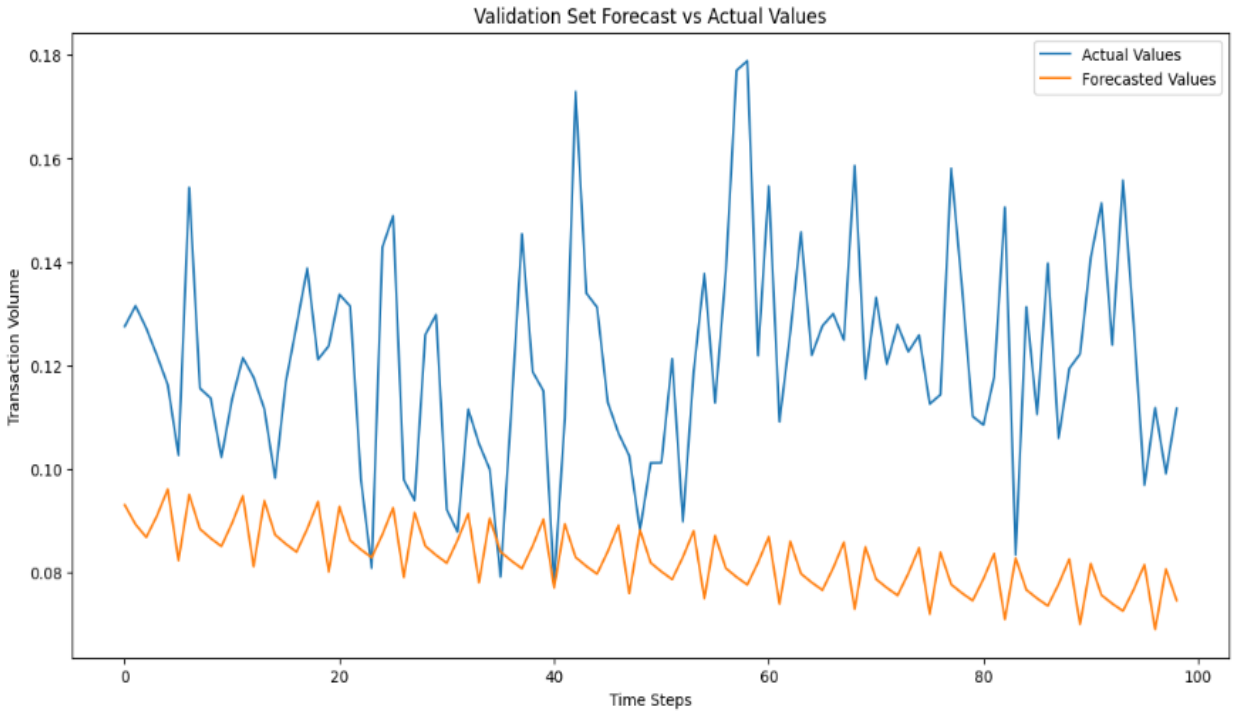

Figure 31 - ARIMA's prediction plot

- **SARIMA:**

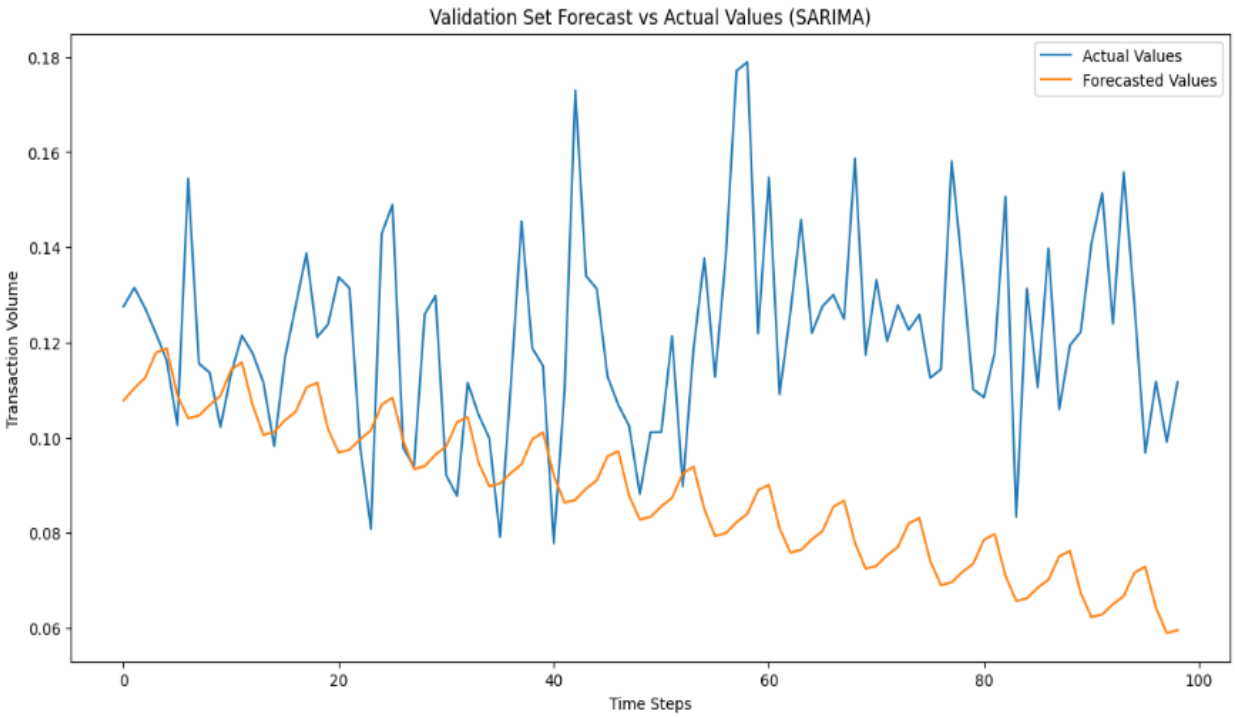

Figure 32 - SARIMA's prediction plot

- **Analysis of results:**

Based on the analysis of the ARIMA and SARIMA models shown in the plots, we can observe the following:

The **ARIMA** model shows a consistent pattern where the forecasted values are noticeably lower than the actual values across the time steps. The predictions fail to capture both the sharp fluctuations and the higher peaks of the actual transaction volumes. This indicates that while

ARIMA is able to identify some underlying trend in the data, it struggles to adapt to the high variability and noise present in the time series. The underperformance suggests that ARIMA may not be suitable for handling complex, non-linear relationships in the data.

The **SARIMA** model, which incorporates a seasonal component, shows some improvement in capturing the periodic trends in the data. However, while it captures some seasonal patterns, it still fails to accurately predict the actual transaction volume's sharp changes and peaks. The forecasted values often appear offset from the actual data, indicating that while SARIMA has better adaptability than ARIMA for seasonal data, it is still not fully capturing the intricate, non-linear patterns and volatility of the series.

Overall, both ARIMA and SARIMA show limitations in accurately modeling the data's complexity, with ARIMA not capturing the variability effectively and SARIMA not aligning well with sharp fluctuations. This reinforces the idea that traditional machine learning techniques may be better suited than deep learning and time series for this problem.

3. Definition of Evaluation Metrics:

3.1 Mean Squared Error (MSE):

The Mean Square Error (MSE) is a fundamental metric in regression analysis and machine learning used to evaluate model accuracy. It measures the average squared difference between the predicted values ($\hat{y}_i$) and actual values (y_i), and is calculated as:

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

Where n represents the total number of data points. The squaring of the errors ensures that larger errors are penalized more heavily, making MSE particularly sensitive to outliers.

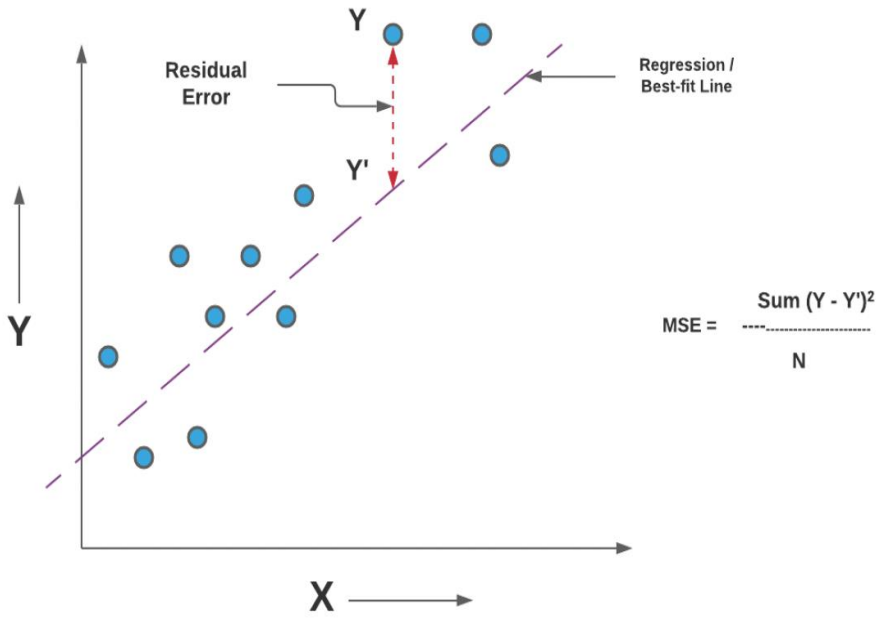

Figure 33 - MSE

MSE is frequently used in various applications:

- **Regression Models**: It evaluates the performance of models like linear and polynomial regression, with a lower MSE indicating more accurate predictions.
- **Model Selection**: When comparing multiple models, the one with the lowest MSE is often preferred as it better fits the data.
- **Feature Selection**: By comparing MSE values with different feature sets, important features contributing to prediction accuracy can be identified.

Despite its usefulness, MSE has certain limitations:

- **Sensitivity to Outliers**: Squaring the errors makes MSE highly sensitive to extreme values, which can disproportionately impact the metric.
- **Scale Dependence**: MSE depends on the scale of the target variable, making it difficult to compare MSE values across different datasets unless standardized.

This makes MSE both a powerful and nuanced tool for evaluating model performance, though it should be used with care in the presence of outliers or varying scales.

3.2 Root Mean Squared Error (RMSE):

Root Mean Square Error (RMSE) is a widely used metric for evaluating the accuracy of predictions in machine learning and regression. It quantifies how far predicted values ($\hat{y}_i$) deviate from the actual values (y_i) by measuring the Euclidean distance between them. RMSE is computed as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where:

- y_i is the true value,
- $\hat{y}_i$ is the predicted value,
- n is the total number of data points.

RMSE is popular in supervised learning applications because it uses true values to assess the accuracy of predictions. Its intuitive nature allows practitioners to have a single number that reflects model performance during training, cross-validation, or post-deployment monitoring.

Figure 34 – RMSE

While RMSE provides a comprehensive error measurement, it has limitations. Squaring the residuals makes RMSE sensitive to outliers, meaning a few poor predictions can disproportionately affect the metric. In cases where such sensitivity is undesirable, alternative metrics like Mean Absolute Error (MAE) or the median error may be more appropriate, as they minimize the impact of outliers.

3.3 Mean Absolute Error (MAE):

Mean Absolute Error (MAE) is a metric used to evaluate the performance of regression models by measuring the average magnitude of errors between predicted values ($\hat{y}_i$) and actual values (y_i). Unlike other error metrics, MAE does not consider the direction of the error (whether over or under-predicted) and focuses solely on the size of the error. It is computed as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

This formula calculates the absolute difference between each predicted value and its corresponding actual value, sums these differences, and divides by the number of data points (n) to get the average.

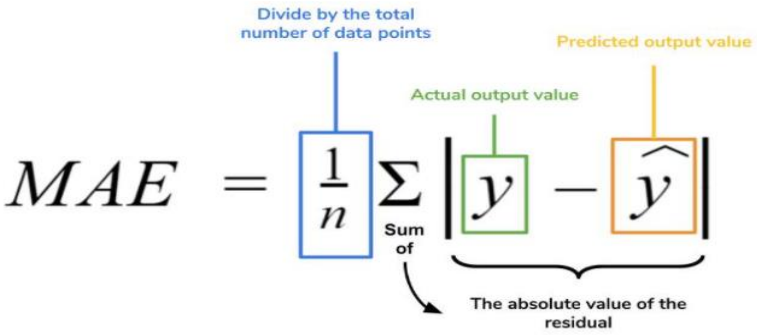

Figure 35 - MAE

- Key Benefits:

- **Resilient to Outliers**: Less affected by extreme values compared to MSE, making it suitable for datasets with outliers.
- **Linear and Intuitive**: All errors are treated equally, providing a simple average error that's easy to understand and compare.
- **Same Units**: MAE is in the same units as the target variable, offering a clear sense of error magnitude.

- Limitations:

- **Insensitive to Large Errors**: MAE doesn't penalize larger errors more, which might be important in some cases.
- **Non-Differentiable at Zero**: This complicates optimization for models that rely on gradient-based learning methods.
- **No Directionality**: MAE provides no indication of whether the model tends to overestimate or underestimate.

MAE is a widely-used, easy-to-interpret metric for regression tasks but lacks the sensitivity to large errors seen in other metrics like MSE.

3.4 R-squared (R^2):

R-squared (R^2) is a statistic that indicates how well the independent variables in a model explain the variation in the dependent variable. It ranges from 0 to 1, where 1 signifies a perfect fit and 0 means no explanatory power.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Where:

- y_i = actual value
- $\hat{y}_i$ = predicted value
- $\bar{y}$ = mean of actual values

- Steps for Calculation:

- Perform regression to find the best-fit line.
- Compute the **unexplained variance** (sum of squared residuals between actual and predicted values).
- Calculate the **total variance** (sum of squared differences between actual values and the mean).
- Subtract the ratio of unexplained variance to total variance from 1 to get R^2.

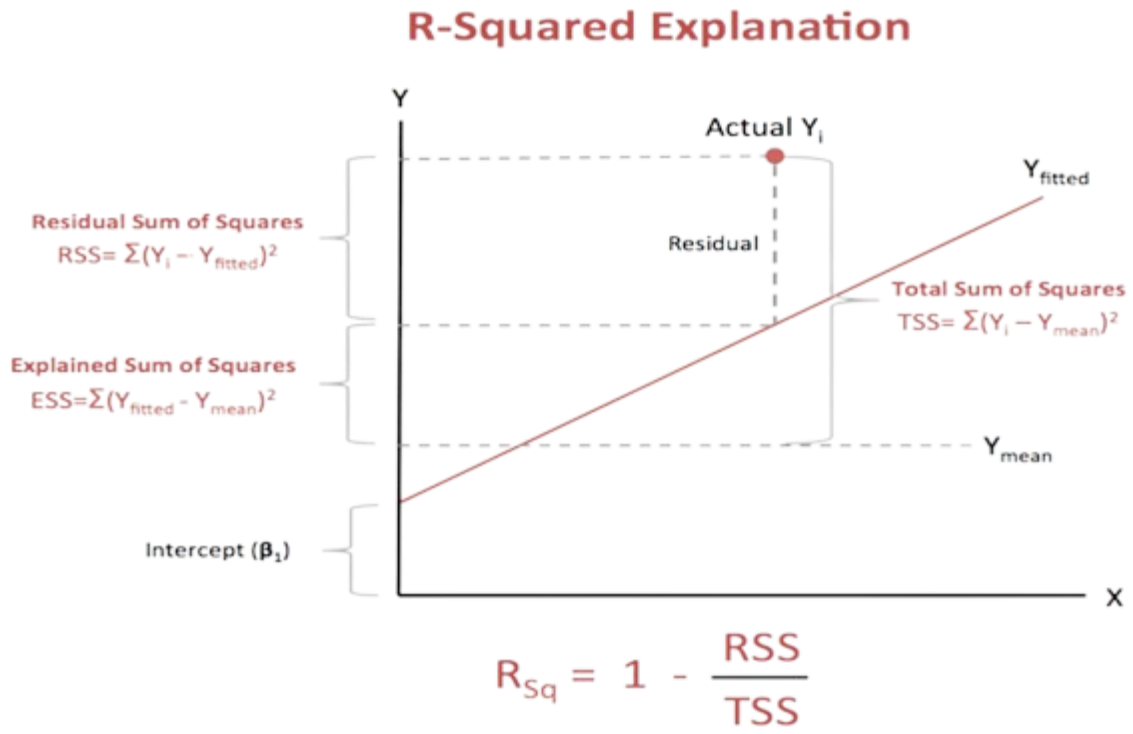

Figure 36 - R^2

- Interpretation:

- **$0 \leq R^2 \leq 1$**: A value closer to 1 means the model explains a larger portion of the variance.
- **0**: The model explains none of the variability in the dependent variable.
- **1**: The model perfectly explains all variability.

R^2 is commonly used in regression analysis and finance to explain the degree to which movements in a dependent variable can be predicted by an independent variable. For instance, an R^2 of 0.80 means 80% of the variation is explained by the model.

- Key Points:

- **Interpretation**: A high R^2 (e.g., 85-100%) suggests a strong correlation between a fund's performance and a benchmark index, while a low R^2 (below 70%) indicates weaker alignment.
- **Limitation**: A high R^2 doesn't necessarily mean a good model; it could signal overfitting.

This metric, while useful, should always be considered with other statistics for a full picture.

4. Comparison of Evaluation Metrics:

In this section, we will concentrate on comparing the machine learning models that have demonstrated robust predictive performance.

Due to the observed underfitting and poor predictions from the deep learning and time series models, which did not meet the desired accuracy and performance criteria, we will exclude them from this comparative analysis.

The focus will instead be on evaluating the machine learning models that achieved satisfactory results, based on their performance metrics, to determine the most effective approach for fuel demand forecasting.

- **The evaluation metrics:**

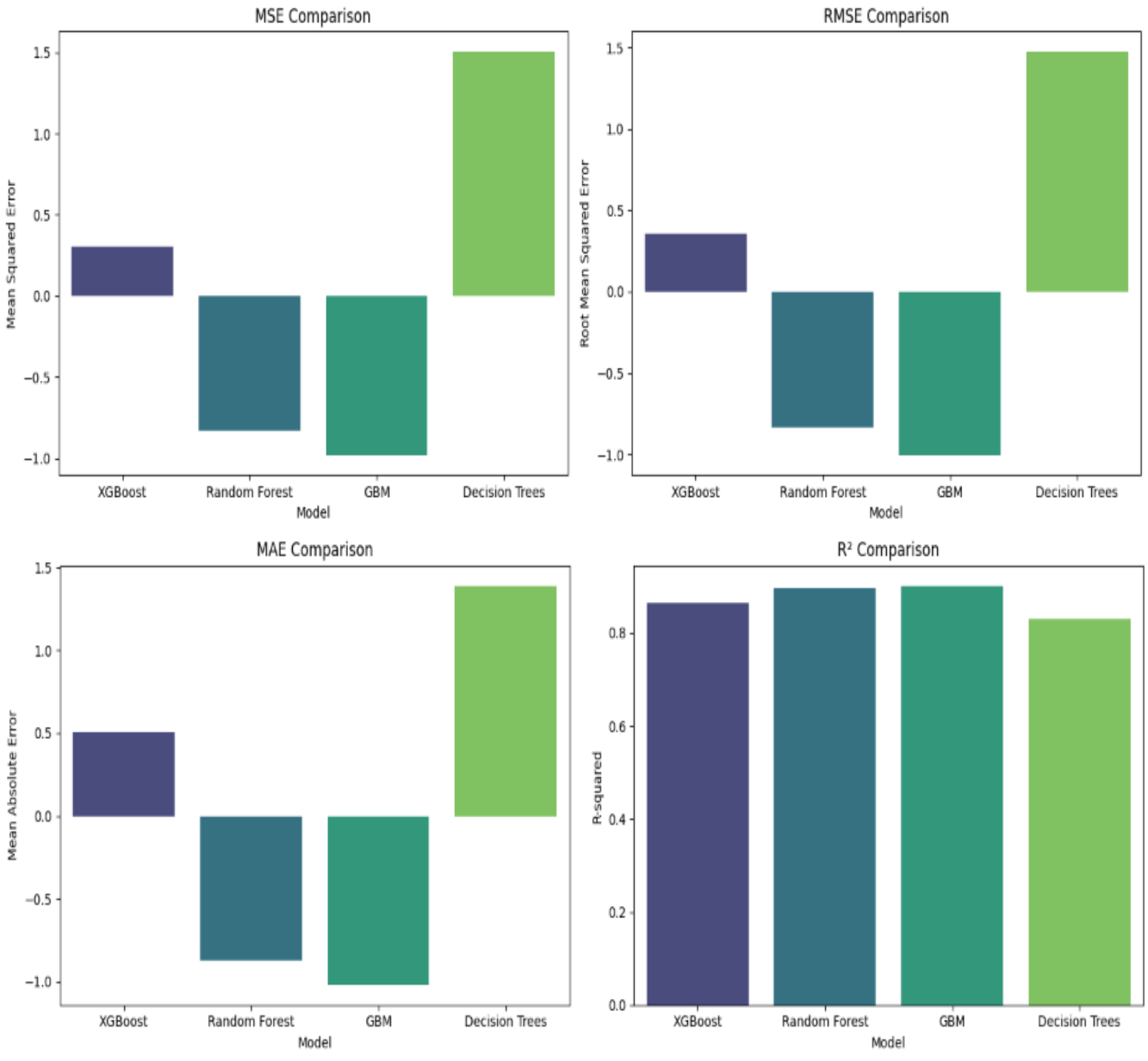

Figure 37 - Evaluation metrics comparison of ML models

- **Analysis of results:**

Before delving into the analysis of these results it must first be noted that **StandardScaler** was used, thus the values are transformed to have a mean of 0 and a standard deviation of 1. This is done using the formula:

$$z = \frac{x - \mu}{\sigma}$$

Where x is the original value, μ is the mean, and σ is the standard deviation.

- **Negative values** mean the model's metric is below the average (better performance).
- **Positive values** mean the model's metric is above the average (worse performance).

So, negative values are normal and simply indicate better-than-average performance.

When comparing the performance of the models using key metrics, each model demonstrates varying levels of accuracy, with some clearly outperforming others.

- **XGBoost**: This model has decent performance but isn't the best across all metrics.
- **Random Forest**: Performs well, with lower MSE, RMSE, and MAE compared to XGBoost and Decision Trees. It also has a solid R^2 value, indicating good predictive accuracy.
- **GBM**: This model shows the best performance, with the lowest MSE, RMSE, and MAE, and the highest R^2, suggesting it predicts the target variable more accurately than the others.
- **Decision Trees**: This model has the highest MSE, RMSE, and MAE, indicating that it is less accurate. The R^2 value is also the lowest, further showing that Decision Trees are the least preferred model in this comparison.

In summary we can say that:

- Best Model: GBM stands out as the top performer, with the lowest error metrics and highest R^2, indicating it fits the data best.

- Next Best: Random Forest also delivers strong performance across all metrics.

- Least Preferred: Decision Trees show the weakest performance, making them the least accurate model in this comparison.

5. Improving Prediction Accuracy: Advanced Methods

After making real predictions on data that the models were not trained on, good results were observed, indicating the generalizability of the models. However, to further enhance accuracy, several advanced techniques were implemented:

5.1 Average Model Prediction:

In this method, predictions from the chosen models are averaged to obtain a final prediction. The equation for this is:

$$\hat{y}_{avg} = \frac{1}{n}\sum_{i=1}^{n} \hat{y}_i$$

Where $\hat{y}_{avg}$ is the averaged prediction, n is the number of models, and $\hat{y}_i$ is the prediction from model i.

5.2 Weighted Average Model Prediction:

Similar to the average model prediction, but each model's prediction is given a weight based on its accuracy. The equation is:

$$\hat{y}_{weighted} = \sum_{i=1}^{n} w\,\hat{y}_i$$

Where w is the weight assigned to model i, and $\hat{y}_i$ is the prediction from model i.

5.3 Ensemble Method with Prediction Corrections:

This technique combines multiple models but applies corrections to the predictions based on residuals or errors from previous predictions. The final prediction adjusts for the biases of individual models.

5.4 Prediction with Min/Max Normalization:

Here, predictions are normalized between a minimum and maximum value (typically 0 and 1) to scale them, which reduces variability. The formula is:

$$\hat{y}_{norm} = \frac{\hat{y} - min(\hat{y})}{max(\hat{y}) - min(\hat{y})}$$

Where $\hat{y}_{norm}$ is the normalized prediction, and $min(\hat{y})$ and $max(\hat{y})$ represent the minimum and maximum predictions, respectively.

5.5 Best Model:

In these advanced methods, only the Random Forest and GBM models were used, and while these methods produced good results by combining predictions and improving overall accuracy, in most cases, the GBM model alone was found to be consistently providing more accurate predictions. Despite the improvements achieved through techniques like averaging or ensemble methods, the GBM often outperformed them on its own, proving to be the most reliable model for this task.

6. Conclusion:

In this chapter, we found that the **GBM** model consistently outperformed others, delivering the most accurate predictions for fuel demand. Although advanced methods like averaging and ensemble techniques improved accuracy, **GBM** alone remained the most reliable.

The next chapter will address the challenges encountered during this project and explore potential solutions.

Chapter 5: Challenges and Solutions

1. Introduction:

This chapter explores the key challenges encountered during the fuel demand forecasting project and presents possible solutions to address these issues. Despite the successful application of various models, several difficulties arose, particularly concerning data limitations, feature constraints and modeling decisions.

2. Challenges:

2.1 Data Limitations:

- **Limited Features**: The dataset used for modeling had a narrow range of features, primarily consisting of station codes, transaction volume, and time-related attributes. The lack of diverse and additional features constrained the ability to capture more complex patterns and dependencies in fuel demand.
- **Insufficient Data**: The data, although aggregated by station and day, may not have included all relevant variables that could affect fuel demand, such as weather conditions, economic factors, or promotional activities.

2.2 Feature Constraints:

- **Sparse Feature Set**: With only a few features available, such as station codes, transaction volume, and time, there was limited scope for creating advanced feature engineering. This restriction impacted the models' ability to leverage additional predictive signals.
- **Lack of External Data**: The absence of external factors or additional contextual information limited the models' capacity to account for external influences on fuel demand.

2.3 Modeling Decision Challenge:

- **Initial Model Approach**: At the project's outset, there was uncertainty about whether to predict fuel demand based on the day or by time periods (morning, afternoon, evening, night). This decision led to extensive model development based on time periods.
- **Switch to Daily Predictions**: After realizing the complexity and potential issues with time-period predictions, the approach was changed to predict by day. This shift required substantial adjustments and reworking of the models.

3. Possible Solutions:

3.1 Enhance Feature Engineering:

- **Generate New Features**: Create additional features from existing data, such as interaction terms and more detailed time-related attributes, to capture better patterns in fuel demand.
- **External Data Integration**: Incorporate external data sources like weather conditions or economic indicators to provide more context and improve predictions.

3.2 Expand Data Collection:

- **Broaden Data Scope**: Collect more granular or diverse data to include additional features affecting fuel demand, and extend the data collection period if possible.
- **Increase Data Volume**: Gather data from more stations or over a longer period to enhance the dataset's representativeness.

3.3 Refine Modeling Approach:

- **Explore Additional Models**: Experiment with more sophisticated models or techniques that can better handle limited feature sets or smaller datasets. Techniques such as transfer learning or hybrid models could be explored.
- **Use Synthetic Data**: Consider generating synthetic data to simulate additional features or scenarios. This approach can help in testing model performance under various conditions.

4. Conclusion:

To overcome the challenges faced in this project, including data limitations, feature constraints, and modeling decision complexities, it is essential to enhance feature engineering, expand data collection, and refine the modeling approach. Addressing these challenges will lead to more accurate and robust fuel demand forecasts.

Conclusion

This report has comprehensively analyzed various models for fuel demand forecasting, including XGBoost, Random Forest, GBM, and Decision Trees. Through detailed evaluation, GBM emerged as the most effective model, consistently demonstrating superior performance across key metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), as well as achieving the highest R^2 value. This indicates that GBM is particularly adept at capturing the complexities in fuel demand data and provides the most reliable predictions.

The project encountered several challenges that influenced the modeling outcomes. Initially, there was significant uncertainty regarding whether to predict fuel demand based on daily data or time periods (morning, afternoon, evening, night). The decision to shift from time-period-based predictions to daily predictions required extensive adjustments and reworking of the models, highlighting the importance of iterative validation and flexibility in model development.

Data limitations also posed a challenge. The dataset, which included only station codes, transaction volume, and time-related features, lacked the diversity needed to fully capture the factors influencing fuel demand. This limitation underscored the need for more comprehensive data and suggested that future efforts could benefit from enhanced feature engineering, expanded data collection, and exploration of additional contextual information.

The internship at AKWA Group was instrumental in improving the company's prediction capabilities and provided valuable experience in working effectively within a team and addressing challenges. This experience has significantly contributed to my professional growth and skill development.

Overall, the findings emphasize the importance of selecting the appropriate model, understanding data limitations, and being adaptable to changes. Future work should focus on addressing these challenges and implementing proposed solutions to further enhance the accuracy and effectiveness of fuel demand forecasting.

References

- Artificial Intelligence in the Power Sector
- Southern Company Subsidiaries Among the First To Use AI To Enhance Worker Safety
- Duke Energy's Hybrid Approach to AI
- Fuel Consumption Prediction Models Based on Machine Learning and Mathematical Methods
- Case Study: How Chevron Leverages AI for Operational Excellence
- Role of Artificial Intelligence in Smart Meters
- GraphCast: AI model for faster and more accurate global weather forecasting
- Google DeepMind: Deep Reinforcement Learning
- Generative artificial intelligence takes Siemens' predictive maintenance solution to the next level
- GE: How Artificial Intelligence (AI) and Machine Learning (ML) Streamlines Renewable Energy Trading
- Decision Tree (DT)
- Random Forest (RF)
- Gradient Boosting Machine (GBM)
- eXtreme Gradient Boosting (XGBoost)
- Recurrent Neural Network (RNN)
- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)
- AutoRegressive Integrated Moving Average (ARIMA)
- Seasonal Auto-Regressive Integrated Moving Average (SARIMA)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- R-squared (R^2)
- Model Averaging
- Weighted Averaging