Petrol Hourly Price Prediction Report

Lintaro Miyashin

Santiago Contreras Vera

1.Exploratory Data Analysis

As the first part of the assignment, we did exploratory data analysis on our dataset.
The following explained each step and what we did in each step.

Initial Exploration:
Upon loading the dataset, we initiated a thorough exploration phase. We observed that the dataset was complete without any missing values.

Feature Engineering:
Understanding the temporal influence on petrol prices, we extracted meaningful features from the 'date_time' column, such as 'day', 'month', 'season', 'weekday', 'weekend', 'hour', and 'am_pm'. Those features offer richer temporal granularity, potentially capturing periodic fluctuations in petrol prices. We also created a new column called 'distance_to_centroid', which calculated the distance of each data point to a central point of all the data points.

Feature Selection:
Post feature extraction, we checked the distribution of values in each column. The 'year' column had a single value, making the column meaningless. Thus, it was dropped. Furthermore, an analysis of correlation metrics in the heatmap showed a high correlation between 'weekday' and 'weekend' with the value of 0.76, suggesting potential multicollinearity. The original 'date_time' column was also dropped. After the feature selection, our dataset looked like below with new features:

self_service	manager	company	station_category	station_name	latitude	longitude	month	day	season	weekday	hour	am_pm	distance_to_centroid
1	manager_62	Esso	on urban street	station_133	22.467250	20.137507	6	10	summer	4	9	AM	4.072209
1	manager_105	Api-Ip	on urban street	station_175	22.424384	20.249712	4	14	spring	3	7	AM	9.040361
1	manager_74	Esso	on urban street	station_145	22.467639	20.141852	4	21	spring	3	8	AM	3.623727
0	manager_33	Agip Eni	on urban street	station_45	22.438954	20.126184	7	11	summer	0	16	PM	6.270802
1	manager_22	Q8	on urban street	station_67	22.444513	20.117991	5	18	spring	2	7	AM	6.702082

Outlier Analysis:
An essential step in ensuring model robustness is handling outliers. For our outlier detection, we used the IQR method and removed outliers from price column. We identified points that were lying outside of lower and upper bounds and removed them from the dataset.

Data Transformation:
With the dataset cleaned, we then separated the columns into numerical and categorical variables. Recognizing the sensitivity of machine learning models to the scale and distribution of input features, we employed standard scaling and robust scaling for numerical attributes. This method ensured our models wouldn't be influenced by features with larger scales. For the categorical attributes, one-hot encoding was introduced, transforming them into a format suitable for model training.

2. Model Setup

In the next phase of the project, we set up the foundation of the modelling process. We decided to use seven different models for this project. The models we chose were linear regression, lasso regression, elastic net regression, kernel ridge regression, gradient boosting regression, XGBoost, and lightGBM. We created two types of models, linear regression models and gradient boosting models. A brief description of each model is below:

1. Linear Regression:
Description: Models the relationship between a dependent variable and one or more independent variables using a linear equation. The goal is to find the best-fitting straight line that predicts the output value within the range.

2. Lasso Regression:
Description: A type of linear regression that includes a regularization term. The regularization term discourages overly complex models which can lead to overfitting.

3. Elastic Net Regression:
Description: Combines the properties of both Ridge and Lasso regression. It works by penalizing the model using both the L2 and L1 regularization.

4. Kernel Ridge Regression:
Description: Combines ridge regression (linear regression with L2 regularization) with the kernel trick.

5. Gradient Boosting Regression:
Description: Builds an additive model in a forward stage-wise manner. It constructs a new tree at each stage that corrects the errors of the combined ensemble of all preceding trees.

6. XGBoost:
Description: An optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable.

7. LightGBM (Light Gradient Boosting Machine):
Description: A gradient boosting framework that uses tree-based algorithms and follows leaf-wise approach with depth limitation.

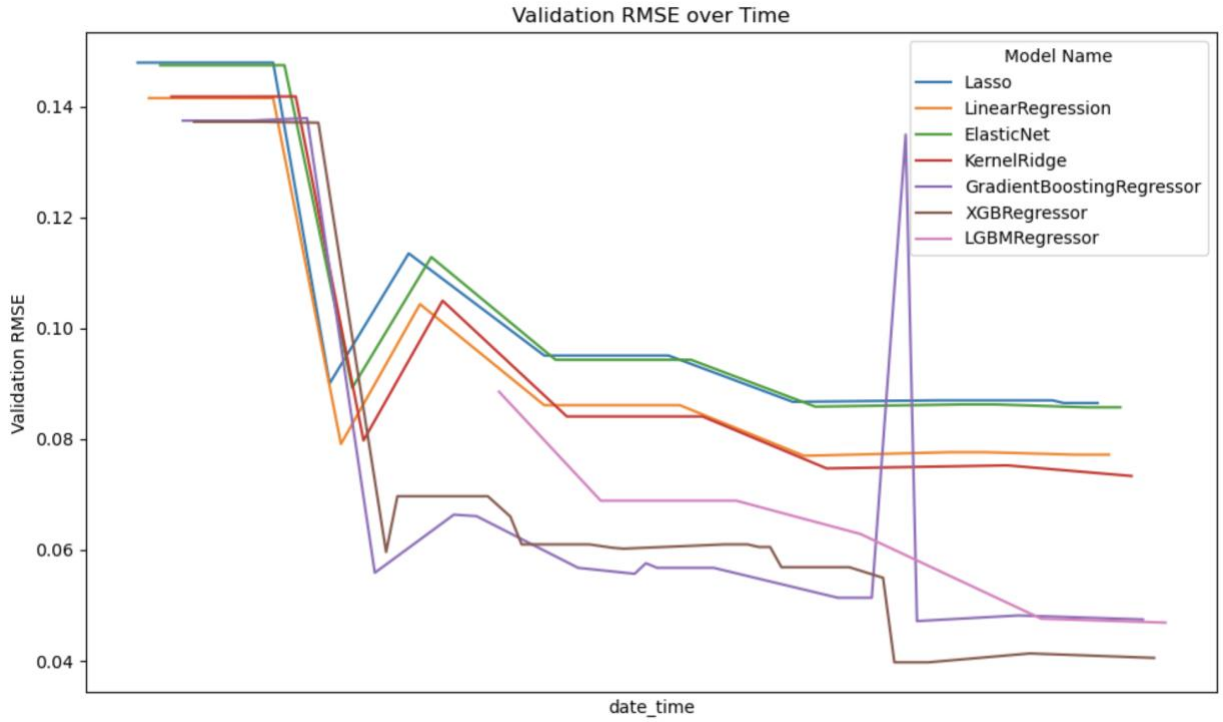
For all the models, we used cross validation and hyperparameters tuning to get the best from each model. We partitioned the data into 5 subsets to perform cross validation. For hyperparameters tuning, we introduced the method called GridSearchCV which performs an exhaustive search over a specified hyperparameter space. Each combination got tested using cross validation and the best combination was selected at the end. We used these methods to ensure that selected models are robust and optimal. We believed that combination of them two would give us the best chance of the model performing well on the validation set.

3. Regression Performance in Terms of RMSE

As the next phase of the project, we then run the regression model for each model and computed the RMSE (Root Mean Square Error).
For each model:
1. We employed the most optimal hyperparameters identified by GridSearchCV.
2. The models then made predictions on the training data.

The RMSE value served as a quantifiable metric, showing the accuracy and reliability of each model. RMSE measures the differences between predicted values and the actual values. The goal of regression task is to minimize the error between the predicted and actual value of the data. Hence, a lower RMSE denotes a superior model for regression tasks.

We stored the result of the RMSE values in the JSON file to keep them in order. Each log stored an essential information about each model and enabled us to see the progress we made throughout the time. The models made progress overtime as our models generated lower RMSE values as time went on.



The graph above depicted the RMSE values of each model on the validation set over time. The graph was generated to help us choose the best models.
For our case, XGBoosting Regressor emerged as the best model for boosting regressor and Kernel Ridge Regressor emerged as the best for linear regression models.

[model_name, train_rmse, val_rmse]
[KernelRidge', 0.0752630768611301, 0.07441234412824163]
['XGBRegressor', 0.015003836886612167, 0.03973501800150991)

The above showed the RMSE values for each model in training dataset and validation dataset. XGBoost had better RMSE numbers than Kernel Ridge Regressor in both datasets, making it the best model. Hence, we chose that model as our model for the Kaggle competition. Our best model achieved the score of 0.12320 in the Kaggle competition, making top 30 among all the groups.