ASSIGNMENT: DESIGN AND APPLICATION OF A MACHINE LEARNING SYSTEM FOR A PRACTICAL PROBLEM

Report

(Word count : 2500)

submitted as part of the assignment

in

CE802 Machine Learning

by

Ganesh Ravindran

(2106136)

School of Mathematical Sciences
University of Essex

January 2021

Introduction :

In this report, We will compare and analyse various machine learning approaches such as Cat boost classifier[1], Decision tree classifier[2] , LightGBM Classifier[3], Random forest classifier[4], Support vector machines[5], XGB Classifier[6], for classification and Gradient boosting regressor[1], Adaboost regressor[2], Random forest regressor[3], Linear regressor[4], Decision tress regressor[5], Support vector regressor[6] Which were taken in both profit ability prediction (Classification problem) and profit prediction problems (Regression problem).

Libraries :

- Numpy
- Pandas
- Seaborn
- Matplotlib
- missingno
- Sci-kit learn
- XGBoost
- Catboost
- LightGBM

Design and Architecture of the machine learning system :

The machine system was implemented following object-oriented principles. The main Mlsystem class takes care of various stages of data-preprocessing such as a function for **feature engineering**, a separate function for **feature selection**, a separate function for **feature scaling** and a function called **modelbuilding** to compare and build the best model. The modelbuilding function takes three params **imputation strategy**, **Scaling strategy** and, models which can be selected from the list. In this way, we can specify whichever model to be compared and analysed, imputation strategy to be used and scaling strategy to be used.

Project life cycle:

The machine learning procedures proposed in these comparative studies have been organized in a logical way based on complete life-cycle of a machine learning system.

- Data Collection
- Data pre-processing
- Model comparison and selecting best model
- Training the best model
- Evaluating the model using performance metrics
- Fine Tuning the model
- Prediction phase

Part 2 :

Objective : To Predict whether a new hotel opened in a given location will be profitable or not?

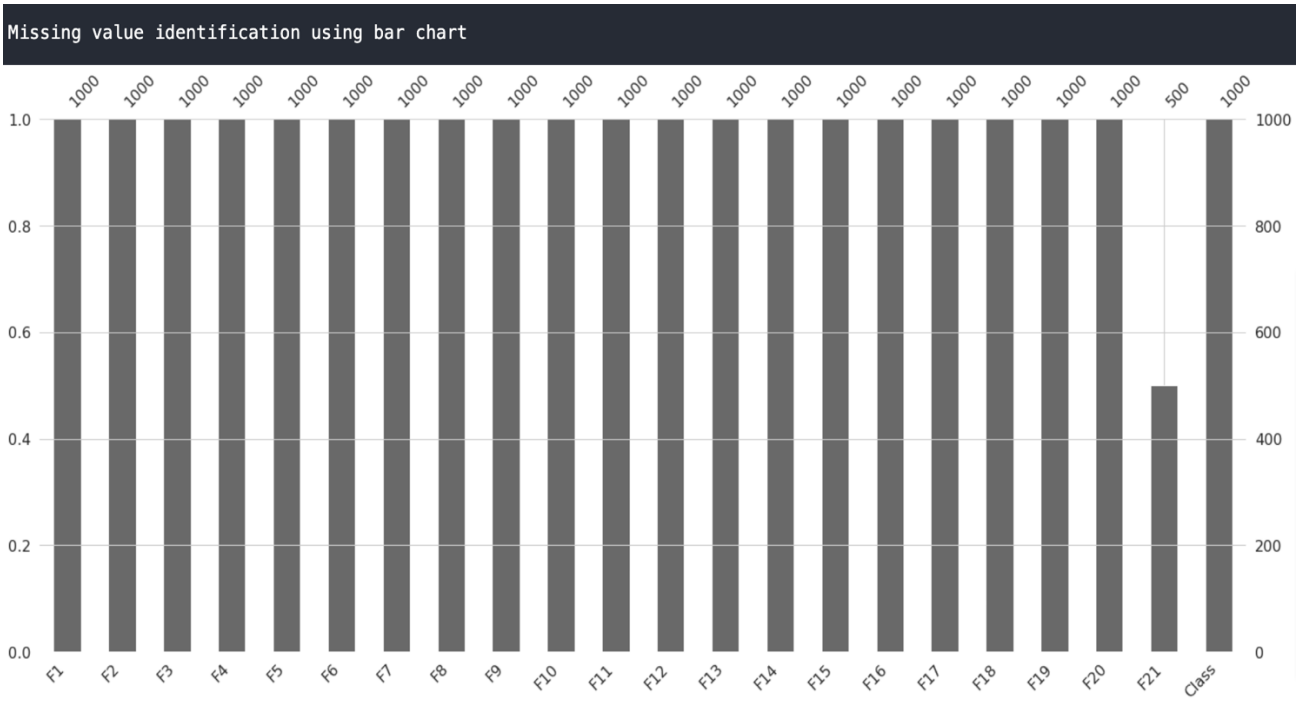
Type of the task : Binary-Classification.

Exploratory Data Analysis and Data Pre-processing :

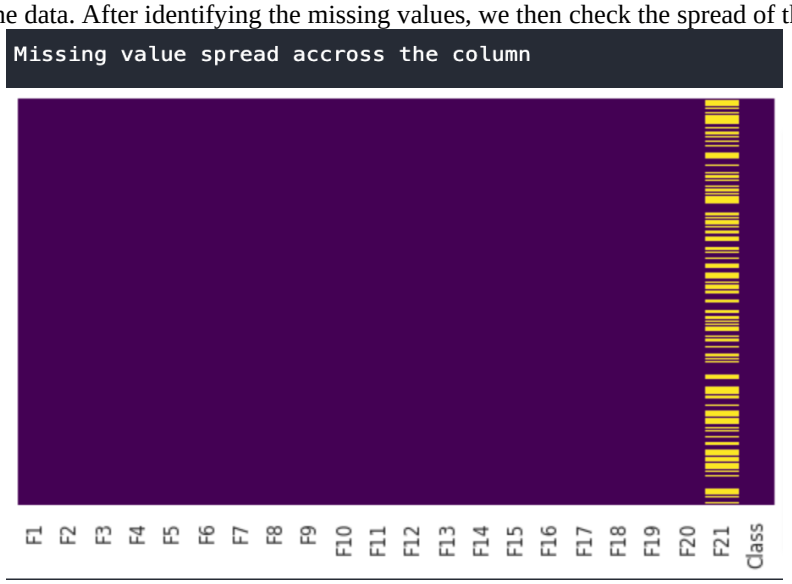
To start with, We first take a look at the descriptive statistical summary and column information of each column in the training dataset.

	count	mean	std	min	25%	50%	75%	max
F1	1000.0	0.507000	0.500201	0.000000	0.000000	1.00000	1.000000	1.00000
F2	1000.0	183.864599	20.571395	163.564660	169.944000	177.72400	189.465000	327.12000
F3	1000.0	-3924.895867	1037.177844	-8482.260000	-4325.040000	-4131.46000	-3811.385000	3847.74000
F4	1000.0	4.778608	0.901204	3.842990	4.101825	4.49725	5.147000	8.39500
F5	1000.0	-2538.698490	702.890861	-6268.740000	-2819.817500	-2677.05000	-2445.365000	2276.26000
F6	1000.0	11.563792	1.842535	9.241776	10.103100	11.16225	12.703500	19.86300
F7	1000.0	-3.004358	0.869678	-6.595000	-3.389500	-2.74905	-2.353325	-2.08222
F8	1000.0	-5.435266	0.899919	-8.862000	-5.918250	-5.14480	-4.733575	-4.46138
F9	1000.0	5161.847898	1494.941683	-4799.760000	5000.140000	5539.14000	5792.546500	13246.24000
F10	1000.0	35436.452131	5064.689411	24137.130000	35234.025000	35290.47300	35348.460000	193598.13000
F11	1000.0	-2.878000	0.998816	-3.820000	-3.820000	-3.82000	-1.820000	-1.82000
F12	1000.0	9025.612611	1058.036210	2731.260000	8615.560000	8786.95000	9097.260000	20155.26000
F13	1000.0	-3.297873	0.902763	-6.832000	-3.694250	-3.01370	-2.626475	-2.34220
F14	1000.0	0.481000	0.499889	0.000000	0.000000	0.00000	1.000000	1.00000
F15	1000.0	0.614264	0.671938	0.130001	0.173685	0.36360	0.769525	5.94800
F16	1000.0	11.735125	14.584924	-45.780000	2.510250	11.85015	20.939250	58.86000
F17	1000.0	-1931.272785	493.077434	-5111.540000	-1958.215000	-1823.60000	-1757.457500	1350.46000
F18	1000.0	-7.410040	2.560369	-18.069000	-8.505000	-6.59625	-5.517000	-4.71387
F19	1000.0	-4384.216246	1534.747192	-14823.750000	-4790.850000	-4417.08450	-3982.050000	7388.25000
F20	1000.0	-15.902875	2.708330	-26.394000	-17.140500	-15.06675	-13.822350	-12.93636
F21	500.0	-49.360560	3.188244	-59.670000	-51.585000	-49.21500	-47.160000	-39.24000

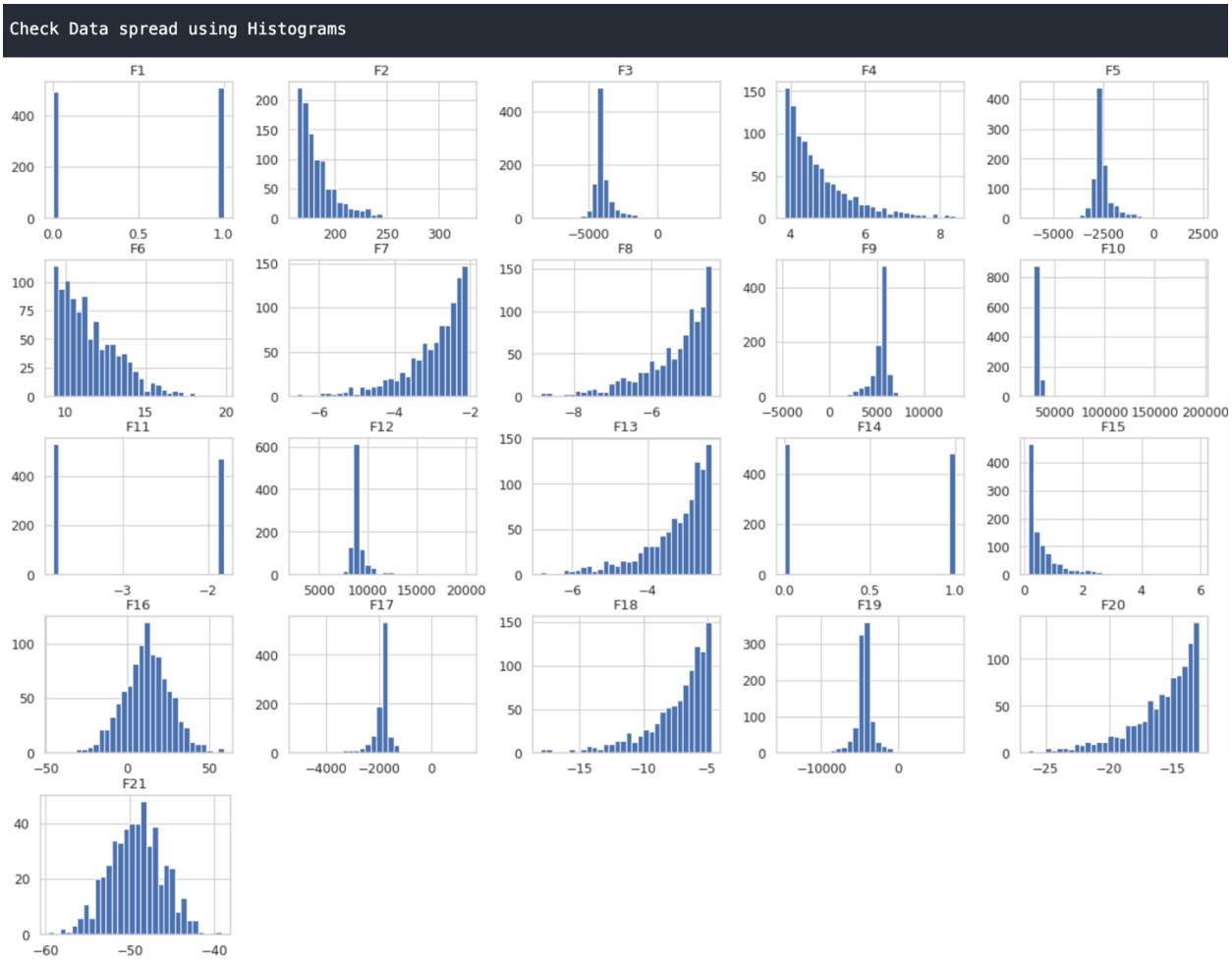
The given dataset contains 21 features and it can be clearly seen that the F21 feature has **some missing values**. We visualize the data to find the missing value frequency of each column.



The above chart depicts the missing value count of each column and it is now evident that the F21 feature has only 50% of the total size of the data. After identifying the missing values, we then check the spread of the missing value in that column.

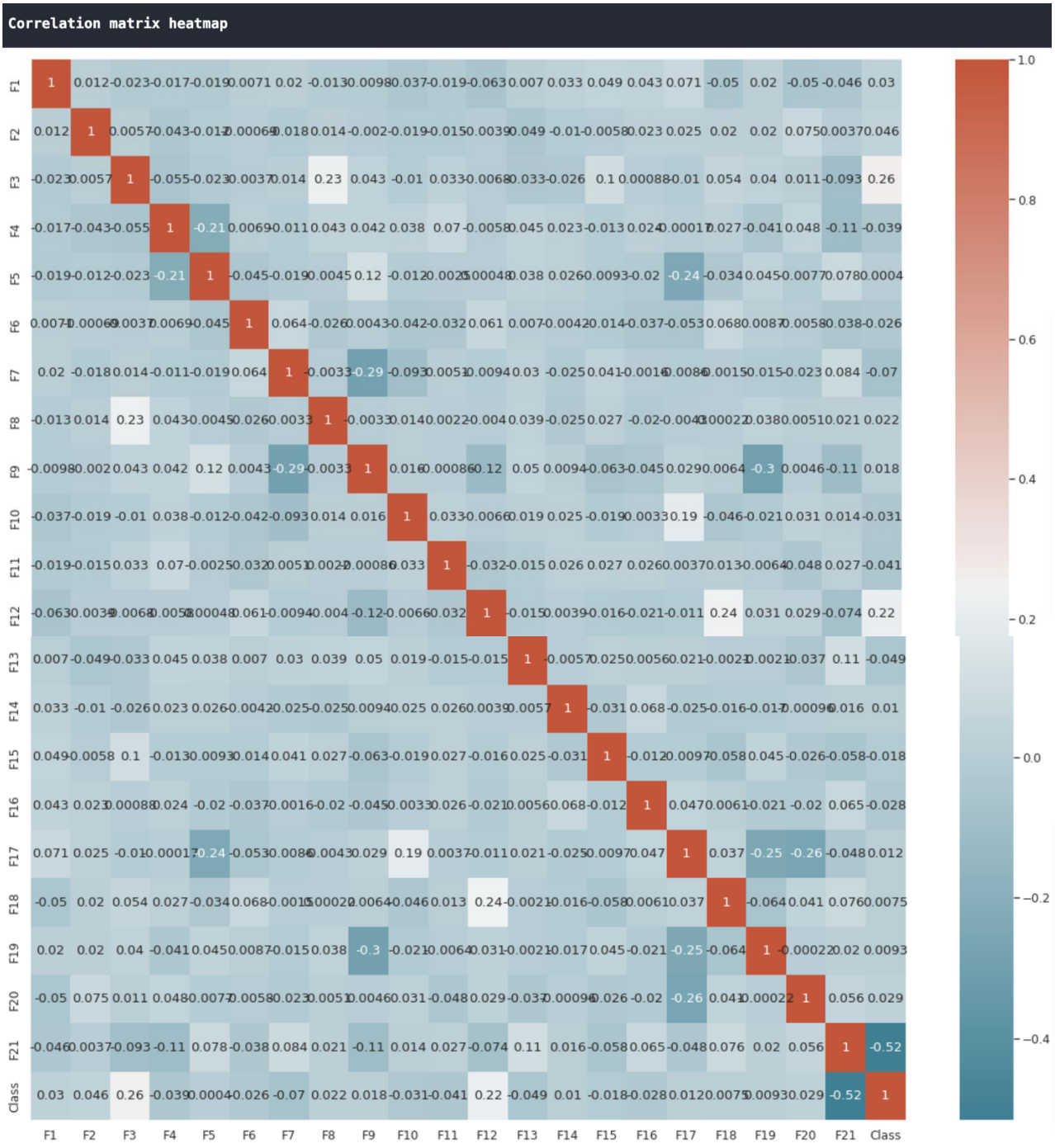


From the above chart, we can assume that the missing values of the F21 feature are of type **Missing at random (MAR)** since it is not missing structurally. After this, We then check the skewness of all the features in the dataset with the help of histograms, which will give us a better understanding if the given dataset.

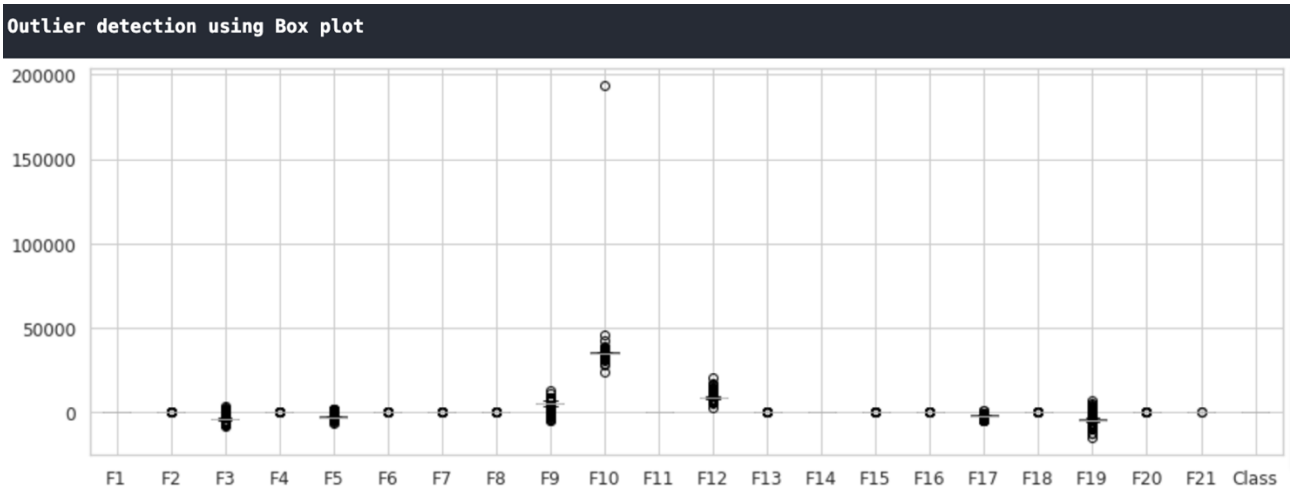


After exploring the data, **Different imputation strategies** were carried out for missing values on the feature F21 such as removing the column, replacing missing values with mean, machine learning approaches such as knn and iterative imputer all of these options were used as part of the machine learning pipeline.

For **feature selection**, we look at the correlation between all the features in the dataset. For that, a heatmap was used with the help of the seaborn library. From the below heatmap it can be seen that the features have less correlation with each other.



After this, We look for outliers in the dataset using a box plot as seen below.



The detected outliers that are above and below 4 * s.d were removed. Moving on, The target column was label encoded replacing True with 1 and false with 0. Next to that, The dataset was separated using sklearn.model_selection.train_test_split with 80% for training and 20% for testing purposes. For Feature scaling, two different approaches were taken which are standard scaler for standardisation, min-max scaler for normalising the data. Function to rescale the values before applying the training process.

- Standard scaler is prune to outliers it rescales the input values as per the given equation:

$$x' = (x - \mu) / \sigma$$

where μ denotes the mean and σ denotes the standard deviation.

- Min-Max Scaler rescale the input values as per the below equation :

$$Xsc = X – Xmin / Xmax – Xmin$$

where Xmin denotes the minimum of the column and Xmax denotes maximum value of the column

Once all these procedures are applied, The data is now prepared for experimentation and evaluation using different machine learning models.

Machine Learning model comparison (Comparative study):

Selected machine learning procedures:

- Catboost classifier
- Decision tree classifier
- eXtreme Gradient boosting classifier
- Random forest classifier
- Support vector machines
- Light Gradient boosted machines classifier

For comparing and picking the best performing model for prediction a machine learning pipeline was created and all the six selected algorithms were fed to the pipeline which was then utilized inside Grid search cv with a cross-validation fold of 5. For comparison, an option is given as models which can be used to compare different models using the same pipeline, that is, instead of comparing all the models at once, we can specify which models to compare and analyse from the list of models. After selecting which models to compare the pipeline and grid search cv compare and analyse all the algorithms given inside the grid

params and will rank all the algorithms based on the mean test score. After this, The grid search cv picks the best model from the list of models in the grid params for the evaluation and prediction phase of the machine learning system.

Before Hyper Parameter Tuning:

classifier	split1 test score	split2 test score	split3 test score	split4 test score	split5 test score	mean test score	std test score	rank
Cat Boost	0.925	0.9125	0.89375	0.93125	0.8875	0.91	0.017048	1
Light GBM	0.89375	0.89375	0.86875	0.91875	0.89375	0.89375	0.015811	2
XGB	0.88125	0.90625	0.8875	0.90625	0.8875	0.89375	0.010458	3
Random Forest	0.84375	0.8375	0.85	0.825	0.85625	0.8425	0.010753	4
Decision Tree	0.84375	0.81875	0.8125	0.85625	0.8125	0.82875	0.017941	5
Support vector mach	0.75625	0.725	0.74375	0.74375	0.7875	0.75125	0.020691	6

The above scores were obtained by the ml pipeline by comparing the models before tuning the hyperparameters. It calculates the accuracy score in each of the five splits and calculates mean test accuracy for each model then ranks them based on it. It is clear that cat boost classifier outperforms every other classifier for the given set of data. Among all models, support vector machines produced the lowest accuracy. However, Tuning the hyperparameters of each model may tell a different story.

After Hyper parameters Optimization:

classifier	split1 test score	split2 test score	split3 test score	split4 test score	split5 test score	mean test score	std test score	rank
Cat Boost	0.9125	0.93125	0.90625	0.91875	0.89375	0.9125	0.0125	1
XGB	0.88125	0.90625	0.8875	0.90625	0.8875	0.89375	0.010458	2
Random Forest	0.86875	0.88125	0.86875	0.88125	0.86875	0.87375	0.006124	3
Decision Tree	0.875	0.8875	0.88125	0.85625	0.83125	0.86625	0.020387	4
Light GBM	0.8875	0.85	0.8625	0.85625	0.86875	0.865	0.01287	5
Support vector mach	0.7625	0.7625	0.74375	0.74375	0.7875	0.76	0.016105	6

After Hyperparameter optimization, As it can be seen in the above table cat boost classifier still holds the best model position and outperforms every other model. However, Fine-tuning parameters increased the accuracy of XGB, Random forest, XGB, SVM Classifiers by a small value.

Comparison of different imputation strategies :

To deal with the null in the F21 feature, The following methods were used :

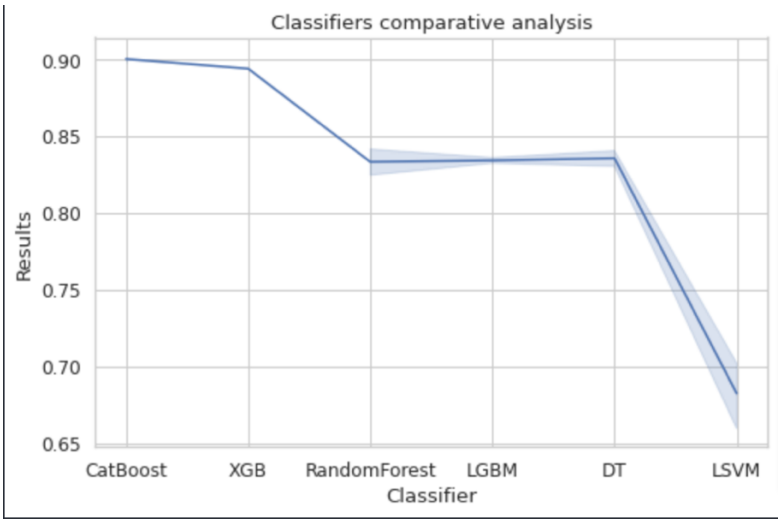
- Method 1: Remove the F21 column from the dataset
- Method 2: Replace the null values with mean imputer
- Method 3: Replace the null values using Iterative imputer
- Method 4: Replace the null values using knn imputer

Each of the above imputation strategy was applied to the data set and have summarized in a table as follows:

Method	Accuracy
missing	0.91
mean	0.91
iterative	0.91
knn	0.895

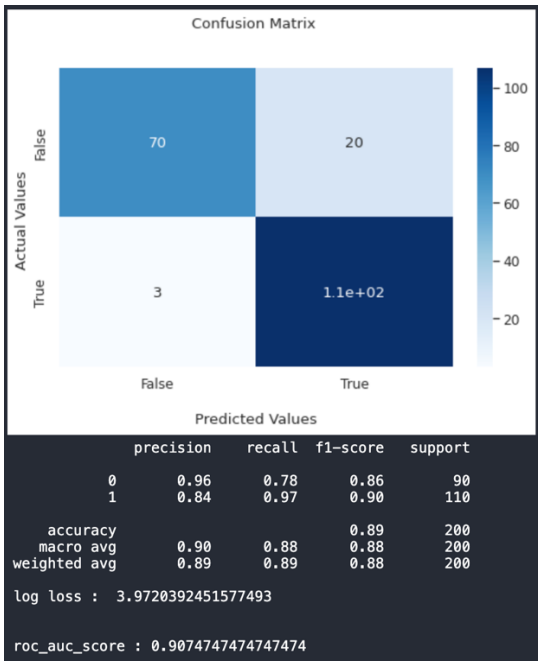
The above table clearly shows that apart from knn all other imputation strategy produces same result with a precision 2.

Discussion, Evaluation metrics and conclusion drawn:



The previous line graph clearly depicts the performance of different regression models on the given dataset. The cat boost model which was specifically built to get better accuracy and time complexity has better performance than eXtreme gradient boosting and other models that had been chosen for this problem.

Based on the above evidence the experiment was then carried on with the best model which was chosen automatically by the grid search cv pipeline that is Cat boost classifier in this case. The chosen model was then trained on training data and was used to make predictions on the test set for performance metrics. Using the best model (cat boost classifier) the below confusion matrix was generated.



	precision	recall	f1-score	support
0	0.96	0.78	0.86	90
1	0.84	0.97	0.90	110
accuracy			0.89	200
macro avg	0.90	0.88	0.88	200
weighted avg	0.89	0.89	0.88	200

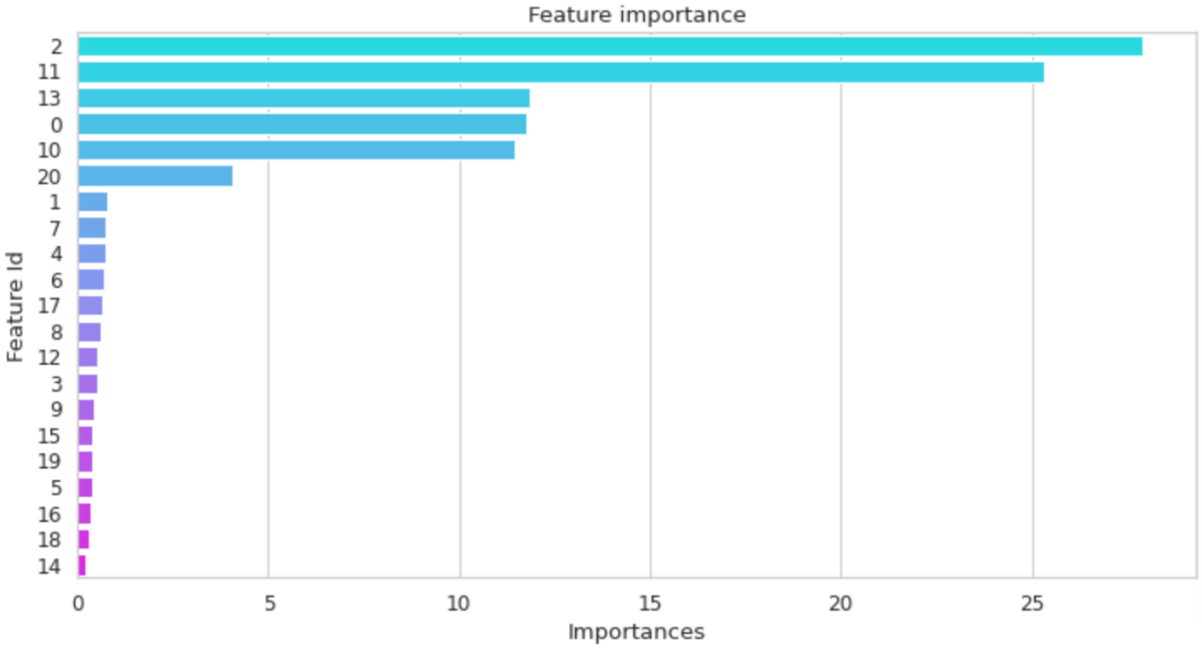
log loss : 3.9720392451577493

roc_auc_score : 0.9074747474747474

To start with, The best model (cat boost) was used for evaluation of the model using model accuracy and confusion matrix which will help to get an idea of number of True positive, true negative, False positive and False negatives. Precision and recall can be calculated using the formulas :

- Precision = TruePositives / (TruePositives + FalsePositives)
- Recall = TruePositives / (TruePositives + FalseNegatives)

Implemented using sklearn.metrics.classification_report Also log loss was used which is one of the most important classification metric based on probabilities. The cat boost classifier also produced feature importance as seen below.



Applying Cat boost classifier on test dataset produces results as follows :

- Number of profitable hotels : 522
- Number of hotels that may not be profitable : 478

After all these procedures, With conclusive evidence, Cat boost classifier with high performance and accuracy of 0.91 evidently performs better than other classification models applied on the given dataset for this predictive task.

Part **3** Comparative Study :

Part 3:

Objective : To predict the profit of the hotel based on historical data.

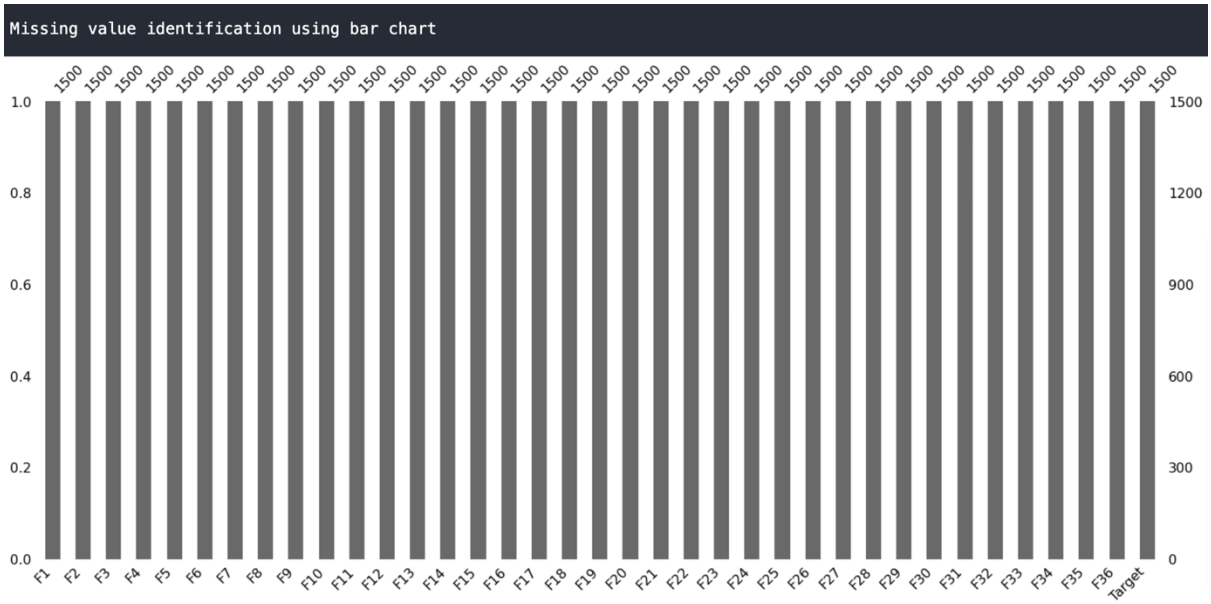
Type of the task : Regression.

Exploratory Data Analysis and Data Pre-processing:

To start with, We first take a look at the descriptive statistical summary and column information of each column in the training dataset.

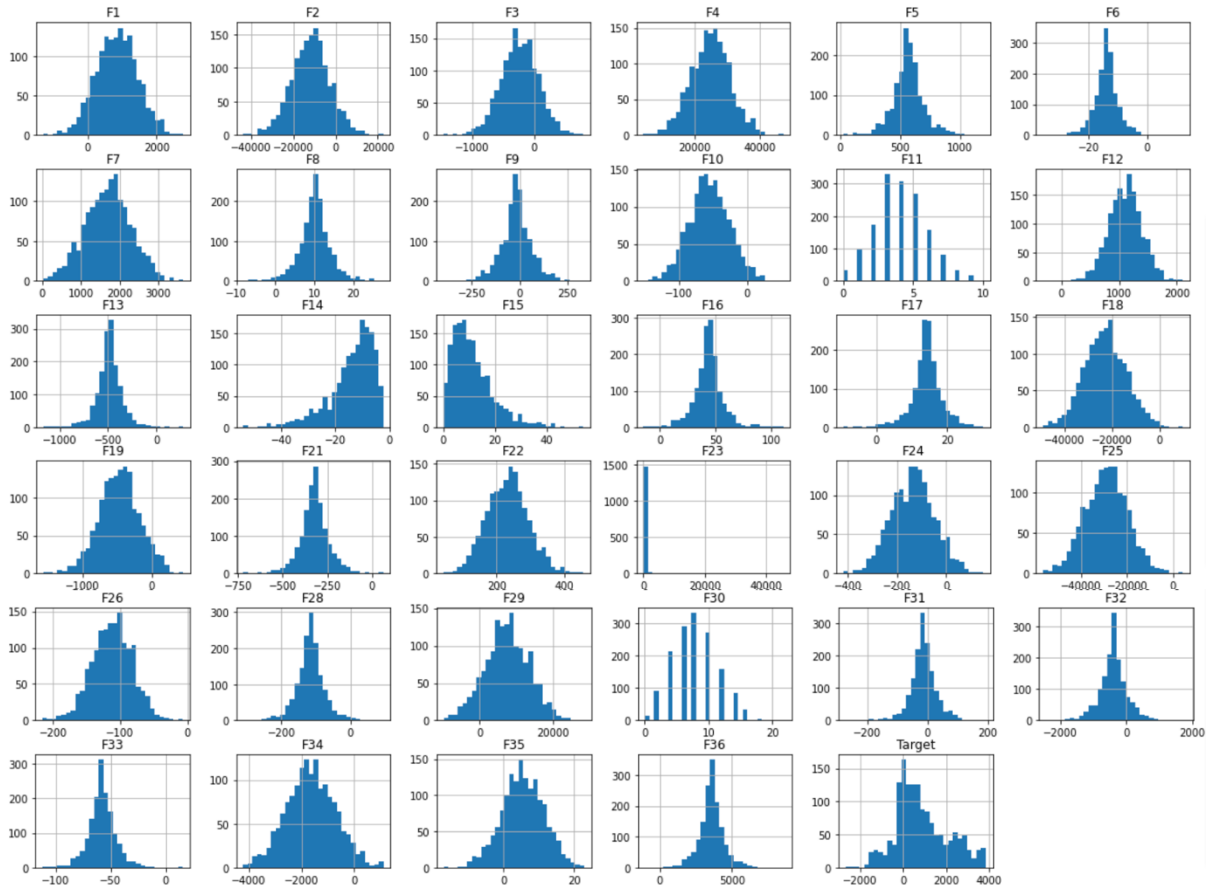
	count	mean	std	min	25%	50%	75%	max
F1	1500.0	847.011520	595.115533	-1271.54	441.4350	858.350	1249.0250	2769.66
F2	1500.0	-11736.908600	9055.633907	-43559.91	-17835.6675	-11634.240	-5991.9900	22545.33
F3	1500.0	-236.076500	299.256174	-1466.30	-430.1900	-236.735	-40.2500	789.33
F4	1500.0	24903.095547	5930.199404	4409.78	20806.3950	25064.170	28904.1000	47069.10
F5	1500.0	569.654240	129.856996	24.18	501.3000	567.705	633.5700	1194.48
F6	1500.0	-13.881793	4.227795	-35.66	-15.8875	-13.905	-11.8350	12.23
F7	1500.0	1701.127373	591.284001	34.98	1297.5900	1716.700	2095.6950	3640.98
F8	1500.0	10.154180	4.214935	-8.26	8.0500	10.180	12.1550	27.75
F9	1500.0	-15.809880	82.215588	-398.68	-57.0150	-16.330	25.9700	331.62
F10	1500.0	-55.084673	29.895706	-152.81	-75.0125	-55.550	-35.1900	53.45
F11	1500.0	3.969333	1.829378	0.00	3.0000	4.000	5.0000	10.00
F12	1500.0	1110.544520	289.526864	-315.85	915.8500	1120.685	1299.6600	2109.45
F13	1500.0	-486.879220	126.902301	-1167.00	-548.6925	-485.745	-422.9475	279.57
F14	1500.0	-13.857060	8.373293	-54.06	-17.4150	-11.880	-7.8225	-2.25
F15	1500.0	11.494480	8.019788	0.09	5.6100	9.795	15.3000	54.18
F16	1500.0	44.098320	12.997569	-14.76	37.8600	44.100	50.2200	110.85
F17	1500.0	14.209387	4.327923	-9.42	12.2700	14.310	16.3900	30.12
F18	1500.0	-22925.390960	8765.709270	-49099.50	-28746.4425	-22964.475	-17054.4150	9825.15
F19	1500.0	-470.113987	293.951559	-1566.03	-665.2425	-462.950	-284.3975	456.56
F21	1500.0	-322.213347	82.371292	-721.12	-366.5400	-322.990	-281.6700	64.38
F22	1500.0	233.007000	60.705946	42.40	190.4250	234.150	271.9400	456.84

F23	1500.0	147.677373	1407.814521	0.00	0.2800	2.170	15.9650	45417.44
F24	1500.0	-135.210200	87.979299	-419.40	-197.0475	-133.350	-77.3325	150.15
F25	1500.0	-28968.562200	9210.333946	-56492.13	-35313.8475	-28771.755	-22766.1150	4351.80
F26	1500.0	-112.058327	30.026542	-215.71	-132.0325	-111.760	-91.2000	-7.11
F28	1500.0	-116.321787	41.155295	-310.52	-137.1150	-116.610	-96.1000	95.31
F29	1500.0	7402.949867	5949.554732	-9729.68	3548.7050	7472.750	11456.3200	28498.44
F30	1500.0	7.954667	3.511623	0.00	6.0000	8.000	10.0000	22.00
F31	1500.0	-12.847560	44.360757	-271.95	-33.3575	-13.585	8.7275	192.92
F32	1500.0	-397.265347	433.264391	-2440.88	-618.3700	-388.210	-182.0650	1832.77
F33	1500.0	-57.938820	12.475389	-112.02	-64.6500	-58.260	-51.3900	15.81
F34	1500.0	-1636.515100	921.611610	-4215.09	-2263.7925	-1655.115	-1013.6475	1111.38
F35	1500.0	5.444893	5.969597	-17.04	1.5950	5.390	9.5850	22.96
F36	1500.0	3598.854547	845.754592	-1023.92	3188.5550	3611.660	4009.6750	8670.12
Target	1500.0	856.493547	1222.860406	-2685.92	17.2350	624.735	1609.1675	3836.44



The above table clearly illustrates that there is no presence of empty/null values. So we do not need to perform imputation. Then, We check for skewness in the given data using histograms.

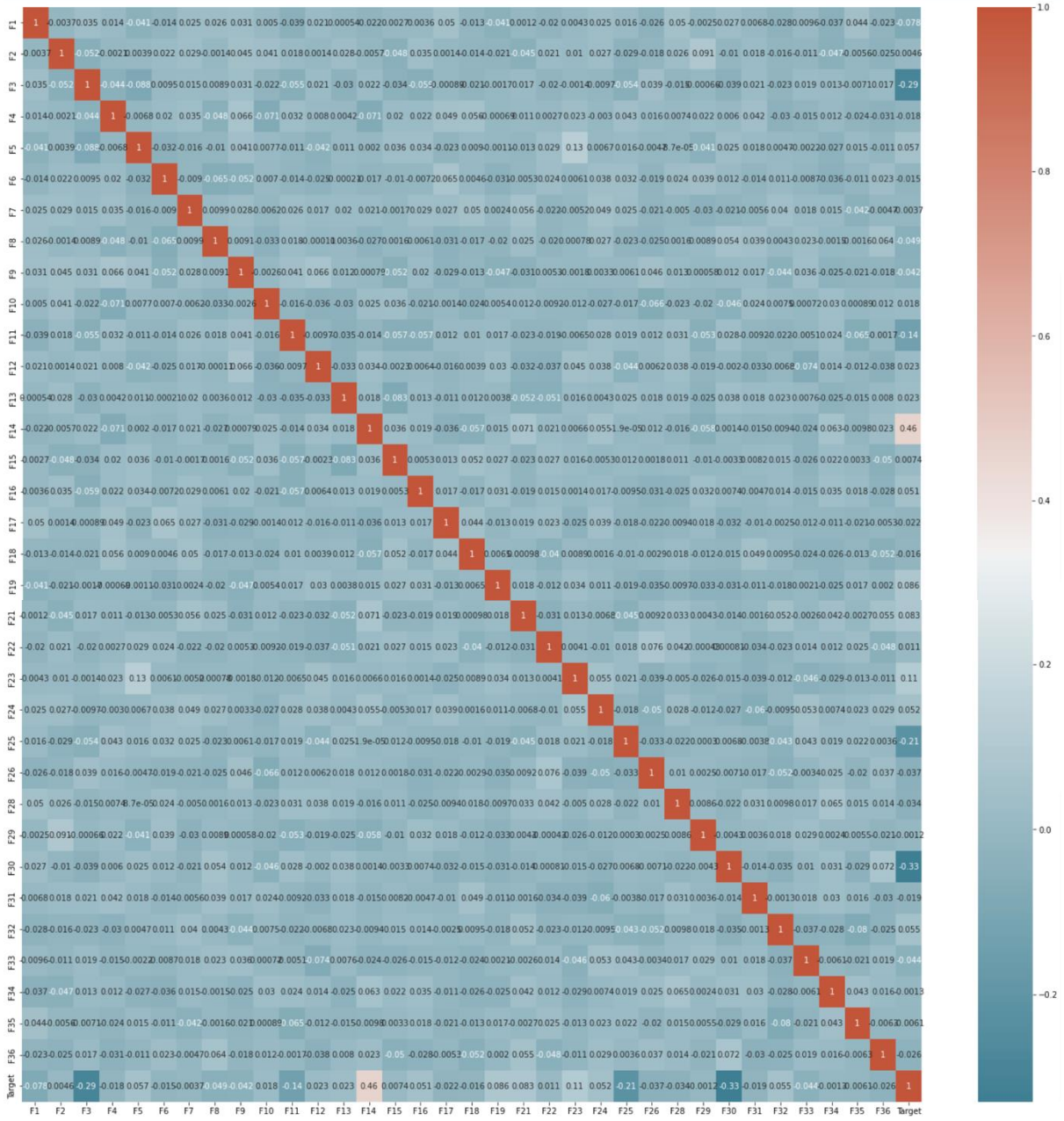
```
Check Data skewness using Bar chart
```



The features F20 and F27 need to be encoded as part of **feature engineering** since the data type of those two columns are objects. We do these by first finding which type of categorical data they are and applying the suitable encoding method. Feature F20 is an ordinal data type which is a categorical, statistical data type where the variables have natural, ordered categories and the distances between the categories are not known. So to encode F20, We use label encoding by replacing **ordinal data** with numerical values in increasing order. Then, We encode F27 using the **one-hot encoding** method in which the categorical data is removed and a new binary variable is added for each unique.

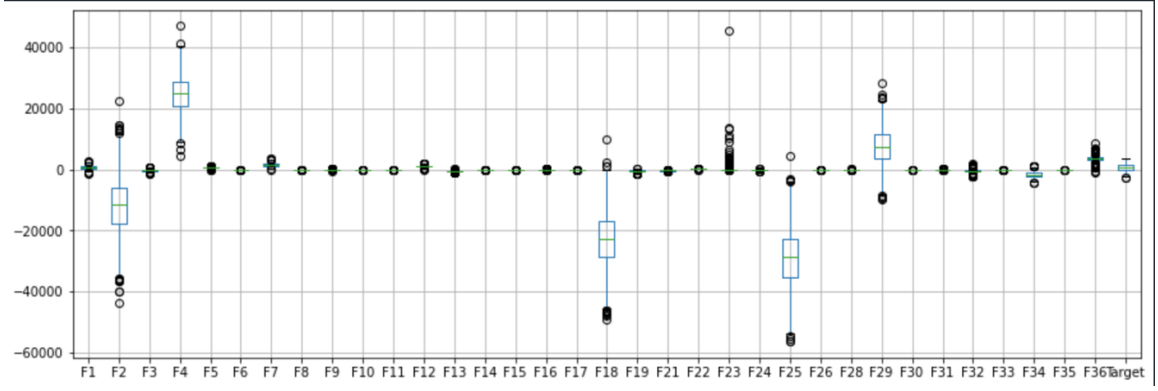
For **feature selection**, the correlation between all the features in the dataset was found using heatmap with the help of the seaborn library. From the below heatmap it can be seen that the features have less correlation with each other.

Correlation matrix heatmap



After this, We look for outliers in our dataset using a box plot as seen below:

Outlier detection using Box plot



Moving on to data splitting and **feature scaling**, The dataset was separated using sklearn.model_selection.train_test_split with 80% for training and 20% for testing purposes. For **Feature scaling**, two different approaches were taken which are standard scaler for **standardisation**, **min-max** scaler for normalising the data. **MinMax** scaler is a Function to rescale the values before applying the training process.

- Standard scaler is prune to outliers it rescales the input values as per the given equation:

$$x' = (x - \mu) / \sigma$$

where μ denotes the mean and σ denotes the standard deviation.

- Minmax Scaler rescale the input values as per the below equation :

$$Xsc = X – Xmin / Xmax – Xmin$$

where Xmin denotes the minimum of the column and Xmax denotes maximum value of the column

Once all these procedures were applied, The data is now ready for experimentation and evaluation of different machine learning models.

Machine Learning model comparison (Comparative study):

Selected machine learning procedures :

- Gradient Boosting Regressor Decision tree classifier
- Linear Regression
- Random forest Regressor
- Ada Boost Regressor
- Support vector Regressor
- Decision Tree Regressor

For comparing and picking the best performing model for prediction, A machine learning pipeline was created using sklearn.pipeline.Pipeline and all the six-candidate algorithms were fed to the pipeline which was then utilized inside Grid search cv with a cross-validation fold of 5.

For comparison, candidate models were given as grid_params which was then used to compare different models using the same pipeline that is instead of comparing all models separately, By using a pipeline we can specify which models to choose from the list of models and compare all at once. The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. After selecting which models to compare the pipeline and grid search cv compare and analyse all the models given inside the grid params and will rank all the algorithms based on the mean test score. After this, The grid search cv picks the best model from the list of models in the grid params for the evaluation and prediction phase of the machine learning system.

Before Hyper Parameter optimization :

Regressor	split1 test score	split2 test score	split3 test score	split4 test score	split5 test score	mean test score	std test score	rank
Gradient Boosting	0.818798	0.836675	0.825081	0.856749	0.818501	0.831161	0.014389	1
Ada Boost	0.702545	0.748863	0.701612	0.746303	0.692376	0.71834	0.024154	2
Random Forest	0.659965	0.715453	0.679671	0.730513	0.622433	0.681607	0.038759	3
Linear Regression	0.66717	0.699979	0.587282	0.683672	0.666638	0.660948	0.038827	4
Descision Tress	0.228542	0.182365	0.148102	0.167657	0.150999	0.175533	0.029234	5
Support vector	0.016912	0.019644	0.016953	0.014465	0.01199	0.015993	0.002587	6

The above results clearly show that Gradient boosting tops other candidate algorithms with a overall mean accuracy of 0.831161 and also in every phase of the cross-validation. While ada-boost holds the second position with a mean test score of 0.71834. The accuracy of random forest and linear regression are relatively less compared to the top-performing model. Among all the candidate models decision tree and support vector regressors produced the worst accuracies for the given data set.

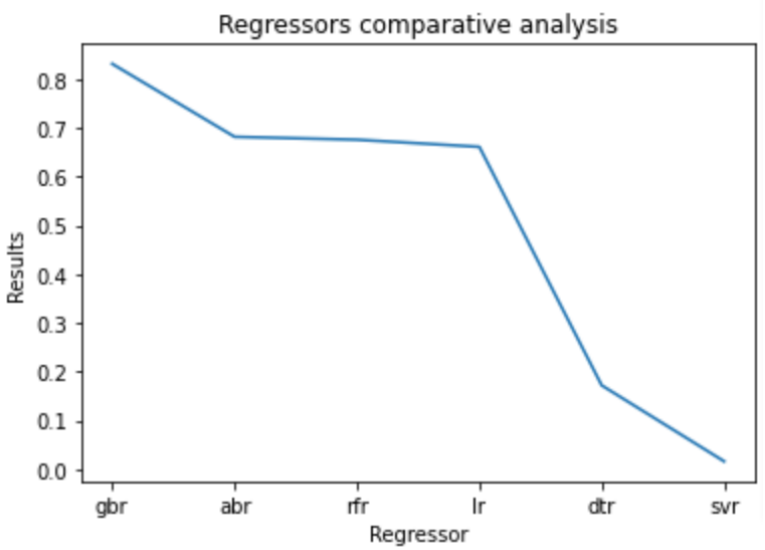
After Hyper parameter optimization :

Regressor	split1 test score	split2 test score	split3 test score	split4 test score	split5 test score	mean test score	std test score	rank
Gradient Boosting	0.83874	0.85072	0.842049	0.8742	0.825	0.8461	0.016276	1
Ada Boost	0.750849	0.71963	0.764692	0.724	0.7382	0.7395	0.017015	2
Random Forest	0.666936	0.715546	0.687706	0.7403	0.6329	0.6887	0.037355	3
Linear Regression	0.66717	0.699979	0.587282	0.68367	0.66664	0.66095	0.038827	4
Decision Tree	0.273494	0.409145	0.383751	0.4101	0.3631	0.3679	0.051708	5
support vector regressor	0.016912	0.019644	0.016953	0.01447	0.01199	0.01599	0.002587	6

Based on the results after hyper parameter optimization we can say that **Gradient boosting regressor's** performance/ mean test score is better than all the other models considered for the given data. In contrast, Support vector regressor has least accuracy on the given data after decision tree and linear regression. With 0.73 as accuracy ada boost has better accuracy than random forest which produced 0.68 as its mean test score.

It is evident that Gradient boosting outperforms all other models during every phase of the cross validation and also on mean test score.

Discussion, Evaluation metrics and Conclusion drawn:



The above line graph indicates that gradient boosting model which handles Handles missing data and with no need for imputation builds trees one at a time, where each new tree helps to correct errors made by previously trained tree. With each tree added, the model becomes even more expressive. There are typically three parameters - number of trees, depth of trees and learning rate, and the each tree built is generally shallow. It performs better and has a high accuracy compared to decision tress and support vector regressor

Based on the evidence drawn from the above metrics the ml pipeline and grid search cv will pick the model with best accuracy which in our case Gradient boosting regressor. Which is then trained using training data and validated using test set. We use metrics such as **R-squared (R2)** which is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.and **Mean Squared Error (MSE**) which measures how close a fitted line is to data points.

- With, MEAN Accuracy of 0. 8468200065353325
- Mean squared error of 30662.21
- And **Coefficient of determination / R Squared** of 0.85

Gradient boosting regressor outperform all of the other candidate models conclusively with a accuracy of 0.84 . Svm in contrast performed worst this is because choosing the kernel for SVMs can be difficult. Also, for any tree ensemble, we have the base learner splitting the solution space into a bunch of hyperrectangles. If this is close to what the underlying truth is, or even an adequate approximation, the ensemble will tend to do very well. Again, And also, kernel selection becomes hard at a higher dimension. it takes a lot less thought and effort to tweak an ensemble of trees than finding the correct kernel for each separate problem.

Applying Gradient boosting regressor to the test dataset produced better and close accurate results than other regression models considered.