University of Essex

School of Computer Science and Electronic Engineering

Assignment Report: Design and Application of a Machine Learning System for a Practical Problem

Naresh Mahendiran (MAHEN92802)

CE802 Machine Learning

Word count : 2714

INTRODUCTION

In this report, we will analyze and examine various Machine Learning(ML) algorithms for classification such as Decision trees, Bayesian Learning, k-nearest neighbors,and regression such as Linear regression, Gradient boosting, Ridge Regression, and Cat Boost.

Which were taken in both risk of developing diabetes prediction (Classification problem) and blood glucose level prediction problems (Regression problem).

Libraries

I implemented this machine learning system in this project using several third-party packages such as.

- Numpy
- Pandas
- Seaborn
- Matplotlib
- Missingno
- Sci-kit learn
- Cat Boost

Design and Architecture of the Code Base:

The implementation of this machine learning system followed object-oriented programming. Methods for preparing data, training, and assessing machine learning models are included in the primary **ModelSelector()** class. Using the **preprocess()** method, you can handle missing data in a variety of ways, including imputation or dropping an entire column of values. The **train_models()** technique applies cross-validation to train the models, and the accuracy, confusion matrix, classification report, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC) measures a performance. Then, based on accuracy, it chooses the top performing model and stores it in **self.current_model** variable. We are able to select the imputation approach to be utilized, as well as the model to compare and analyze. This expedites the process of testing and training a new model.

Project life cycle:

The proposed machine learning techniques in these comparison studies have been logically arranged based on the entire machine learning system life cycle.

- Data pre-processing
- Model training
- Model evaluation using performance metrics
- Model comparison
- Optimal model selection as well as the prediction phase

Part 2 : Classification model

Objective *: To Predict whether the person will develop diabetes or not*

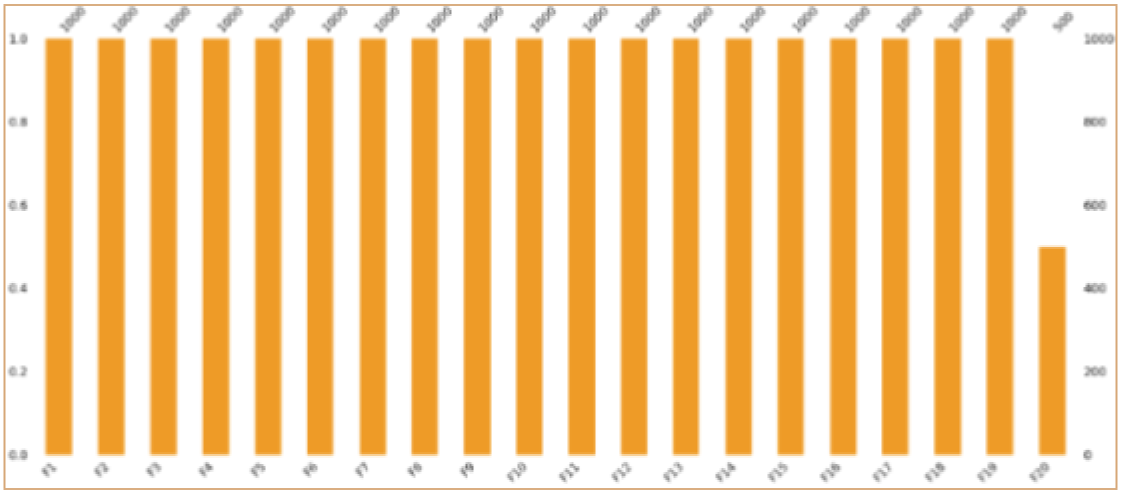
Type of the task : Binary-Classification :

Analyzing exploratory data and preprocessing it : To begin, We first examine the descriptive statistical summary and column data for each column in the training dataset.

	count	mean	std	min	25%	50%	75%	max
F1	1000.0	3.484358	0.869678	2.562220	2.833325	3.22905	3.869500	7.075000
F2	1000.0	8.460027	1.706913	6.662580	7.198000	7.91750	9.190000	15.566000
F3	1000.0	0.516000	0.499994	0.000000	0.000000	1.00000	1.000000	1.000000
F4	1000.0	2818.788916	1587.054314	-6622.740000	2203.710000	2460.79500	2926.260000	19513.260000
F5	1000.0	1.753968	0.765717	1.230002	1.278987	1.44840	1.943675	7.309000
F6	1000.0	-7264.855469	2108.672583	-18454.980000	-8108.212500	-7679.91000	-6984.855000	7180.020000
F7	1000.0	-9.151815	1.819802	-18.252000	-10.157250	-8.73705	-7.700400	-6.870168
F8	1000.0	-2081.014585	511.582397	-6005.170000	-2215.070000	-2070.05850	-1945.470000	1398.830000
F9	1000.0	-52.012299	10.285697	-123.640000	-54.812500	-48.94200	-45.052000	-41.862328
F10	1000.0	-4.318608	0.901204	-7.935000	-4.687000	-4.03725	-3.641825	-3.382990
F11	1000.0	6.635745	1.805525	4.724400	5.292950	6.06740	7.428500	13.704000
F12	1000.0	-1.467000	0.500201	-1.960000	-1.960000	-1.96000	-0.960000	-0.960000
F13	1000.0	-6635.307898	1494.941683	-14719.700000	-7266.006500	-7012.60000	-6473.600000	3326.300000
F14	1000.0	0.508000	0.500186	0.000000	0.000000	1.00000	1.000000	1.000000
F15	1000.0	-4.185798	2.699756	-14.466000	-5.634750	-3.31440	-2.080725	-1.264140
F16	1000.0	3036.192399	3111.533532	-10635.900000	1835.760000	2416.50000	3376.725000	26354.100000
F17	1000.0	12929.041422	3376.459608	5396.160000	12794.090000	12831.72200	12870.380000	118370.160000
F18	1000.0	12.632875	2.708330	9.666360	10.552350	11.79675	13.870500	23.124000
F19	1000.0	-137.867215	493.077434	-3419.600000	-311.682500	-245.54000	-110.925000	3042.400000
F20	500.0	21.840200	2.311671	15.120000	20.240000	21.80000	23.390000	28.600000

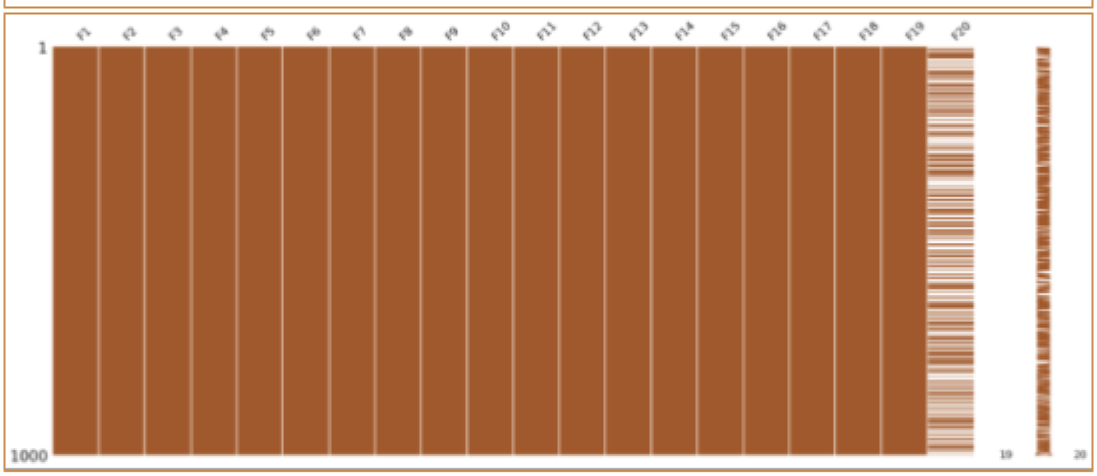
There are 20 features in the provided dataset, and it is evident that the **F20** feature has some **missing values**. To determine the frequency of each column's missing values, we visualize the data.

Missing value identification using bar chart



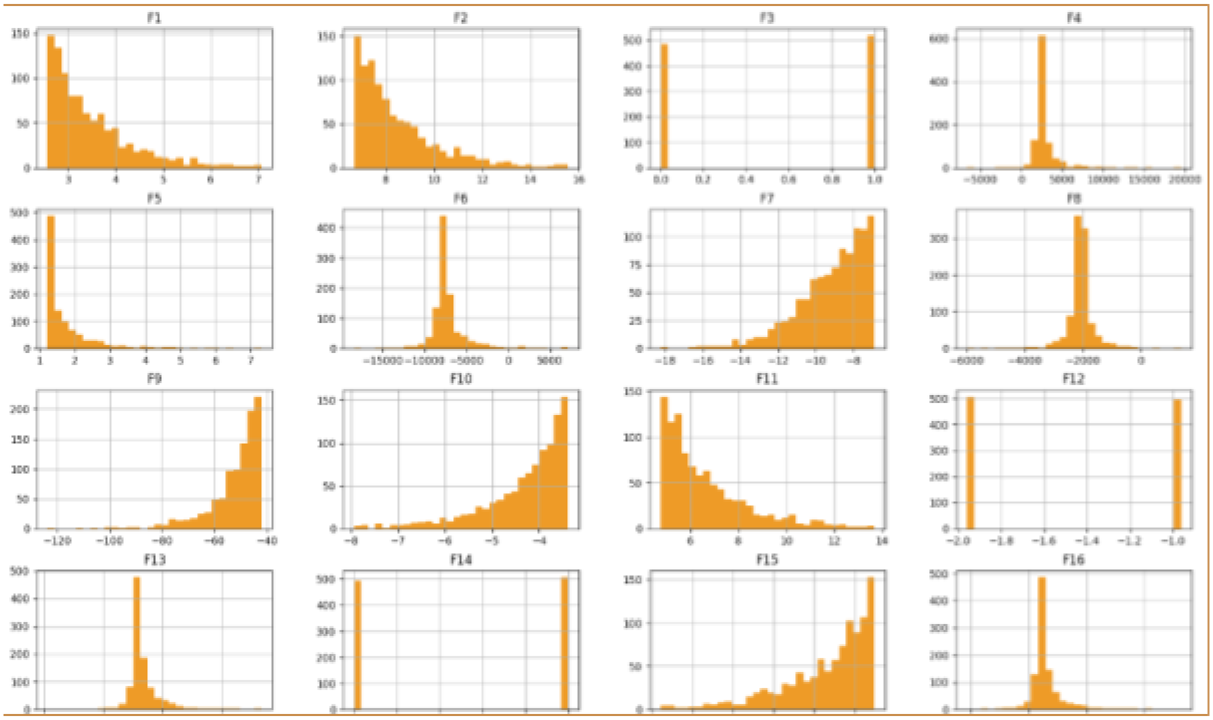
The missing value count for each column is shown in the chart above, and it is clear that only **50%** of the data's entire size is represented by the F20 feature. We examine the spread of the missing value in that column after locating the missing data.

Missing value spread across the column



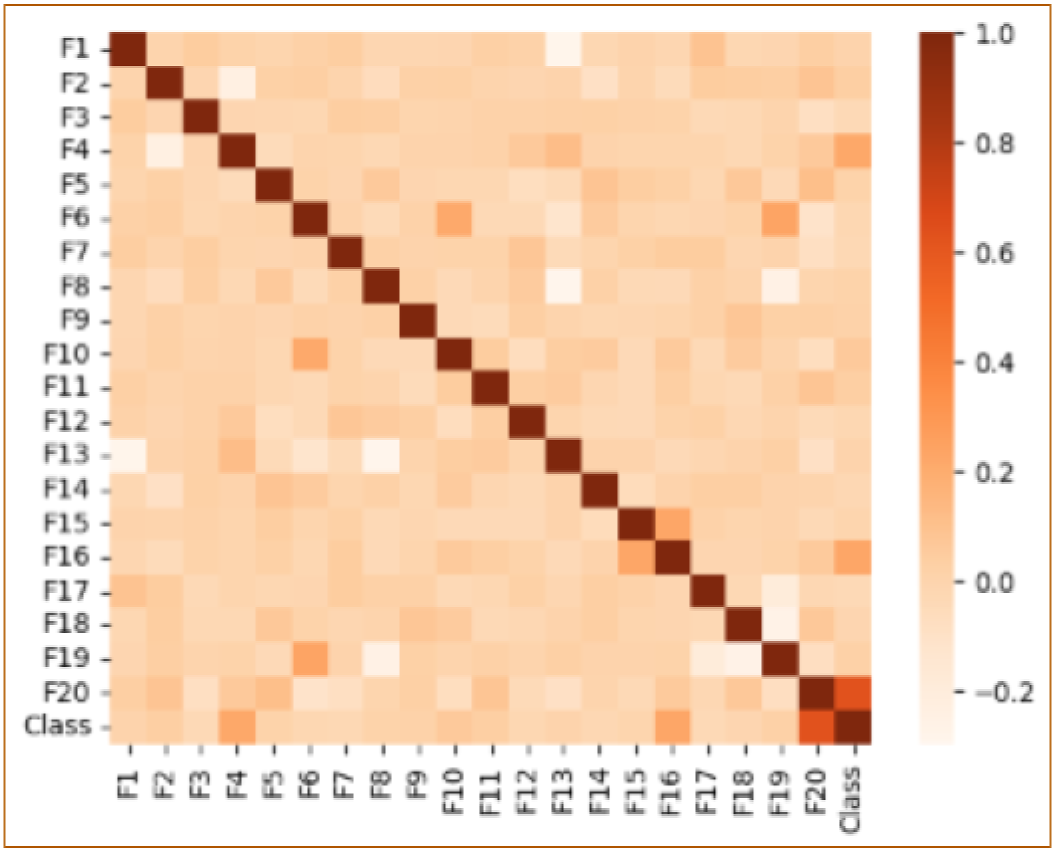
Since the missing values for the F20 feature are not structurally missing, we can infer from the above chart that they are **Missing at Random (MAR)** values. Following that, we use histograms to check the skewness of all the features in the dataset, which will help us determine whether the provided dataset is appropriate.

Data spread check using Histograms



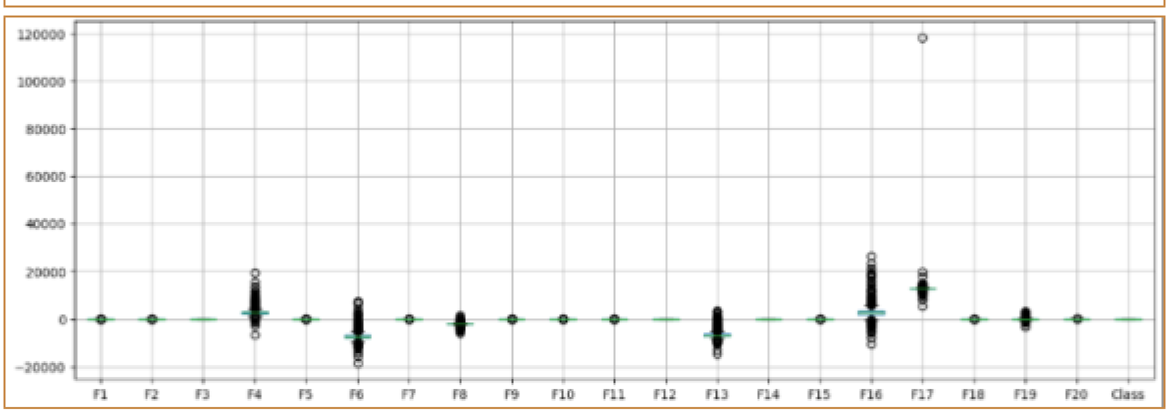
After exploring the data, various imputation strategies were used for missing values on the feature F20, including removing the column, replacing missing values with different values, and machine learning techniques like KNN and iterative imputer. We consider the correlation between each feature in the dataset while choosing a feature. A heatmap was employed for that purpose with the aid of the seaborn library. The heatmap below shows that there is less association between the features.

Correlation Matrix Heatmap



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

Outlier detection using Box plot



We have found one outlier. Next, the target column's label was encoded, with True replaced by 1 and false by 0. Additionally, the dataset was divided into **20%** for testing and **80%** for training using sklearn.model_selection.train_test_split. Standard scaler for standardization in feature scaling. Before using the training procedure, rescale the values using this function.

The input values are rescaled using the following equation by the standard scaler, which is pruned to outliers:

$$Z = x - \mu / \boldsymbol{\sigma}$$

- where µ stands for the standard deviation and for the mean.

The data is now ready for experimentation and assessment using various machine learning models after all these steps have been applied.

Machine Learning model comparison (Comparative study):

Selected machine learning procedures:

- Decision Tree Classification
- Bayesian learning
- Support vector machine
- k-nearest neighbors

A Python class called MyModelSelecter() was developed for comparing and selecting the best performing model for prediction. The class includes a number of techniques for pre-processing the data, dividing it into training and testing sets, training the models, and assessing the performance of the models. The **preprocess()** method can handle missing values in a variety of ways, such as **baseline** (dropping the enter column with missing values), **linear regression imputation**, and **KNN imputation**. Using **StandardScaler**, the **train_test_split()** method divides the data into training and testing sets. Using K-fold cross-validation, the **train_models()** method trains the models and computes a number of performance metrics, including accuracy, confusion matrix, classification report, and **ROC-AUC score**.

Preprocess(), train_test_split(), and train_models() methods are called by the **evaluate()** function, which also returns the performance metrics for each model. Additionally, based on accuracy, the method returns the model that is best suited for this machine learning system's prediction phase.

The performance of several models is finally visualized using the **plot_report()** method based on performance reports. It displays all models' precision, recall, and accuracy for each data preprocessing technique.

Comparison of several imputation techniques :

The following techniques were employed to address the null in the F20 feature :

- Method 1(Baseline): Remove the F20 column from the dataset.
- Method 2: Using linear regression Iterative imputer to replace the null values
- Method 3: Using KNN Iterative imputer to replace the null values

Why don't we use mean, medium, and mode imputation techniques?

1000 rows in the dataset have 500 missing values, which indicates that more than half of the data in that column is missing. Since they are all based on the values of the non-missing data points, using mean, median, or mode imputation in this situation could potentially introduce bias and skew the true distribution of the data.

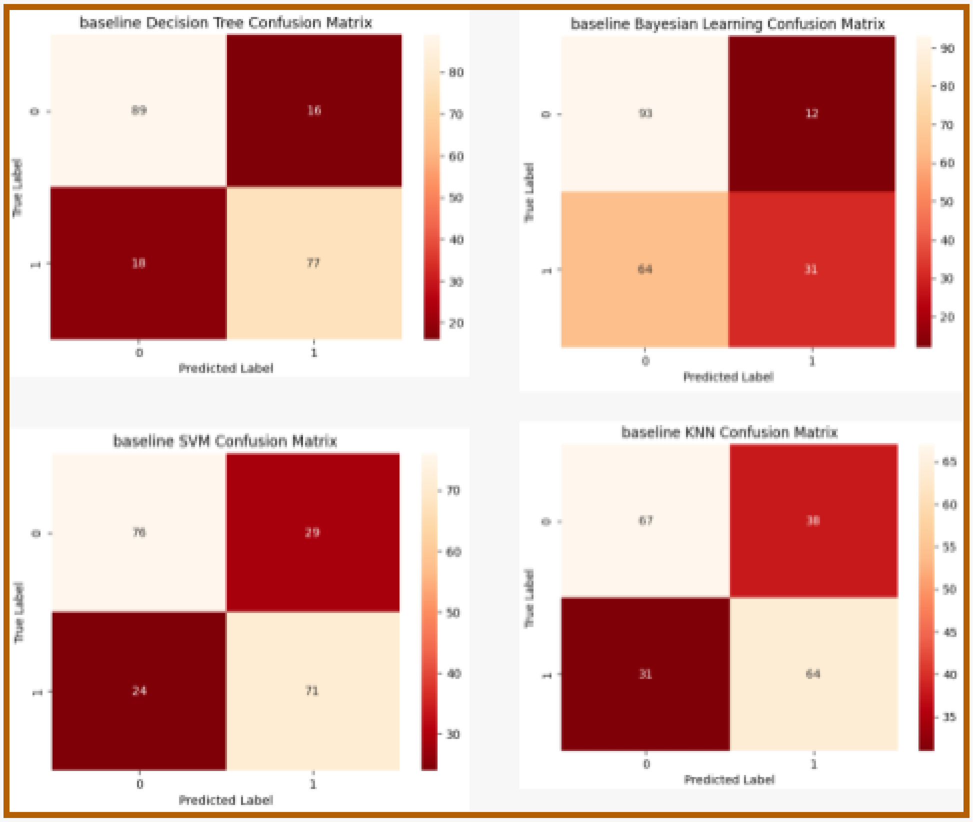
Method 1 - Remove the F20 column from the dataset :

Baseline Approach Report

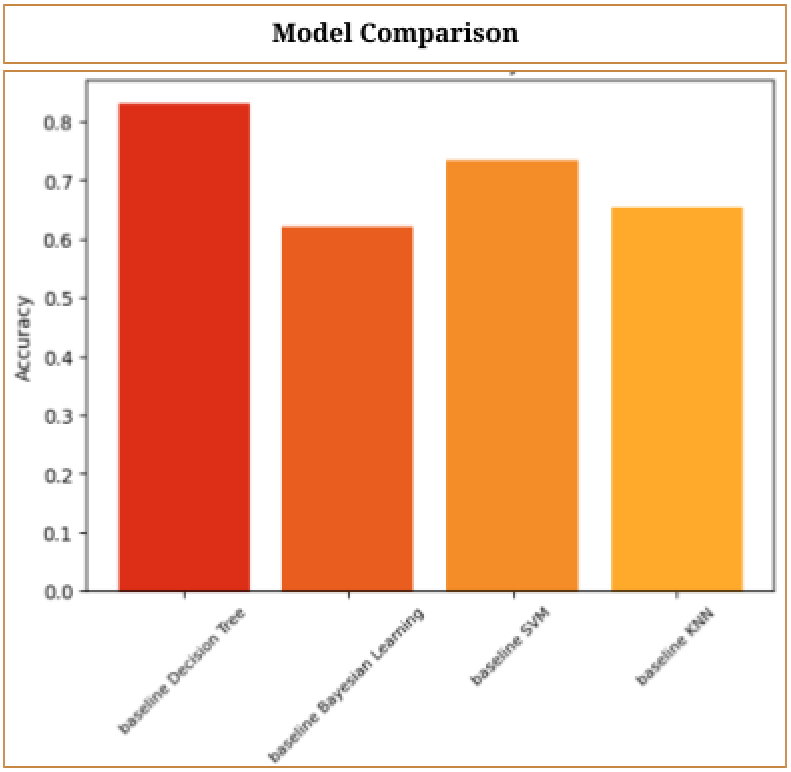
Classifier	Accuracy	Roc_Auc_Score	Cross validation	Precision		Recall		F1-Score		Support	
				True	False	True	False	True	False	True	False
Decision	0.83	0.83	0.82	0.83	0.83	0.81	0.85	0.82	0.84	95	105
Bayesian Learning	0.62	0.61	0.54	0.72	0.59	0.33	0.89	0.45	0.71	95	105
Support Vector Machine	0.73	0.74	0.68	0.71	0.76	0.75	0.72	0.73	0.74	95	105
K-nearest neighbour	0.66	0.66	0.59	0.68	0.68	0.67	0.64	0.65	0.66	95	105

The effectiveness of four ML algorithms on a binary classification. Each model comes with a classification report, ROC AUC score, cross-validation score, and accuracy score.

In the baseline method, the **decision tree** model performs well with an accuracy of **0.83** and has the highest cross validation and ROC AUC score. Of all the models, the Bayesian learning model performs the least well. Despite having similar other metrics, the SVM model is more accurate than the KNN model.



The confusion matrices demonstrate how effectively the Decision Tree model classifies both positive and negative cases. The SVM model tends to **misclassify negative** cases as positive, whereas the Bayesian Learning approach tends to **misclassify positive** situations as negative. The KNN model is more likely to incorrectly classify situations. When assessing methods of classification, it's crucial to take accuracy and the confusion matrix into account.



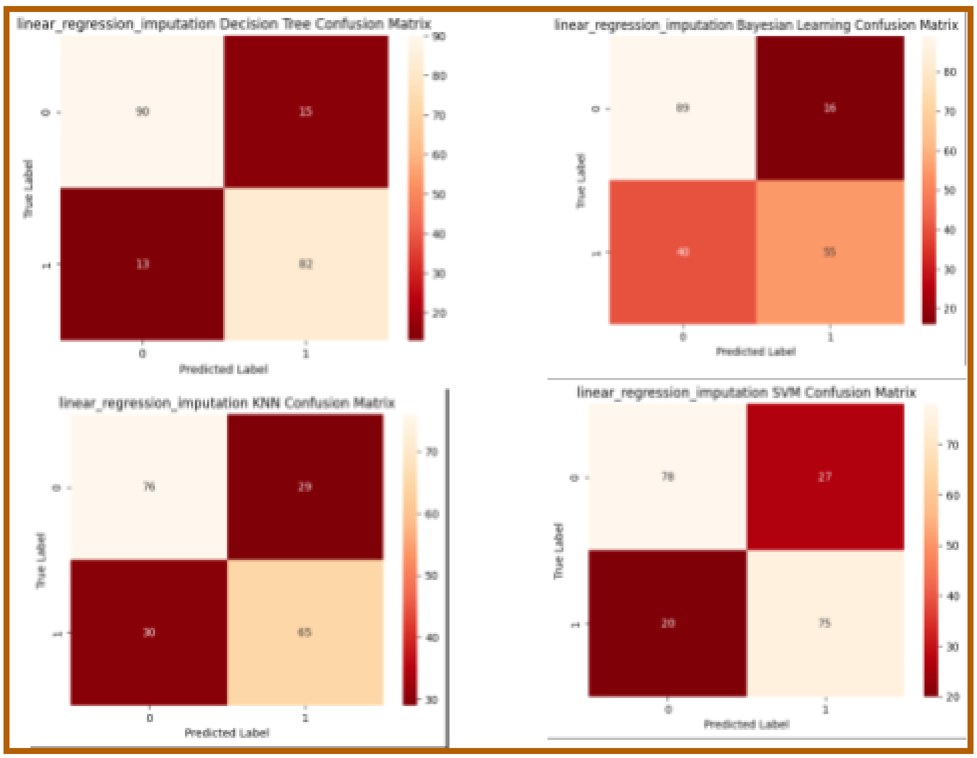
Overall, it appears that the **decision tree** model is the best-performing with **83%** accuracy in this method.

Method 2 - Using linear regression Iterative imputer to replace the null values :

Linear Regression Imputation Report

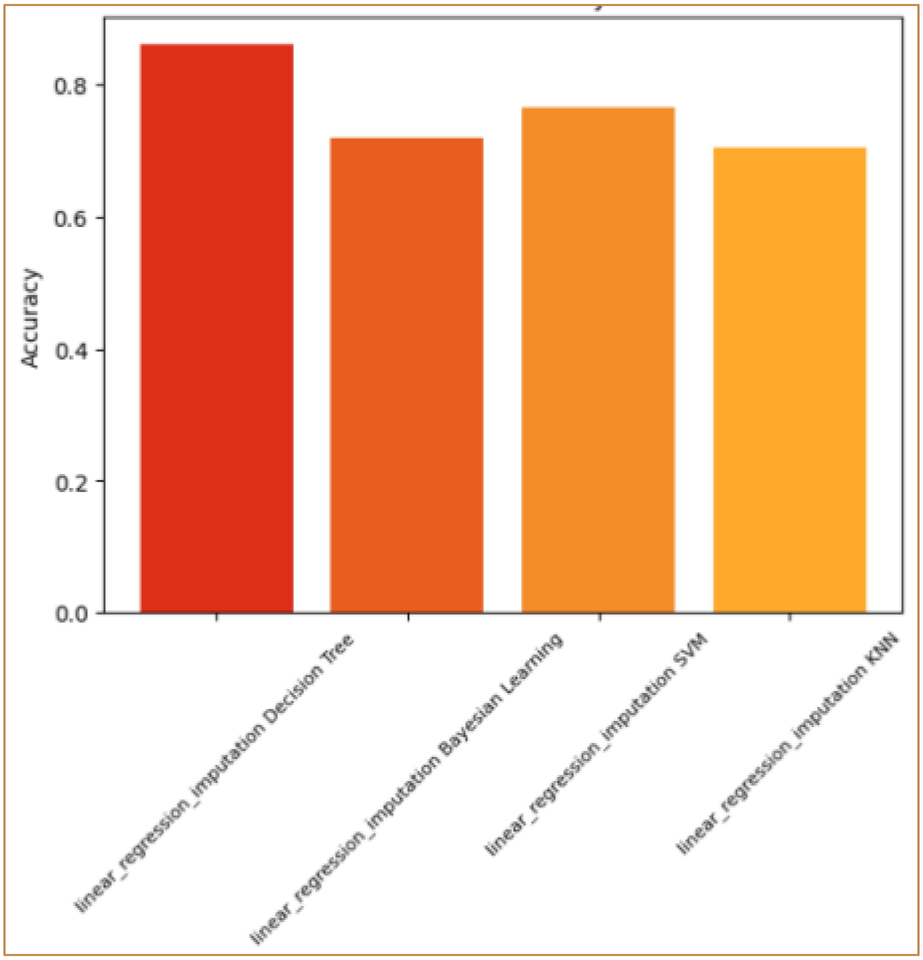
Classifier	Accuracy	Roc_Auc_Score	Cross validation	Precision		Recall		F1-Score		Support	
				True	False	True	False	True	False	True	False
Decision	0.86	0.86	0.85	0.85	0.87	0.86	0.86	0.85	0.87	95	105
Bayesian Learning	0.72	0.71	0.63	0.77	0.69	0.58	0.85	0.66	0.76	95	105
Support Vector Machine	0.77	0.77	0.72	0.74	0.80	0.79	0.74	0.76	0.77	95	105
K-nearest neighbour	0.70	0.70	0.64	0.68	0.72	0.68	0.72	0.69	0.72	95	105

Utilizing the linear regression imputation method to handle missing values, four classification models were assessed. Following the Decision Tree in terms of accuracy and roc_auc_score were the SVM, Bayesian Learning, and KNN models. Additionally, **Decision Tree had the highest cross-validation accuracy**, while Bayesian Learning had the lowest. Overall, Decision Tree performed better than the other models, with **Bayesian Learning performing the worst**.



The Decision Tree model has the most **true positives and true negatives**, whereas Bayesian Learning has the **highest false negatives**, according to the confusion matrices. SVM produced more erroneous positives, while KNN produced more false positives and false negatives. In terms of appropriately identifying both positive and negative cases, Decision Tree did the **best overall**. However, when assessing the effectiveness of classification models, it's crucial to take into account both accuracy and the confusion matrix.

Model Comparison



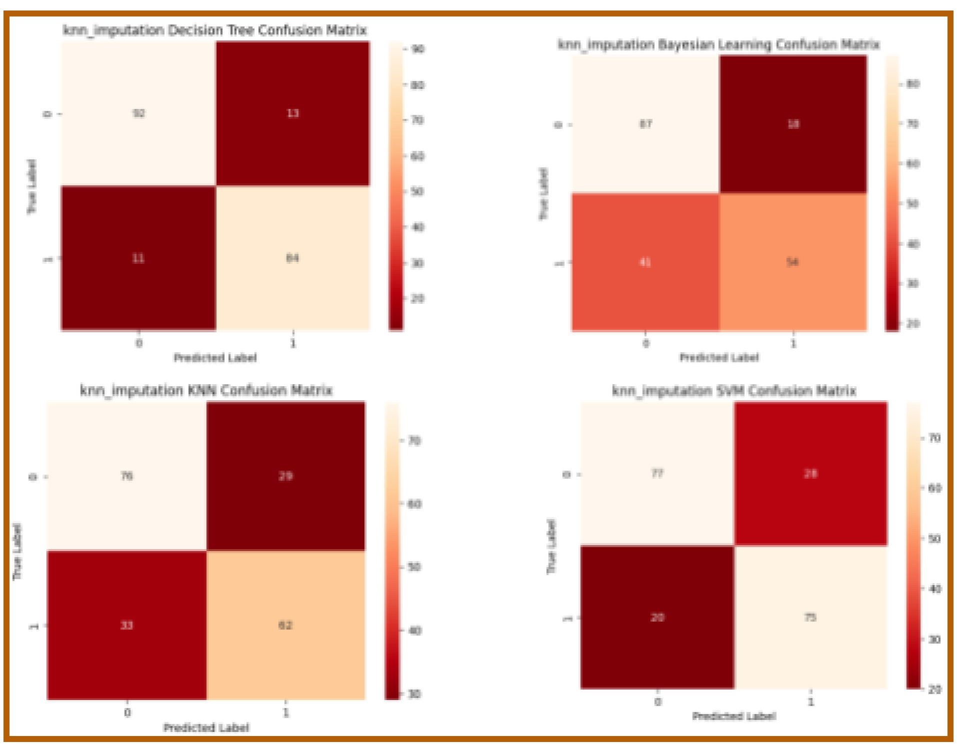
Based on these findings, the **Decision Tree classifier** seems to be the most promising with **86% accuracy**, although additional research and testing could be required to find the model that works best with this dataset.

Method 3 - Using KNN Iterative imputer to replace the null values :

KNN Imputation Report

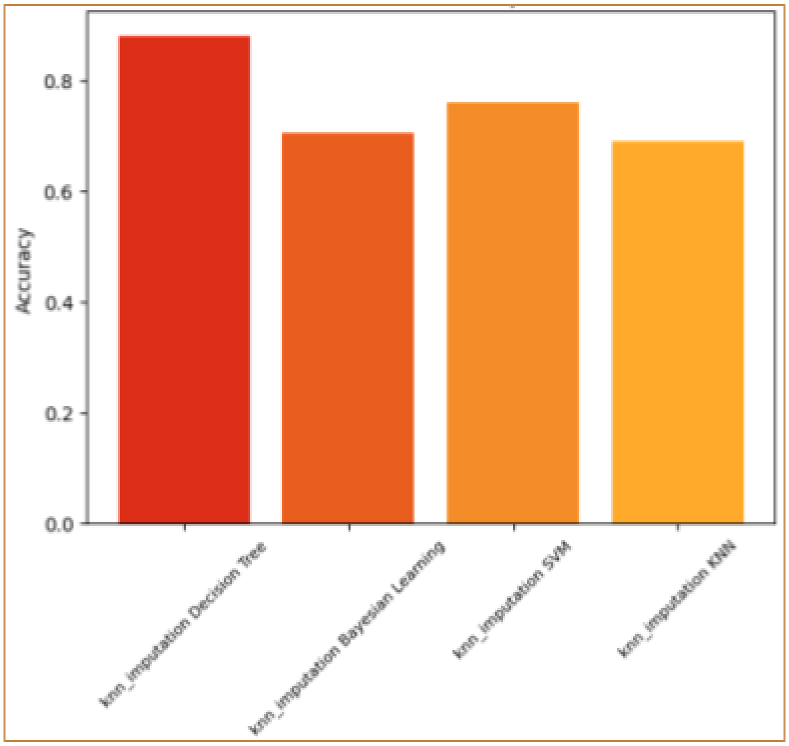
Classifier	Accuracy	Roc_Auc_Score	Cross validation	Precision		Recall		F1-Score		Support	
				True	False	True	False	True	False	True	False
Decision	0.88	0.88	0.83	0.87	0.89	0.88	0.88	0.87	0.88	95	105
Bayesian Learning	0.70	0.70	0.63	0.75	0.68	0.57	0.83	0.65	0.75	95	105
Support Vector Machine	0.76	0.76	0.71	0.73	0.79	0.79	0.73	0.76	0.76	95	105
K-nearest neighbour	0.69	0.69	0.63	0.68	0.70	0.65	0.72	0.67	0.71	95	105

These reports include classification reports for four models (Decision Tree, Bayesian Learning, SVM, and KNN) that were trained using KNN imputed data, as well as accuracy, ROC AUC scores, cross-validation accuracy, and classification reports. The model with the highest accuracy and ROC AUC score was the **Decision Tree model (0.88)**. The cross-validation accuracy and accuracy for the Bayesian Learning model were both 0.63 and 0.70, respectively.



Looking at the matrices, the **Decision Tree model performed the best**, followed by Bayesian Learning, SVM, and KNN.

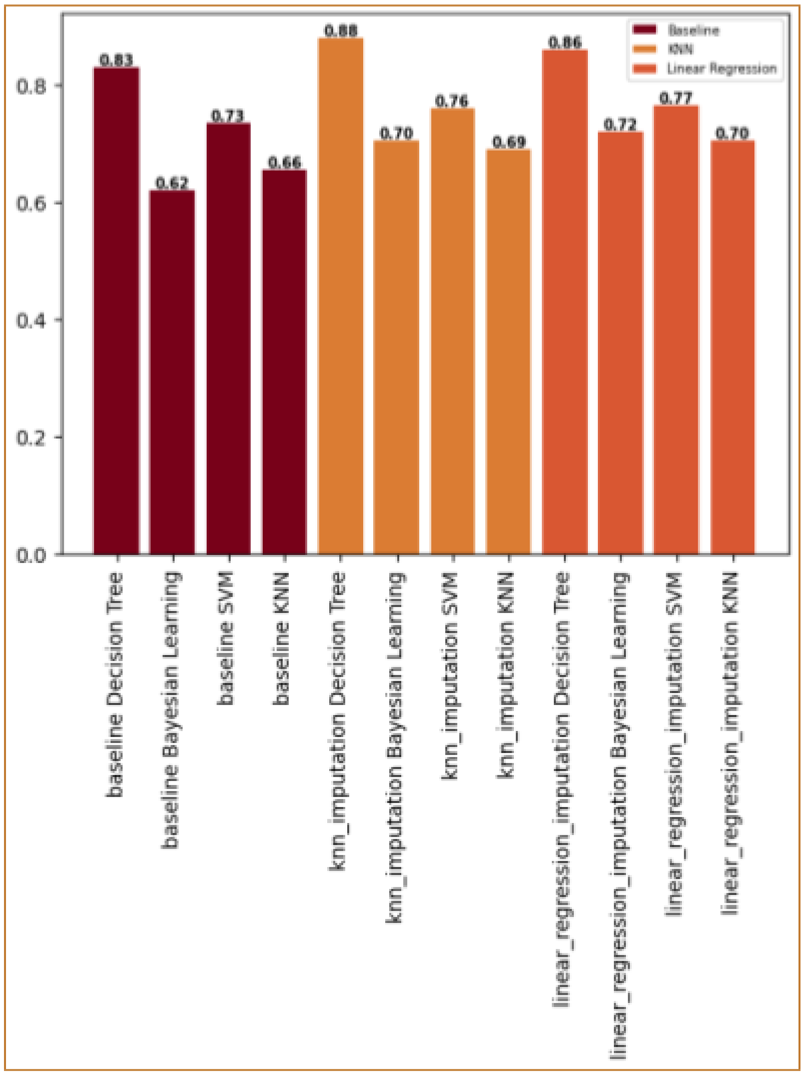
Model Comparison



In general, the **Bayesian Learning and Decision Tree models are more precise** and accurate than the SVM and KNN models. The KNN model, however, has the **highest sensitivity** of the four, which means that while it accurately identifies more true positives, it also has a larger rate of false positives.

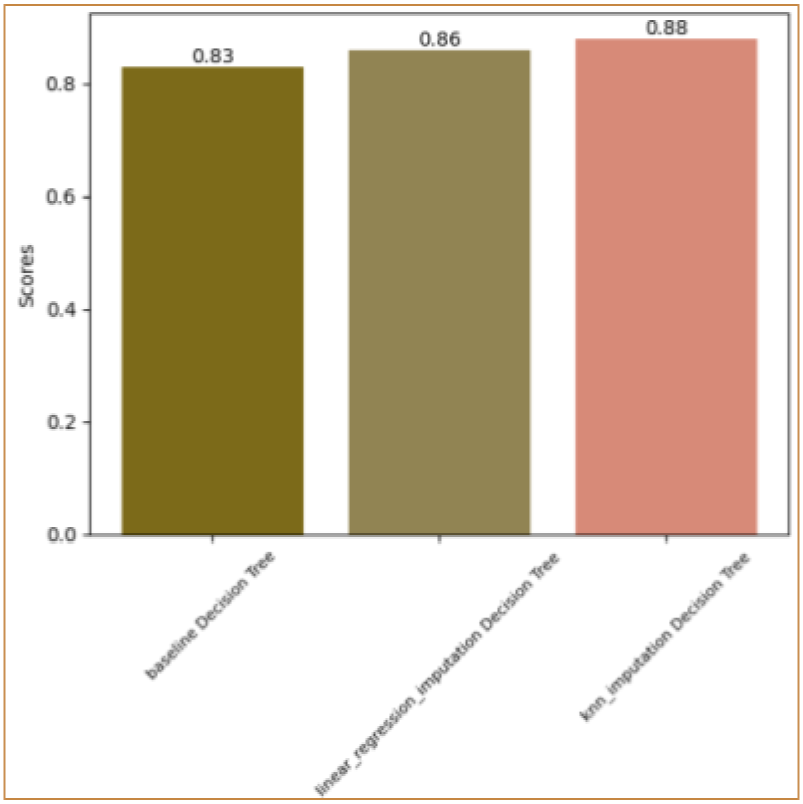
Final Comparison of all the models:

All Model Comparison



There is little doubt that the **Decision Tree Classifier** utilizing the **KNN Imputer** Method **outperforms** all the models we have trained, followed by the Linear Regression Imputer and the Baseline Approach. In comparison to the other two methods, **the Bayesian learning, KNN classifier performs the worst.**

Best Model Comparison



This graph displays the top performers across all imputation techniques. In every imputation approach, the decision tree classifier **outperforms** all other classifiers. In this bar plot, the baseline technique is performing the worst. With an accuracy score of **83%,** the linear regression imputer is extremely close to the best performer. Last but not least, a decision tree classifier that employs a KNN imputer outperformed every other classifier with an accuracy of **88%**.

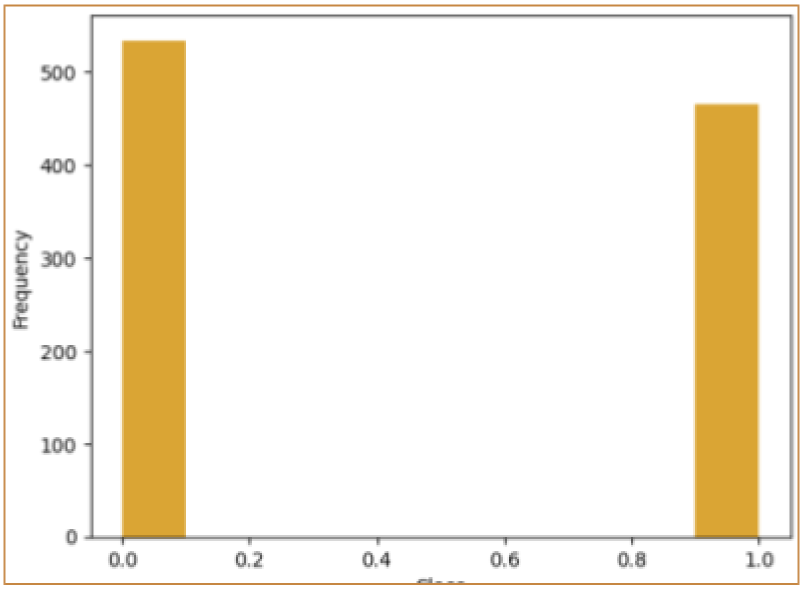
Finally, after all the report and model visualization. We can categorically state that the decision tree classifier is the best for this dataset when used with a KNNimputer to handle missing values. Thus, we will perform

Final Prediction using best model:

Following are the outcomes of applying a decision tree classifier to the test dataset:

- The following patients have a high chance of getting diabetes : **466**
- The following patient population has a low chance of getting diabetes: **534**

Predicted Class Distribution



Conclusion:

After all these steps, it is clear from the evidence that the decision tree classifier, which has the best performance and accuracy on the available dataset for this predictive job, outperforms other classification models.

Part 3 : Regression model

Objective : To Predict whether patient's average blood glucose level exceeds the diagnostic threshold or not

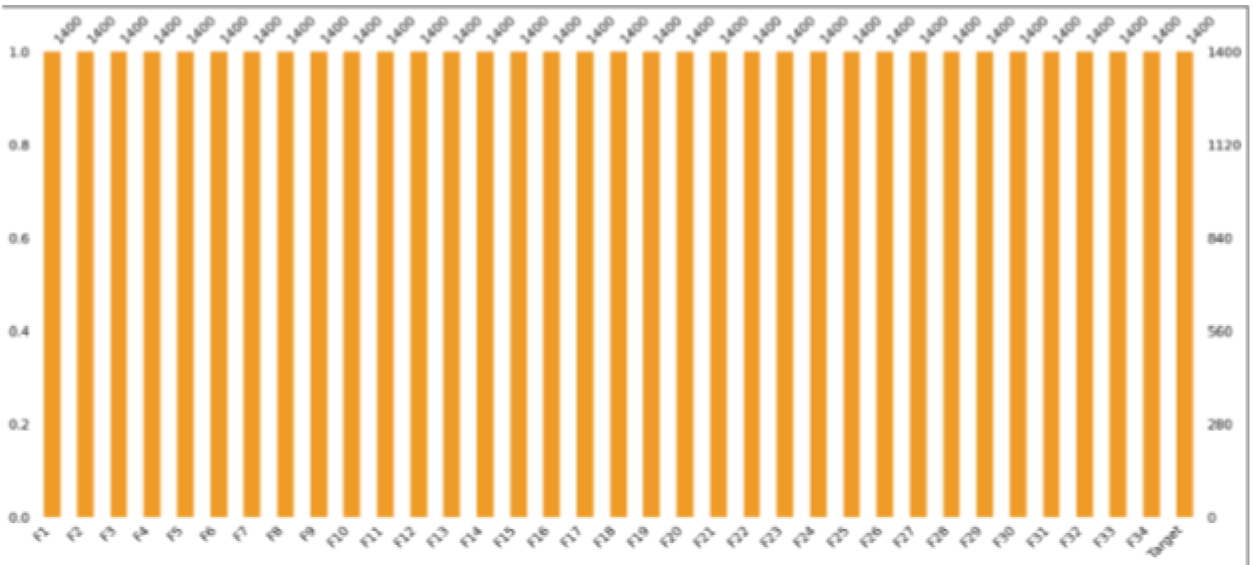
Type of the task : Regression :

Analyzing exploratory data and preprocessing it : To begin, We first examine the descriptive statistical summary and column data for each column in the training dataset.

	count	mean	std	min	25%	50%	75%	max
F1	1185.0	-772.598397	623.838276	-2534.18	-1206.68	-769.90	-358.42	1726.48
F2	1185.0	40.901494	13.079752	-26.34	34.68	40.89	47.07	111.60
F3	1185.0	4.004160	2.840863	0.05	1.91	3.29	5.39	17.34
F4	1185.0	5084.650557	2969.806866	-5835.15	3117.14	5121.30	7156.40	15040.56
F6	1185.0	-17.599080	4.126263	-40.88	-19.63	-17.60	-15.57	1.74
F7	1185.0	254.207046	295.150797	-680.11	52.39	242.31	459.96	1209.88
F8	1185.0	-85.524489	30.142736	-191.71	-104.61	-84.38	-65.76	16.16
F9	1185.0	4.603679	7.293075	0.00	0.18	1.16	5.48	34.14
F10	1185.0	-680.146228	134.876096	-1406.04	-742.20	-677.49	-614.49	43.98
F11	1185.0	-17.764329	12.882687	-78.78	-24.18	-17.76	-11.46	61.80
F12	1185.0	8.006751	3.661333	0.00	6.00	8.00	10.00	20.00
F13	1185.0	-860.281688	587.479731	-2791.70	-1266.20	-863.34	-464.54	1111.60
F14	1185.0	60.120759	12.767226	-48.60	53.94	60.51	66.21	117.99
F15	1185.0	1874.799460	846.432925	-2687.40	1443.24	1869.36	2307.22	5095.44
F16	1185.0	16828.077224	3974.966656	1088.42	14823.88	16828.18	18733.57	32769.68
F17	1185.0	1661.143544	2966.991333	-7898.33	-386.35	1594.59	3708.59	10498.02
F18	1185.0	-905.680219	603.672434	-3546.50	-1326.90	-906.08	-500.08	1327.20
F19	1185.0	11.721924	8.478195	0.09	5.67	9.66	15.66	66.81
F20	1185.0	3377.326025	915.185142	623.58	2751.21	3371.97	3968.16	6934.35
F22	1185.0	239.358506	130.384447	-447.09	177.81	238.65	304.47	762.09
F23	1185.0	-2220.088844	594.954932	-4398.76	-2631.42	-2219.82	-1815.42	-461.80
F24	1185.0	-244.242278	91.543221	-525.60	-306.63	-244.56	-179.64	49.65
F25	1185.0	-12.625865	3.957113	-31.82	-14.83	-12.66	-10.57	2.55
F26	1185.0	3504.438076	819.947355	218.74	3097.94	3518.32	3920.70	7651.10
F27	1185.0	2742.789924	894.069416	-101.85	2177.79	2773.80	3346.23	5768.58
F28	1185.0	8.156962	3.561099	0.00	6.00	8.00	10.00	22.00
F29	1185.0	-0.245823	9.164916	-27.57	-6.72	-0.33	6.24	26.19
F30	1185.0	-15.496278	29.879791	-117.71	-35.33	-15.36	3.62	80.22
F31	1185.0	-346.846886	92.492701	-656.64	-408.63	-346.68	-284.04	-53.37
F32	1185.0	-2344.213485	436.463839	-4860.34	-2556.00	-2337.16	-2117.19	-240.81
F33	1185.0	-8.553198	2.914239	-18.27	-10.48	-8.40	-6.67	0.59
F34	1185.0	-57.475865	8.173798	-112.76	-61.74	-57.42	-53.62	-25.28
Target	1185.0	182.207401	207.771531	-8.93	-8.93	115.07	302.66	750.78

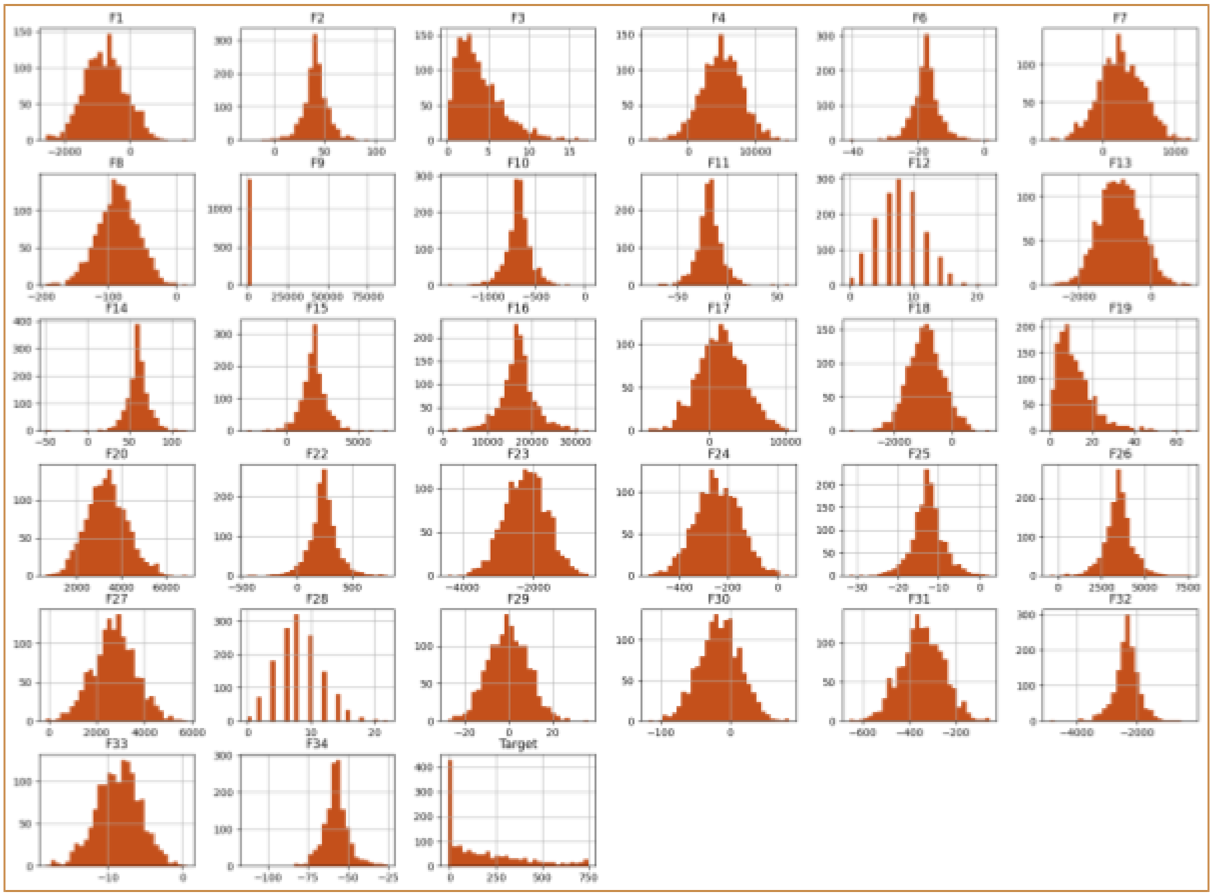
34 features are included in the dataset, and we must look for any missing information. To determine the frequency of each column's missing values, we visualize the data.

Missing value identification using bar chart



The preceding plot makes it abundantly obvious that there are **no empty** or null values. So, there is no need for imputation. Then, we use histograms to look for skewness in the provided data.

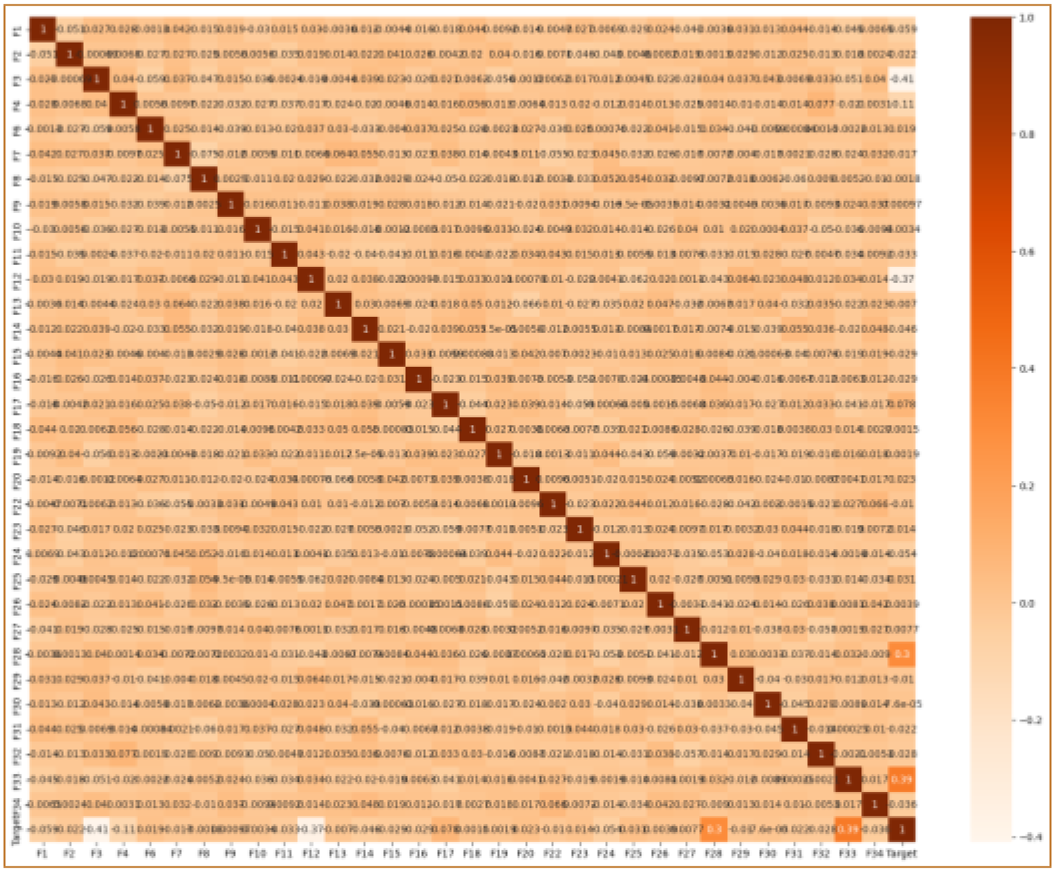
Data spread check using Histograms



The data must be encoded in order to prepare it for training on **F5** and **F21**. These two columns include object data types (categorical data). F5 is an **ordinal categorical data**, meaning that the categories are ranked or have a natural order. Since F21 is a **nominal categorical data**, there is no natural hierarchy or order among the categories or values. We can encode these category data into new binary data using the **one-hot encoding** method so that they can be used in ML models.

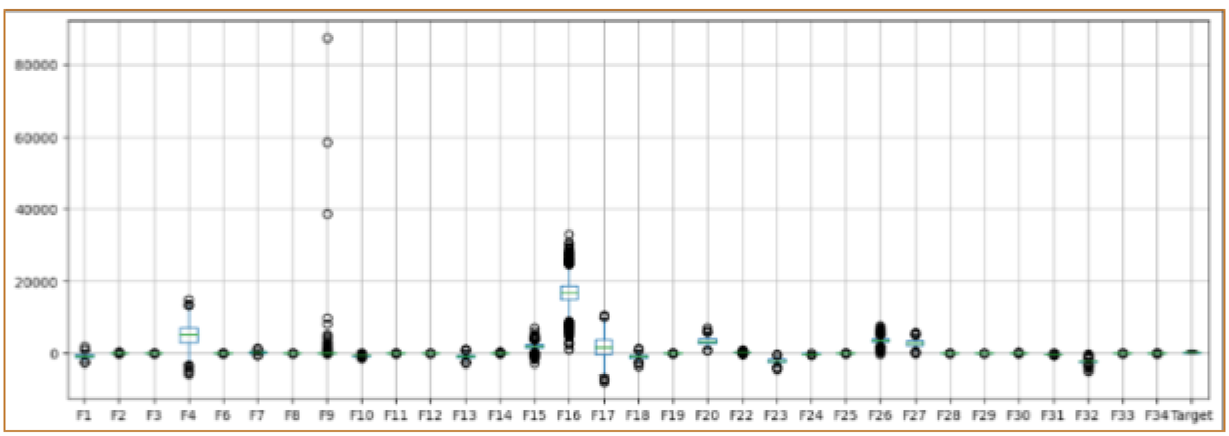
With the use of the Seaborn library, the correlation between each feature in the dataset was discovered using a heatmap. The heatmap below shows that there is less association between the features.

Correlation Matrix Heatmap



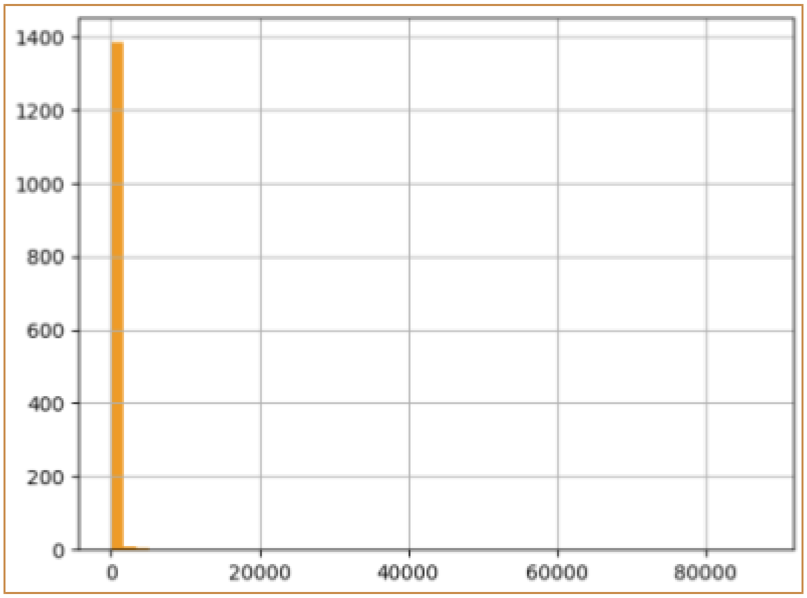
After this, We are checking if the dataset has any outlier using box plot

Outlier detection using Box plot



With above plot, we can clearly see that there is a **outlier** in **F9**, so we need to take care of this before train model with this dataset

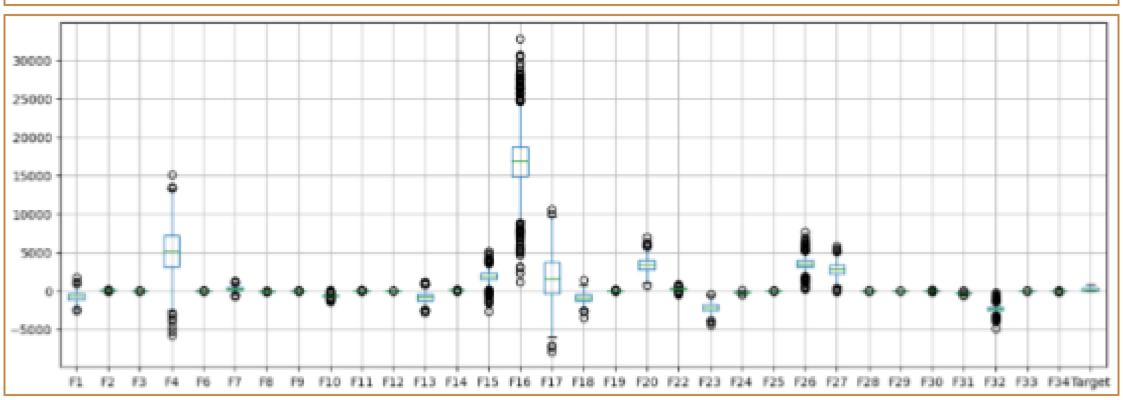
Data distribution



The date is **left skewed**. Interquartile range (IQR), which is the difference between the highest and lowest values in the dataset, allows us to quantify variability in data that is less impacted by extreme values or outliers.

Finding probable outliers in a dataset can be done by computing the interquartile range (IQR). The IQR, which is the difference between a dataset's maximum and minimum values, is a measure of variability that is less impacted by extreme values or outliers than the range. The data appears good after handling the anomaly.

Outlier detection using Box plot



Moving on to data splitting and feature scaling, the dataset was divided into **80%** for training and **20%** for testing using sklearn.model_selection.train_test_split. One distinct approaches were used for feature scaling: **standard scaler** for standardization

The input values are rescaled using the following equation by the standard scaler, which is pruned to outliers:

$$Z = x - \mu / \boldsymbol{\sigma}$$

- where **µ** stands for the standard deviation and for the mean.

The data is now ready for experimentation and assessment using various machine learning models after all these steps have been applied.

Machine Learning model comparison (Additional Comparative study):

Selected machine learning procedures:

- Linear Regression
- Gradient Boosting
- Ridge Regression
- Cat Boost

A machine learning pipeline was made using sklearn.pipeline for the purpose of comparing and selecting the best performing model for prediction.All four algorithms were fed into the pipeline, which then underwent 5 Kfold cross-validation. Then, using test data, evaluate the trained data. Finally, the model with the highest performance rating is chosen.

Design and Architecture of the Code Base:

The implementation of the machine system followed object-oriented ideas. Methods for preparing data, training, and assessing machine learning models are included in the primary **ModelSelector()** class. Categorical data encoding and feature scaling are done using the **preprocess()** technique. The **evaluate_models()** method applies cross-validation and pipeline training to the models and uses model scores to assess their performance. The scores are all stored in the **self.scores** dictionary. Finally, the best performing model and data preprocessing can be obtained using the **get_best_model()** method. The **show_results()** method shows a bar chart with the scores for each metric, while the **show_scores()** method outputs the scores of all models. These details help us develop the **best_pipeline**, which makes predictions on test data. Plotting **feature_importance** with feature id is the final step.

In general, this class automates the selection and evaluation of models for a given dataset, making it simpler for a data scientist to select the most appropriate model for their issue.

Model Reports

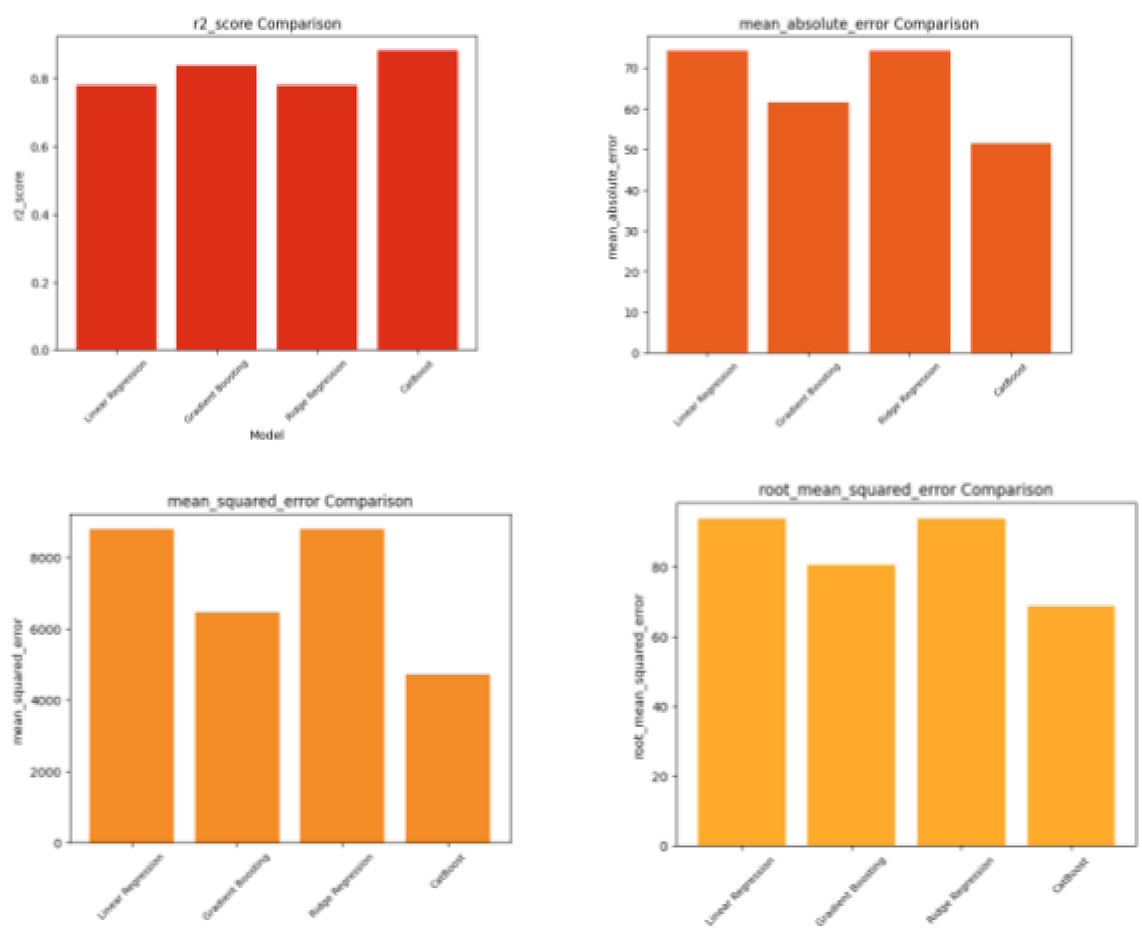
ML Algorithm	Cross-validation score	R2 Score	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error
Linear Regression	10281.784 (+/- 944.181)	78	74	87.84	93
Gradient Boosting	7419.943 (+/- 619.311)	83	61	64.67	80
Ridge Regression	10280.652 (+/- 942.919)	78	74	87.84	93
Cat Boost	5499.190 (+/- 515.797)	88	51	47.25	68

The effectiveness of four distinct regression models, as measured by various criteria, on a particular dataset. The models are Ridge Regression, CatBoost, Gradient

Boosting, and Linear Regression.

With the lowest cross-validation score, highest r2_score, and lowest values for mean_absolute_error, mean_squared_error, and root_mean_squared_error, CatBoost clearly had the best performance in terms of all measures.

Model Scores

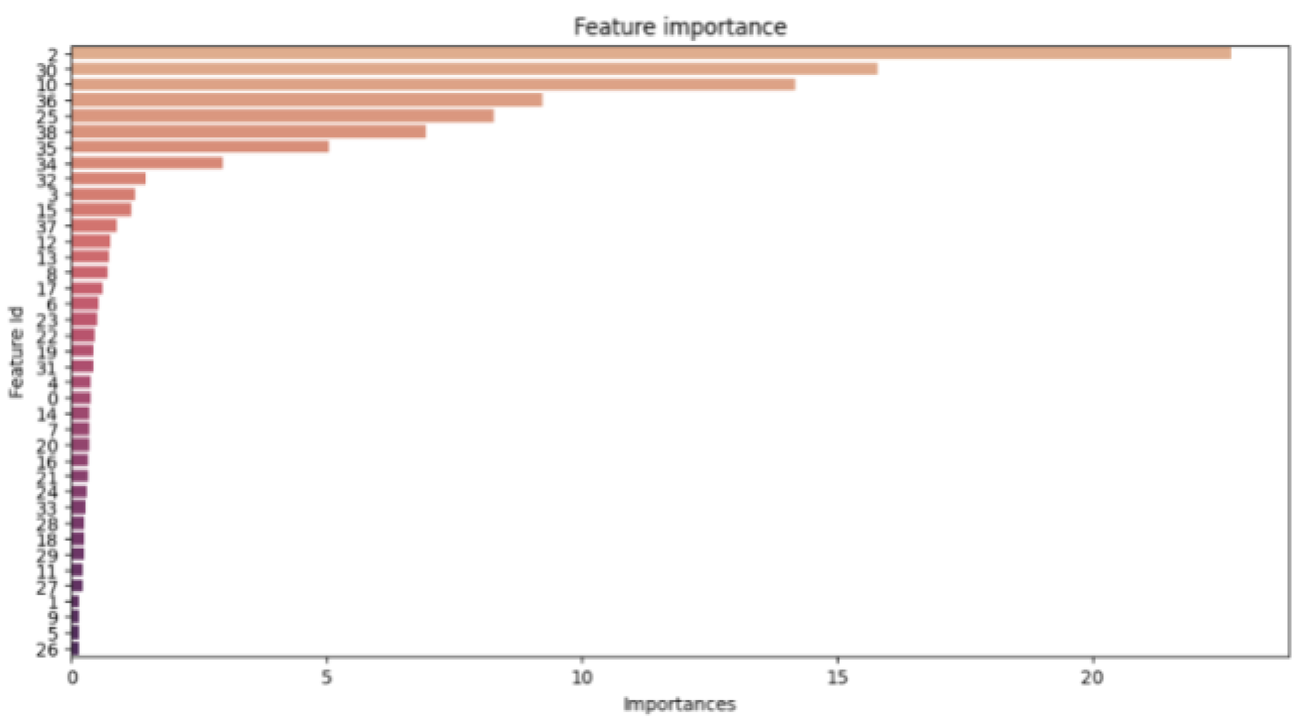


Overall, this report offers a helpful comparison of various regression models, which may be used to guide the choice of a suitable model for a certain dataset and issue.

Finally, after all the report and model visualization. We can categorically state that the **Cat Boost Regressor** with **88% accuracy** is the best for this dataset when used with a **One Hot Encoder** to categorical values. Thus, we will perform

Final Prediction using best model:

The ML pipeline model with the best accuracy will be employed in our cat boost to make a prediction on test data based on the evidence derived from the aforementioned metrics. The test dataset's use of Cat boost led to **more accurate** findings than other regression models that were taken into consideration.

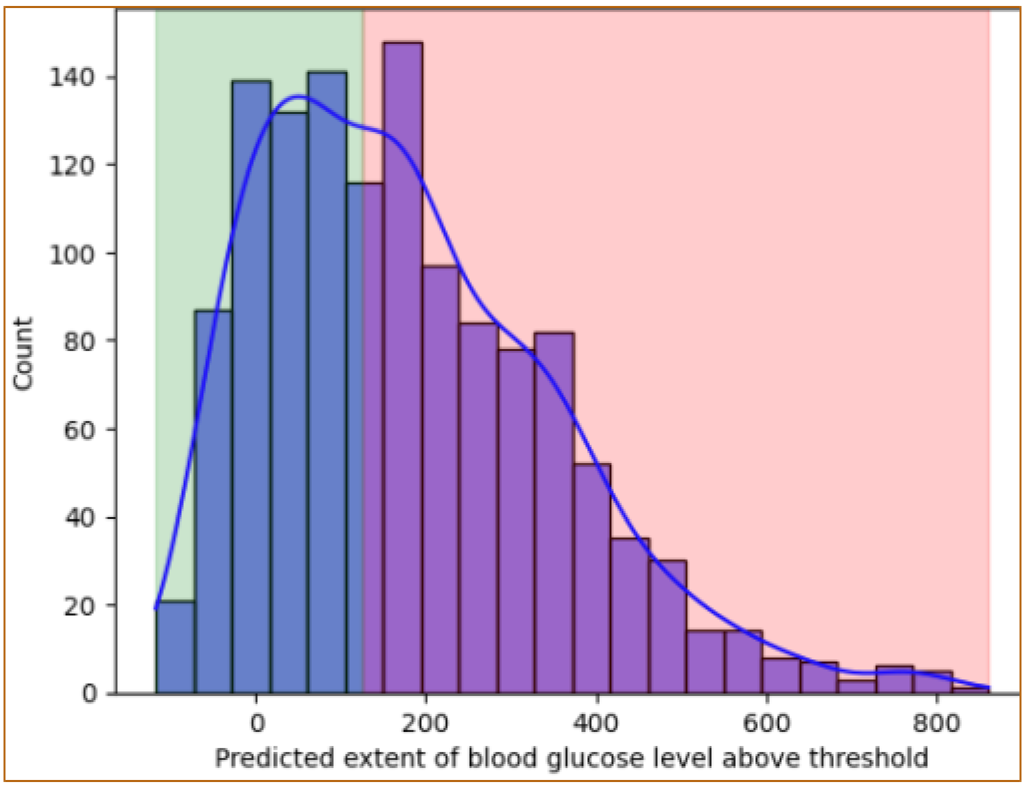


After all these steps, the **Cat boost classifier**, which has a high performance and accuracy of **83%**, clearly outperforms other models when used on the provided dataset for this predicting task.

Following are the outcomes of applying a cat boost regressor to the test dataset:

- The following patient's average blood glucose level exceeds the diagnostic threshold : **735**
- The following patient's average blood glucose level does not exceeds the diagnostic threshold: **565**

Distribution of Predicted values



Conclusion :

In order to compare and choose the best regression model for estimating the degree to which a patient's blood glucose level exceeds the diagnostic threshold, the code constructed an object-oriented machine learning pipeline. With **88%** accuracy, CatBoost outperformed the other models. The final CatBoost prediction showed **565** patients with normal blood glucose levels and **735** patients with high blood glucose levels. Overall, this pipeline offers a useful manual for choosing the best regression model for a specific dataset and problem.

Reference:

- *Machine learning for diabetes clinical decision support: a review :* *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9006199/*
- *A survey on diabetes risk prediction using machine learning approaches :* *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10041290/*