

Credit Default Prediction Using Machine Learning Algorithms

Machine Learning Project Report
April 8th, 2022

TABLE OF CONTENT

S. No	TOPICS	Page No.
1	INTRODUCTION	03
2	DATASET	03
3	EXPLORATORY DATA ANALYSIS	04
4	DATA PREPROCESSING	08
4	METHODOLOGY	08
5	MODEL EVALUATION AND OPTIMIZATION	09
6	FEATURE SELECTION	13
7	MODEL IMPLEMENTATION	16
8	RESULT AND CONCLUSION	20

INRODUCTION

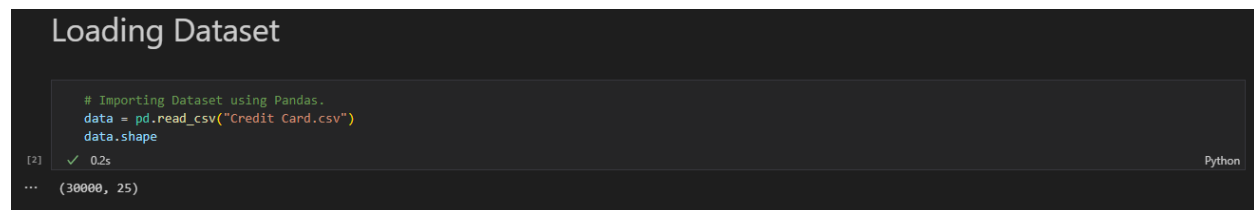
The ability to accurately predict the likelihood of a customer defaulting on their payments is a critical task for financial institutions. This machine learning project tries to create a model for predicting which clients are likely to fall behind on their payments. A significant collection of finance data, including demographic data, credit history, and payment information, will be used to train the model. The ultimate objective of this project is to provide financial institutions with the information they need to manage credit risk and make optimum decisions about lending, which will reduce losses and improve their overall financial health.

This project is a binary class classification problem where our target variable “default” is having only two classes which are 0 and 1. For the accurate prediction of our target variable, we are using different classifiers including decision tree, random forest, naïve bayes classifier, KNN, boosting models etc. At first, we are performing data preprocessing such as cleaning etc. and undertaking exploratory data analysis (EDA) to gain useful insights into our data.

Our goal is to build a robust and accurate prediction model that can help financial institutions identify customers who are at risk of defaulting on their payments. So, for that reason, we are using different optimization techniques such as K fold cross validation to improve accuracy of the model and to validate our models and ensure they are not overfitting the data and Grid search CV to get best hyperparameters in this project. To get best performing features we are selecting features with different methods of feature selection such as using random forest feature selector, using information gain approach, lasso regularization, and correlation matrix.

DATASET: -

The dataset we are using in this project is from credit card data collected from the banking industry (private bank) and contains information of customers. Dataset contains 30000 customer records and 25 features. We are using “Default payment next month” as our target variable. It is a Binary class classification problem, where value of class is 0 and 1 (0 means not default and 1 means default). The data is slightly imbalanced and consists of 24 attributes other than target attribute.



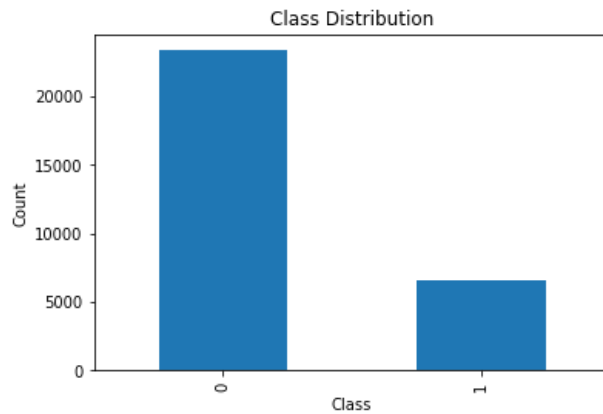
```
Loading Dataset

# Importing Dataset using Pandas.
data = pd.read_csv("Credit Card.csv")
data.shape

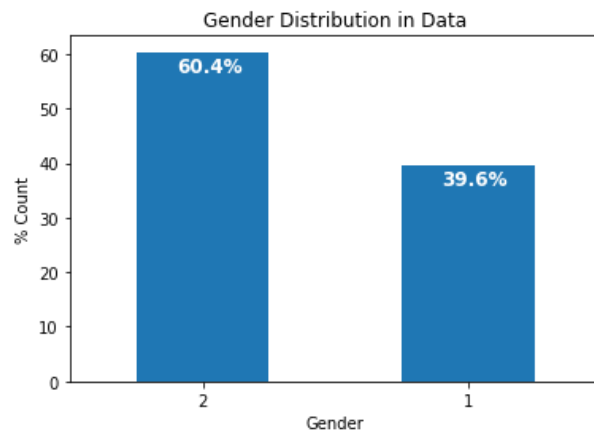
[2] ✓ 02s Python
... (30000, 25)
```

EXPLORATORY DATA ANALYSIS

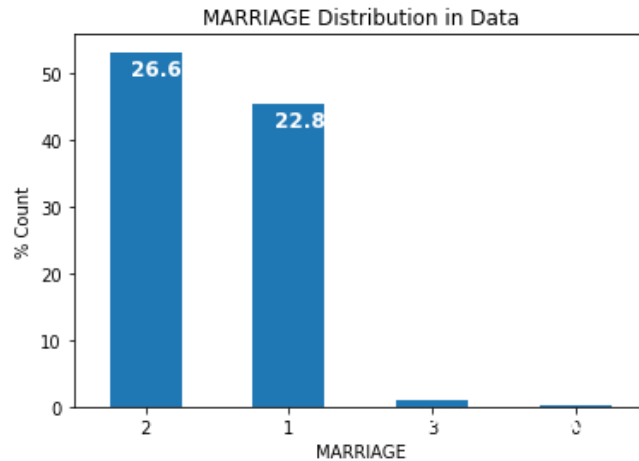
In this project, before performing data preprocessing, we are conducting EDA to get deep insights from our data. We are using different visualizations libraries such seaborn, matplotlib, and plotly etc. for this purpose.



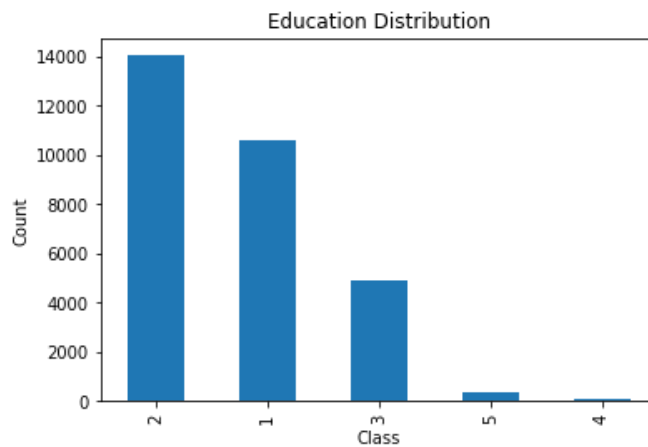
In this first plot, we have plotted a distribution of our target variable. We can see that distribution of class label is imbalanced in our dataset. There are 23,364 records of non-default and 6,636 records of default payments. We have 2 classes so it's a binary class classification task. Now plotting this distribution in graphical form.



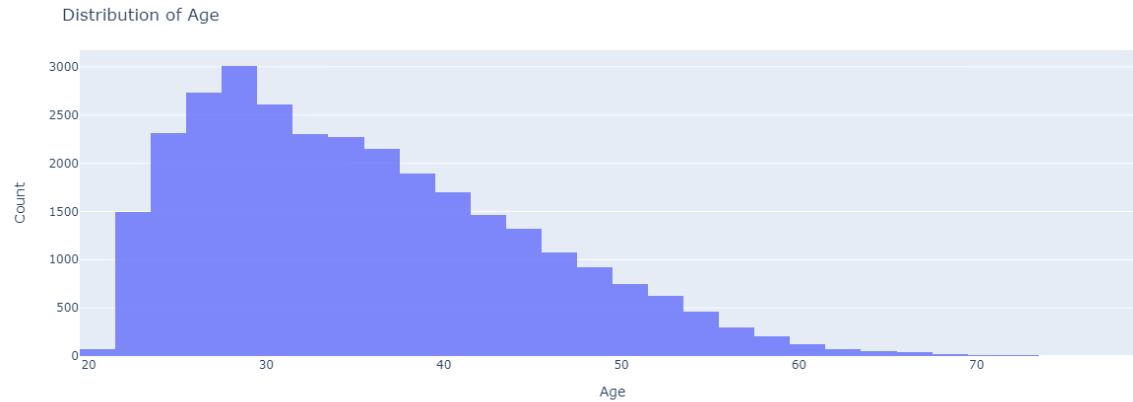
In the second plot, we are plotting distribution of gender in the dataset (Here 1 = male, 2 = female). We can observe clearly from the bar chart that female are more as compare to the males in the data.



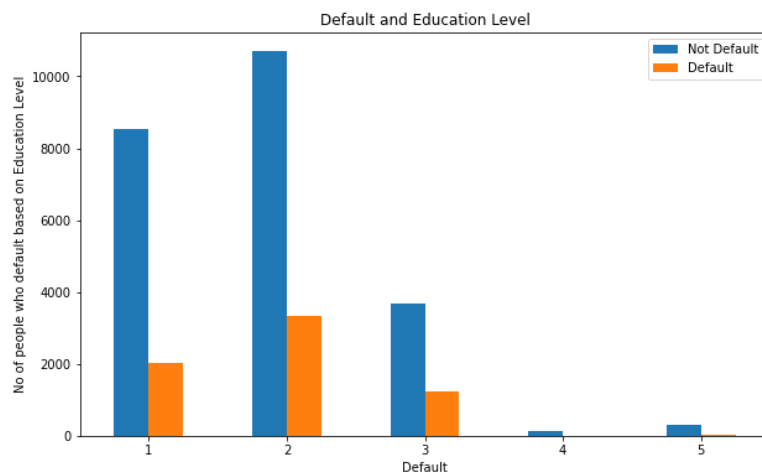
Now we have plotted distribution with respect to marriage. To check out the marital status of the members (Here 2 = Single, 1 = Married, and 3 = Others). Most of the records in the dataset are from the people having marital status as single followed by married people.



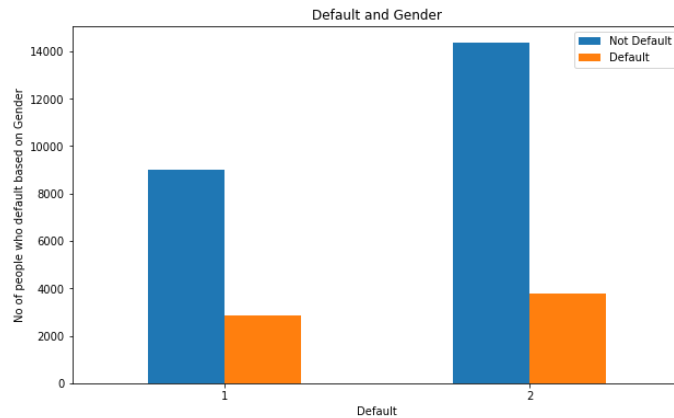
Now we have plotted the bar chart of Education level of the members. To check out the education status of the members (Here 2 = University Level, 1 = Graduate School, 3 = High School, 4 = Others, 5 = Unknown). Most of the people in the dataset are university level graduate. There are also many people who are having graduate school level education followed by high school level people.



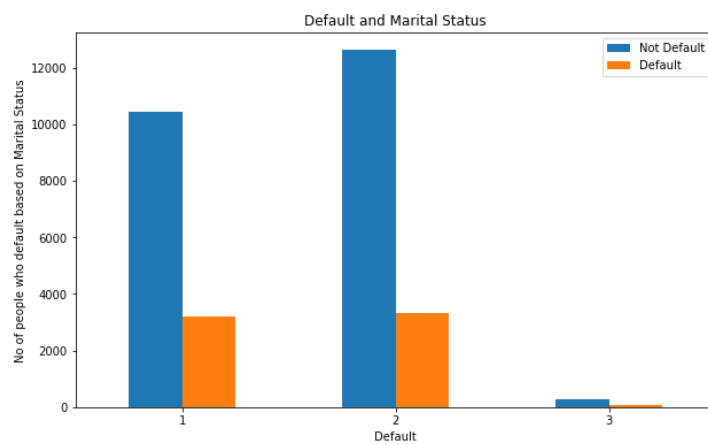
According to the distribution plot. Most of the people in the dataset are of the age group 25-35.



Now we have plotted the bar chart of education level along with the Default target to check out the relation between both and which education level are defaulting more. As we can see, most of the default payments are from university level educated people but the rate of default is almost same in category 1 ,2 ,3.



Now we have plotted the bar chart of Gender along with the Default target to check out the relation between both and which gender are defaulting more. (Here 2 = Female and 1 = Male). From above graph we can observe that rate of default payment is high in Male members as compared to the female members.



From the above bar chart, we can observe that the rate of default payment is higher in married people as compared to the single.

So, after doing the EDA, we have got the necessary information about the dataset and insights about the dependency and contribution of different features. Now we will proceed with data preprocessing.

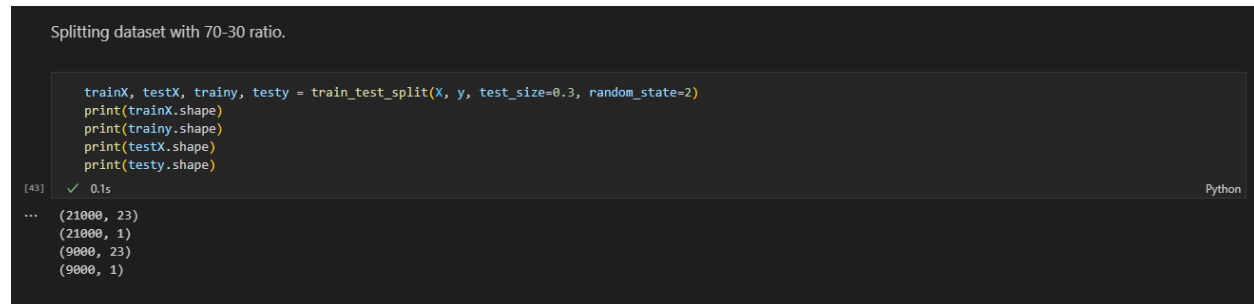
Data Preprocessing

Before using the data in the machine learning algorithms, preparing the data is a crucial step. Data preprocessing is a data mining technique used to turn the raw data into a format that is both practical and effective.

For our dataset, we also must perform some data pre-processing. First, we load our dataset using pandas and check shape of our dataset which is 30000 and 25 columns. Dataset is normalized and contain both integer and float values. We then checked for null values, but data don't have any null values. After that we dropped some unnecessary columns such user id.

Train – Test Split

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. Here we are splitting our dataset into train dataset and test dataset with the ratio of 70-30% where 70% is our train dataset and 30 is our test dataset.



```
Splitting dataset with 70-30 ratio.

trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.3, random_state=2)
print(trainX.shape)
print(trainy.shape)
print(testX.shape)
print(testy.shape)

[43]: ✓ 0.1s Python

... (21000, 23)
(21000, 1)
(9000, 23)
(9000, 1)
```

METHODOLOGY

In this project, we use various machine learning algorithms to predict the target variable default payment and compare the outcomes of each approach to see which model fits and performs best on the training data set. To boost the model's performance, we are also using various model enhancement strategies. These methodologies include voting classifiers, K-fold cross validation and grid search for hyperparameter tuning of models. Our only goal is to increase the model's performance with highest accuracy.

We first implement machine learning models on the full data with all the features and then apply different feature selection techniques to select top performing features and to reduce dimensionality of the data.

Model Evaluation: -

We are evaluating the performance of our model's using accuracy, AUC graph, precision score, sensitivity, specificity, miss rate, precision, and recall values. Before we move forward to model implementation, it is necessary to know the following terms.

1) Precision: -

Precision is the ratio between the True Positives and all the Positives. For our problem statement, that would be the measure of cases that we correctly identify having a disease out of all the cases having it.

2) Recall: -

The recall is the measure of our model correctly identifying True Positives. Thus, for all the cases who have disease, recall tells us how many we correctly identified as having a disease.

3) Sensitivity: -

How well a machine learning model can identify positive examples is measured by its sensitivity. The true positive rate (TPR) or recall are other names for it. Sensitivity helps us to determine how many cases the model was able to accurately recognize, which is why it is utilized to assess model performance.

4) Specificity: -

Specificity is the proportion of true negatives that are correctly predicted by the model.

5) Miss Rate: -

The false negative rate – also called the miss rate – is the probability that a true positive will be missed by the test. It's calculated as $FN / (FN + TP)$, where FN is the number of false negatives and TP is the number of true positives (FN+TP being the total number of positives).

6) False Positive Rate: -

The false positive rate is calculated as $FP / (FP + TN)$, where FP is the number of false positives and TN is the number of true negatives (FP+TN being the total number of negatives). It's the probability that a false alarm will be raised: that a positive result will be given when the true value is negative.

7) Confusion Matrix: -

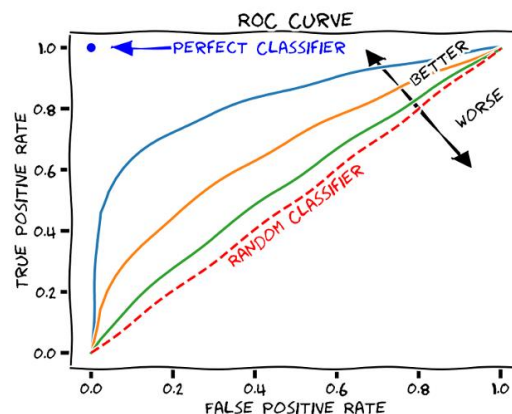
A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

8) ROC – AUC Curve: -

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the ‘signal’ from the ‘noise’. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes. When $AUC = 1$, then the classifier can perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.



K fold Cross Validation

K-fold cross-validation is a technique used to evaluate the performance of a machine learning model by dividing the dataset into K subsets (or folds) of roughly equal size. The model is trained and evaluated K times, with each fold used as the test set once and the remaining K-1 folds used as the training set.

K-fold cross-validation is a commonly used technique for model evaluation because it provides a more accurate estimate of model performance than a single train-test split. By using multiple train-test splits, K-fold cross-validation reduces the risk of overfitting and provides a more reliable estimate of how the model will perform on new, unseen data.

In our project, we are using K fold cross validation with 10 folds. It tells about how our model will perform on the actual test data.

```
# K - FOLD Cross Validation.

kfold = KFold(n_splits=5, random_state=100, shuffle=True)

DT = DecisionTreeClassifier(max_depth=200)
results_kfold = cross_val_score(DT, trainX, trainy.values.ravel(), cv=10)
print(results_kfold)
print("Accuracy of DECISION TREE KFold with k=10: %.2f%%" % (results_kfold.mean()*100.0))

pred = cross_val_predict(DT, trainX, trainy.values.ravel(), cv=5)
conf_mat = confusion_matrix(trainy, pred)
print(conf_mat)
disp = ConfusionMatrixDisplay.from_predictions(trainy, pred)
```

Python

Grid Search Hyper Parameter Tuning

To determine the ideal set of hyperparameters for a particular model, we implemented the grid search hyperparameter tuning technique. Hyperparameters are model parameters that are selected by the user prior to training rather than being learned from the data. In our project, we are using Grid search technique to get the best possible hyper parameters for the model to increase its efficiency.

Applying Grid Search CV for best parameters of Random forest.

```
rf_cv = RandomForestClassifier()
param_grid = {
    'max_depth': [2, 5, 10, 20, 30, 40, 50],
    'n_estimators': [100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600]
}
grid_search = GridSearchCV(estimator=rf_cv, param_grid=param_grid, cv=3, verbose=2)
grid_search.fit(trainX, trainy.values.ravel())
best_grid = grid_search.best_estimator_
print(best_grid)
```

Python

Applying Grid Search CV for best parameters of Gradient Boosting.

```
gb_cv = GradientBoostingClassifier()
param_grid = {
    'learning_rate': [0.01, 0.05, 0.1],
    'n_estimators': [100, 150, 200, 250, 300]
}
grid_search = GridSearchCV(estimator=gb_cv, param_grid=param_grid, cv=3, verbose=2)
grid_search.fit(trainX, trainy.values.ravel())
best_grid = grid_search.best_estimator_
print(best_grid)
```

Python

Feature Selection

In this project, we are first implementing machine learning models on the full feature dataset and after that performing different feature selection methods to select the best performing features.

The selection methods we have used are:

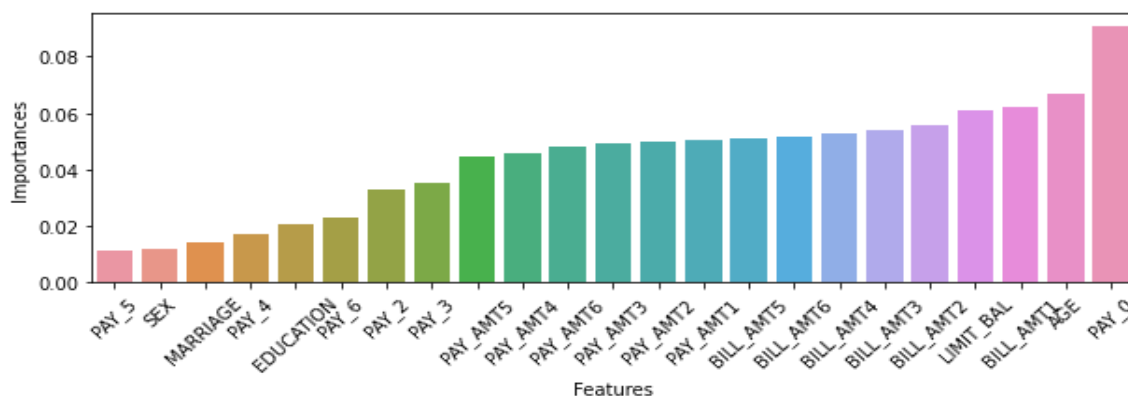
- 1- Feature Selection using Random Forest.
- 2- Feature Selection using Lasso Regularization.
- 3- Feature Selection using Information Gain.
- 4- Feature Selection using correlation matrix.

We are selecting features using all these methods and checking the impact on the model's performance and to find out which method is selecting top impactful features.

a) Feature Selection Using Random Forest

Feature selection using random forest involves using a random forest model to identify the most impactful features in a data. Random forest is an ensemble learning algorithm that works by forming multiple decision trees and combining their outputs predictions. In feature selection using random forest, the model determines the importance of each feature by calculating the decrease in impurity (e.g., Gini index or entropy) that is achieved when that feature is used to split the data at a node in the decision tree.

The random forest algorithm builds the decision trees and then determines the average decrease in impurity for each feature across all the trees. Characteristics are more significant if their average impurity decrease is higher. A subset of the most crucial traits might be chosen for additional study based on this rating. Because of the decreased overfitting and enhanced interpretability, this can improve model performance.



Selected features after random forest feature selection are:

```
# These are the important features after RFE.
selected_cols = [column for column in trainX.columns if column in trainX.columns[sel_rfe_tree.get_support()]]
selected_cols

... ['LIMIT_BAL',
      'AGE',
      'PAY_0',
      'BILL_AMT1',
      'BILL_AMT2',
      'BILL_AMT3',
      'BILL_AMT4',
      'BILL_AMT5',
      'BILL_AMT6',
      'PAY_AMT1',
      'PAY_AMT2',
      'PAY_AMT3',
      'PAY_AMT4',
      'PAY_AMT6']
```

b) Feature Selection Using Lasso Regularization

Feature selection using Lasso regularization involves using a logistic regression model with L1 regularization to identify and select the most important features in a dataset.

In Lasso regularization, a penalty term proportional to the total of the absolute values of the coefficients is included in the objective function for the logistic regression model. By forcing some coefficients to be exactly zero, this penalty term promotes sparsity in the coefficient values, essentially removing the corresponding features from the model.

```
coeff = lasso.coef_
coeff

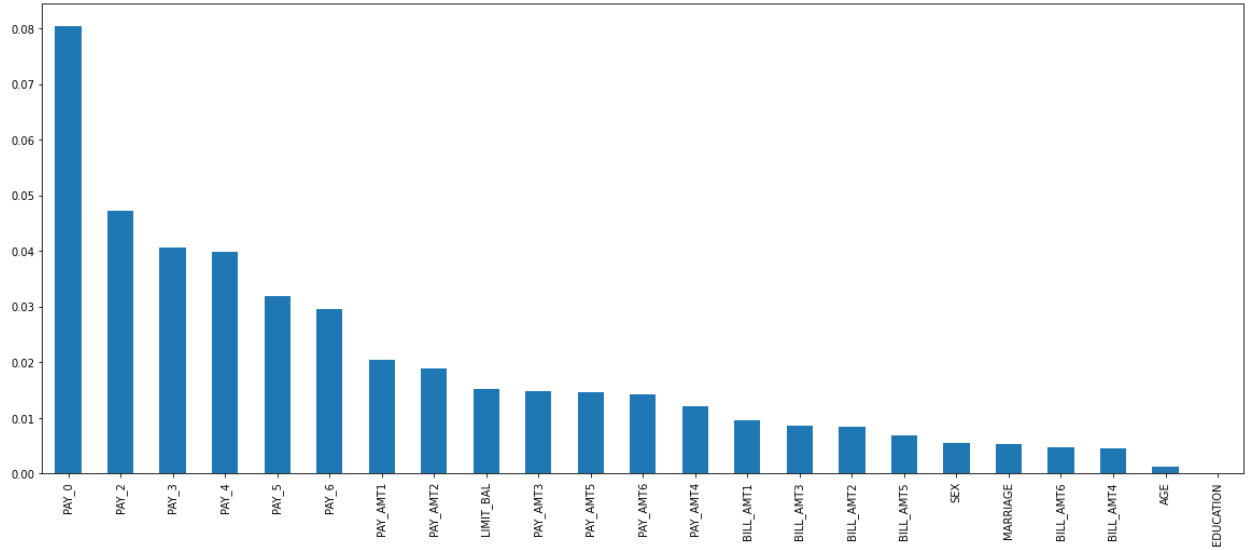
... array([-3.79085909e-07, -0.00000000e+00, -0.00000000e+00, -0.00000000e+00,
           6.18301841e-05,  5.58917340e-02,  0.00000000e+00,  0.00000000e+00,
           0.00000000e+00,  0.00000000e+00,  0.00000000e+00, -4.42671829e-07,
           4.05465980e-07, -5.29279758e-08, -4.38872983e-08,  1.07059842e-07,
           3.69002490e-07, -1.15833055e-06, -9.03468581e-08, -2.19027230e-07,
           -5.61345857e-07, -5.53642704e-07, -1.85586095e-07])
```

Here we will remove all the features whose lasso coefficient is shrink to 0.

c) Feature Selection Using Information Gain

First, we will apply all machine learning models on all our 23 features then after that we will select some features fusing Information Gain approach and apply models on selected features.

Information gain calculates the reduction in entropy or surprise from transforming a dataset in some way. Information gain is used for feature selection, by evaluating the gain of each variable in the context of the target variable. In this slightly different usage, the calculation is referred to as mutual information between the two random variables.

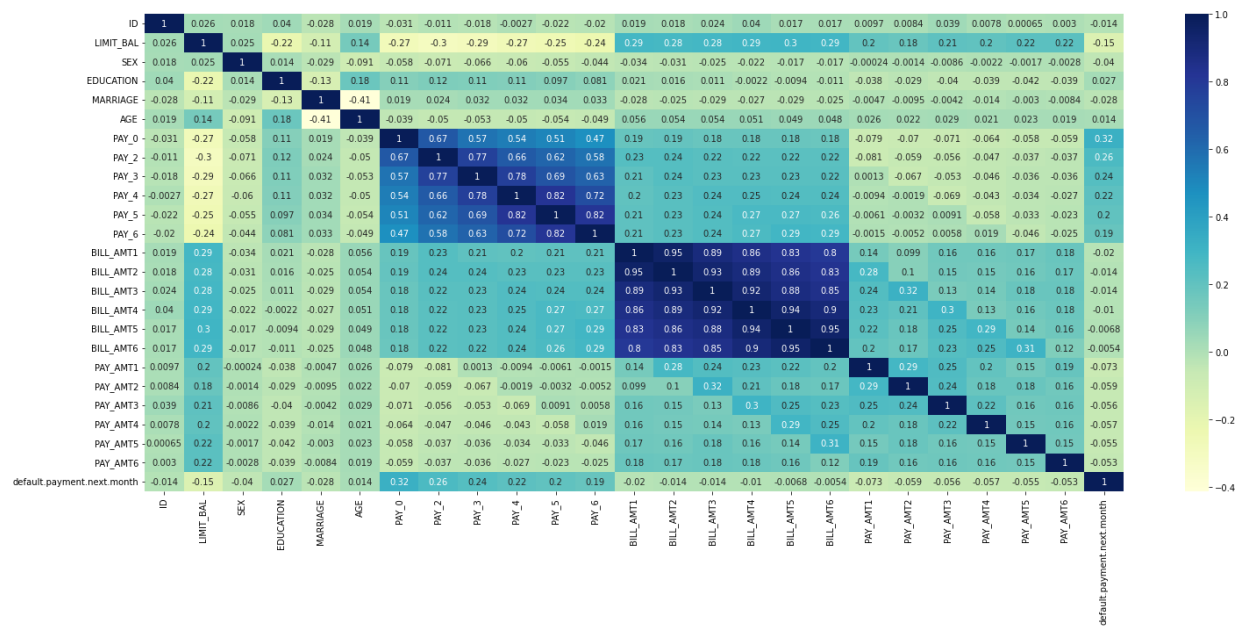


d) Feature Selection Using Correlation matrix.

Feature selection using a correlation matrix involves identifying pairs of features in a dataset that are highly correlated with each other and selecting a subset of features based on their correlation with the target variable.

A correlation matrix is a table that shows the correlation coefficients between all pairs of variables in a dataset. The correlation coefficient measures the strength of the linear relationship between two variables, ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation).

We first determine the correlation matrix for each feature in the dataset before doing feature selection using it. Afterwards, we search for feature pairings that are highly associated with one another (e.g., correlation coefficient greater than a threshold value such as 0.8). We choose the feature that is most strongly positively correlated with the target variable out of each pair of highly correlated characteristics and remove the other one. By doing so, we can simplify the dataset's dimensions and get rid of extraneous data.



Here in the correlation plot we can, Repayments features are positively correlated with the target variable. PAY_0 has the highest positive correlation with the target variable. which means it will contribute much to the prediction of target variable. Limit Balance is the variable which is highly negatively correlated with target. Multicollinearity is also present in the data as we can see the blue block in matrix.

Model Implementation: -

At first, we are implementing different machine learning classifier on our dataset first without any feature selection and then after doing selection with different methods of feature selection. We have 25 attributes at first, and our data is 70-30% divided into train and test sets.

1- Decision Tree

The first model we implemented is the decision tree classifier. A decision tree is a supervised machine learning model that is used for classification and regression tasks. It is a tree-like model that consists of nodes and branches, where each node represents a feature or attribute, and each branch represents a decision rule or condition based on that feature or attribute.

In a classification task, the goal of a decision tree is to divide the dataset into smaller and more homogeneous groups, where each group corresponds to a particular class or category.

The decision tree gives us accuracy of 72% on our full feature dataset, but its accuracy increases after reducing dimension and use only top performing features. The highest accuracy obtained by decision tree is after selecting feature using correlation matrix that in 82.28%.

2- Random Forest Classifier

The second model we are implementing after decision tree is random forest classifier. A Random Forest Classifier is a supervised machine learning algorithm that uses an ensemble of decision trees to perform classification tasks. The random forest algorithm is an extension of the decision tree algorithm, which overcomes some of the limitations of decision trees such as overfitting and instability.

On our dataset, random forest is performing well, giving accuracy of 81.98% accuracy on full feature dataset and 82.24% accuracy after feature selection.

3- Gradient Boosting Classifier

One of the famous boosting algorithms, which combines different weak learners into strong by minimizing the loss function. It is an ensemble technique. The goal of gradient boosting classifier is to iteratively add new models to the ensemble that correct the errors of the previous models.

In our project Gradient boosting is performing well on both full feature and selected feature dataset. It is getting the highest accuracy of 82.22% after feature selection.

4- XGBoost Classifier

A well-known machine learning technique that is a member of the gradient boosting family is called XGBoost (Extreme Gradient Boosting) Classifier. It is a development of the conventional gradient boosting approach, which minimizes a loss function to combine several weak models into a strong one.

High performance and accuracy are hallmarks of XGBoost, particularly when applied to structured data issues like feature engineering and tabular data. In comparison to other gradient boosting methods, XGBoost key benefit is its scalability, which enables it to handle huge datasets with millions of rows and columns.

In our project Xtreme Gradient boosting is performing better than all the other models, maybe because of the reason that XGBoost can capture non-linear relationships between the features and the target variable. It is getting the highest accuracy of 82% after feature selection.

5- Naïve Bayes Classifier

Naive Bayes is a simple yet effective machine learning algorithm for classification tasks. It is based on the Bayes theorem and assumes that the features are conditionally independent given the class variable. This assumption is known as the naive assumption, which makes the algorithm computationally efficient and easy to implement.

For our dataset, naïve bayes is not performing very well and its accuracy was below expectations.

6- K Nearest Neighbor

K Nearest Neighbor (KNN) is a simple machine learning algorithm used for classification tasks. The algorithm works by finding the K nearest neighbors of a new data point based on some distance metric, where K is a hyperparameter specified by the user. The class of the new data point is then determined by majority vote among its K nearest neighbors.

In our project, KNN is working fine, and its accuracy was 77% with full feature dataset and 78% after doing feature selection.

Voting Classifier (Ensemble Model)

Voting classifier is basically an ensemble technique which unites different base classifiers and based on the result of these classifiers it computes new result. It is a technique that may be used to improve model performance, ideally achieving better performance than any single model used in the ensemble. We are using a soft voting technique here which computes probability based on probabilities and weights associated with different classifiers. In voting classifier, we are using our best models.

We are using 3 models which are random forest models, gradient boosting model and XGBOOST which were giving us best results. So, after combining all our best performing models we create a voting classifier model, which gives us the best result better than all other base models.

Voting Classifier (Best Model)

```
r0 = GradientBoostingClassifier(learning_rate=0.02, max_depth=6, n_estimators=150)
r1 = RandomForestClassifier(max_depth=12, n_estimators = 147)
r2 = XGBClassifier(learning_rate = 0.014, max_depth = 5)
r3 = DecisionTreeClassifier(max_depth=8, max_features='auto')

# In soft voting, every individual classifier provides a probability value that a specific data point belongs to a particular target class
voting = VotingClassifier([('gb1', r0), ('gb2', r1), ('gb3', r2), ('gb4', r3)], voting='soft')
voting.fit(trainX, trainy.values.ravel())
y_pred_voting = voting.predict(testX)
```

Results

After implementing all the models, we have got the following results which are:

Credit Card Data						
Models	Accuracy of the Model (%)					
	Full Features	Feature Selection (Random Forest)	Lasso Regularization Feature Selection	Feature Selection using Information Gain	Feature Selection Using Correlation Matrix	
Random Forest	81.98	82.12	82.17	82.04	82.24	
Gradient Boosting	82.13	82.14	82.21	82.22	82	
Decision Tree	72.53	82.05	81.12	81.71	82.28	
XGBoost Classifier	82.24	82.26	82.25	82.25	82.21	
K-Nearest Neighbor	77.34	77.94	77.94	77.97	78.46	
Naïve Bayes	38.96	-	-	-	-	
Voting Classifier	82.15	82.11	82.06	82.12	83.33 (Best Accuracy)	

Conclusion

After performing a deep analysis on the dataset, we have successfully built a voting classifier model that has proven to be the best in predicting the outcome of our target variable. Through rigorous testing and evaluation, we have determined that the voting classifier model provides the most accurate and reliable predictions, surpassing all other models that we have explored.

We have used different feature selection techniques such as recursive feature selection, feature selection based on Information Gain, using confusion matrix, feature selection using lasso regularization or by removing multi collinear features. Other than that, we have used different techniques to increase the efficiency of the models such as K-Fold CV, Grid Search CV Hyperparameter Tuning and ensemble technique.

Our best model (Voting classifier) is getting highest accuracy of 83.33%.