Credit Card Default Prediction

Lakshmi Keerthana

Tito Varghese

Data Science Trainees,
Alma Better , Banglore

Abstract

Credit card default happens when you have become severely delinquent on your credit card payments. Default is a serious credit card status that affects not only your standing with that credit card issuer but also your credit standing in general and your ability to get approved for other credit-based services

How credit card default happens

When you accept a credit card, you agree to certain terms. For example, you agree to make your minimum payment by the due date listed on your credit card statement. If you miss the minimum payment six months in a row, your credit card will be in default. Your credit card issuer will likely close your account and report the default to the credit bureaus.

In the months leading up to a default, your (late) payment status will be reported to the three major credit bureaus, and your credit score will be

impacted by the lateness of your payments. If you apply for any new credit cards or loans after a default, your application will likely be denied because creditors think you are at risk of defaulting on any new credit obligations. In fact, some lenders will not approve you at all until you have cleared up the default balance (or it drops off your credit report).

By the time your credit card defaults, you have likely accumulated hundreds of dollars in fees and interest charges. Unfortunately, your options for clearing up the credit card default may be limited because of the number of payments you have missed on your account. Had you

contacted your credit card issuer sooner, you may have been able to work out an arrangement to make payments on the past due balance and bring your account back into good standing. At this point, your credit card issuer will expect the account to be paid in full.

The following are the limited ways in which you can deal with credit card default:

- **Pay the account in full** (if you have the money). First, try negotiating a pay for delete where the credit card issuer removes the account from your credit report in exchange for payment. Some creditors may agree while others will not, but you will not know if you do not ask.

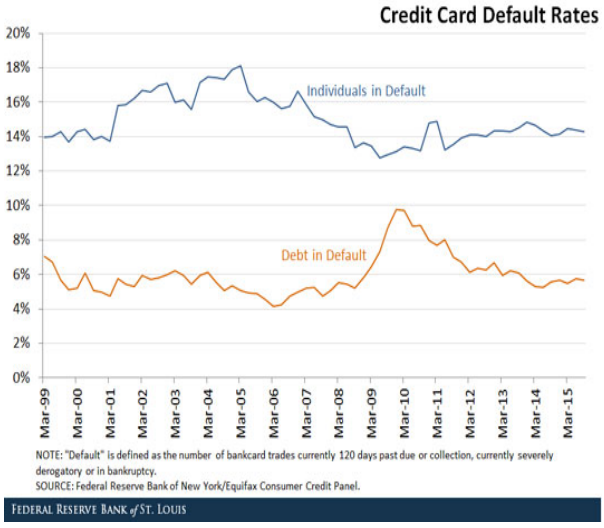
- **Settle the account for less than the amount due**. It may be possible to settle the debt. The creditor does not have to accept an amount lower than the balance due, but some can be persuaded.

- **File for bankruptcy**. Depending on the extent of the default and any other debts you have, you may consider filing bankruptcy to either restructure your debt and make it more affordable or to have it discharged. Note that bankruptcy stays on your credit report for 7 to 10 years, so it is not a decision to enter lightly.

- **Do nothing**. You can try ignoring the account, but note that the creditor can still pursue you for the debt, list it on your credit report, and may even sue you as long as the statute of limitations is in effect.

Problem Statement

This is our ML Supervised Classfication Project.We will be looking at data set of credit card clients. We will predict if the client will default or not.The dataset contains information about default payments,credit data, history of payments, bill statements of credit card of the client in Taiwan.



Data Fields

- ID: ID of each client
- LIMIT_BAL: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- SEX: Gender(1=male,2=female)
- EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown,6=unknown)
- MARRIAGE:Marrital status (1=married,2=single,3=others)
- AGE:Age (year)
- PAY_0,PAY_2,PAY_3,PAY_4,PAY_5: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; . . .;X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
- BILL_AMT1,BILL_AMT2,BILL_AMT3,BILL_AMT4,BILL_AMT5,BILL_AMT6: Amount of bill statement (NT dollar). X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; . . .; X17 = amount of bill statement in April, 2005.
- PAY_AMT1,PAY_AMT2,PAY_AMT3,PAY_AMT4,PAY_AMT5,PAY_AMT6:Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .;X23 = amount paid in April, 2005.
- default.payment.next.month: Default payment(1=Yes, 0= No)

Introduction

This is our Classification Capstone Project, hence we will be looking into multiple classification models and try to come up with a best model at the end of this project. We are only focussing on all that algorithm which has been taught to us till now in our class. SVM, KNN, Gradient Naive Bayes and a few more algorithm we have implemented in this capstone project.

ML pipeline to be followed

- Dataset Inspection
- Data Cleaning
- Exploratory Data Analysis
- Feature Engineering
- Handling Class Imbalance
- One Hot Encoding
- Baseline Model with default parameters
- Performance Metrics
- Optimization of the Model
- Feature Importance
- Hyperparameter Tuning
- Analyze Results

Dataset Inspection

We loaded the dataset using the given csv files. We checked the general information about data. We observe that the data contains 30001 records and 25 features.

Initial datatype of all the features were category in nature. After going throught the dataset,we initially performed few actions as follows-

1.Renamed the columns using the first record given in the dataset and dropped the first record.

2.Renamed our target variable to ‘defaulter’ and PAY_0 column to ‘PAY_1’

3.Converted the datatypes of all columns from object to int datatype because all columns contains numerical values.

4.Drop the ID column from the dataset,since its not an influential feature in our modeling

The Dataset Inspection Summary

1. The average credit card limit/consumer credit amount is 167484.32(NT Dollars)
2. The maximum number of credit card holders were females in Taiwan.
3. The given dataset consist of 30000 rows and 24 columns
4. The feature 'defaulter' is our dependent/target feature
5. The most number of credit card holders were having university degree education.
6. The most of the customers marriage status was Single, who carries a credit card.

Data Cleaning

Data cleaning is one of the important parts. We remove the unwanted observations, fix the structural errors, manage the unwanted outliers and handle the missing data.

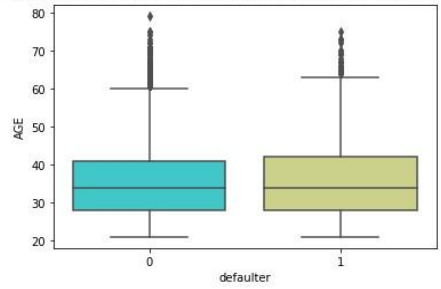
We had received a clean dataset without any missing values but consist of few duplicate rows (35 records).We have successfully dropped the duplicated rows .

One of the most important steps as part of data preprocessing is detecting and treating the outliers as they can negatively affect the analysis and the training process of a machine learning algorithm resulting in lower accuracy.

An outlier may occur due to the variability in the data, or due to experimental error/human error.Box plots are a visual method to identify outliers. It is a data visualization plotting function. It shows the min, max, median, first quartile, and third quartile

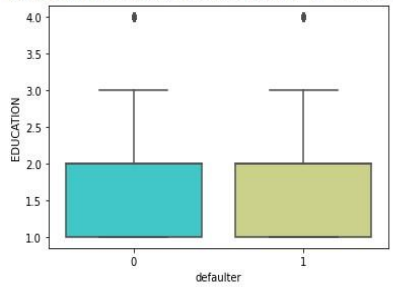
```
sns.boxplot(x='defaulter',y='AGE',data=credit_df,palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f21457bc490>
```



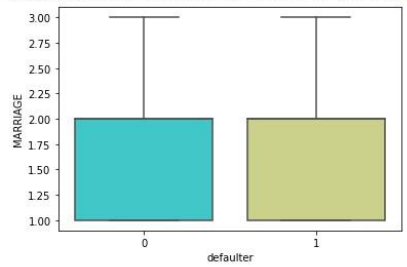

```
sns.boxplot(x='defaulter',y='EDUCATION',data=credit_df,palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2148683f90>
```



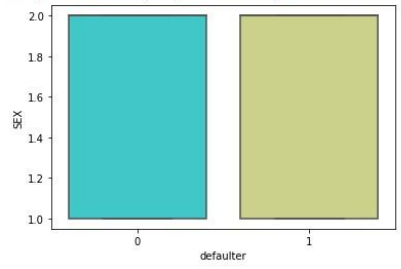

```
sns.boxplot(x='defaulter',y='MARRIAGE',data=credit_df,palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2148d5f650>
```



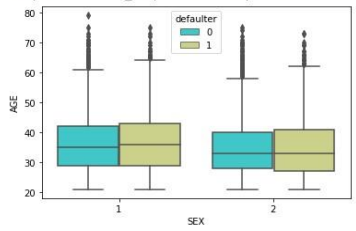

```
sns.boxplot(x='defaulter',y='SEX',data=credit_df,palette='rainbow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2145a1a590>
```



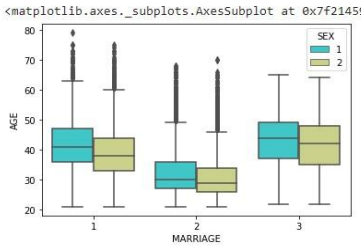

```
sns.boxplot(x='SEX',hue='defaulter', y='AGE',data=credit_df,palette="rainbow")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f213a31e090>
```




```
sns.boxplot(x='MARRIAGE',hue='SEX', y='AGE',data=credit_df,palette="rainbow" )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f214599d310>
```



Outliers can be visualised as the dots outside the whiskers in the boxplots. The majority of the continuous and discrete variables seem to contain outliers. In addition, the majority of the variables are not normally distributed.

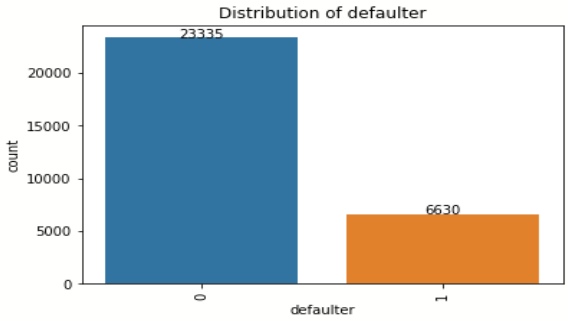
We are not removing any of the outliers as of now because it may lead to information loss since our dataset is small and our model will not be able to train well if we remove few records due to outliers .

It's not always adviced to remove outliers inorder to increase the accuracy of the model,here our data records are compartively low in number.Hence we have decided not to remove the outliers as of now.

Finally we also find out that our target variable is highly imbalance.

```
#Check for imbalance data in Target Feature
(credit_df['defaulter'].value_counts()/len(credit_df['defaulter'])*100)
```

```
0    77.874187
1    22.125813
Name: defaulter, dtype: float64
```



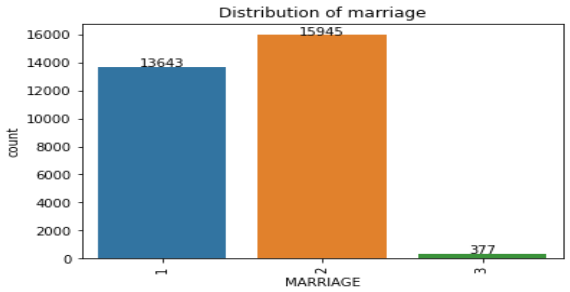
Distribution of target classes is highly imbalanced, non-defaults far outnumber defaults. This is common in these datasets since most people pay credit cards on time

The target variable consist of imbalance data with 77.87% Non defaulters(0 value) and 22.12% 1 defaulters (1 value).

Exploratory Data Analysis

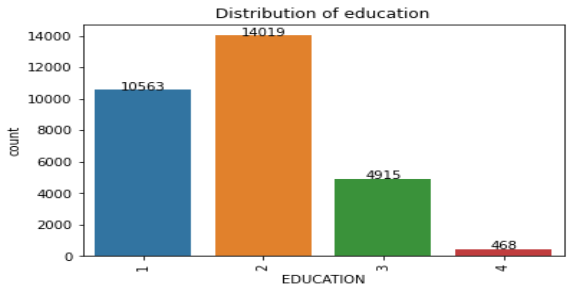
Univariate Analysis

MARRIAGE



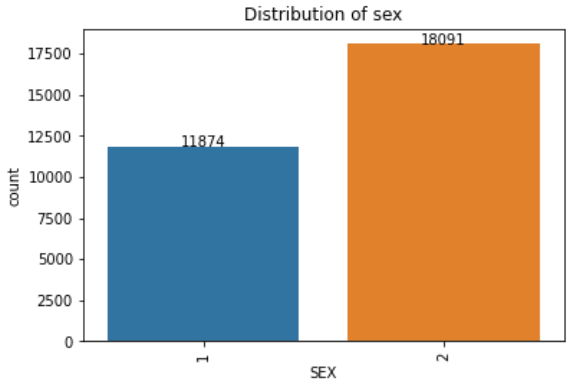
From the above graph,we can conclude that the most number of credit card holders were not married(Single)

EDUCATION



The above graph conveys us that the most number of customers were holding a university degree as their educational qualification followed by graduates degree holders

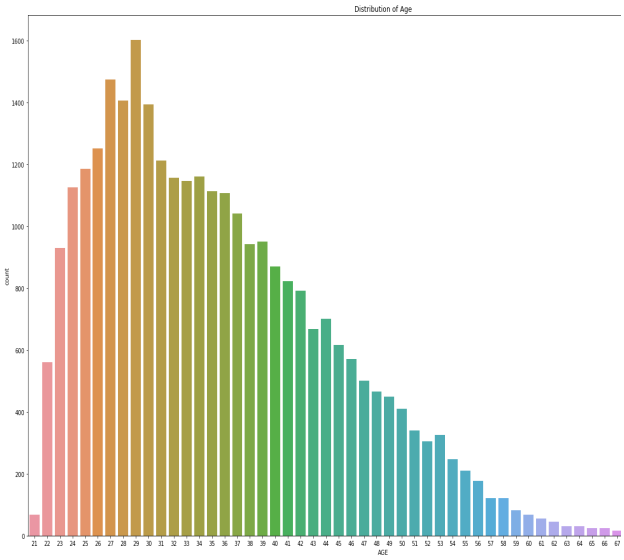
SEX



The above figure tells us that the most number of credit card holders in Taiwan were females.

Hence proved the famous saying,the females usually do more shopping compared to mens from the above given data

AGE



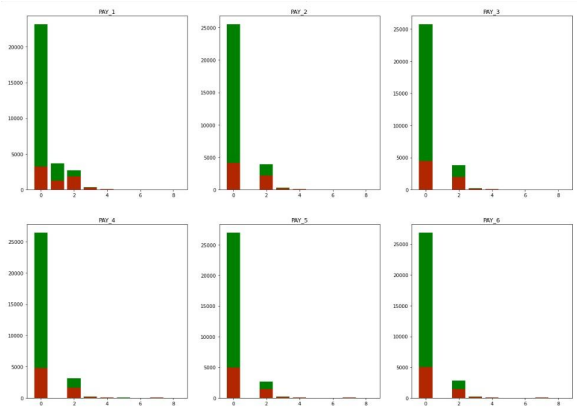
The above figure shows the coutplot of Age column

The most frequent age of the credit card holder is between 25-34.

The highest proportion of credit card holders were youth in the age of 29.

Hence we can understand that mostly credit cards were popular among youths of taiwan than the older people.

Repayment Status (Payment History)



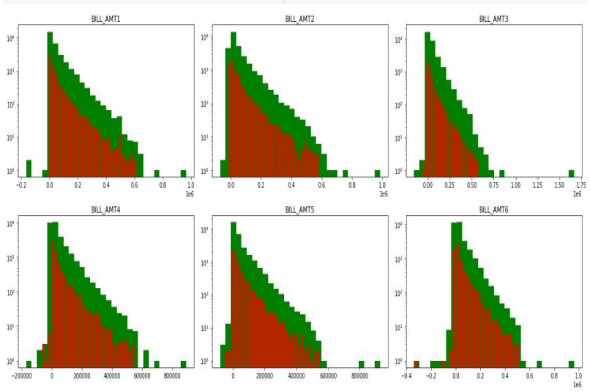
The above figure shows the bar plot of payment history status for past six months starting from September to April , which show the count of defaulters and non-defaulter

The payment history status consist of unique values like 0- No delay in payment 2- payment delay of 2 months 4 - payment delay of 4 months 6- payment delay of 6 months 8- payment delay of 8 months

The green bins shows the count of payment status of all the customers (both defaulters and non-defaulters). On the otherhand, the red bins shows the count of payment status explicitly for the customers who were defaulters.

From the above graph, we can conclude that if the payment status is greater than 2 months,then there is a 90% chance of the customer to default the payment

Previous Amount Paid (PAY_AMT)



The above histogram shows the distribution of Bill amount generated for each month explicitly for defaulters

The green bins in the histograms shows the bill_amount for all the customers from September to April Month.

The red bins in the histogram tells the payment amount of customers who were actually a defaulter from September to April Month.

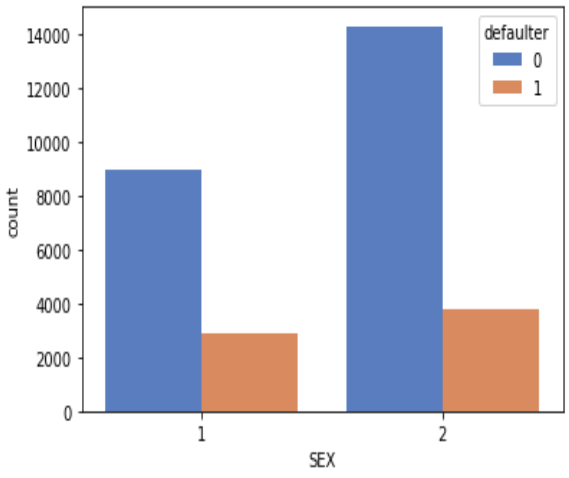
Above plot indicates that there is higher proportion of clients for whom the bill amount is high but payment done against the same is very low.

From the above plot , we can say that bill amount variable based on different months from April to September not able to give us a clear insight from the plot whether the customer will be defaulter or not compared to payment amount variable.

Hence payment amount features are more significant variables compared to the bill amount features.

BI-VARIATE ANALYSIS

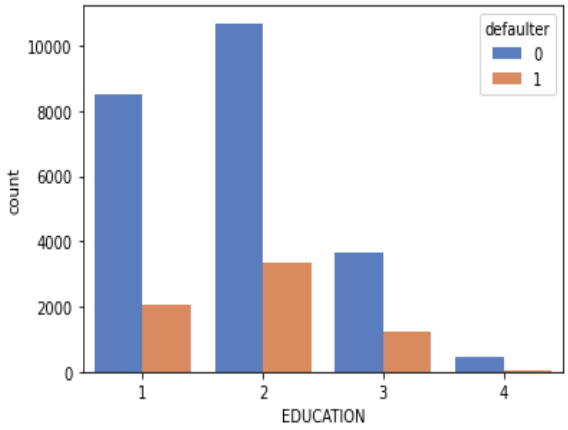
Defaulter vs Sex



It is evident from the above count plot output that males have overall less default payment rate w.r.t females

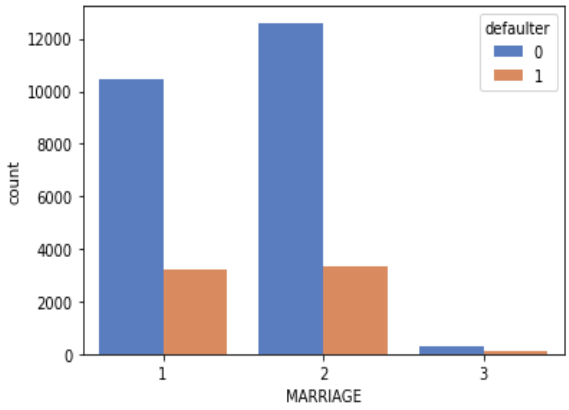
Both in Defaulter and Non-Defaults count, females were having higher proportion (Sex=2)

Defaulter vs Education



The credit card holders with a university degree were the customers with the highest number of default payment rate compared to other degree holders.

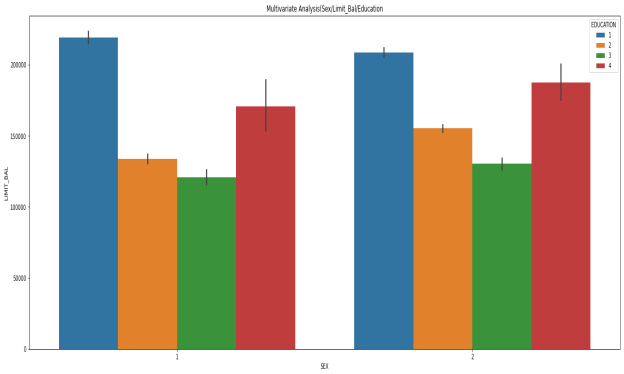
Defaulter vs Marriage



It is evident from the above plot that both the credit card holders who were singles and married used to do default in payments.

Multi-variate Analysis

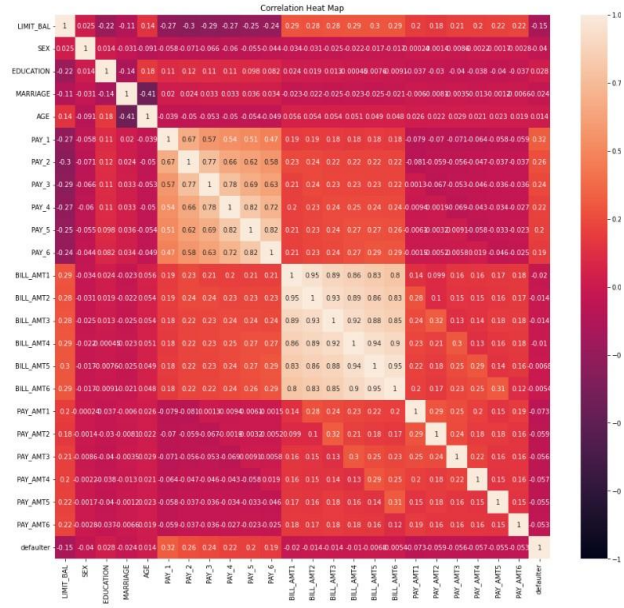
SEX VS CREDIT LIMIT VS EDUCATION



The above figure tells us that the highest LIMIT_BAL/credit limit amount is given to the graduate education credit card holders in both the sex.

On the contrary, the least credit limit amount is given to the high school education credit card holders in both the sex.

Heat Map



We can see that next month default prediction is dependent on payment repayment status of past six months of all the features given to us. But there is multicollinearity between the Payment Repayment Status features.

We will first train model wil all the features and try reducing the non-important features.

Feature Engineering

Step-1.Buliding New Features

We have derived few features from bill amount and pay amount columns

Few new features derived are-

Total bill amount feature ,total paid amount and pending payment amount.

```
credit_df['Total_Bill_AMT']=credit_df['BILL_AMT1'] + credit_df['BILL_AMT2']+ credit_df['BILL_AMT3'] + credit_df['BILL_AMT4'] + credit_df['BILL_AMT5'] + credit_df['BILL_AMT6']
credit_df['Total_Paid_AMT']= credit_df['PAY_AMT1'] + credit_df['PAY_AMT2']+ credit_df['PAY_AMT3'] + credit_df['PAY_AMT4'] + credit_df['PAY_AMT5'] + credit_df['PAY_AMT6']
credit_df['Pending_Payment_AMT']= credit_df['Total_Bill_AMT']- credit_df['Total_Paid_AMT']
```

We have derived a new feature (total bill amount) using the available features in our dataset bill amount for past six months.

We have also derived new features like (total paid amount) by adding pay amount for past six months

Another important feature derived was pending payment amount by subtracting

newly created feature total bill amount and total paid amount .

We have also implemented binning on one feature i.e AGE column inorder to better train our model efficiently .Then we have encoded the column and given a new name to the feature as AGE_Encoded.

```
# Bin 'AGE' data to 6 groups
bins= [21,30,40,50,60,70,80]
labels = list(range(6))
credit_df['AGE'] = pd.cut(credit_df['AGE'],bins=bins, labels=labels,right=False)

from sklearn.preprocessing import LabelEncoder
# creating instance of labelencoder
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
credit_df['AGE_Encoded'] = labelencoder.fit_transform(credit_df['AGE'])
credit_df['AGE_Encoded'].value_counts()
```

```
1    11226
0     9603
2     6456
3     2341
4      314
5       25
Name: AGE_Encoded, dtype: int64
```

Step-2 Replacing few undefined categories/classes in different independent features

We have came across few features like marriage and education with undefined categories present inside its records. We have decided to put these undefined classes into a related/similar group of classes in the respective columns.

Replaced 0 class/category in marriage column into class/category 3='others' because 0 class is not defined in the marriage variable.

Similarly, we replaced few undefined classes like 0, 5 and 6 in education column into class 4='others' using a fuction name education.

```
credit_df['MARRIAGE'].replace(to_replace=0,value=3,inplace=True)
```

```
def education(value):
  if value> 4:
    value = 4
  elif value==0:
    value= 4
  else:
    value
  return value
```

Replaced negative values in payment history columns

```
payment_list = ['PAY_1','PAY_2','PAY_3','PAY_4','PAY_5','PAY_6']
```

```
for var in payment_list:
  credit_df.loc[(credit_df[var] == -1) | (credit_df[var]==-2),var]=0
```

We have grouped all the negative class in the payment history column to class 0,which tells the payment status as no delay in payment.

Thought Process

What actually will drive the customer to default the payment?

Do we think of all the given variables when we are trying to find out a defaulter? e.g. When we think about the defaulter, do we care about its 'Male or Female'.Do gender has any infulence on our target variable?.

If so, how important would this variable be? e.g. What is the impact of having that SEX variable? Whether the variable has a postive or negative correlation with our target variable defaulter.

Is this information already described in any other variable?

These are few questions you ask before predicting whether the customer will be defaulter or not -

What is the customer past payment history?

What is his credit limit and whether he is paying the bills in a timely manner?

What is his/her age and educational background?

The steps followed in this Classification Project were as follows –

Handling Class Imbalance

Baseline Model with all features

Performance Metrics

Optimization of the Model

Feature Importance/Selection

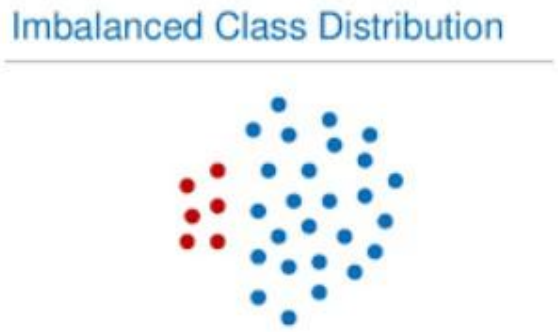
Hyperparameter Tuning

Balance Dataset

Imbalanced Dataset: — If there is the very high different between the positive values and negative values. Then we can say our dataset in Imbalance Dataset.

Handling Imbalanced Target Variable using S.M.O.T.E

Balanced Dataset: — Let's take a simple example if in our data set we have positive values which are approximately same as negative values. Then we can say our dataset in balance



SMOTE (synthetic minority oversampling technique) is one of the most commonly used oversampling methods to solve the imbalance problem.

It aims to balance class distribution by randomly increasing minority class examples by replicating them.

SMOTE synthesises new minority instances between existing minority instances. It generates the **virtual training records by linear interpolation** for the minority class. These synthetic training records are generated by randomly selecting one or more of the k-nearest neighbors for each example in the minority class. After the oversampling process, the data is reconstructed and several classification models can be applied for the processed data.

The Original dataset shape before sampling - **29965**

The Resampled dataset shape **- - 46670**

We again split the resampled dataset to train and test data. Thereby performed scaling using Standard Scalar before training the model.

One Hot Encoding

One hot encoding is one method of converting data to prepare it for an algorithm and get a better prediction. With one-hot, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns. Each integer value is represented as a binary vector. All the values are zero, and the index is marked with a 1.

Take a look at this chart for a better understanding:

Type
AA
AB
CD
AA

Onehot encoding →

Type	AA_Onehot	AB_Onehot	CD_Onehot
AA	1	0	0
AB	0	1	0
CD	0	0	1
AA	0	0	0

One hot encoding is useful for data that has no relationship to each other. Machine learning algorithms treat the order of numbers as an *attribute of significance*. In other words, they will read a higher number as better or more important than a lower number.

While this is helpful for some ordinal situations, some input data does not have any ranking for category values, and this can lead to issues with predictions and poor performance. That's when one hot encoding saves the day.

One hot encoding makes our training data more useful and expressive, and it can be rescaled easily. By using numeric values, we more easily

determine a probability for our values. In particular, one hot encoding is used for our output values, since it provides more nuanced predictions than single labels.

From our dataset, we have applied one-hot encoding on following features-

MARRIAGE,SEX,EDUCATION,

PAY_(0-6)

Model Buliding

1.Data Preparation for Model Buliding

Data Preparation is the process of cleaning and transforming raw data to make predictions accurately through using ML algorithms. Although data preparation is considered the most complicated stage in ML, it reduces process complexity later in real-time projects.

Everyone must explore a few essential tasks when working with data in the data preparation step. These are as follows:

Data cleaning: This task includes the identification of errors and making corrections or improvements to those errors.

Feature Selection: We need to identify the most important or relevant input data variables for the model.

Data Transforms: Data transformation involves converting raw data into a well-suitable format for the model.

Feature Engineering: Feature engineering involves deriving new variables from the available dataset.

Dimensionality Reduction: The dimensionality reduction process involves converting higher dimensions into lower dimension features without changing the information.

Each machine learning project requires a specific data format. To do so, datasets need to be prepared well before applying it to the projects. Sometimes, data in data sets have missing or incomplete information, which leads to less accurate or incorrect predictions. Further, sometimes data sets are clean but not adequately shaped, such as aggregated or pivoted, and some have less business context. Hence, after collecting data from various data sources, data preparation needs to transform raw data. Below are a few significant advantages of data

preparation in machine learning as follows:

- It helps to provide reliable prediction outcomes in various analytics operations.
- It helps identify data issues or errors and significantly reduces the chances of errors.
- It increases decision-making capability.
- It reduces overall project cost (data management and analytic cost).
- It helps to remove duplicate content to make it worthwhile for different applications.
- It increases model performance.

Our approach in this project will be to initially to take into consideration all the features present in the dataset .

The next step will be to split the independent and dependent variable in the ratio of 70:30 to training and test data.

Then we will perform Standard Scaler on the Independent feature inorder to bring all the variables into a common scale by initially using fit_transform on training data and lateron transform the test data.

After splitting and scaling the dataset, we build a function to train the model using multiple supervised learning algorithms and classify the different models on the basis of their test accuracy.

2.Spliting the dataset into train and test(70:30)

For ML models to give reasonable results, we not only need to feed in large quantities of data but also have to ensure the quality of data.

Though making sense out of raw data is an art in itself and requires good feature engineering skills and domain knowledge (in special cases), the quality data is of no use until it is properly used. The major problem which ML/DL practitioners face is how to divide the data for training and testing. Though it

seems like a simple problem at first, its complexity can be gauged only by diving deep into it.

Poor training and testing sets can lead to unpredictable effects on the output of the model. It may lead to overfitting or underfitting of the data and our model may end up giving biased results.

The data should ideally be divided into 3 sets – namely, train, test, and holdout cross-validation set.

Train Set:

The train set would contain the data which will be fed into the model. In simple terms, our model would learn from this data. For instance, a Regression model would use the examples in this data to find gradients in order to reduce the cost function. Then these gradients will be used to reduce the cost and predict data effectively.

Cross validation Set:

The development set is used to validate the trained model. This is the most important setting as it will form the basis of our model evaluation. If the difference between error on the training set and error on the dev set is huge, it means the model as high variance and hence, a case of over-fitting.

Test Set:

The test set contains the data on which we test the trained and validated model. It tells us how efficient our overall model is and how likely is it going to predict something which does not make sense. There are a plethora of evaluation metrics (like precision, recall, accuracy, etc.) which can be used to measure the performance of our model.

3. Standardization (Scaling the data)

As we know most of the supervised and unsupervised learning methods make decisions according to the data sets applied to them and often the algorithms calculate the distance between the data

points to make better inferences out of the data.

In the machine learning algorithms if the values of the features are closer to each other there are chances for the algorithm to get trained well and faster instead of the data set where the data points or features values have high differences with each other will take more time to understand the data and the accuracy will be lower.

So if the data in any conditions has data points far from each other, scaling is a technique to make them closer to each other or in simpler words, we can say that the scaling is used for making data points generalized so that the distance between them will be lower.

As we know, most of the machine learning models learn from the data by the time the learning model maps the data points from input to output. And the distribution of the data points can be different for every feature of the data. Larger differences between the data points of input variables increase the uncertainty in the results of the model.

The machine learning models provide weights to the input variables according to their data points and inferences for output. In that case, if the difference between the data points is so high, the model will need to provide the larger weight to the points and in final results, the model with a large weight value is often unstable. This means the model can produce poor results or can perform poorly during learning.

In this project we have used Standard Scalar inorder to perform the scaling.

```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train[features]=sc.fit_transform(X_train[features]) # fit on training data columns and transform the training data columns
X_test[features]=sc.transform(X_test[features]) # transform the testing data columns
```

Building different models using default parameter and selecting the best model after tunning

We have used a function to compare different models based on the test accuracy of different models.We have following algorithm in our project to build the model

Logistic Regression, Gaussian Naive Bayes, Support Vector Machines, KNN , XGBoost and Random Forest to select the best baseline model.

```
LogReg  training accuracy score: 0.8718
LogReg test accuracy score: 0.8755
SVM  training accuracy score: 0.8714
SVM test accuracy score: 0.8719
KNN  training accuracy score: 0.8918
KNN test accuracy score: 0.8434
XGBoost  training accuracy score: 0.8724
XGBoost test accuracy score: 0.8757
RF  training accuracy score: 0.9983
RF test accuracy score: 0.8784
NB  training accuracy score: 0.6444
NB test accuracy score: 0.644
```

We can clearly classify based on above test accuracy score of different models that the best baseline model without doing any hyperparametric tunning is the Random-Forest model and the worst baseline model is the Naive Bayes Model.

But two of our models were overfitting,i.e the KNN model and Random Forest model.We cannot take an overfit model for production and handling the overfit is really important.We can handle them using hyperparameteric tuning but it will compromise our accuracy score as well.Hence Optimization of model is one of the key task.

But before getting into Optimization,let dive into various evaluation metrics of our XGBoost Model inorder to get a better insight from our baseline model.

Optimizing the Recall score

What are we optimizing for?

Using accuracy score as a evaluation metrics for such highly imbalanced dataset is not a good measure of classifier performance.

In such cases, evaluation metrics like ROC-AUC curve are a good indicator of

classifier performance. It is a measure of how good model is at distinguishing between various class. Higher the ROC-AUC score, better the model is at predicting 0s as 0s and 1s as 1s. Just to remind, ROC is a probability curve and AUC represents degree or measure of separability. Apart from this metric, we will also check on recall score, false-positive (FP) and false-negative (FN) score as we build our classifier

Ideally, we do not want to miss any potentially defaults to fall through the cracks, so our optimal model will minimize False Negatives (optimize Recall Score)

Our aim will be to reduce the false negative count because it will enhance the recall evaluation metric of our model. Since our problem statement is to predict the credit card defaulter,hence recall is our key parameter to focus among the evaluation metrics.

The optimization of recall value basically help us in not missing any credit card defaulter from our analaysis by reducing the false negative rate in our model.

Inorder to optimize our model recall,we follows -

Hyperparameteric Tunning using RandomizedCV and Grid SearchCV using a cross validation value of three.

HyperParameter Tuning

A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.

There is another kind of parameter, known as **Hyperparameters**, that cannot be directly learned from the regular training process. They are usually fixed before the actual training process begins. These parameters express important properties of the model such as its complexity or how fast it should learn.

Models can have many hyperparameters and finding the best combination of parameters can be treated as a search problem. The two best strategies for Hyperparameter tuning are:

GridSearchCV

In GridSearchCV approach, the machine learning model is evaluated for a range of hyperparameter values. This approach is called GridSearchCV,

because it searches for the best set of hyperparameters from a grid of hyperparameters values.

In this project,we have used GridSearchCv and RandomisedCV for finding the best parameters for our classification model algorithm like RandomForest , KNN,SVM and Xgboost model to find the best parameter inorder to get a optimal and best model.

Logistic Regression

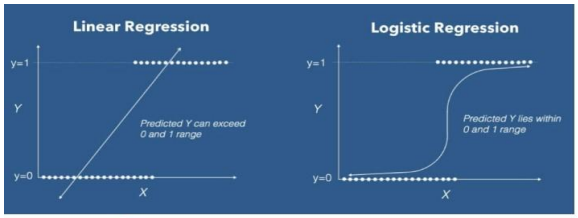
Logistic Regression is a “Supervised machine learning” algorithm that can be used to model the probability of a certain class or event. It is used when the data is linearly separable and the outcome is binary or dichotomous in nature.

That means Logistic regression is usually used for Binary classification problems.

Binary Classification refers to predicting the output variable that is discrete in two classes.

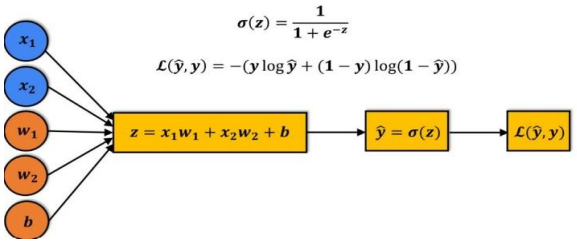
A few examples of Binary classification are Yes/No, Pass/Fail, Win/Lose, Cancerous/Non-cancerous, etc.

A linear equation (z) is given to a sigmoidal activation function (σ) to predict the output (ŷ).



Source

The image that depicts the working of the Logistic regression model



To evaluate the performance of the model, we calculate the loss. The most commonly used loss function is the mean squared error.

But in logistic regression, as the output is a probability value between 0 or 1,

mean squared error wouldn't be the right choice. So, instead, we use the cross-entropy loss function.

The cross-entropy loss function is used to measure the performance of a classification model whose output is a probability value.

Inorder to optimize our logistic regression model we have used hyperparameter tuning using GridSearchCV with a cross validation value of 3.

The parameters we have selected in logistic regression model for tuning were penalty,C_value,class weight and we have used saga as the solver.

The best parameter after tunning were as follows-

```
{'C': 0.001, 'class_weight': {0: 10, 1: 15}, 'penalty': 'l2'}
```

The accuracy ,precision,recall ,f1score and ROC score with default parameter and after hyperparameter tuning as follows-

	Model	Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.875509	0.942163	0.800798	0.865748	0.875696
1	Logistic Regression Tuned	0.866152	0.902126	0.822172	0.860295	0.866263

The confusion and classification report after tuning as follows-

Check for Overfit

```
[93] y_pred_lr1= grid_lr.predict(X_test)  # predicting on test data
     y_pred_train_lr1 = grid_lr.predict(X_train) # predicting on train data

     print(f'Train Accuracy: {accuracy_score(y_train,y_pred_train_lr1)}')
     print(f'Test Accuracy: {accuracy_score(y_test, y_pred_lr1)}')
     print(f'Area Under Curve: {roc_auc_score(y_test, y_pred_lr1)}')

     Train Accuracy: 0.8633566990112951
     Test Accuracy: 0.8661524176844511
     Area Under Curve: 0.8662626375082555
```

Classification Report

```
[94] print(classification_report(y_test, y_pred_lr1))

                  precision    recall  f1-score   support

               0       0.84      0.91      0.87      6983
               1       0.90      0.82      0.86      7018

        accuracy                           0.87     14001
       macro avg       0.87      0.87      0.87     14001
    weighted avg       0.87      0.87      0.87     14001
```

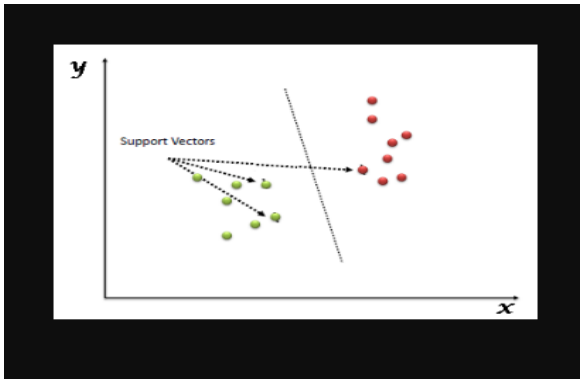
Confusion Matrix

```
cm=confusion_matrix(y_test, y_pred_lr1)
print(cm)

[[6357  626]
 [1248 5770]]
```

Support Vector Machine (SVM)

Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification

or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).

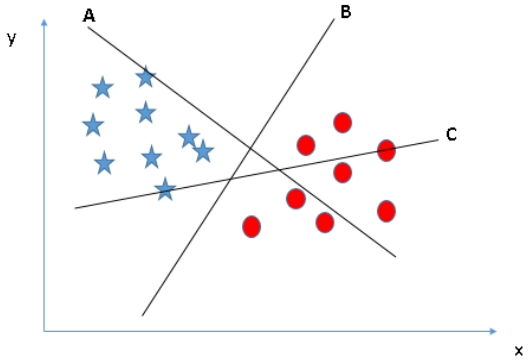


Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

"How can we identify the right hyper-plane?".

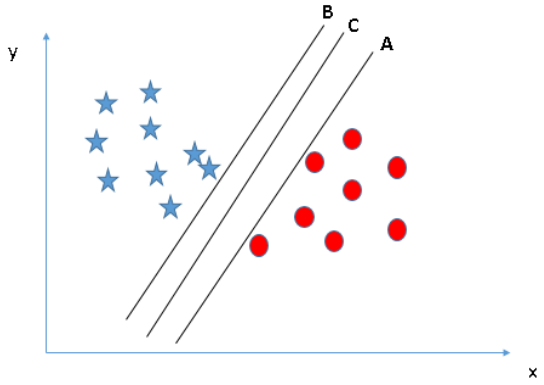
Let's understand:

Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.

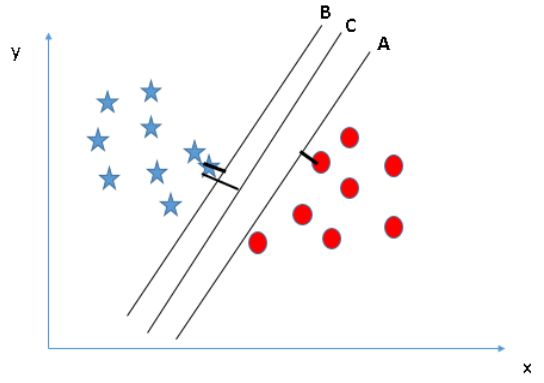


You need to remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.

Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B, and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

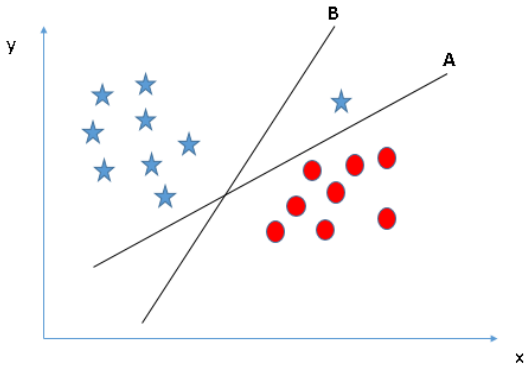


Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin. Let's look at the below snapshot:



Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

Identify the right hyper-plane (Scenario-3):Hint: Use the rules as discussed in previous section to identify the right hyper-plane



Some of you may have selected the hyper-plane B as it has higher margin compared to A. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

Inorder to optimize our support vector machine model we have used hyperparameter tuning using

GridSearchCV with a cross validation value of 3.

The parameters we have selected in the svm model for tuning were C_value,gamma and we have used rbf as the kernel.

The best parameter after tunning were as follows-

```
Best Parameters: {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
```

The accuracy ,precision,recall ,f1score and ROC score with default parameter and after hyperparameter tuning as follows-

Support Vector Machine	0.871866	0.943463	0.791821	0.861016	0.872067
Support Vector Machine Tuned	0.874795	0.938239	0.803078	0.865413	0.874974

The confusion and classification report after tuning as follows-

Check for Overfit

```
[ ] y_pred_test_svm1 = grid_svm.predict(X_test)
    y_pred_train_svm1 = grid_svm.predict(X_train)
    print(f'Train Accuracy: {accuracy_score(y_train, y_pred_train_svm1)}')
    print(f'Test Accuracy: {accuracy_score(y_test, y_pred_test_svm1)}')
    print(f'Area Under Curve: {roc_auc_score(y_test, y_pred_test_svm1)}')

    Train Accuracy: 0.8788453885946922
    Test Accuracy: 0.8747946575244625
    Area Under Curve: 0.8749743861522264
```

Classification Report

```
[ ] print(classification_report(y_test,y_pred_test_svm1))

                  precision    recall  f1-score   support

               0       0.83      0.95      0.88      6983
               1       0.94      0.80      0.87      7018

        accuracy                           0.87     14001
       macro avg       0.88      0.87      0.87     14001
    weighted avg       0.88      0.87      0.87     14001
```

Confusion Matrix

```
[ ] cm = confusion_matrix(y_test, y_pred_test_svm1)
    print(cm)

    [[6612  371]
     [1382 5636]]
```

K Nearest Neighbour Classifier(KNN)

K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbor) suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new Datapoint.

The algorithm's learning is:

1. Instance-based learning: Here we do not learn weights from training data to predict output (as in model-based algorithms) but use entire training instances to predict output for unseen data.

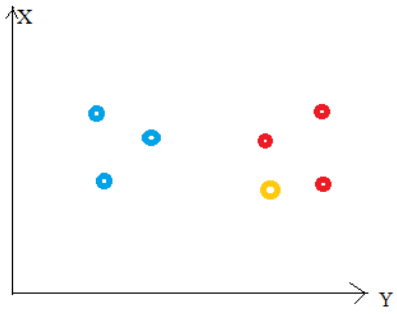
2. Lazy Learning: Model is not learned using training data prior and the learning process is postponed to a time when prediction is requested on the new instance.

3. Non -Parametric: In KNN, there is no predefined form of the mapping function.

Principle:
Consider the following figure. Let us say we have plotted data points from our training set on a two-dimensional feature space. As shown, we have a total of 6 data points (3 red and 3 blue). Red data points belong to ‘class1’ and blue data points belong to ‘class2’. And yellow data point in a feature space represents the new point for which a class is to be predicted. Obviously, we say it belongs to ‘class1’ (red points)

Why?

Because its nearest neighbors belong to that class!



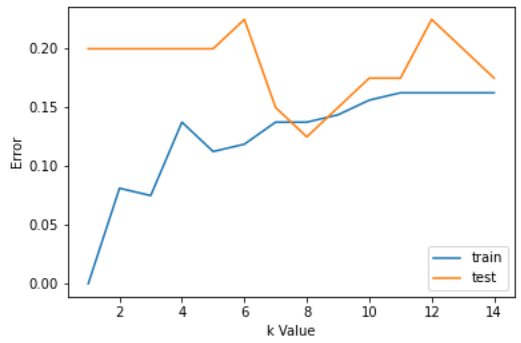
Yes, this is the principle behind K Nearest Neighbors. Here, nearest neighbors are those data points that have minimum distance in feature space from our new data point. And K is the number of such data points we consider in our implementation of the algorithm. Therefore, distance metric and K value are two important considerations while using the KNN algorithm. Euclidean distance is the most popular distance metric. You can also use Hamming distance, Manhattan distance, Minkowski distance as per your need. For predicting class/ continuous value for a new data point, it considers all the data points in the training dataset. Finds new data point’s ‘K’ Nearest Neighbors (Data points) from feature space and their class labels or continuous values.

For classification: A class label assigned to the majority of K Nearest Neighbors from the training dataset is considered as a predicted class for the new data point

Here, we do not learn weights and store them, instead, the entire training dataset is stored in the memory. Therefore, model representation for KNN is the entire training dataset.

K is a crucial parameter in the KNN algorithm. Some suggestions for choosing K Value are:

1. Using error curves: The figure below shows error curves for different values of K for training and test data.



Choosing a value for K

At low K values, there is overfitting of data/high variance. Therefore test error is high and train error is low. At K=1 in train data, the error is always zero, because the nearest neighbor to that point is that point itself. Therefore though training error is low test error is high at lower K values. This is called overfitting. As we increase the value for K, the test error is reduced.

But after a certain K value, bias/ underfitting is introduced and test error goes high. So we can say initially test data error is high(due to variance) then it goes low and stabilizes and with further increase in K value, it again increases(due to bias). The K value when test error stabilizes and is low is considered as optimal value for K. From the above error curve we can choose K=8 for our KNN algorithm implementation.

2. Also, domain knowledge is very useful in choosing the K value.

3. K value should be odd while considering binary(two-class) classification.

Inorder to optimize our KNN model we have used hyperparameter tuning using GridSearchCV with a cross validation value of 3.

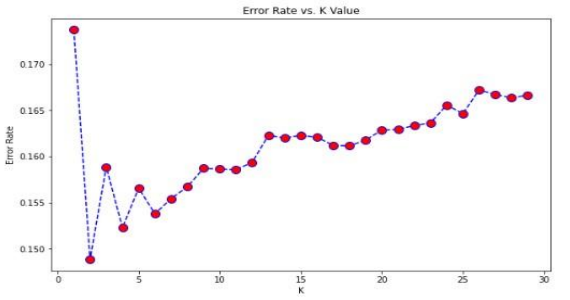
The parameters we have selected in svm model for tuning were n_neighbors and we have used uniform as the weights.

The best parameter after tunning were as follows-

```
{'n_neighbors': 15, 'weights': 'uniform'}
```

The accuracy ,precision,recall ,f1score and ROC score with default parameter and after hyperparameter tuning as follows-

K-Nearest Neighbour	0.843440	0.866161	0.813337	0.838918	0.843515
K-Nearest Neighbour Model Tuned	0.837726	0.886860	0.775150	0.827251	0.837883



The confusion and classification report after tuning as follows-

Check for Overfit

```
[ ] y_pred_test_knn = grid_knn.predict(X_test)
    y_pred_train_knn = grid_knn.predict(X_train)
    print(f'Train Accuracy: {accuracy_score(y_train, y_pred_train_knn)}')
    print(f'Test Accuracy: {accuracy_score(y_test, y_pred_test_knn)}')
    print(f'Area Under Curve: {roc_auc_score(y_test, y_pred_test_knn)}')

    Train Accuracy: 0.858857020416909
    Test Accuracy: 0.8377258767230912
    Area Under Curve: 0.83788269822894
```

Classification Report

```
print(classification_report(y_test, y_pred_test_knn))

              precision    recall  f1-score   support

           0       0.80      0.90      0.85      6983
           1       0.89      0.78      0.83      7018

    accuracy                           0.84     14001
   macro avg       0.84      0.84      0.84     14001
weighted avg       0.84      0.84      0.84     14001
```

Confusion Matrix

```
[ ] cm = confusion_matrix(y_test, y_pred_test_knn)
    print(cm)

    [[6289  694]
     [1578 5440]]
```

XGBoost Classifier

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

XGBoost (Extreme Gradient Boosting) belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework at its core. It is an optimized distributed gradient boosting library.

Inorder to optimize our XGBoost model we have used hyperparameter tuning using GridSearchCV with a cross validation value of 3.

The parameters we have selected in svm model for tuning were learning rate,gamma,max_depth,n_estimator,max_features,min_child weight and we have used reg_alpha.

The best parameter after tunning were as follows-

```
#Checking for best estimator
grid_search_xgb.best_estimator_
```

```
XGBClassifier(gamma=1, learning_rate=0.6, max_depth=8, max_features='sqrt',
              min_child_weight=10, n_estimators=153, random_state=42,
              reg_alpha=0.5)
```

The accuracy ,precision,recall ,f1score and ROC score with default parameter and after hyperparameter tuning as follows-

XGBoost	0.875080	0.930544	0.811342	0.866865	0.875240
XGBOOST Tuned	0.862938	0.894843	0.823311	0.857588	0.863038

The confusion and classification report after tuning as follows-

```
y_pred_test_rf = rf.predict(X_test)
y_pred_train_rf = rf.predict(X_train)
print(accuracy_score(y_train, y_pred_train_rf))
print(accuracy_score(y_test, y_pred_test_rf))
```

```
0.9983164467844133
0.8770087850867795
```

Classification Report

```
[121] print(classification_report(y_test, y_pred_test_rf))
```

```
              precision    recall  f1-score   support

           0       0.85      0.91      0.88      6983
           1       0.91      0.84      0.87      7018

    accuracy                           0.88     14001
   macro avg       0.88      0.88      0.88     14001
weighted avg       0.88      0.88      0.88     14001
```

Confusion Matrix

```
[122] cm = confusion_matrix(y_test, y_pred_test_rf)
      print(cm)
```

```
[[6368  615]
 [1107 5911]]
```

Feature Importance using Embedded Method in XGBoost

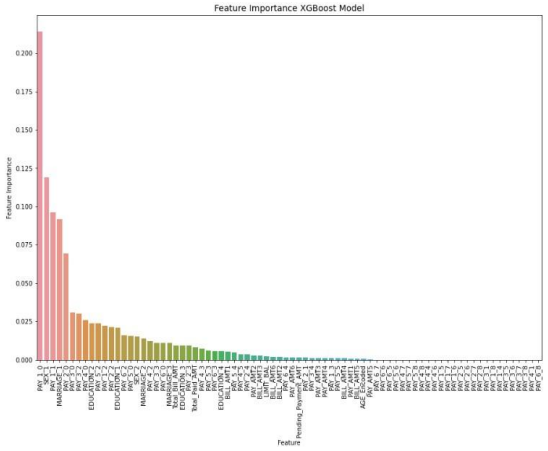
Feature (variable) importance indicates how much each feature contributes to the model prediction. Basically, it determines the degree of usefulness of a specific variable for a current model and prediction. For example, if we want to predict the weight of a person based on height, age, and name, it's obvious that the variable height will have the

strongest influence, while the variable name is not even relevant to the person's weight.These methods encompass the benefits of both the wrapper and filter methods, by including interactions of features but also maintaining reasonable computational cost. Embedded methods are iterative in the sense that takes care of each iteration of the model training process and carefully extracts those features which contribute the most to the training for a particular iteration.

There are many feature selection scores one could perform to determine which features are most useful. For this case, we will use Feature Importance.

Feature Importance is the process of assigning scores to each feature, depending on how useful it is in predicting the target variable

If we remove those features which are least important and keep the most important ones, this might allow us to better predict our target variable.



Random Forest

Random forest is a technique used in modeling predictions and behavior analysis and is built on decision trees. It contains many decision trees representing a distinct instance of the classification of data input into the random forest. The random forest technique considers the instances individually, taking the one with the majority of votes as the selected prediction.

The random forest algorithm is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees.Using RandomForest Ensemble technique to predict the trip_duration after applying

hyperparameter tuning with help of RandomizedSearchcv (cross-validation technique).

Inorder to optimize our Random Forest model we have used hyperparameter tuning using GridSearchCV with a cross validation value of 3.

The parameters we have selected in random forest model for tuning were max_depth,max_features,min_samples _leaf,min_samples_split,n_estimators and we have used gini as the criterion.

The best parameter after tunning were as follows-

```
#Checking for best estimator
grid_search.best_estimator_
```

```
RandomForestClassifier(max_depth=10, max_features=0.4, min_samples_leaf=5,
                       n_estimators=800)
```

The accuracy ,precision,recall ,f1score and ROC score with default parameter and after hyperparameter tuning as follows-

Random Forest	0.878437	0.908546	0.842263	0.874150	0.878528
Random Forest Classifier Tuned	0.868367	0.916600	0.811200	0.860685	0.868510

The confusion and classification report after tuning as follows-

```
y_pred_train_rf1 = grid_search.predict(X_train)
y_pred_test_rf1=grid_search.predict(X_test)
print(accuracy_score(y_train, y_pred_train_rf1))
print(accuracy_score(y_test, y_pred_test_rf1))
```

```
0.8864060730355995
0.868366545246768
```

Classification Report

```
print(classification_report(y_test, y_pred_test_rf1))
```

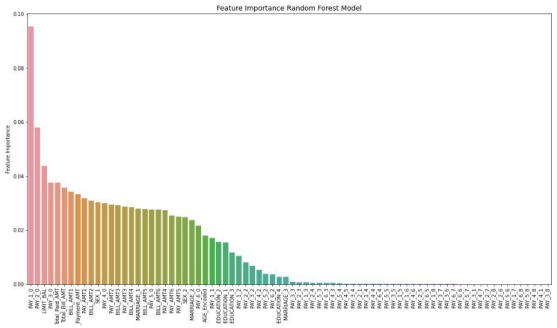
```
              precision    recall  f1-score   support

           0       0.83      0.93      0.88      6983
           1       0.92      0.81      0.86      7018

    accuracy                           0.87     14001
   macro avg       0.87      0.87      0.87     14001
weighted avg       0.87      0.87      0.87     14001
```

Confusion Matrix

```
cm = confusion_matrix(y_test, y_pred_test_rf1)
print(cm)
```

```
[[6465  518]
 [1325 5693]]
```

Feature Importance using Embedded Method in Random Forest



Feature (variable) importance indicates how much each feature contributes to

the model prediction. Basically, it determines the degree of usefulness of a specific variable for a current model and prediction. For example, if we want to predict the weight of a person based on height, age, and name, it's obvious that the variable height will have the strongest influence, while the variable name is not even relevant to the person's weight.

Overall, we represent feature importance using a numeric value that we call the score, where the higher the score value has, the more important it is. There are many benefits of having a feature importance score. For instance, it's possible to determine the relationship between independent variables (features) and dependent variables (targets). By analyzing variable importance scores, we would be able to find out irrelevant features and exclude them. Reducing the number of not meaningful variables in the model may speed up the model or even improve its performance.

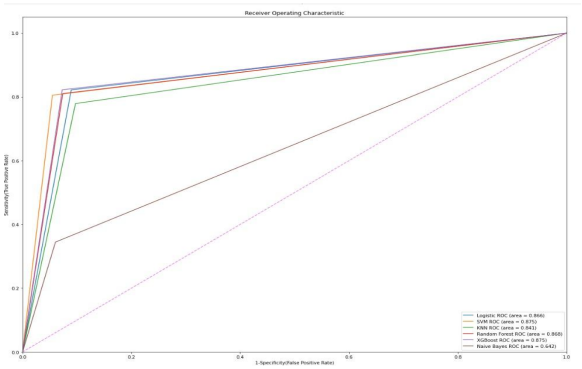
Also, feature importance is commonly used as a tool for ML model interpretability. From the scores, it's possible to explain why the ML model makes particular predictions and how we can manipulate features to change its predictions.

AUC_ROC Graph

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.

AUC-ROC curve is a performance measurement for the *classification problems* at *various threshold* settings. ROC is a probability curve, and AUC represents the degree or measure of separability. It tells how much model is

capable of distinguishing between classes. Higher the AUC, better the model is at *predicting 0s as 0s and 1s as 1s*. By analogy, Higher the AUC, better the model is at distinguishing between *patients with the disease and no disease*.



Performance Metrics Summary

	Model	Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.875223	0.940940	0.801368	0.865564	0.875408
1	Logistic Regression Tuned	0.866081	0.902237	0.821887	0.860189	0.866192
2	Support Vector Machine	0.872438	0.943089	0.793388	0.861786	0.872636
3	Support Vector Machine Tuned	0.875009	0.936526	0.805215	0.865921	0.875184
4	K-Nearest Neighbour	0.843725	0.864583	0.816044	0.839613	0.843795
5	K-Nearest Neighbour Model Tuned	0.840654	0.889504	0.778854	0.830510	0.840809
6	XGBoost	0.874295	0.930137	0.810060	0.865956	0.874456
7	XGBOOST Tuned	0.874509	0.918936	0.822172	0.867865	0.874640
8	Random Forest	0.877866	0.906681	0.843118	0.873745	0.877953
9	Random Forest Classifier Tuned	0.868009	0.916801	0.810202	0.860212	0.868154
10	Naive Bayes	0.641311	0.851408	0.344543	0.490566	0.642055

Conclusion

1. The maximum number of credit card holders in Taiwan were females and the average credit card limit provided by the credit card company to their respective customers was 167484.32(NT Dollars).

2. The most number of credit card holders were having university degree education and the most of the customers marriage status was Single, who carries a credit card in Taiwan.

3. The highest proportion of credit card holders were youth in the age of 29,thus we can conclude that mostly credit cards were popular among youths of taiwan than the older people.

4. The Correlation between features and target variable tells us the level of education and financial stability of the customers had high impact on the default rate.

5. The data also conveys us that the best indicator of delinquency is the behavior of the customer, which has been predominately seen in the past couple of months payment repayment

status .The heat map shows us the high correlation of payment repayment status with the target variable.

6. The females of age group 20-30 have very high tendency to default payment compared to males in all age brackets.

7. Compartively after hyperparameter tuning the XGBoost Model comes out to be the best model in terms of its AUC_ROC score(0.875) and Recall score(0.82) and we can predict with 87.45% accuracy, whether a customer is likely to default next month.

The reason is, XGBoost has high predictive power and is almost 10 times faster than the other gradient boosting techniques. It also includes a variety of regularization which reduces overfitting and improves overall performance.

8. The Second best model was the Support Vector Machine with a AUC_ROC score of 0.875 and a Recall score of 0.805 and we can predict with 87.5% accuracy, whether a customer is likely to default next month.

SVM model perform well when we select the proper kernel and the risk of overfitting is compartively less. We have used rbf kernel and after fine tunning the model we got Cost C parameter best value as 100 and gamma as 0.01.

9. But it would be worth using Logistic Regression model for production since we do not just need a reliable model with good ROC_AUC Score but also a model that is **quick and less complex.

10. Except Naive Bayes model,all the models have got really good ROC_AUC scores with a probability of 0.85 on an average.

11.The Random Forest and KNN models were really overfitting with default parameters and we handle the overfit in both these model by fine tuning the model.

12.Demographics: we see that being Female, More educated, Single and between 30-40years old means a customer is more likely to make payments on time.

References

https://www.stepchange.org/debt-info/debt-collection/default-notices-and-missed-payments.aspx

https://www.geeksforgeeks.org/splitting-data-for-machine-learning-models/

https://analyticsindiamag.com/why-data-scaling-is-important-in-machine-learning-how-to-effectively-do-it/

https://www.ibm.com/cloud/learn/random-forest

https://www.geeksforgeeks.org/xgboost/

https://medium.com/analytics-vidhya/what-is-balance-and-imbalance-dataset-89e8d7f46bc5

https://www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/

https://www.educative.io/blog/one-hot-encoding

Challenges

- Extracting new features from existing features were a bit tedious job to do.
- Handling Imbalance data in Target features
- There were few undefined data records present in the dataset and few duplicate records
- The Hyperparametric tuning using GridSearch cv was really time consuming task and required lots of patience until it get executed
- Selecting different parameter for hyperparameter tuning and finding the best parameters has to follow a trial n error technique,which is again a challenging job.