PGPDSE FT Capstone Project – Interim Report

Project Group Info:

BATCH DETAILS	DSE Offline Bangalore June 2022
TEAM MEMBERS	• Abhishek Roy • Bharath T • Dhinesh Reddy • Irfan Thanveer • Swathy CS • Vignesh S
DOMAIN OF PROJECT	Finance
PROJECT TITLE	Credit Risk Analysis
GROUP NUMBER	GROUP - 7
TEAM LEADER	Irfan Thanveer
MENTOR NAME	Pratik Sonar

CREDIT RISK ANALYTICS

INDEX:

INTRODUCTION:

Consumer credit is the personal debt taken on to purchase goods and services. Although any type of personal loan could be labeled consumer credit, the term is more often used to describe unsecured debt that is taken on to buy everyday goods and services. However, consumer debt can also include collateralized consumer loans like mortgage and car loans.

Consumer Credit is one of the main sources of income for a bank. But that does not mean there is no inherent risk in the consumer credit business, as it is dependent on various factors apart from the consumer's credit history, like the current Economic climate, Government debt and various other trickle down effects that may put pressure on the clients ability to pay back their debts. Our goal is to reduce this outside risk by creating a model that will only allow us to provide a line of credit to the consumers who are less likely to default on their obligations in tougher times than to provide loans to people who are more likely to default under tougher circumstances.

The current solution to this problem is to build a model which predicts the consumer's creditworthiness based on their past banking history, thus classifying them into 'GOOD/BAD' investments. But this can be optimized in a way such that, we can make use of the past data to build models that can help classify new bank clients into good/bad investments based on the trends, characteristics and similarities from the previous client data, thus helping us predict the creditworthiness of the new client with no banking history.

DATASET INFORMATION:

This dataset is from a midsized Brazilian bank which contains the banking details of its clients who are applying for a loan. The dataset contains of 50,000 client records and 54 information variables that explains the attributes of each client. The target variable is a categorical variable which consists of two classes 1/0, indicating that the client is either GOOD/BAD investments based on their creditworthiness determined by their characteristics and trends from their history of transactions.

PROBLEM STATEMENT:

Credit risk or credit default risk is a type of risk faced by the lenders. Credit risk arises because a debtor can always default on their debt payments, thus causing the lending institutions to write off the loan or to bear the loss on its balance sheet. Financial Institutions do credit risk analytics to prevent this to a great extent.The majority of a Bank's income comes in the form of interest payments and fees from its loans and credit lending programs. The objective is to reduce the possibility of credit risk so that the institutions can make money on their loans and credit programs which will help them pay their interest payments on the bank deposit liabilities and to become a more profitable business.

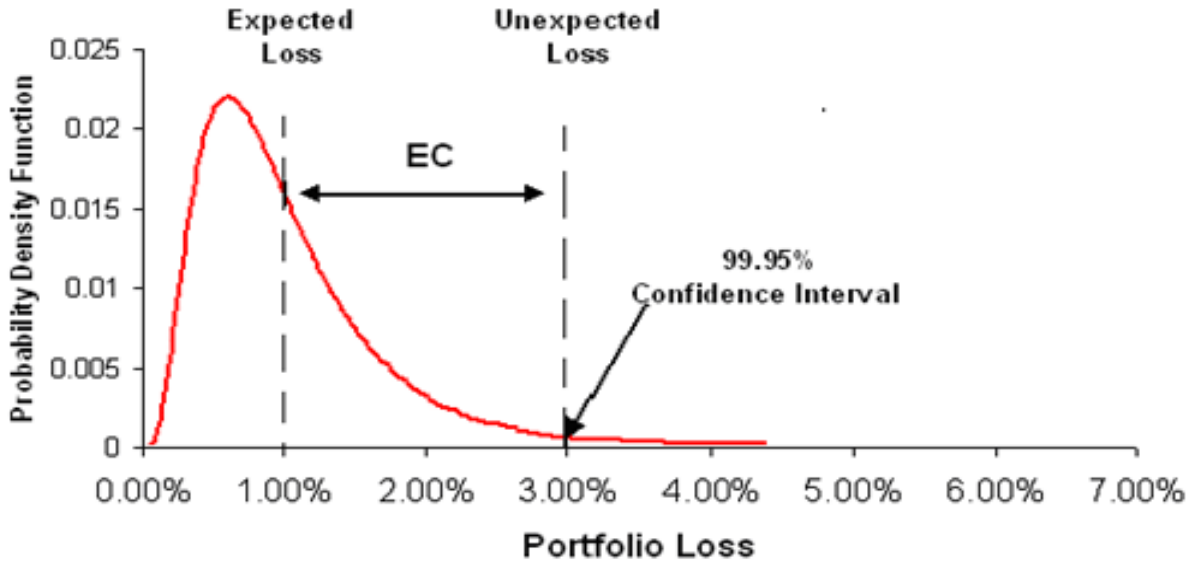


Fig-1: Explanation of the portfolio risk caused by mass credit defaults

One of the approaches to this type of problem is to classify the bank clients who are coming in for a loan into good or bad investments based on their creditworthiness. This means taking into account various personal, banking and credit details and using them to form a model to predict if the future customer is creditworthy or not. Credit risk analysis is assessing the possibility of the borrower's repayment failure and the loss caused to the financial institution when the borrower does not for any reason repay the loan obligations. The cash flow of the institution is impacted when the interest accrued and principal amounts are not paid. Though, there is a grey area in guessing who and when will default on borrowings, with the help of credit analysis we

can help mitigate the severity of complete loss of the borrowings and aid in the principal recovery process.

VARIABLE CATEGORIZATION WITH DESCRIPTION:

The dataset used in the project consists of 53 variables. Out of these 53 variables, we have our target variable which is a categorical variable, Binary class classified to be precise. There are 10 numeric variables and 43 categorical variables which also includes the encoded categories.

Var_Id	Var_Title	Var_Description
1	CLERK_TYPE	Not informed
2	PAYMENT_DAY	Day of the month for bill payment, chosen by the applicant
3	APPLICATION_SUBMISSION_TYPE	Indicates if the application was submitted via the internet or in person/posted
4	QUANT_ADDITIONAL_CARDS	Quantity of additional cards asked for in the same application form
5	POSTAL_ADDRESS_TYPE	Indicates if the address for posting is the home address or other. Encoding not informed.
6	SEX	
7	MARITAL_STATUS	Encoding not informed
8	QUANT_DEPENDANTS	
9	EDUCATION_LEVEL	Educational level in gradual order not informed
10	STATE_OF_BIRTH	
11	CITY_OF_BIRTH	
12	NATIONALITY	Country of birth. Encoding not informed but Brazil is likely to be equal 1.
13	RESIDENCIAL_STATE	State of residence
14	RESIDENCIAL_CITY	City of residence
15	RESIDENCIAL_BOROUGH	Borough of residence
16	FLAG_RESIDENCIAL_PHONE	Indicates if the applicant possesses a home phone
17	RESIDENCIAL_PHONE_AREA_CODE	Three-digit pseudo-code
18	RESIDENCE_TYPE	Encoding not informed. In general, there are the types: owned, mortgage, rented, parents, family etc.

19	MONTHS_IN_RESIDENCE	Time in the current residence in months
20	FLAG_MOBILE_PHONE	Indicates if the applicant possesses a mobile phone
21	FLAG_EMAIL	Indicates if the applicant possesses an e-mail address
22	PERSONAL_MONTHLY_INCOME	Applicant's personal regular monthly income in Brazilian currency (R$)
23	OTHER_INCOMES	Applicant's other incomes monthly averaged in Brazilian currency (R$)
24	FLAG_VISA	Flag indicating if the applicant is a VISA credit card holder
25	FLAG_MASTERCARD	Flag indicating if the applicant is a MASTERCARD credit card holder
26	FLAG_DINERS	Flag indicating if the applicant is a SINERS credit card holder
27	FLAG_AMERICAN_EXPRESS	Flag indicating if the applicant is an AMERICAN EXPRESS credit card holder
28	FLAG_OTHER_CARDS	Despite being label "FLAG", this field presents three values not explained
29	QUANT_BANKING_ACCOUNTS	
30	QUANT_SPECIAL_BANKING_ACCOUNTS	
31	PERSONAL_ASSETS_VALUE	Total value of the personal possessions such as houses, cars etc. in Brazilian currency (R$).
32	QUANT_CARS	Quantity of cars the applicant possesses
33	COMPANY	If the applicant has supplied the name of the company where he/she formally works
34	PROFESSIONAL_STATE	State where the applicant works
35	PROFESSIONAL_CITY	City where the applicant works
36	PROFESSIONAL_BOROUGH	Borough where the applicant works
37	FLAG_PROFESSIONAL_PHONE	Indicates if the professional phone number was supplied
38	PROFESSIONAL_PHONE_AREA_CODE	Three-digit pseudo-code
39	MONTHS_IN_THE_JOB	Time in the current job in months
40	PROFESSION_CODE	Applicant's profession code. Encoding not informed

41	OCCUPATION_TYPE	Encoding not informed
42	MATE_PROFESSION_CODE	Mate's profession code. Encoding not informed
43	EDUCATION_LEVEL	Mate's educational level in gradual order not informed
44	FLAG_HOME_ADDRESS_DOCUMENT	Flag indicating documental confirmation of home address
45	FLAG_RG	Flag indicating documental confirmation of citizen card number
46	FLAG_CPF	Flag indicating documental confirmation of tax payer status
47	FLAG_INCOME_PROOF	Flag indicating documental confirmation of income
48	PRODUCT	Type of credit product applied. Encoding not informed
49	FLAG_ACSP_RECORD	Flag indicating if the applicant has any previous credit delinquency
50	AGE	Applicant's age at the moment of submission
51	RESIDENCIAL_ZIP_3	Three most significant digits of the actual home zip code
52	PROFESSIONAL_ZIP_3	Three most significant digits of the actual job zip code
53	TARGET_LABEL_BAD=1	Target Variable: BAD=1, GOOD=0

TARGET VARIABLE:

The target variable of the above data set is `TARGET_LABEL_BAD=1`, which is a binary classification variable (1/0). Our objective is to predict whether the client is creditworthy or not.

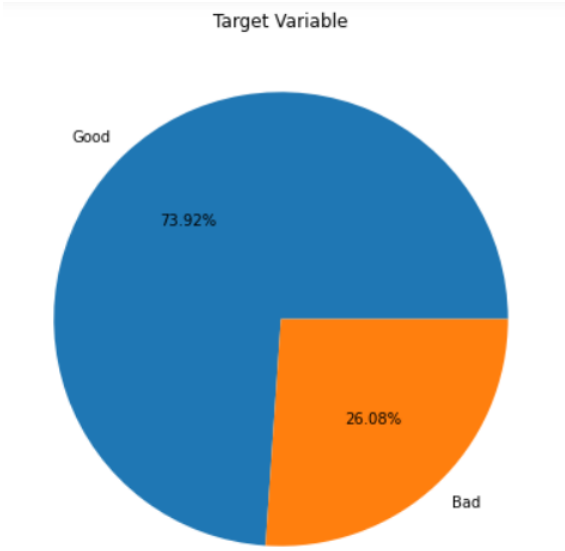


Fig-2: This image depicts the amount of YES vs No (1/0) in the target variable

In the above dataset, 73.9% of the clients are not credit worth and 26.1% of the clients are applicable for the loan approval. **We observe that there is a presence of moderate amount of class imbalance.**

DATA-PRE PROCESSING:

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data pre-processing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasksfor cleaning the data and making it suitable for a machine learning model which also increases theaccuracy and efficiency of a machine learning model.

The data consists of 50,000 rows and 54 columns. Out of these we have 44 categorical columnsand the rest as numerical.

Data types of the variables are as follows:

```
CLERK_TYPE                          object
PAYMENT_DAY                          int64
APPLICATION_SUBMISSION_TYPE         object
POSTAL_ADDRESS_TYPE                  int64
SEX                                 object
MARITAL_STATUS                       int64
QUANT_DEPENDANTS                     int64
STATE_OF_BIRTH                      object
CITY_OF_BIRTH                       object
NATIONALITY                          int64
RESIDENCIAL_STATE                   object
RESIDENCIAL_CITY                    object
RESIDENCIAL_BOROUGH                 object
FLAG_RESIDENCIAL_PHONE              object
RESIDENCIAL_PHONE_AREA_CODE         object
RESIDENCE_TYPE                     float64
MONTHS_IN_RESIDENCE                float64
FLAG_MOBILE_PHONE                   object
FLAG_EMAIL                           int64
PERSONAL_MONTHLY_INCOME            float64
OTHER_INCOMES                      float64
FLAG_VISA                            int64
FLAG_MASTERCARD                      int64
FLAG_DINERS                          int64
FLAG_AMERICAN_EXPRESS                int64
FLAG_OTHER_CARDS                     int64
QUANT_BANKING_ACCOUNTS               int64
QUANT_SPECIAL_BANKING_ACCOUNTS       int64
PERSONAL_ASSETS_VALUE              float64
QUANT_CARS                           int64
COMPANY                             object
PROFESSIONAL_STATE                  object
FLAG_PROFESSIONAL_PHONE             object
PROFESSIONAL_PHONE_AREA_CODE        object
MONTHS_IN_THE_JOB                    int64
PROFESSION_CODE                    float64
OCCUPATION_TYPE                    float64
PRODUCT                              int64
FLAG_ACSP_RECORD                    object
AGE                                  int64
RESIDENCIAL_ZIP_3                   object
PROFESSIONAL_ZIP_3                  object
TARGET_LABEL_BAD=1                   int64
CLIENT_ID                            int64
```

The above shows the data types of the various variables, but there are also categorical variables that are already encoded for us during the data collection process, hence we can use them as they are for our model building, but the other categorical variables need to be encode to their numeric form before the model building process.

NULL VALUE IMPUTATION:

The next step of data pre-processing is to handle missing data in the datasets. If our dataset containssome missing data, then it may create a huge problem for our machine learning model. Hence it isnecessary to handle missing values present in the dataset. The following are the null value percentages:

PAYMENT_DAY	0.000
APPLICATION_SUBMISSION_TYPE	0.000
SEX	0.000
MARITAL_STATUS	0.000
QUANT_DEPENDANTS	0.000
STATE_OF_BIRTH	0.000
CITY_OF_BIRTH	0.000
NATIONALITY	0.000
RESIDENCIAL_STATE	0.000
RESIDENCIAL_CITY	0.000
RESIDENCIAL_BOROUGH	0.000
FLAG_RESIDENCIAL_PHONE	0.000
RESIDENCE_TYPE	2.698
MONTHS_IN_RESIDENCE	7.554
FLAG_MOBILE_PHONE	0.000
FLAG_EMAIL	0.000
PERSONAL_MONTHLY_INCOME	0.000
OTHER_INCOMES	0.000
FLAG_VISA	0.000
FLAG_MASTERCARD	0.000
FLAG_DINERS	0.000
FLAG_AMERICAN_EXPRESS	0.000
FLAG_OTHER_CARDS	0.000
QUANT_BANKING_ACCOUNTS	0.000
QUANT_SPECIAL_BANKING_ACCOUNTS	0.000
PERSONAL_ASSETS_VALUE	0.000
QUANT_CARS	0.000
COMPANY	0.000
FLAG_PROFESSIONAL_PHONE	0.000
MONTHS_IN_THE_JOB	0.000
PROFESSION_CODE	15.512
OCCUPATION_TYPE	14.626
PRODUCT	0.000
FLAG_ACSP_RECORD	0.000
AGE	0.000
RESIDENCIAL_ZIP_3	0.000
PROFESSIONAL_ZIP_3	0.000
TARGET_LABEL_BAD=1	0.000
CLIENT_ID	0.000

Here there are only 4 variables with null values, hence we try and impute them where ever possible. We can impute this using mean, median, mode, b fill, f fill or based on the dataset, you can impute the best possible value such that there is not much of a deviation in the summary of the variable from before the imputation.

- Here, for the variable, Residence type, we have imputed the value using the mode value because the value for the mode is substantially high from the other categories as shown below:

```
>> X['RESIDENCE_TYPE'].value_counts()
```

1.0	41572
2.0	3884
5.0	1983
0.0	760
4.0	311
3.0	141

- For the variable, Months in residence, we have imputed the value with the median as it is a numeric variable.

- For the variable, Profession code, we have again imputed the null value with the mode as there is a large disparity in the mode value comparing to the rest of the classes.

```
>> X['PROFESSION_CODE'].value_counts()
```

9.0	30092
11.0	3545
0.0	3540
2.0	2827
12.0	489
10.0	425
16.0	344
13.0	313
7.0	216
8.0	144
6.0	136
15.0	63
17.0	35
4.0	27
3.0	18
5.0	12
14.0	9
1.0	8
18.0	1

- For the variable occupation type, we do the same null imputation with mode as there is a disparity in the mode V/s the other classes.

```
>> X['OCCUPATION_TYPE'].value_counts()
```

2.0	16947
1.0	8742
4.0	7000
5.0	6891
0.0	2788
3.0	319

CHECKING FOR OUTLIERS:

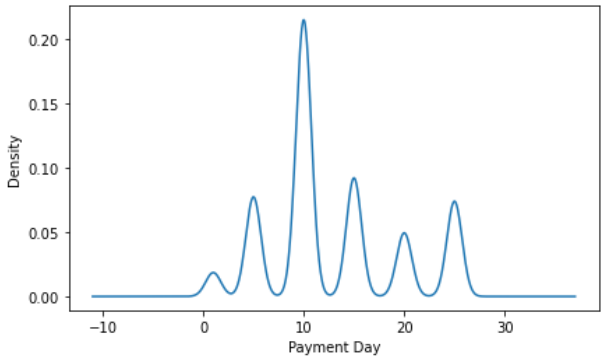
Data has outliers present in each of the numerical columns. For making the base model, we should not perform any outlier treatment and retain all the rows present in the data. The outlier treatment is done after the base model building.

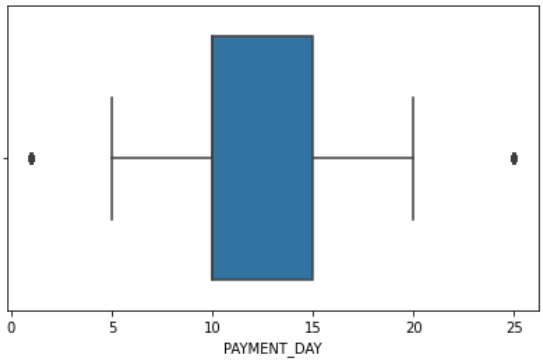
EXPLORATORY DATA ANALYSIS:

Univariate Analysis:

For Numerical Variables: - We plot the distribution curve and box plot to study the variation of the numerical data.

1. Payment day:

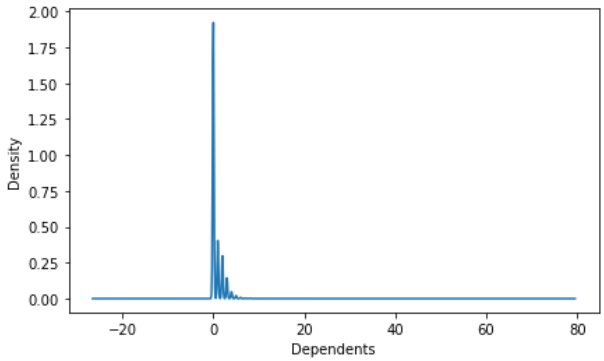


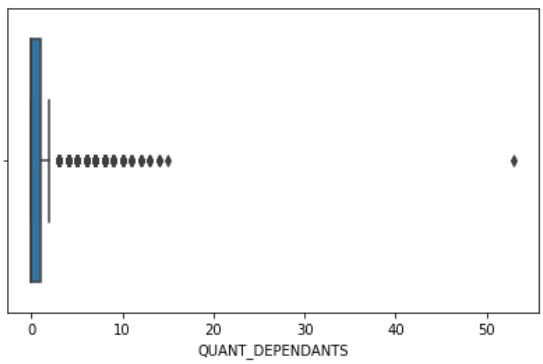



```
Skew:  0.5071180550490278
Kurtosis:  -0.5988538279497719
```

- Payment day is almost normally distributed.
- It is mesokurtic
- IQR of payment day lies from 5-20. Outliers are present.

2. Quantity of Dependents:

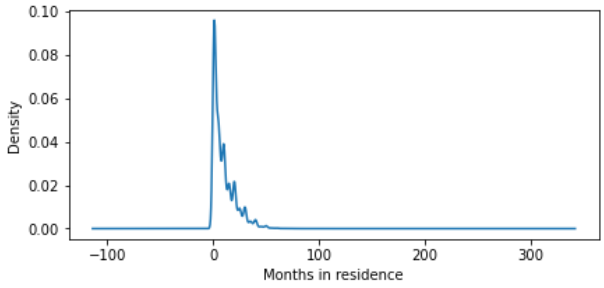


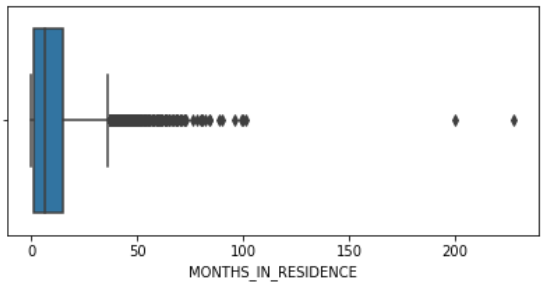



```
Skew:  4.077756812678044
Kurtosis:  83.1914311606446
```

- Quant_dependents is right skewed.
- It is leptokurtic
- IQR of lead dependents lies from 0-1. Outliers are present.
- Highest frequency is at 0

3. Months in Residence:

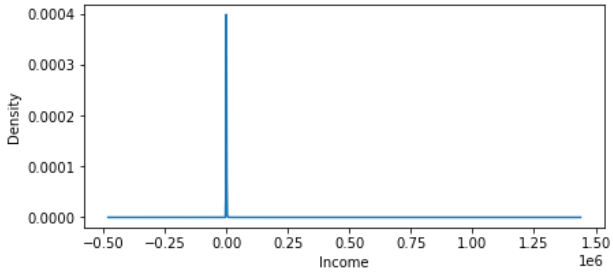


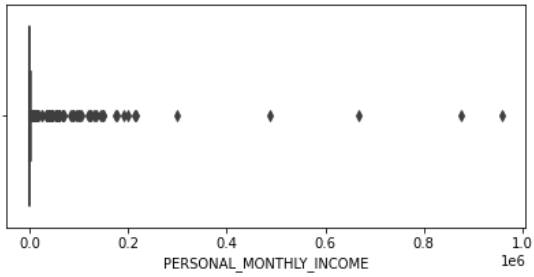



```
Skew:  1.9036703408954152
Kurtosis:  9.129723205274011
```

- Months in residence are right skewed.
- It is leptokurtic
- IQR of month's in residence lies from 0-15. Outliers are present.

4. Personal monthly Income:

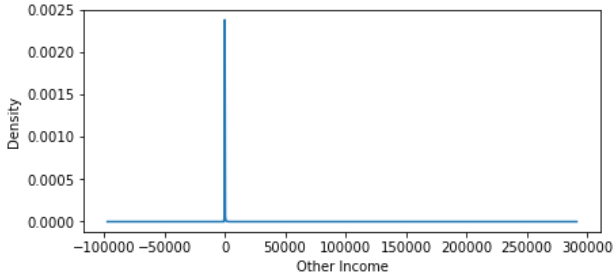


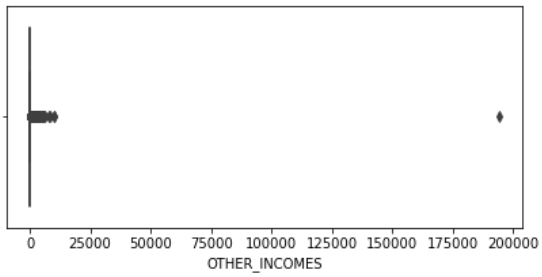



```
Skew:  85.7057480035924
Kurtosis:  8975.879000318488
```

- Personal monthly Income is right skewed.
- It is leptokurtic
- IQR of monthly income lies from 0-800. Outliers are present.

5. Other Income:

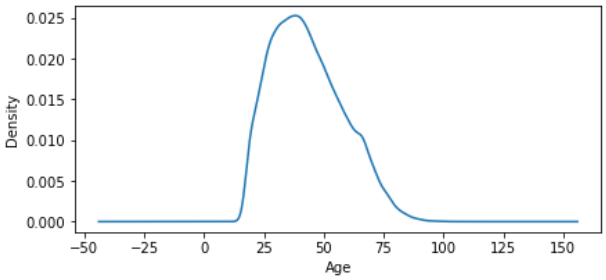


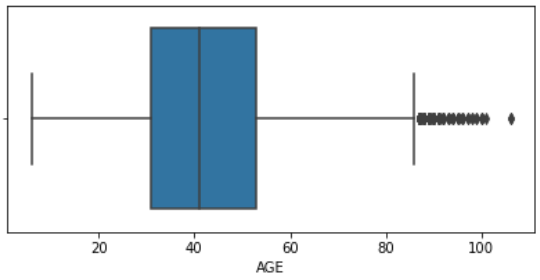



```
Skew:  207.2713503426558
Kurtosis:  45136.46300246626
```

- Other Income is right skewed.
- It is leptokurtic
- Outliers are present.

6. Age:






```
Skew:  0.4731456810100421
Kurtosis:  -0.4050732444053988
```

- Age is right skewed.
- It is mesotokurtic
- IQR of lead time lies from 0-53. Outliers are present.

For Categorical Variables – We plot a combination of bar graph and pie chart to understand thedistribution of categorical data in the dataset.

1. Application Submission Type:

   There are 3 different category classes in the types of submissions, they are web, carga and offline. We see that there are more web based applications compared to the others.

2. Sex:

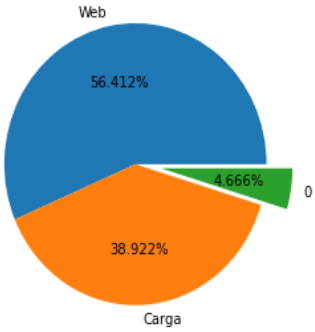
   There are 3 different categories in the gender class, Male, Female and Not mentioned. We see that there are more female applicants than there are male applicants.

3. Company:

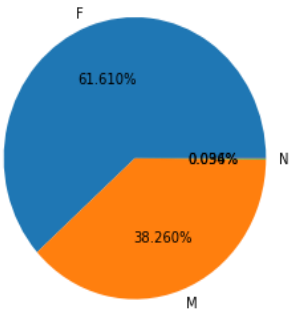
   This is a binary class (yes/no). We can see that there are more unemployed people coming in for a loan than people with a job.

4. Flag professional phone:

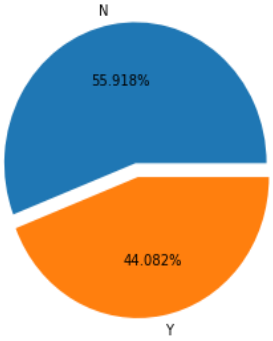
   This is also a binary class variable. We can see that the number of No's exceed the number of Yes', which means that the professional phone Is not flagged in many cases.



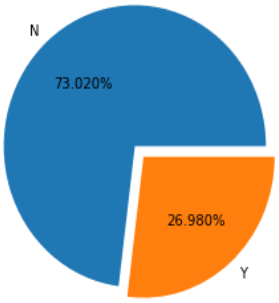
APPLICATION_SUBMISSION_TYPE



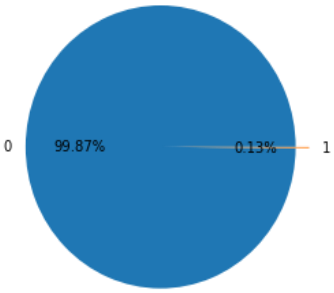
SEX



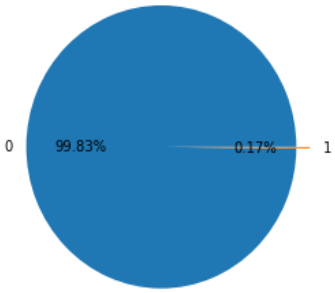
COMPANY



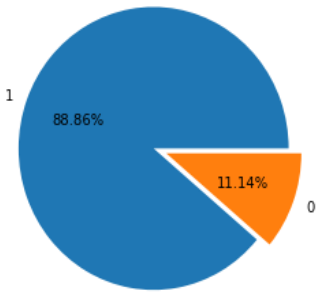
FLAG_PROFESSIONAL_PHONE



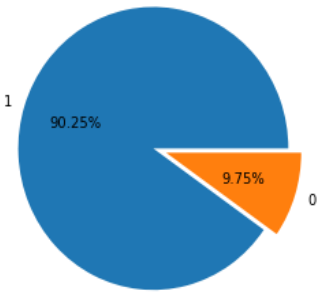
FLAG_DINERS



FLAG_AMERICAN_EXPRESS



FLAG_VISA

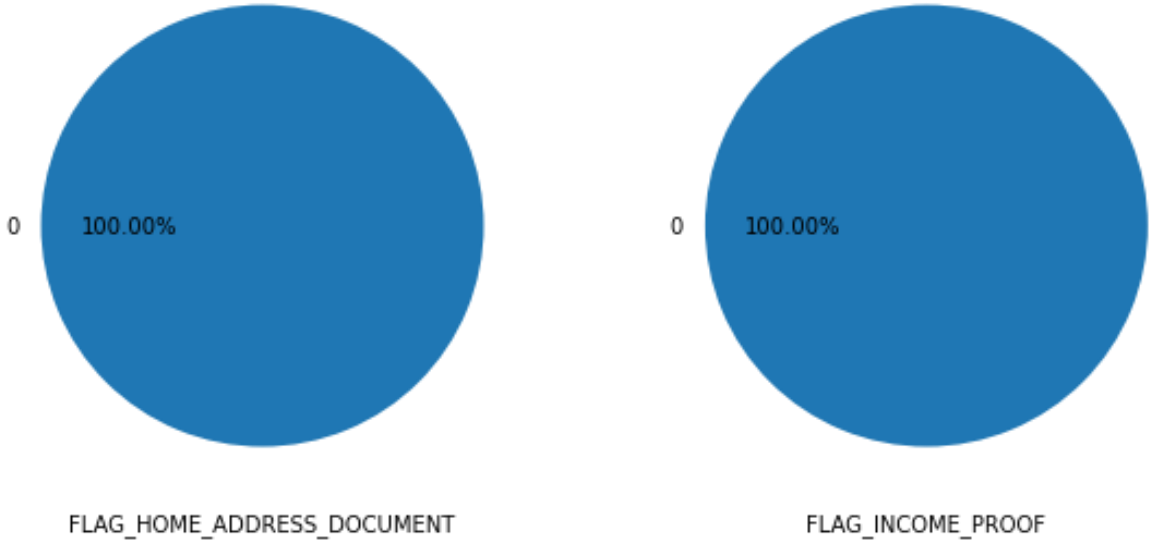


FLAG_MASTERCARD

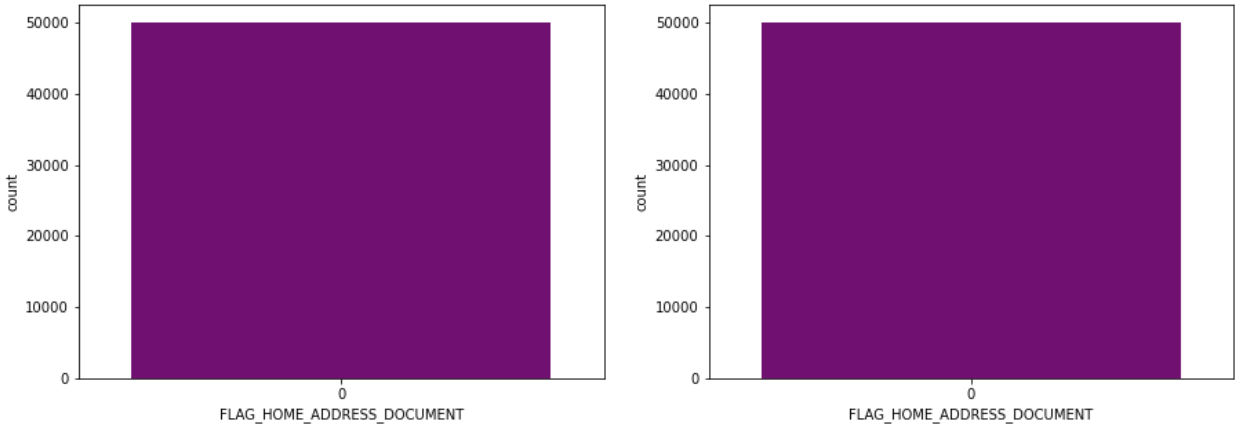
5. Flagged Cards:

From the above images, we see that there is more number of flagged cards compared to non flagged card holders with diners coming first, closely followed by Amex. Visa and M asterCard's have comparatively less flagged cards compared to Diners and Amex, none t he less, they also have high number of flagged cards.

6. Flagged Proof:

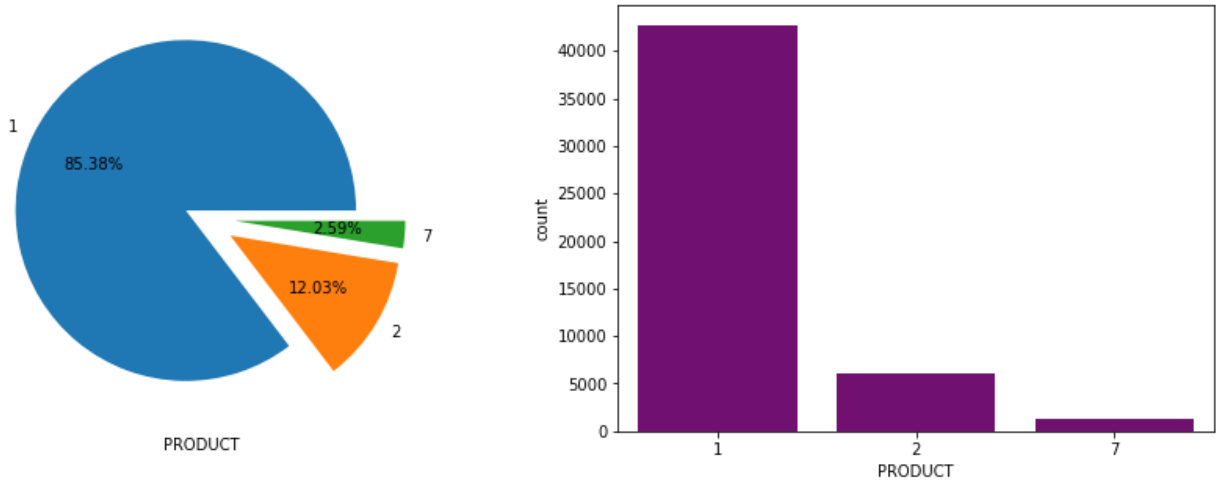


From the image above, we can see that there are no flagged documents during verificati on, which means there is no variance in these columns. Lets also cross check the claim w ith the help of the count plots of the document verification.



Here we can cross verify that there is no additional class in the variables.

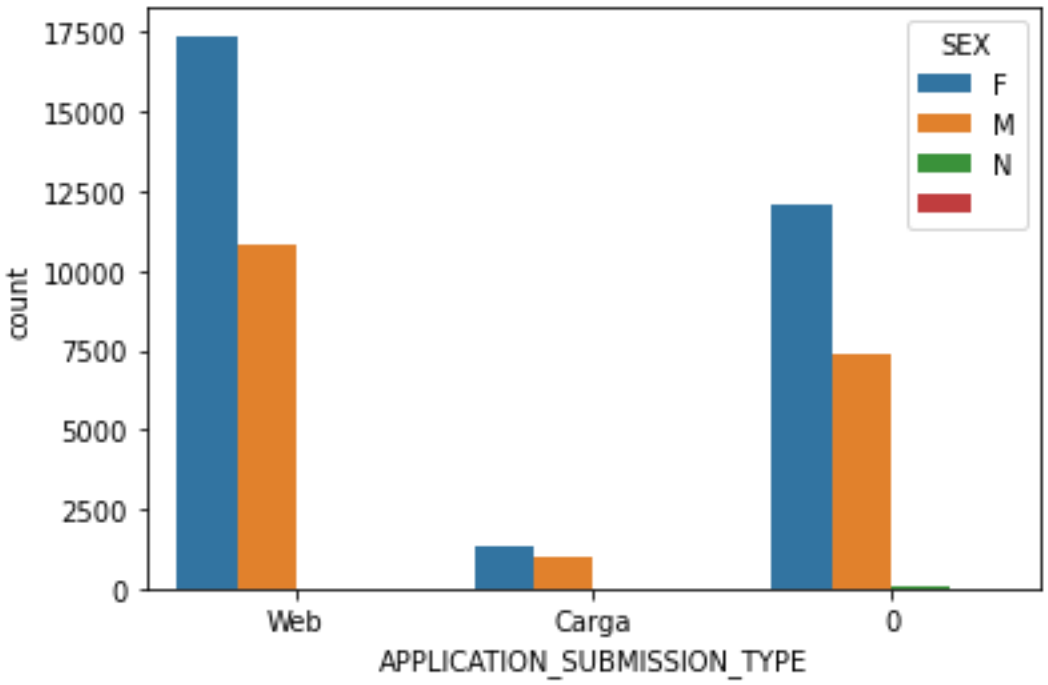
7. Product:



The above image shows us that there are 3 different types of product in the variable. Th e product 1 is having the most amount of client count compared to product 2 and produ ct 7.

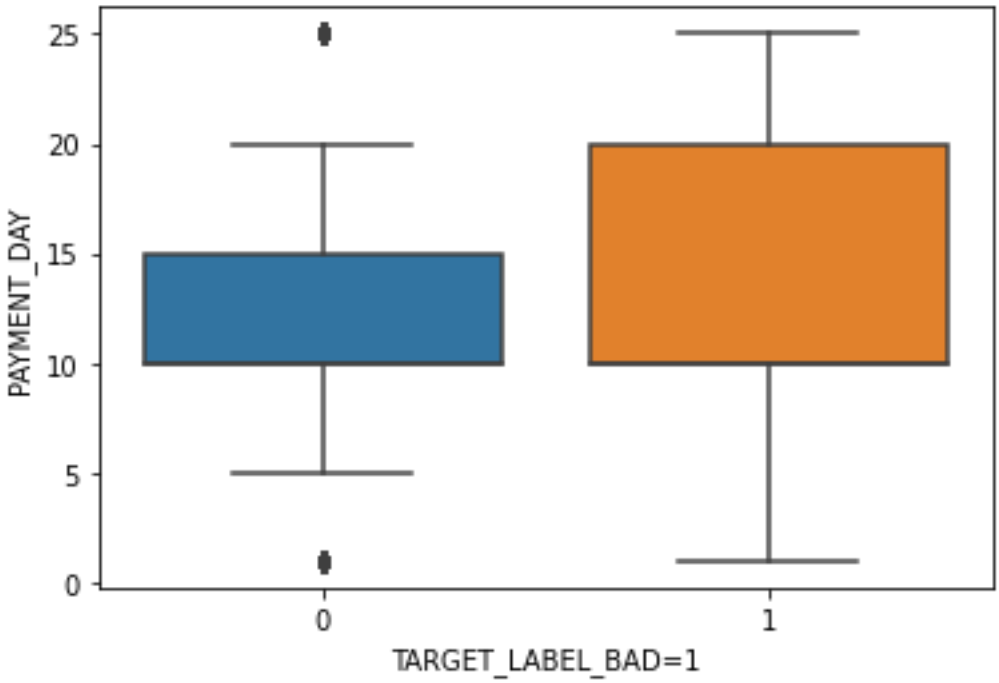
Multi-Variate Analysis:

1. Application Type VS Gender:



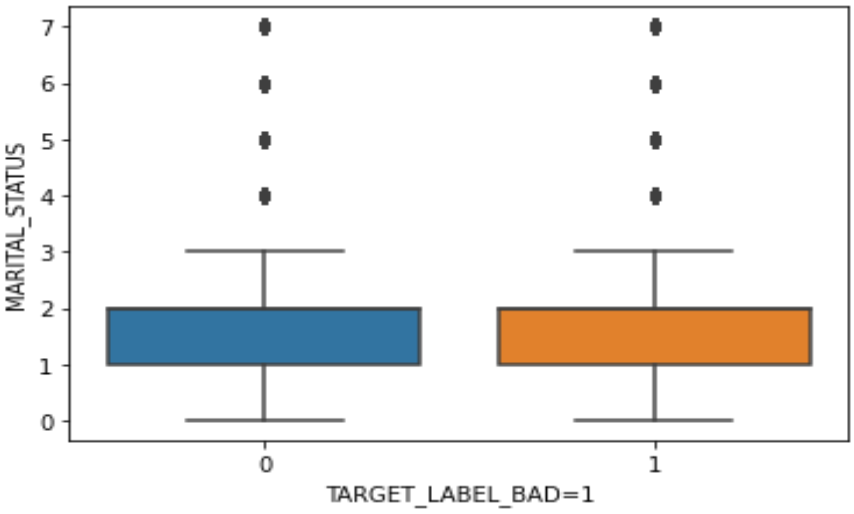
This shows that Web application has the most count and female applicants have the overall highest count.

2. Payment Day VS Target:



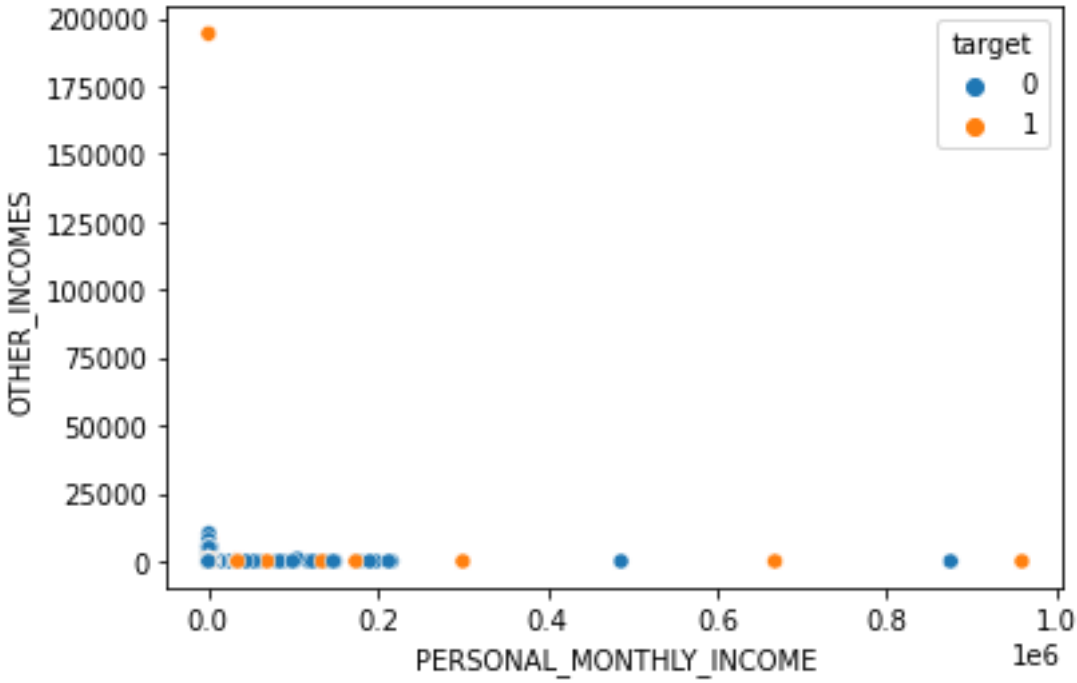
This shows us that there is almost no difference in the payment days for good and bad clients, but the 75th percentile for the bad clients is 5 days away from the good clients.

3. Marital Status VS Target:



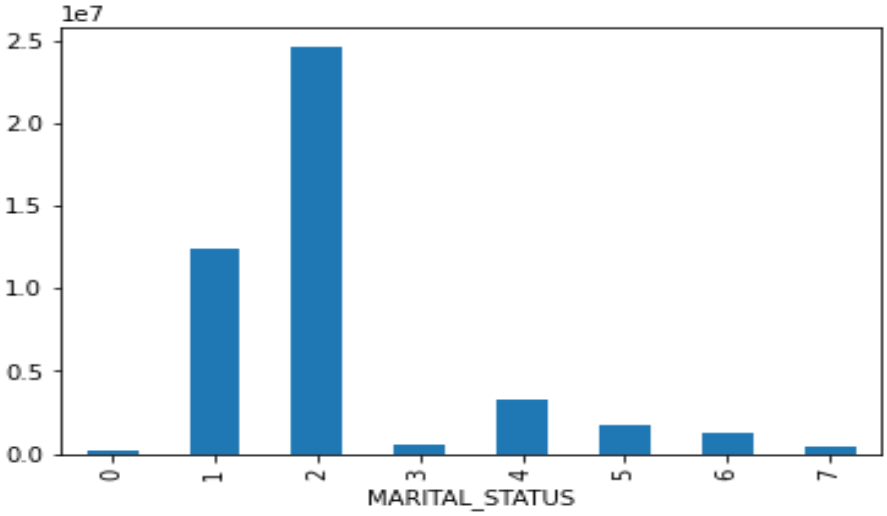
From the above image, we can see that the marital status has little to no effect on th e target variable.

4. Income VS Other Incomes with Target as Hue:



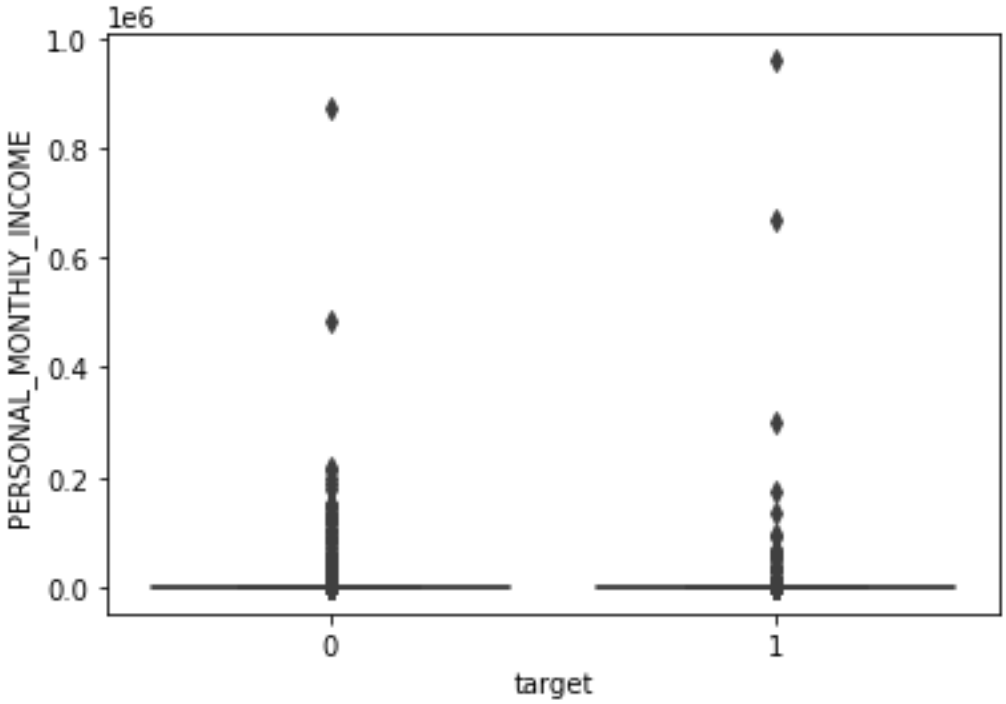
This shows that people with almost no personal income have some other sources of income and as the monthly income increases, there are almost no other sources of i ncome. It also shows that there is a large amount of personal income lies between 0-25000.

5. Marital Status VS Personal Income:



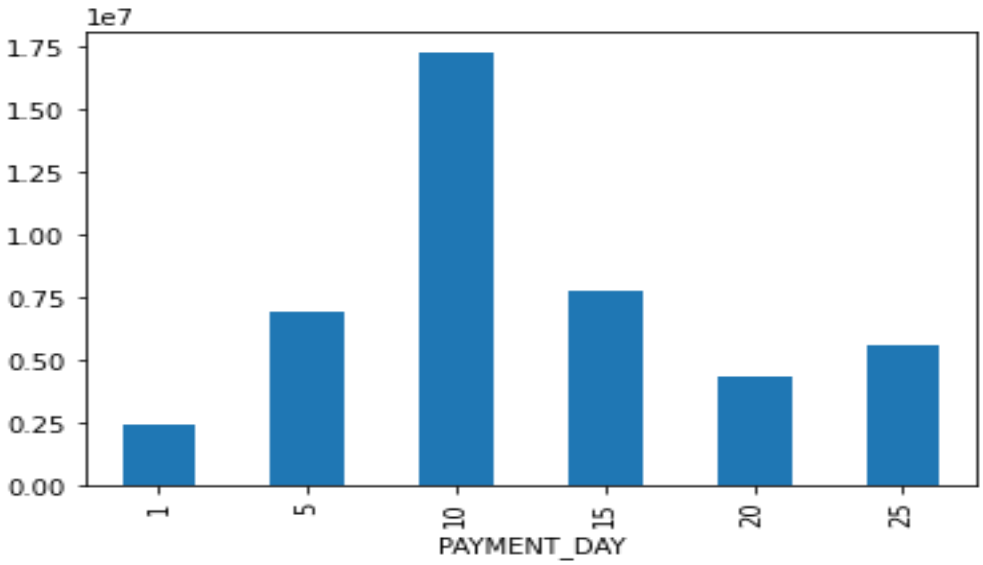
Here we can see that the class 2 has the highest amount of personal income vs the cl ass 1. The rest have significantly less personal income.

6. Income VS Target:



There is some discrepancies in the data for bad clients and income.

7. Payment Day VS Personal Income:



From the above image, we see that the people getting paid on the 10th day of the m onth have relatively higher incomes.

CORRELATION MATRIX:

Heat-Map - Pearson Correlation Matrix

(Assumption: For the Pearson correlation, both variables should be normally distributed. Other assumptions include linearity and homoscedasticity)

It gives a measure of how much two numeric variables are linearly correlated. It tries to obtain a best fit line between two numeric variables and how close the points are to a fitted line.

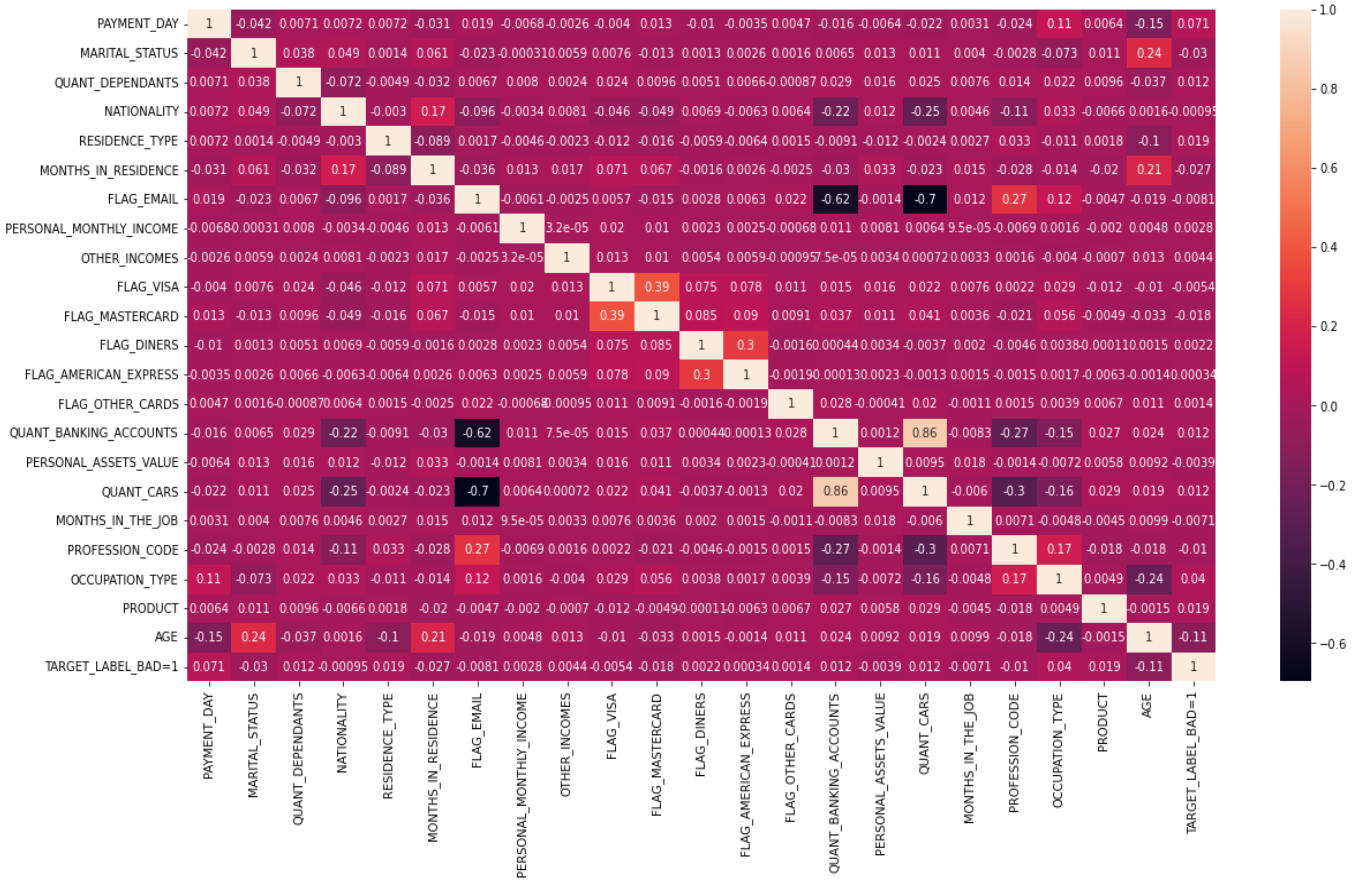


Fig- Heat map: Pearson Correlation Matrix

From the heat map we can understand:

- There is high positive correlation between quantity of cars and the quantity of bank accounts.
- Flagged email and quantity of cars have strong negative correlation.
- Flagged email and quantity of bank accounts also have a strong negative correlation

PAIR PLOT:

Fig- Pair plot of all the numeric variables

Base Model:

The base model is the model which we build before fine tuning our final model. We build the base model with the existing data without tuning any hyper parameters to roughly understand how our model will perform after perfecting it

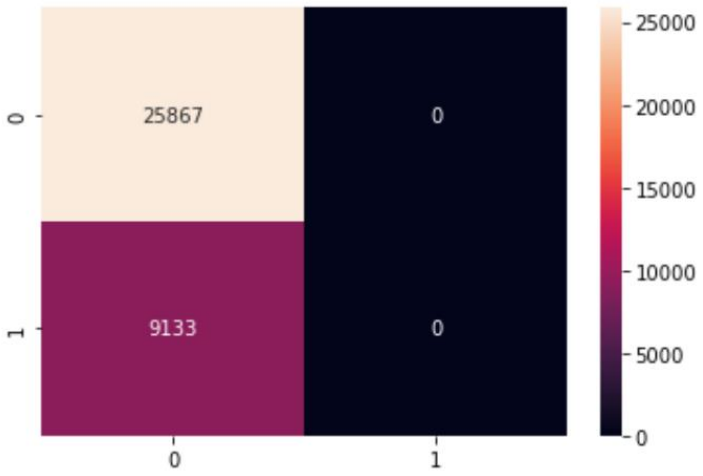
Logistic Regression:

Logistic regression is an example of supervised learning. It is **used to calculate or predict the probability of a binary (yes/no) event occurring**.

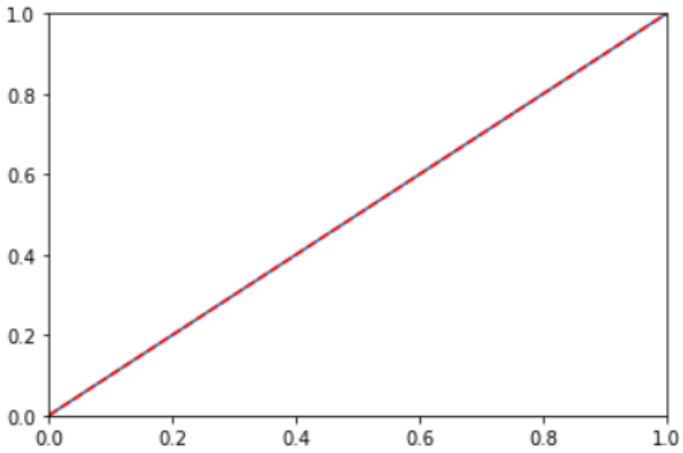
Train:

	precision	recall	f1-score	support
0	0.74	1.00	0.85	25867
1	0.00	0.00	0.00	9133
accuracy			0.74	35000
macro avg	0.37	0.50	0.42	35000
weighted avg	0.55	0.74	0.63	35000

From the classification report, we can see that the accuracy is 74% while the precision for our 1 class is 0% which is the key indicator in our model prediction.



The confusion matrix also shows that the false positive and true positive predictions are 0, which means that the positives have not been predicted at all.

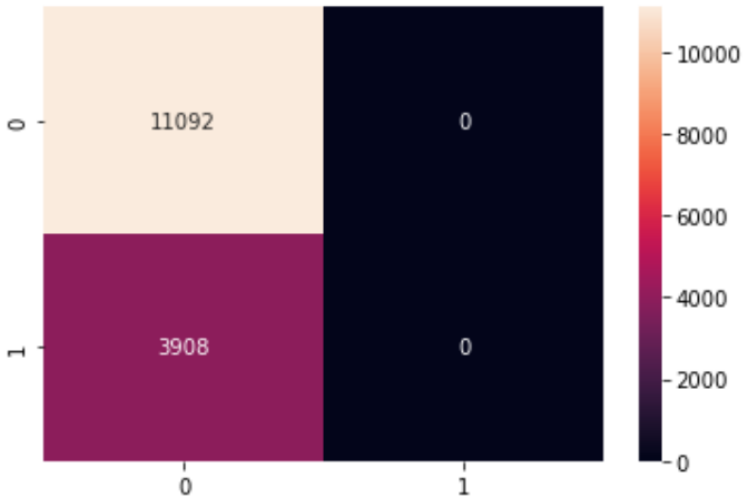


The ROC curve also shows that there is no significant prediction of the class 1.

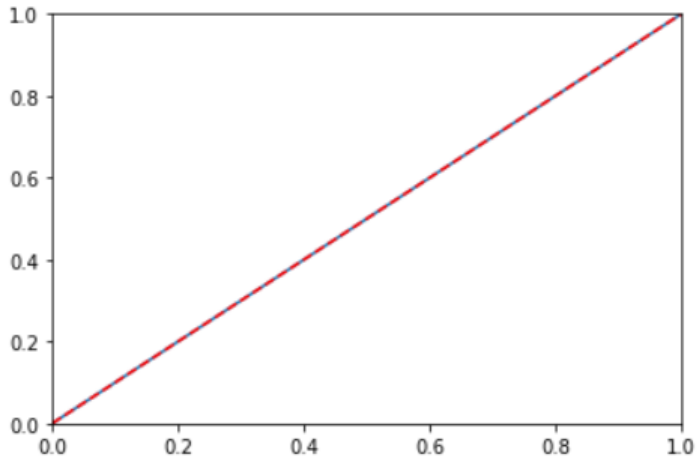
Test:

	precision	recall	f1-score	support
0	0.74	1.00	0.85	11092
1	0.00	0.00	0.00	3908
accuracy			0.74	15000
macro avg	0.37	0.50	0.43	15000
weighted avg	0.55	0.74	0.63	15000

In the test also, we can observe that the model is under fit for predicting the 1 class.



The confusion matrix also shows that the false positive and true positive predictions are 0, which means that the positives have not been predicted at all.



There is no difference in the ROC curve from the train data.

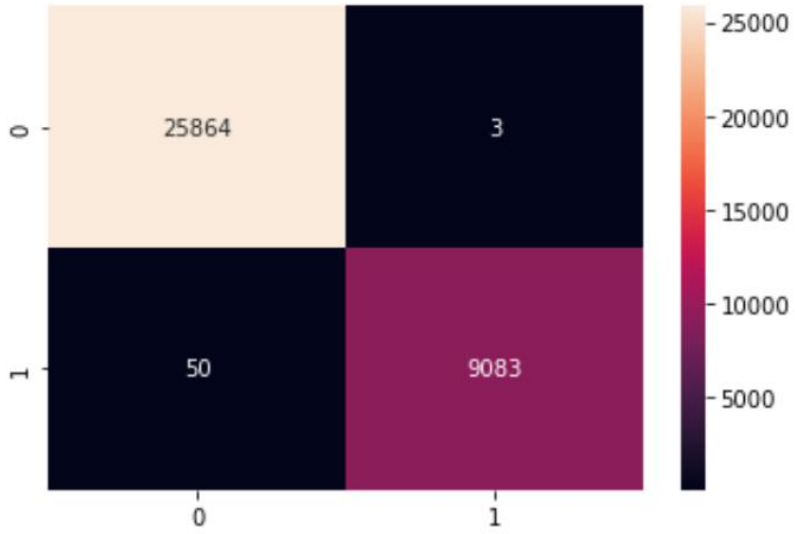
Decision Tree:

Decision trees **use multiple algorithms to decide to split a node into two or more sub-nodes**

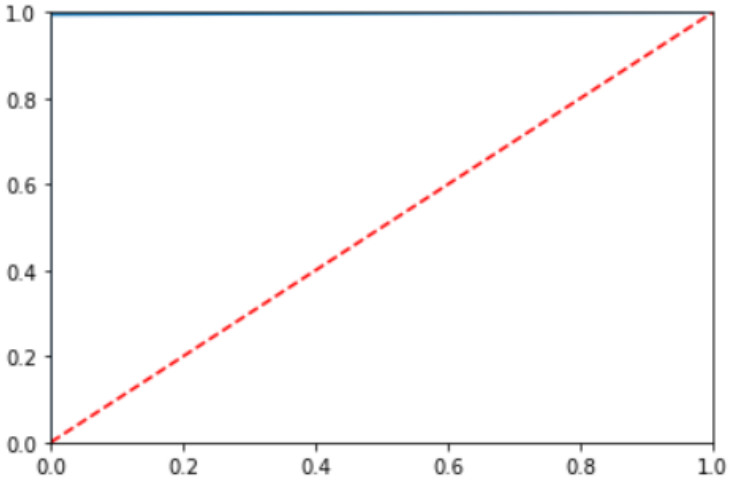
Train:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	25867
1	1.00	0.99	1.00	9133
accuracy			1.00	35000
macro avg	1.00	1.00	1.00	35000
weighted avg	1.00	1.00	1.00	35000

The decision tree model gives us an over fit model as we know that the dataset is imbalanced in the target variable. The accuracy, precision, recall and f1-score are all 100% in the train data depicting a clear over fit of the model. Next we will build a model for the test data to cross verify whether there is an over fit in the model or not.



The confusion matrix shows the separation clearly of the true positives, true negatives, false positives and false negatives. Here the false negatives and false positives classifications are minimal, this could be because of the over fit. We will have to build a model with the test data to verify.

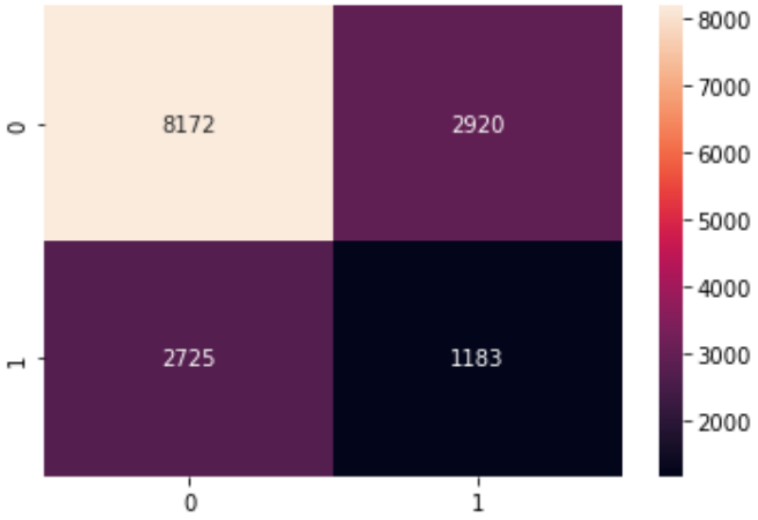


From the ROC curve, we can interpret that the model prediction for true positive rate vs false positive rate is 100%. This implies 100% classification for both 0 and 1 classes. Lets verify this claim with the help of the test data split.

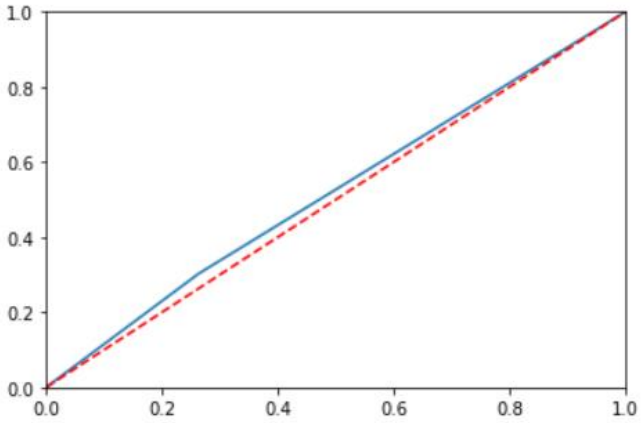
Test:

	precision	recall	f1-score	support
0	0.75	0.74	0.74	11092
1	0.29	0.30	0.30	3908
accuracy			0.62	15000
macro avg	0.52	0.52	0.52	15000
weighted avg	0.63	0.62	0.63	15000

The accuracy for the test model is 62% where in the train data, it was 100% which clearly states an over fit scenario. We also see that the precision for class 1 is 29% where as in the train model, it was 100%. This further solidifies the claim of over fit.



Shows huge imbalance in the false positives and false positives compared to the train model.



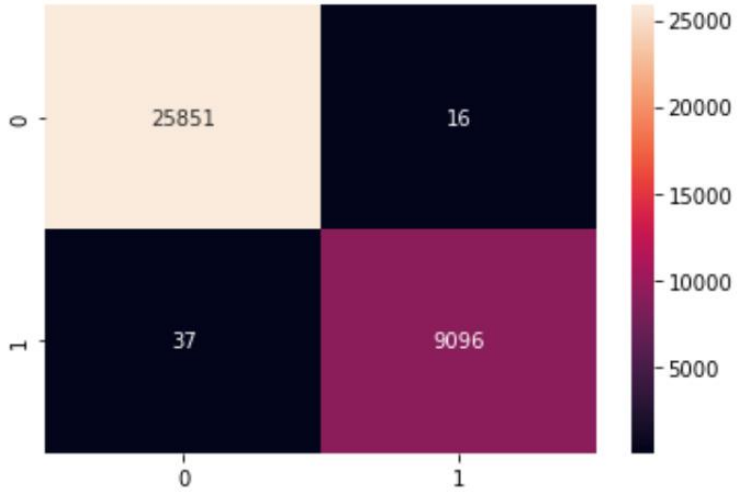
Random Forest:

Random forest is **a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems**. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression

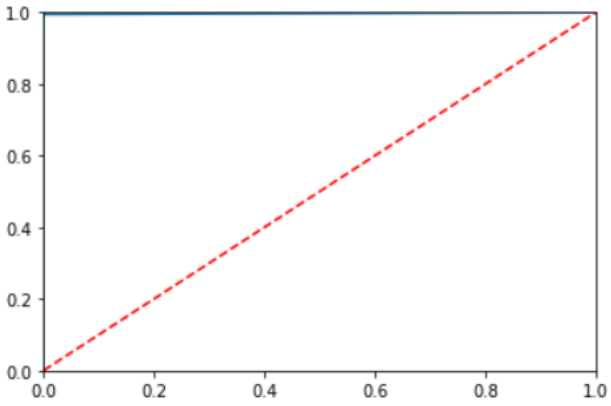
Train:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	25867
1	1.00	1.00	1.00	9133
accuracy			1.00	35000
macro avg	1.00	1.00	1.00	35000
weighted avg	1.00	1.00	1.00	35000

The random forest model gives us an over fit model as we know that the dataset is imbalanced in the target variable. The accuracy, precision, recall and f1-score are all 100% in the train data depicting a clear over fit of the model. Next we will build a model for the test data to cross verify whether there is an over fit in the model or not.



The confusion matrix shows the separation clearly of the true positives, true negatives, false positives and false negatives. Here the false negatives and false positives classifications are minimal, this could be because of the over fit. We will have to build a model with the test data to verify.

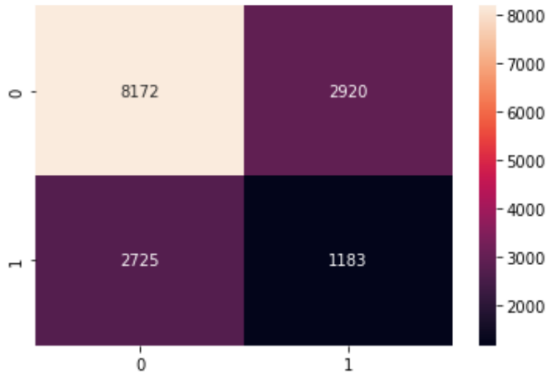


From the ROC curve, we can interpret that the model prediction for true positive rate vs false positive rate is 100%. This implies 100% classification for both 0 and 1 classes. Lets verify this claim with the help of the test data split.

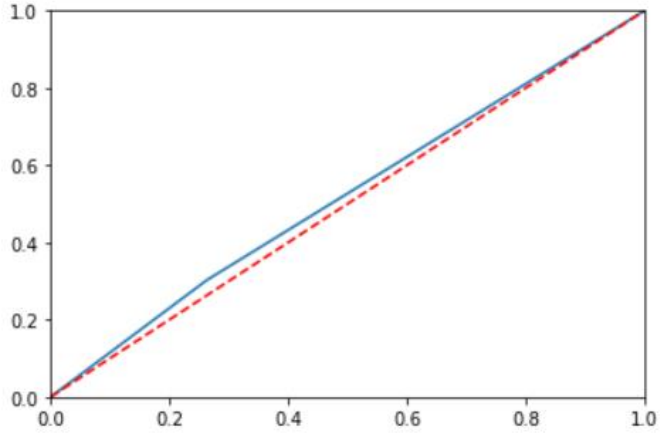
Test:

	precision	recall	f1-score	support
0	0.75	0.74	0.74	11092
1	0.29	0.30	0.30	3908
accuracy			0.62	15000
macro avg	0.52	0.52	0.52	15000
weighted avg	0.63	0.62	0.63	15000

The accuracy for the test model is 62% where in the train data, it was 100% which clearly states an over fit scenario. We also see that the precision for class 1 is 29% where as in the train model, it was 100%. This further solidifies the claim of over fit.



Shows huge imbalance in the false positives and false positives compared to the train model.



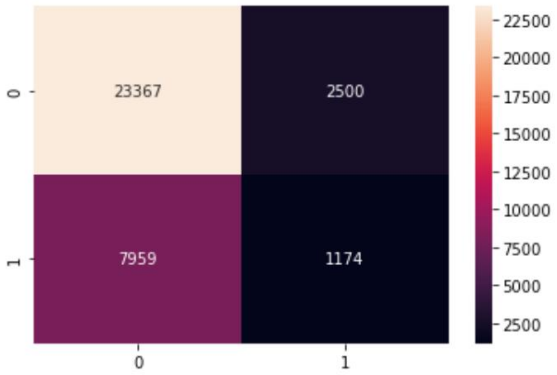
Naïve Bayes:

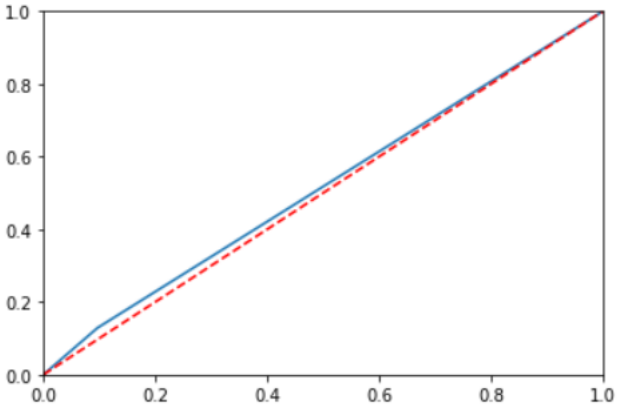
Naive Bayes classifiers are **a collection of classification algorithms based on Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Train:

	precision	recall	f1-score	support
0	0.75	0.90	0.82	25867
1	0.32	0.13	0.18	9133
accuracy			0.70	35000
macro avg	0.53	0.52	0.50	35000
weighted avg	0.63	0.70	0.65	35000

This is an under fit model as it cannot properly classify the 1 class even on the test data.

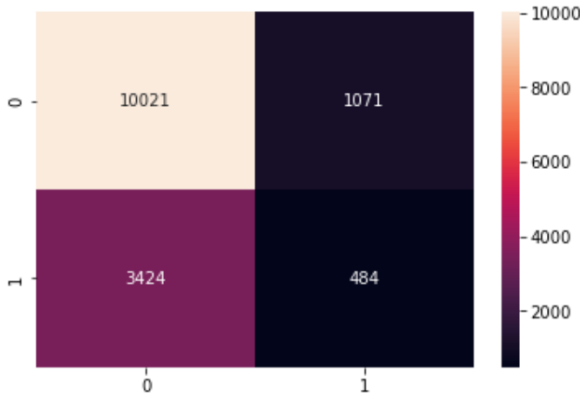


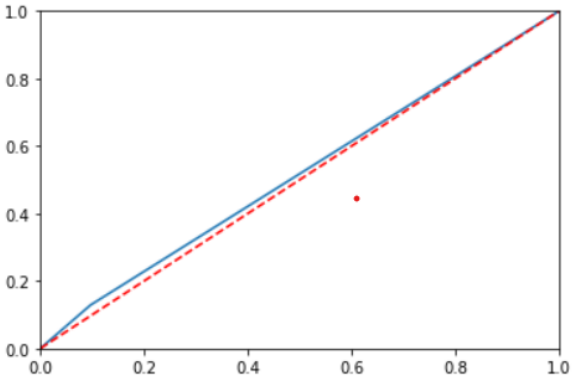


Test:

	precision	recall	f1-score	support
0	0.75	0.90	0.82	11092
1	0.31	0.12	0.18	3908
accuracy			0.70	15000
macro avg	0.53	0.51	0.50	15000
weighted avg	0.63	0.70	0.65	15000

Almost no difference from the train and test models.



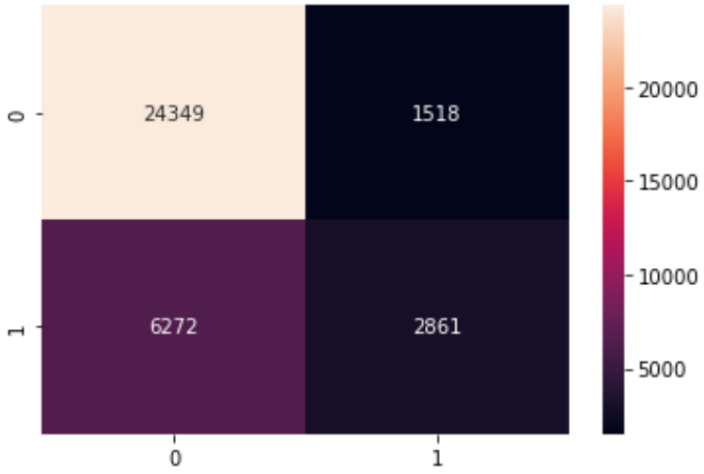


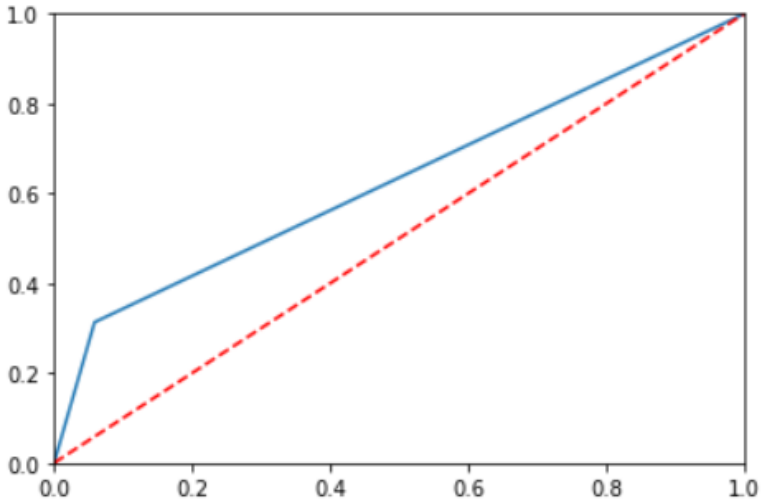
K-Nearest Neighbors:

K-NN algorithm stores all the available data and classifies a new data point based on the similarity.

Train:

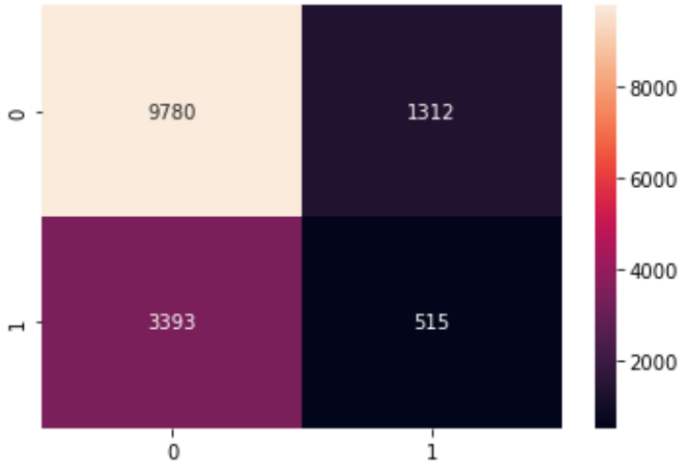
	precision	recall	f1-score	support
0	0.80	0.94	0.86	25867
1	0.65	0.31	0.42	9133
accuracy			0.78	35000
macro avg	0.72	0.63	0.64	35000
weighted avg	0.76	0.78	0.75	35000

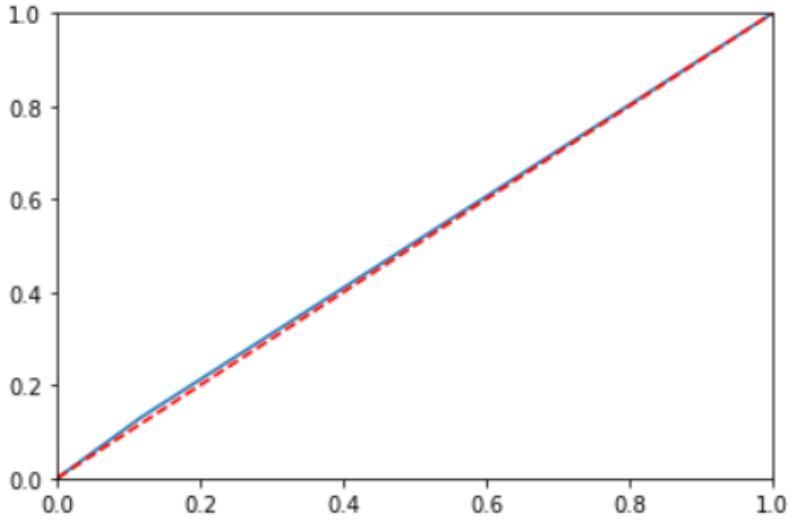




Test:

	precision	recall	f1-score	support
0	0.74	0.88	0.81	11092
1	0.28	0.13	0.18	3908
accuracy			0.69	15000
macro avg	0.51	0.51	0.49	15000
weighted avg	0.62	0.69	0.64	15000





We see significant difference in the train and test models. The classification is not precise and this is due to the imbalance in the classes of the target variable.

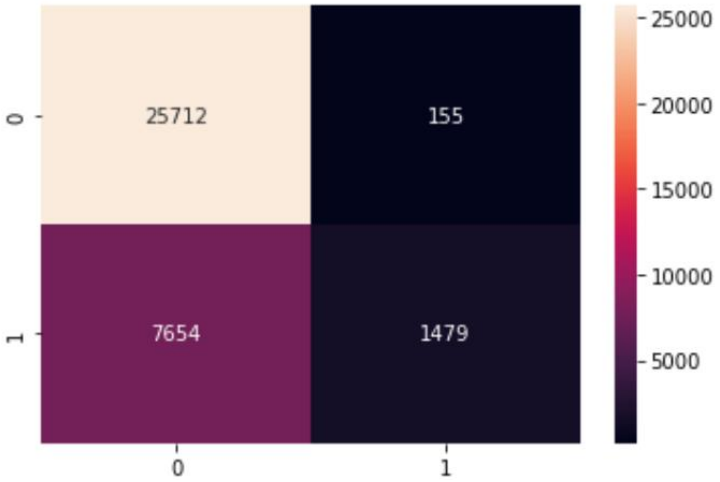
Boosting:

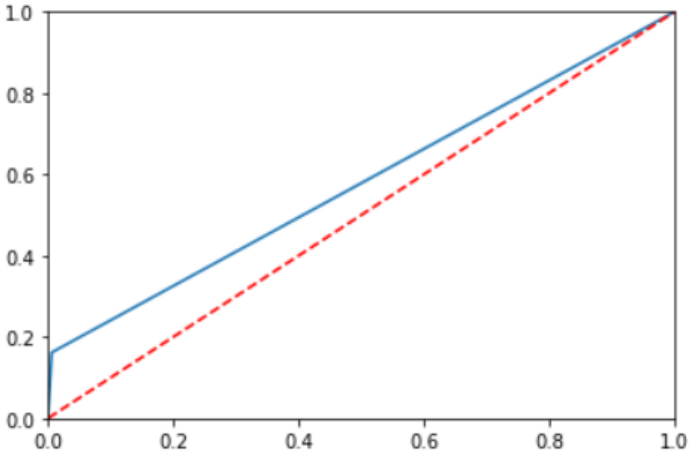
Boosting is an ensemble modeling, technique that attempts to build a strong ***classifier*** from the number of weak ***classifiers***.

XG-Boost:

Train:

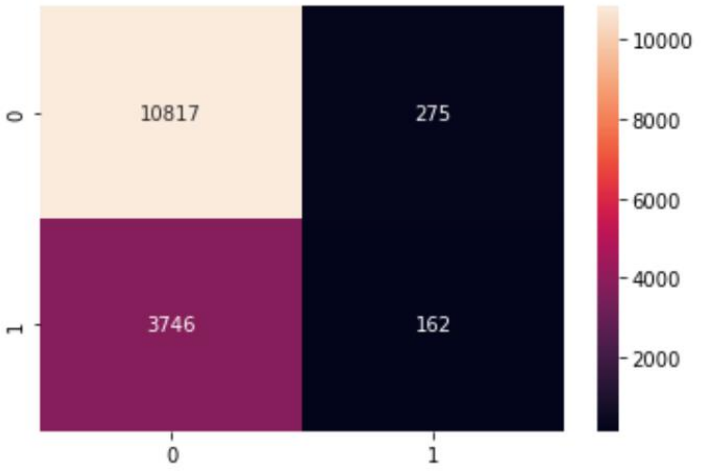
	precision	recall	f1-score	support
0	0.77	0.99	0.87	25867
1	0.91	0.16	0.27	9133
accuracy			0.78	35000
macro avg	0.84	0.58	0.57	35000
weighted avg	0.81	0.78	0.71	35000

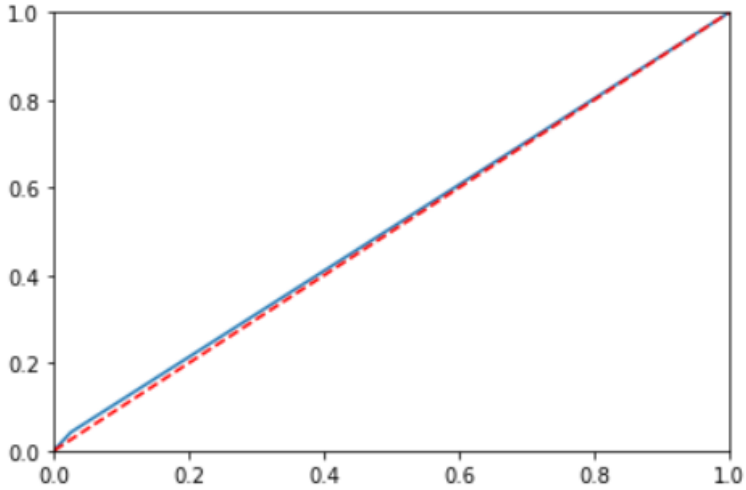




Test:

	precision	recall	f1-score	support
0	0.74	0.98	0.84	11092
1	0.37	0.04	0.07	3908
accuracy			0.73	15000
macro avg	0.56	0.51	0.46	15000
weighted avg	0.65	0.73	0.64	15000



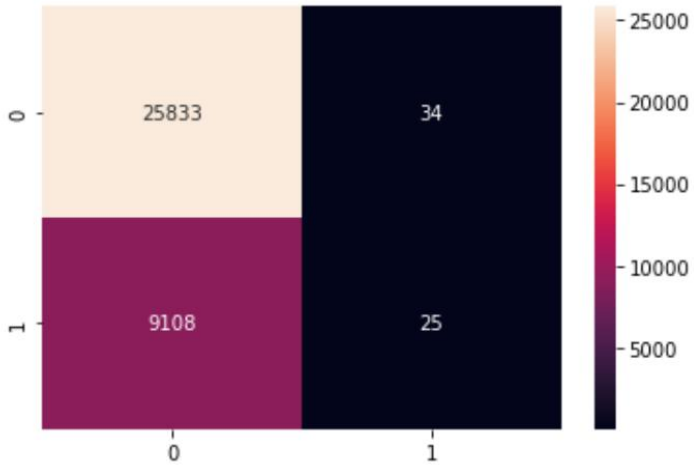


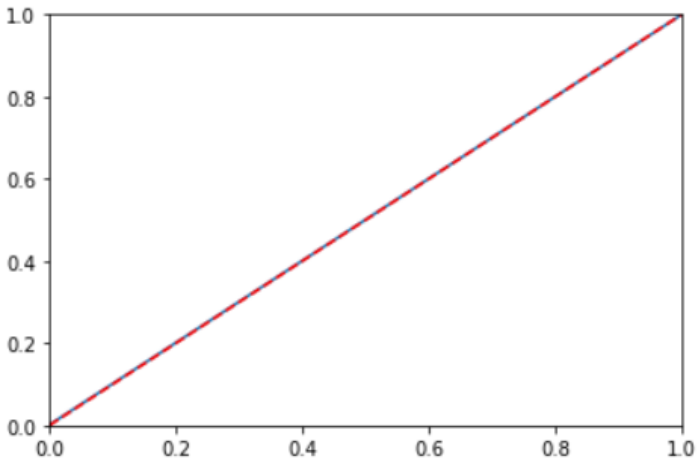
The train data performs moderately in classification of the classes, where as the test does not perform better than the train.

Ada-Boost

Train:

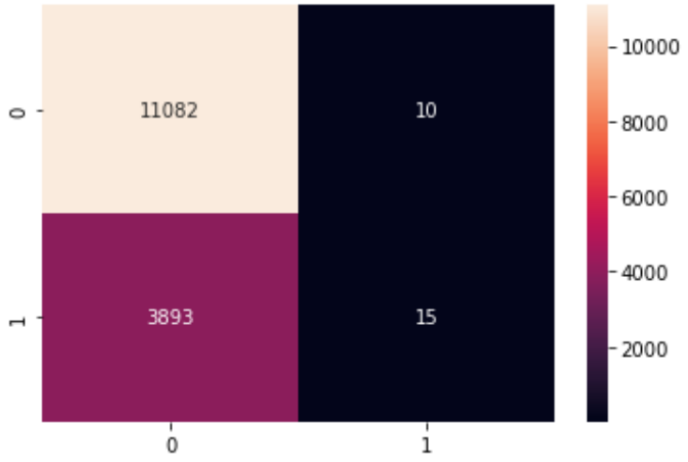
	precision	recall	f1-score	support
0	0.74	1.00	0.85	25867
1	0.42	0.00	0.01	9133
accuracy			0.74	35000
macro avg	0.58	0.50	0.43	35000
weighted avg	0.66	0.74	0.63	35000

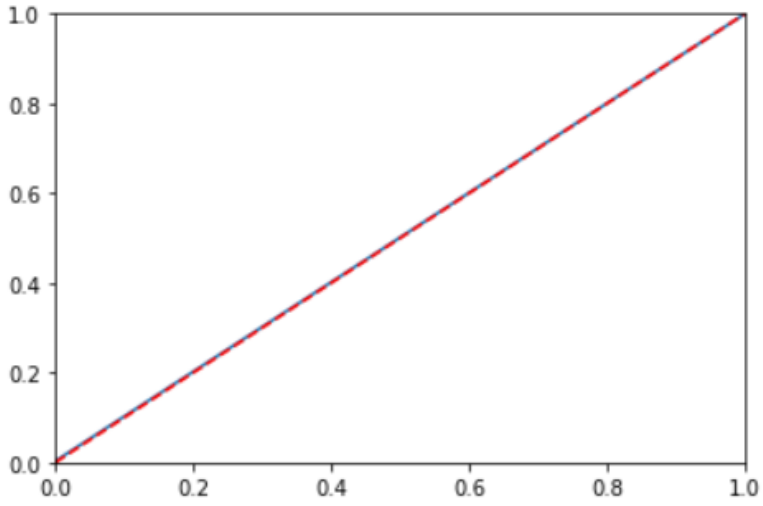




Test:

	precision	recall	f1-score	support
0	0.74	1.00	0.85	11092
1	0.60	0.00	0.01	3908
accuracy			0.74	15000
macro avg	0.67	0.50	0.43	15000
weighted avg	0.70	0.74	0.63	15000





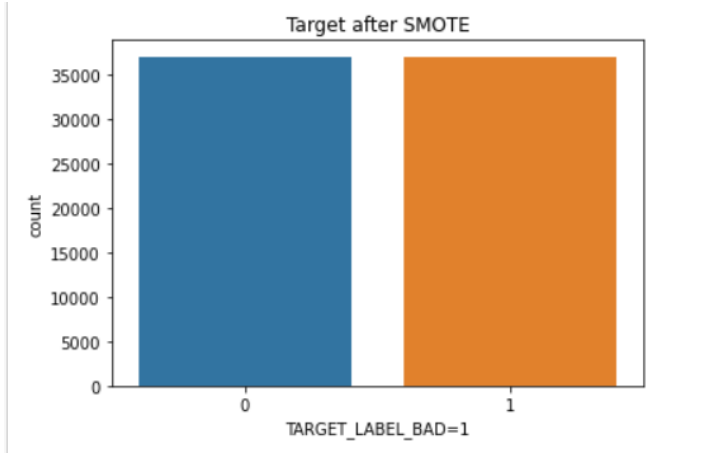
We can see that the positive classes are not being predicted properly compared to other models built previously.

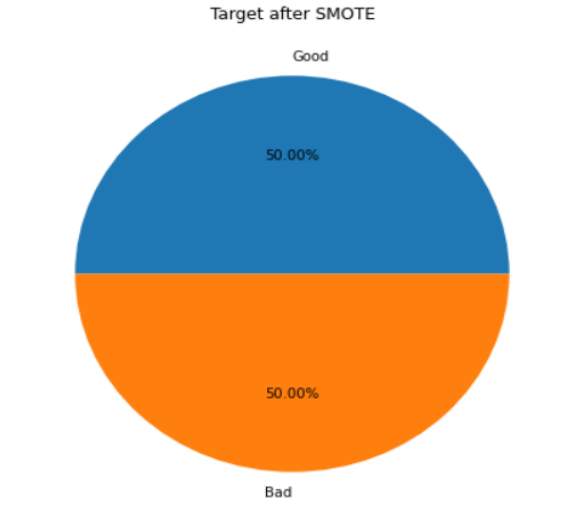
Inference from Base Model:

From all the above models, we can see that there is a severe drawdown in the train prediction and test prediction scores. This is mainly because of the imbalance in the target variable. This is evident from the classification report as the 0 class is predicted very well in some models, where as the 1 class is not predicted as expected. This is because there is a huge imbalance of 40% in the dataset, where the 0 class has 74% of the datapoints and the 1 class only has 26% of the data points. This can be rectified using the SMOTE function in python.

SMOTE

Smoting is the technique used to rectify any imbalance in the dataset. We use this for our dataset as there is a severe imbalance in the 0 and 1 class





From the above images, we can see that the imbalances are rectified and hence we can use this smoted dataset for out further models.

REBUILDING THE MODELS

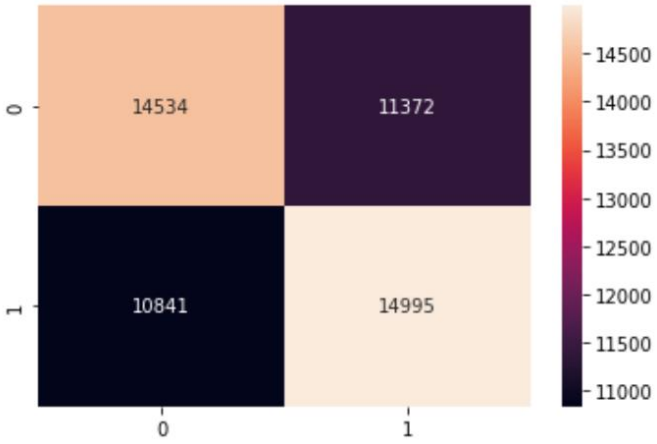
After smoting the data, we will rebuild the models to check the performance of the models.

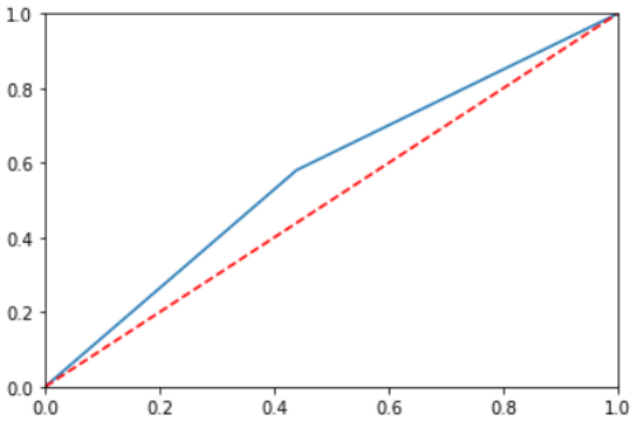
Logistic Regression:

Train:

	precision	recall	f1-score	support
0	0.57	0.56	0.57	25906
1	0.57	0.58	0.57	25836
accuracy			0.57	51742
macro avg	0.57	0.57	0.57	51742
weighted avg	0.57	0.57	0.57	51742

The classification report shows better scores after the smoting of the data.

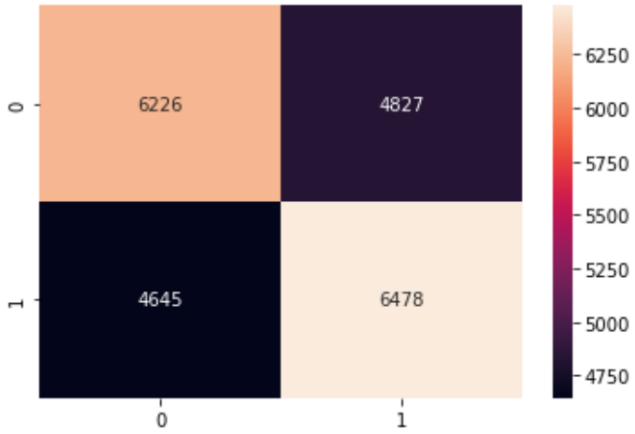




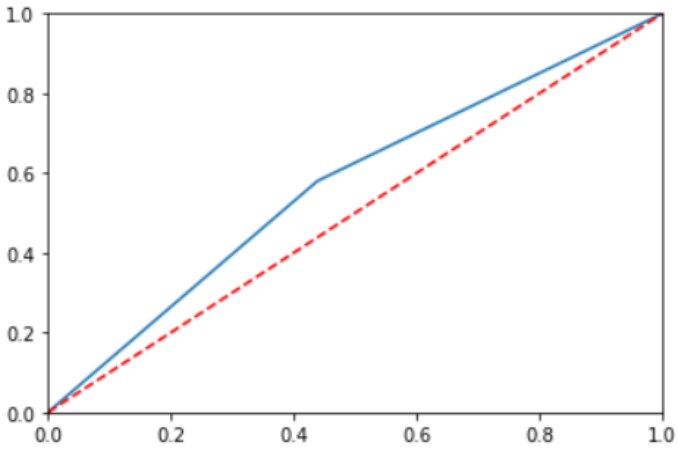
Test:

	precision	recall	f1-score	support
0	0.57	0.56	0.57	11053
1	0.57	0.58	0.58	11123
accuracy			0.57	22176
macro avg	0.57	0.57	0.57	22176
weighted avg	0.57	0.57	0.57	22176

We see that the test model performs better than the model before smoting. We can see the clear classification of class 1 and class 0.



The confusion matrix and ROC curve also shows better classification of the true positives and true negatives.

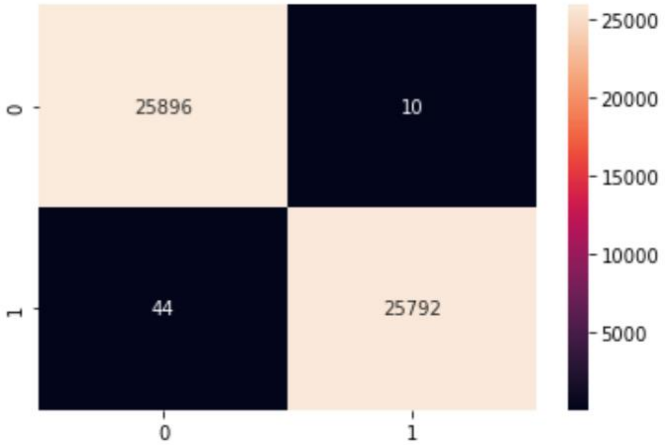


Decision Tree:

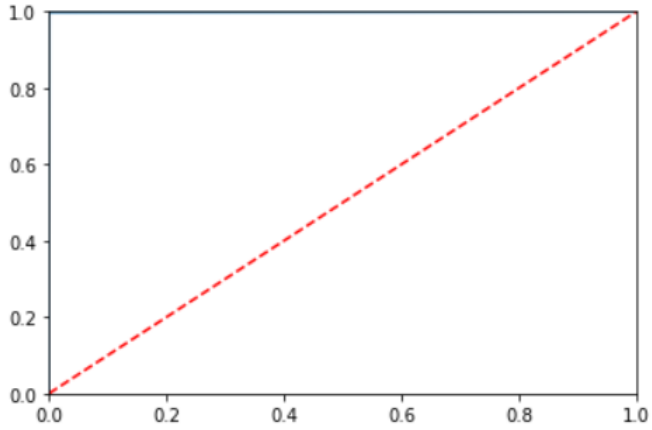
Train:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	25906
1	1.00	1.00	1.00	25836
accuracy			1.00	51742
macro avg	1.00	1.00	1.00	51742
weighted avg	1.00	1.00	1.00	51742

The classification report shows 100% classification results, which might be the case because of over fit. We will check it out by running a test model.



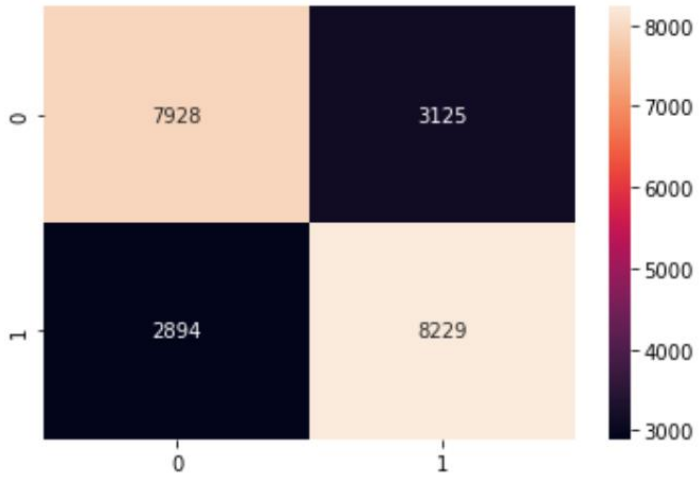
The confusion matrix also shows good classification of true positives and negatives with minimal negatives.



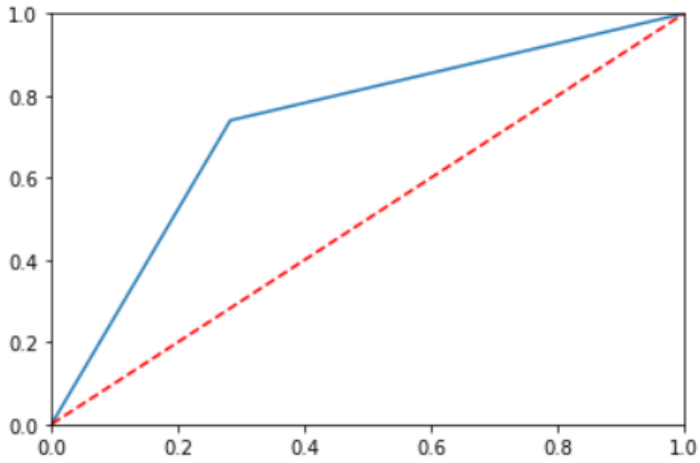
Test:

	precision	recall	f1-score	support
0	0.73	0.72	0.72	11053
1	0.72	0.74	0.73	11123
accuracy			0.73	22176
macro avg	0.73	0.73	0.73	22176
weighted avg	0.73	0.73	0.73	22176

We see that the classification report shows a decline in the scores from the test model vs the train model. This might be a case of over fit. Lets build more models to check if there is a better fit model.



The confusion matrix and ROC curve also shows a better prediction for the data that is not smoted.

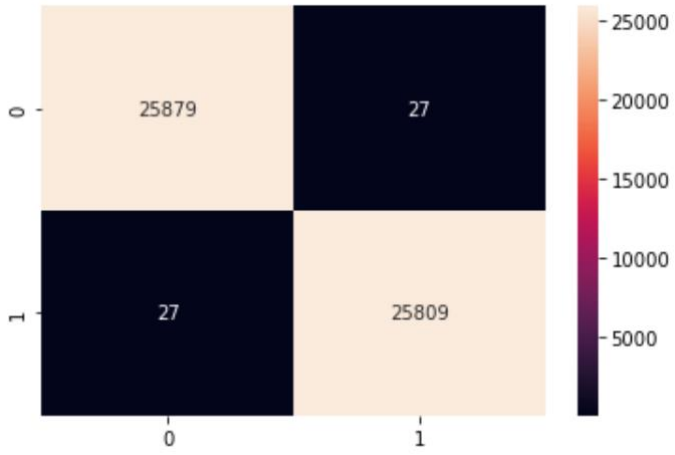


Random Forest:

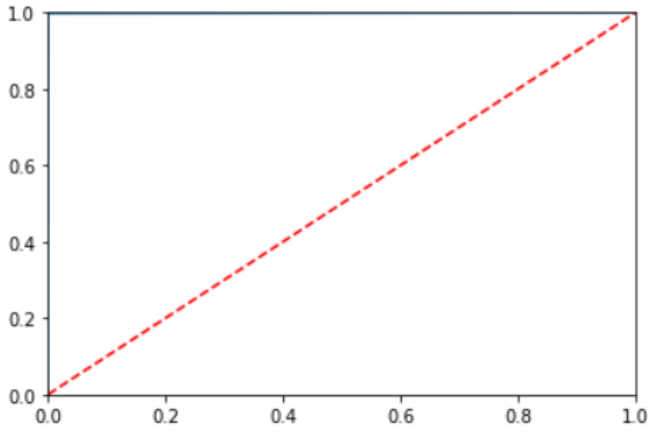
Train:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	25906
1	1.00	1.00	1.00	25836
accuracy			1.00	51742
macro avg	1.00	1.00	1.00	51742
weighted avg	1.00	1.00	1.00	51742

The classification report shows 100% classification results, which might be the case because of over fit. We will check it out by running a test model.



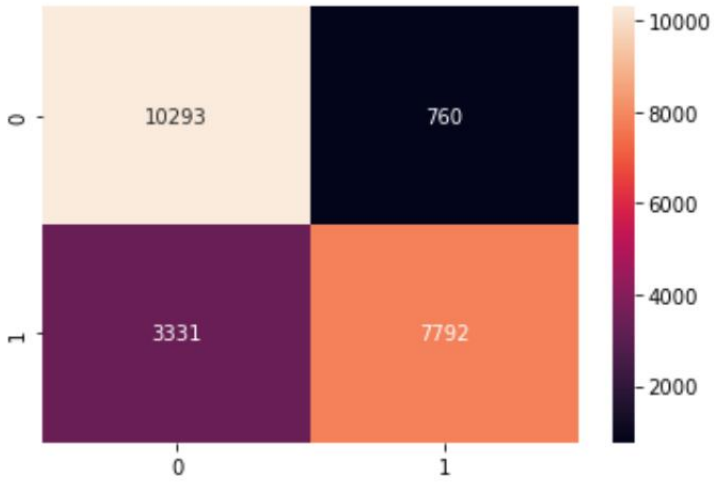
The confusion matrix also shows good classification of true positives and negatives with minimal negatives.



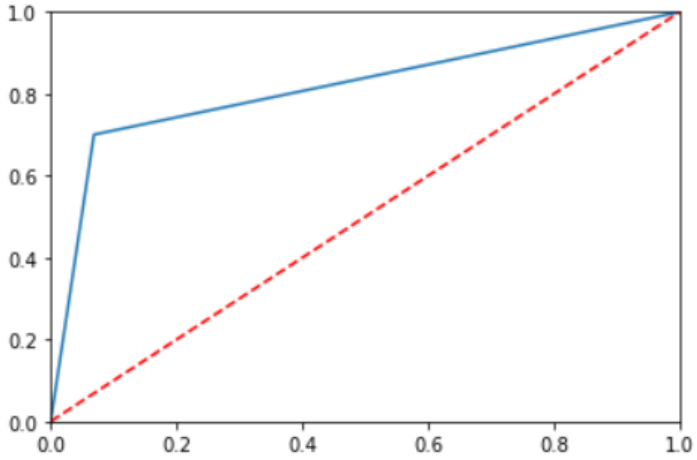
Test:

	precision	recall	f1-score	support
0	0.76	0.93	0.83	11053
1	0.91	0.70	0.79	11123
accuracy			0.82	22176
macro avg	0.83	0.82	0.81	22176
weighted avg	0.83	0.82	0.81	22176

We see that the classification report shows better scores from the test model vs the test model from the decision tree model.



The confusion matrix and ROC curve also shows a better prediction for the data that is not smoted.

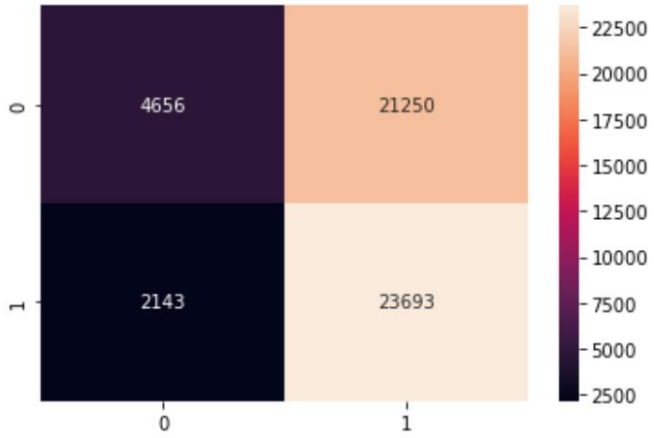


Naïve Bayes:

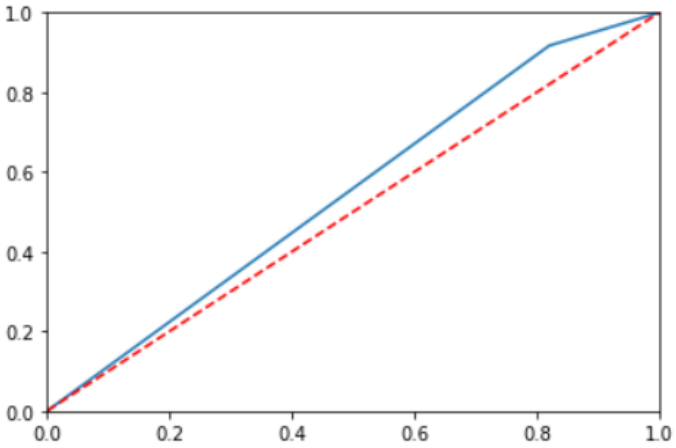
Train:

	precision	recall	f1-score	support
0	0.68	0.18	0.28	25906
1	0.53	0.92	0.67	25836
accuracy			0.55	51742
macro avg	0.61	0.55	0.48	51742
weighted avg	0.61	0.55	0.48	51742

The model is not classifying the target variable as good as the decision tree or random forest. Hence, this might not the model we will go for.



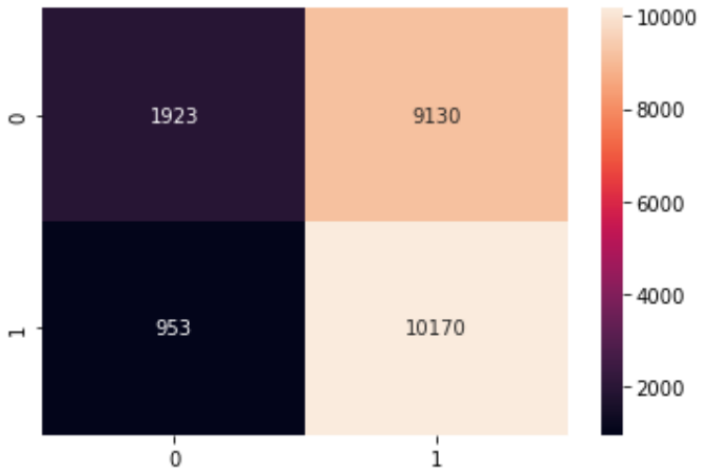
The confusion matrix and ROC curve also shows the bad performance of the model, even on the train data.



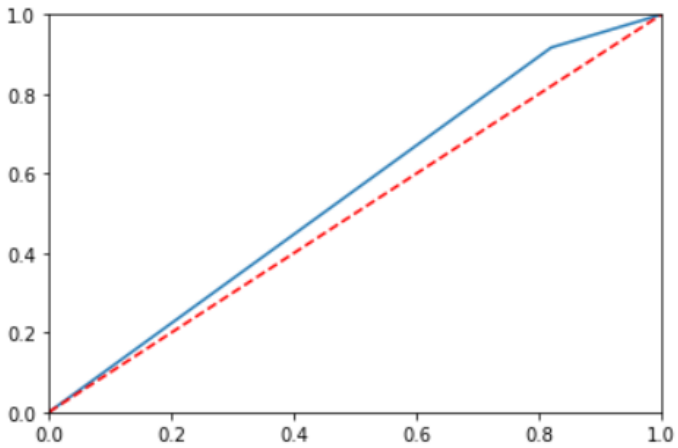
Test:

	precision	recall	f1-score	support
0	0.67	0.17	0.28	11053
1	0.53	0.91	0.67	11123
accuracy			0.55	22176
macro avg	0.60	0.54	0.47	22176
weighted avg	0.60	0.55	0.47	22176

The model performs the same as the train data.



The confusion matrix and the ROC curve proves the same.

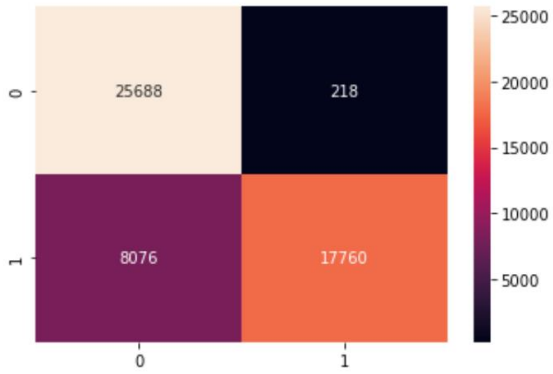


XG – Boost

Train:

	precision	recall	f1-score	support
0	0.76	0.99	0.86	25906
1	0.99	0.69	0.81	25836
accuracy			0.84	51742
macro avg	0.87	0.84	0.84	51742
weighted avg	0.87	0.84	0.84	51742

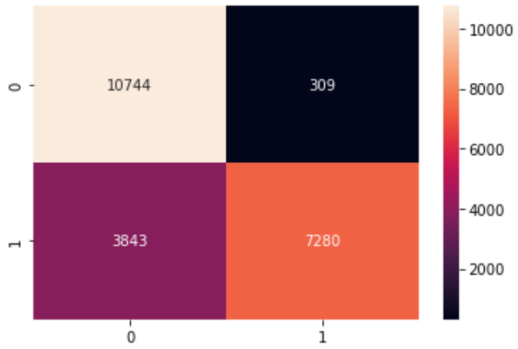
The XG-Boost shows no signs of over fit, and it also shows good classification of the 1 class. We can observe this with the help of the precision score.



The confusion matrix shows low false positive rate which is important in our model. The ROC curve also seems to show good plot on the false positive vs true positive rates.

Test:

The test model also gives good scores and the classification of the class 1. This model can be a viable option for our problem.



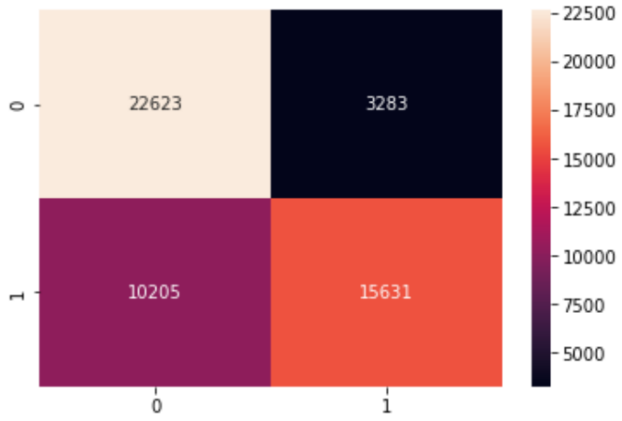
The confusion matrix and ROC curve also show better classification of the 1 class compared to the other over fit models.

Ada Boost:

Train:

	precision	recall	f1-score	support
0	0.69	0.87	0.77	25906
1	0.83	0.61	0.70	25836
accuracy			0.74	51742
macro avg	0.76	0.74	0.73	51742
weighted avg	0.76	0.74	0.73	51742

This model does fairly good compared to the other models and there are no signs of over fit.

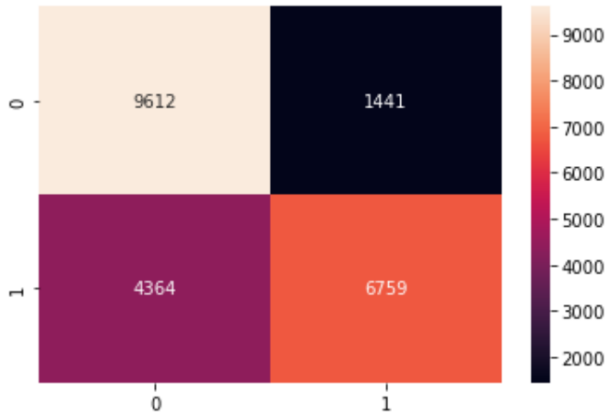


The confusion matrix is better than that off the non smoted models.

Test:

	precision	recall	f1-score	support
0	0.69	0.87	0.77	11053
1	0.82	0.61	0.70	11123
accuracy			0.74	22176
macro avg	0.76	0.74	0.73	22176
weighted avg	0.76	0.74	0.73	22176

The test model also performs better than the model from the non smoted model.



Inference from models after smoting:

From the above models, we can see that there is a significant improvement in the scores after we have fixed the imbalance in the target variable. We can also see that the Random forest and XG-Boost have good scores compared to the rest of the models on both train and test data

Hyper – Parameter Tuning:

Now that we have built the base model and fixed the data imbalance, the next step is to tune the model such that we get better results in our prediction. **Gradient Descent** method for only Random Forest Classifier as it gave the best accuracy after smoting

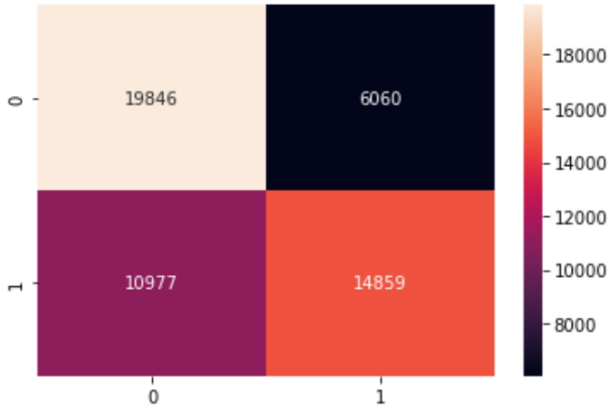
```
rf_grid=grid.fit(X1,y1)
print(rf_grid.best_params_)
```

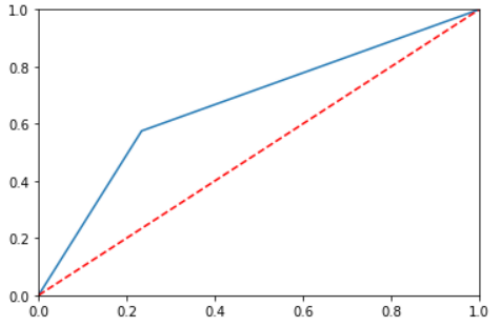
```
{'criterion': 'entropy', 'max_leaf_nodes': 9, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estim
ators': 10}
```

Random Forest Model using Best Parameters:

Train:

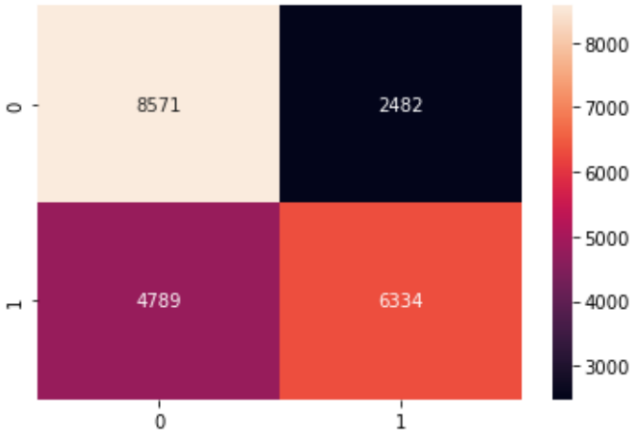
	precision	recall	f1-score	support
0	0.64	0.77	0.70	25906
1	0.71	0.58	0.64	25836
accuracy			0.67	51742
macro avg	0.68	0.67	0.67	51742
weighted avg	0.68	0.67	0.67	51742

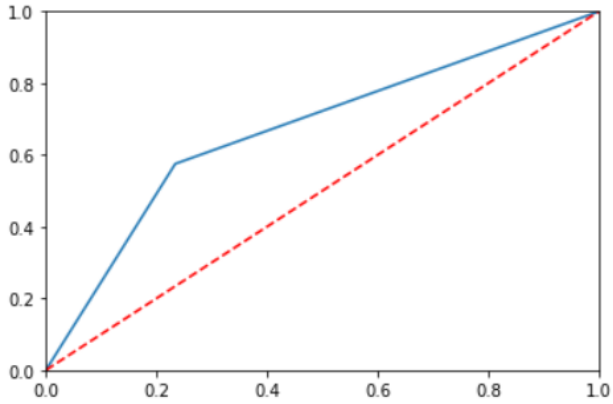




Test:

	precision	recall	f1-score	support
0	0.64	0.78	0.70	11053
1	0.72	0.57	0.64	11123
accuracy			0.67	22176
macro avg	0.68	0.67	0.67	22176
weighted avg	0.68	0.67	0.67	22176





We see that there is a significant degrade in the model performance compared to the base model random forest. Hence we will go with the base random forest and check for other ways to tune the model

Model with Significant Variables:

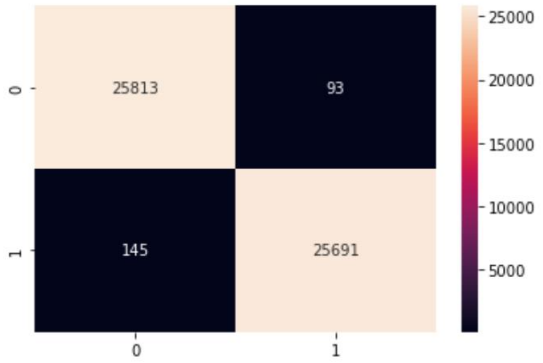
Next we try building the model with the significant variables, ie, using the variables which have p-values less than 0.05

```
sig_df=smote_df.drop(['RESIDENCIAL_STATE','RESIDENCE_TYPE','FLAG_DINERS','FLAG_AMERICAN_EXPRESS','FLAG_PROFESSIONAL_PHONE'
                      ,'PRODUCT'],axis=1)
```

Random Forest:

Train:

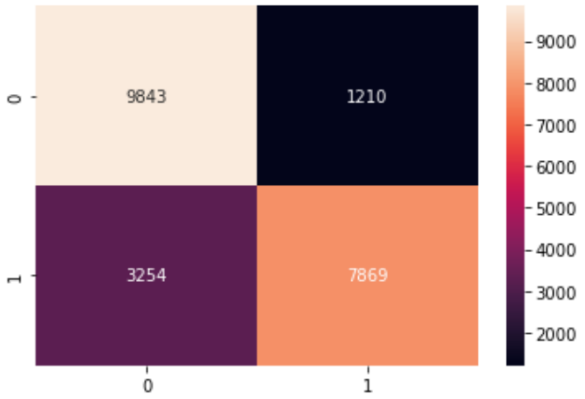
	precision	recall	f1-score	support
0	0.99	1.00	1.00	25906
1	1.00	0.99	1.00	25836
accuracy			1.00	51742
macro avg	1.00	1.00	1.00	51742
weighted avg	1.00	1.00	1.00	51742



From the above metrics, we can see that the model over fits on the train data. Lets check the performance on the test data for further inference.

Test:

	precision	recall	f1-score	support
0	0.75	0.89	0.82	11053
1	0.87	0.71	0.78	11123
accuracy			0.80	22176
macro avg	0.81	0.80	0.80	22176
weighted avg	0.81	0.80	0.80	22176

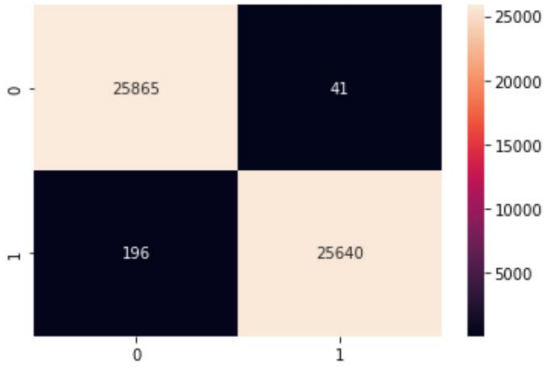


The model performs good on the test data but it might be subject to over fit due to the models performance on the train data.

Decision Tree:

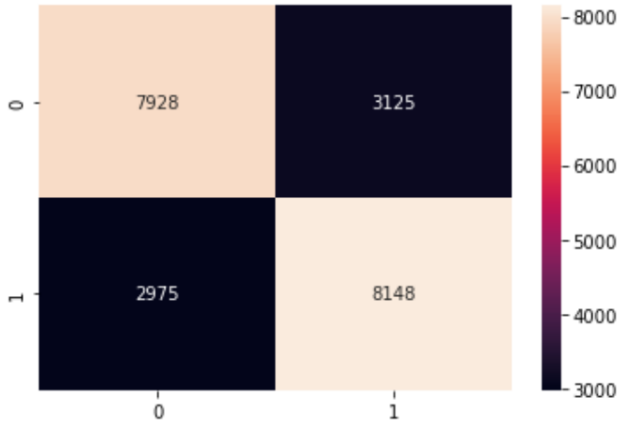
Train:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	25906
1	1.00	0.99	1.00	25836
accuracy			1.00	51742
macro avg	1.00	1.00	1.00	51742
weighted avg	1.00	1.00	1.00	51742



Test:

	precision	recall	f1-score	support
0	0.73	0.72	0.72	11053
1	0.72	0.73	0.73	11123
accuracy			0.72	22176
macro avg	0.72	0.72	0.72	22176
weighted avg	0.72	0.72	0.72	22176



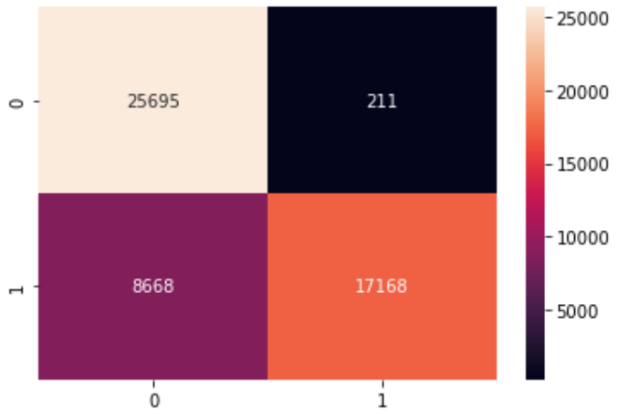
This is also an over fit model. Hence, we might have to look at the other models as the model is a great fit on the train data but is performing relatively bad on the test dat compared to the normal smoted model.

XG-Boost:

Train:

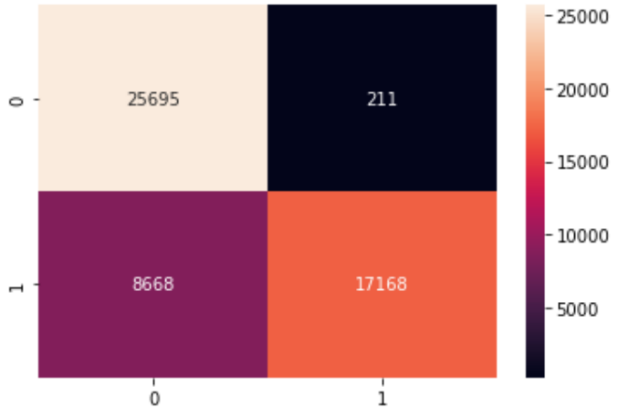
	precision	recall	f1-score	support
0	0.75	0.99	0.85	25906
1	0.99	0.66	0.79	25836
accuracy			0.83	51742
macro avg	0.87	0.83	0.82	51742
weighted avg	0.87	0.83	0.82	51742

We see that in the XG-Boost model, we do not have any kind of over fit and the classifications are also relatively good for the class 1.



Test:

	precision	recall	f1-score	support
0	0.73	0.98	0.83	11053
1	0.97	0.64	0.77	11123
accuracy			0.81	22176
macro avg	0.85	0.81	0.80	22176
weighted avg	0.85	0.81	0.80	22176



From all the above models, we see that the random forest is performing worse after removing all the insignificant columns but the XG-Boost works better compared to the random forest. Hence, we might select the XG-Boost as our model as it performs better on the smoted data as well as the model after removing the insignificant variables.

Model Interpretation:

From all the above models, we can see that the XG-Boost model gives the best fit for the data after smoting as well as removing the insignificant variables. The Random Forest and Decision Tree models also give good results on the smoted data, but the model over fits on the data and hence giving us sub-optimal classification on the test data compared to the train data. Hence we go ahead with the XG-Boost model to avoid over fitting and to keep intact the precision of the model.

REFERENCES:

- CFI-Institute: https://corporatefinanceinstitute.com/resources/knowledge/credit/
- PAKKD Data Mining: https://github.com/deepanshu88/Datasets/blob/master/CreditData/PAKDD%202010.zip
- Brandon Foltz: https://www.youtube.com/c/BrandonFoltz/playlists