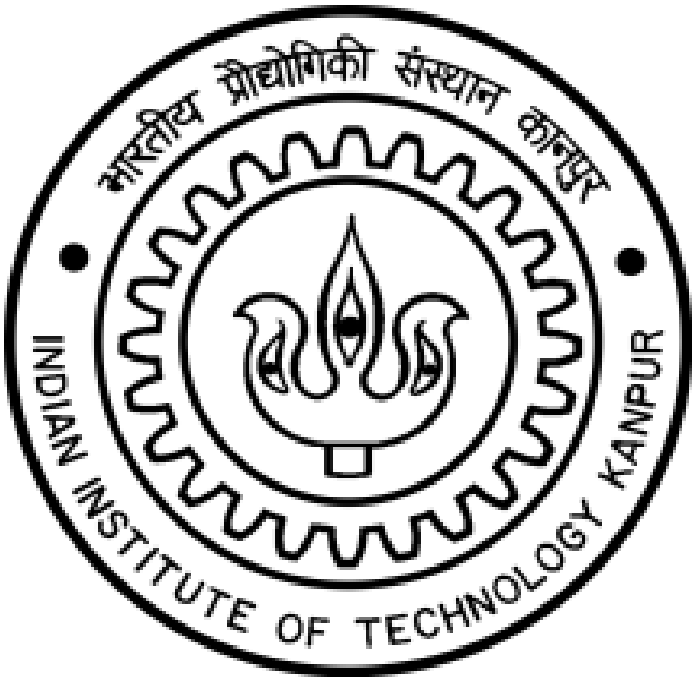


Project Report for: *IME 673A*

Applied Machine Learning: Prediction of Vehicle insurance purchase

Under the Supervision of:

Dr. Veena Bansal

PREPARED BY: Prince k Singh (M.TECH 2nd sem)

roll no- 20114014

Problem Statement:

Our client is an Insurance company that has provided Health Insurance to its customers now they need your help in building a model to predict whether the policyholders (customers) from past year will also be interested in Vehicle Insurance provided by the company.

An insurance policy is an arrangement by which a company undertakes to provide a guarantee of compensation for specified loss, damage, illness, or death in return for the payment of a specified premium. A premium is a sum of money that the customer needs to pay regularly to an insurance company for this guarantee.

For example, you may pay a premium of Rs. 5000 each year for a health insurance cover of Rs. 200,000/- so that if, God forbid, you fall ill and need to be hospitalized in that year, the insurance provider company will bear the cost of hospitalization etc. for up to Rs. 200,000. Now if you are wondering how can company bear such high hospitalization cost when it charges a premium of only Rs. 5000/-, that is where the concept of probabilities comes in picture. For example, like you, there may be 100 customers who would be paying a premium of Rs. 5000 every year, but only a few of them (say 2-3) would get hospitalized that year and not everyone. This way everyone shares the risk of everyone else.

Just like medical insurance, there is vehicle insurance where every year customer needs to pay a premium of certain amount to insurance provider company so that in case of unfortunate accident by the vehicle, the insurance provider company will provide a compensation (called 'sum assured') to the customer.

Building a model to predict whether a customer would be interested in Vehicle Insurance is extremely helpful for the company because it can then accordingly plan its communication strategy to reach out to those customers and optimize its business model and revenue.

Now, in order to predict, whether the customer would be interested in Vehicle insurance, you have information about demographics (gender, age, region code type), Vehicles (Vehicle Age, Damage), Policy (Premium, sourcing channel) etc.

Business Goal:

Building a model to predict whether a customer would be interested in Vehicle Insurance is extremely helpful for the company because it can then accordingly plan its communication strategy to reach out to those customers and optimize its business model and revenue.

Now, in order to predict, whether the customer would be interested in Vehicle insurance, you have information about demographics (gender, age, region code type), Vehicles (Vehicle Age, Damage), Policy (Premium, sourcing channel) etc.

2. Methods and Data

We used the dataset from the Kaggle: https://www.kaggle.com/arashnic/imbalanced-data-practice?select=aug_train.csv

Coding part is from different source like Different Library documentation, different sites like Analytic Vidhya , Toward Data science, Kaggle and By Self
Detailed information of data is provided under the above link but here we give a summary of datasets and how we cleaned up the data.
The data includes two datasets but we are using training data set only

- Training data
- Test data

Each row of these datasets carries information about one customer information regarding vehicle insurance purchase and response for health insurance purchase . There are 12 features (independent variables) and 1 dependent variables from which we are going to predict 1 (purchase of health insurance) and 0 (not interested to purchase of health insurance). Training data has 381109 rows (data points).
For our analysis, we divided our training data to two sets: training and testing.

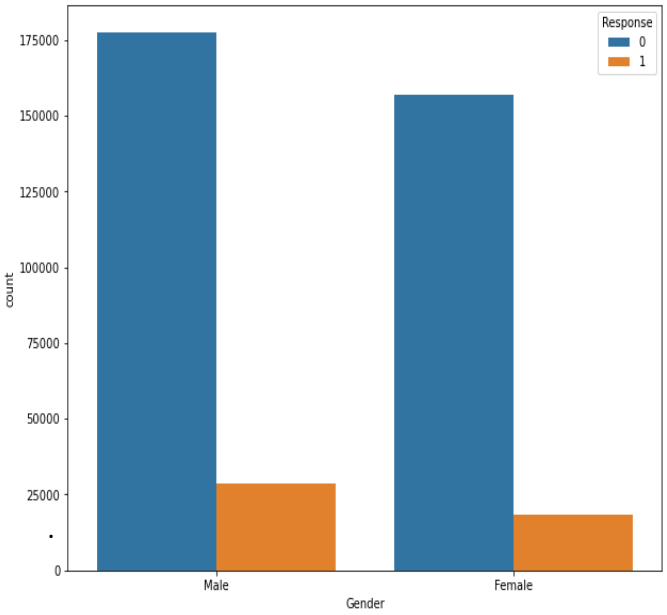
3. Data Cleaning and Exploratory Data Analysis

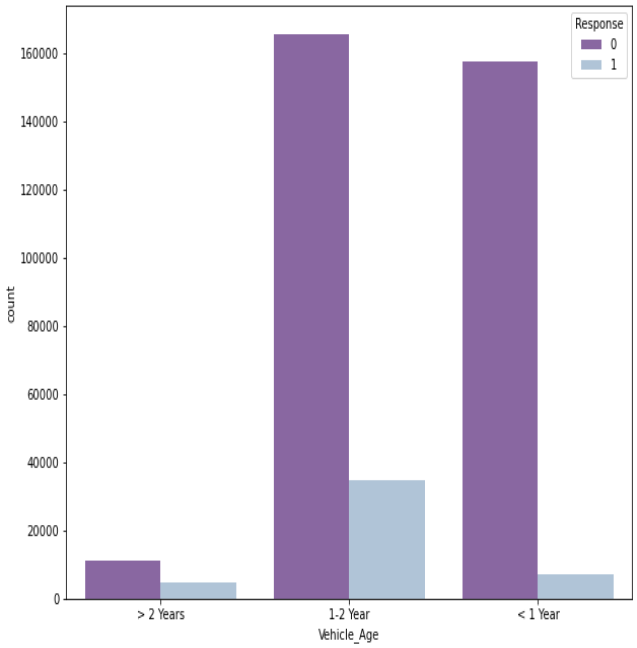
This dataset needed some cleanings and modification. Besides some feature representation should be done. Here is the complete list of features and brief description on how we modified them:

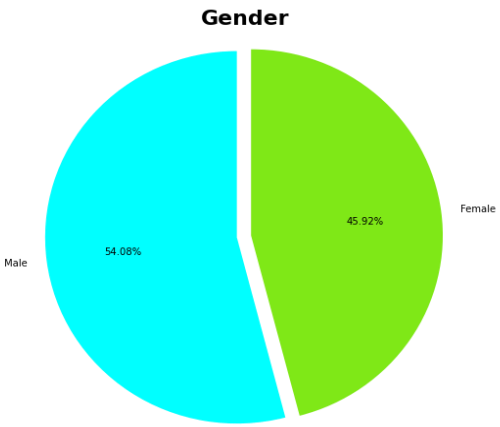
- First we check any missing value and find no null value in our dataset
- Response value count shows data is imbalance

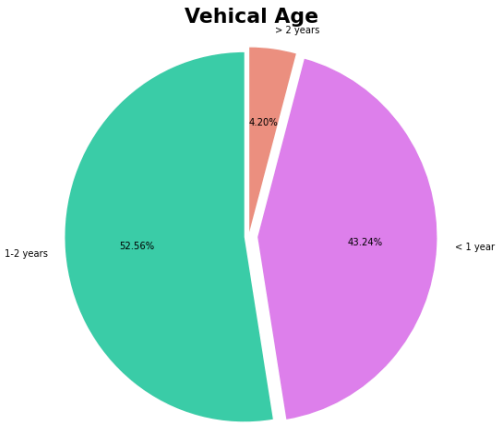
  0: 334399 1: 46710 Name: Response, dtype: int64

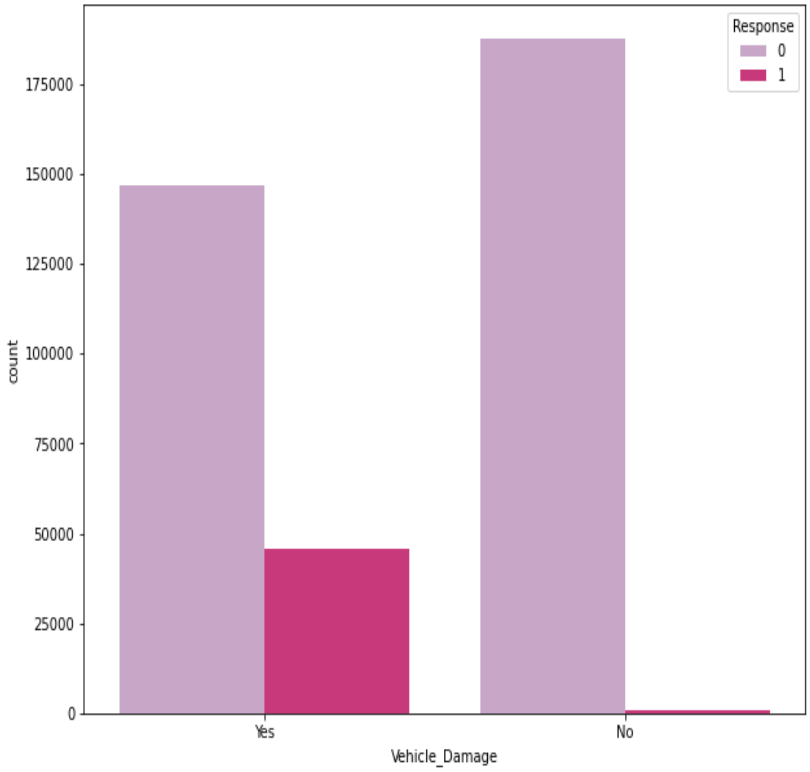
- Three categorical feature 'Gender', 'Vehicle_Age', 'Vehicle_Damage' we did univariate analysis of each categorical variable and count plot and pie chart plot and find that data is imbalance and also find individual count corresponding Responses

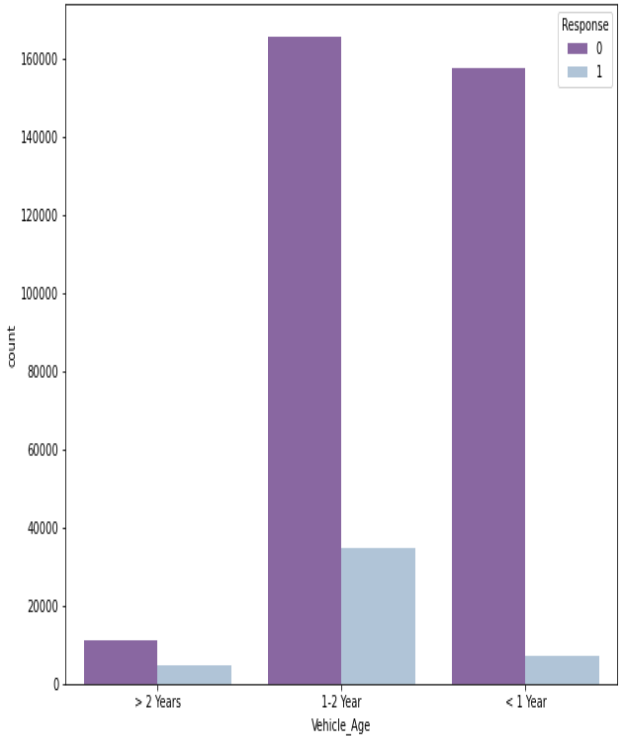








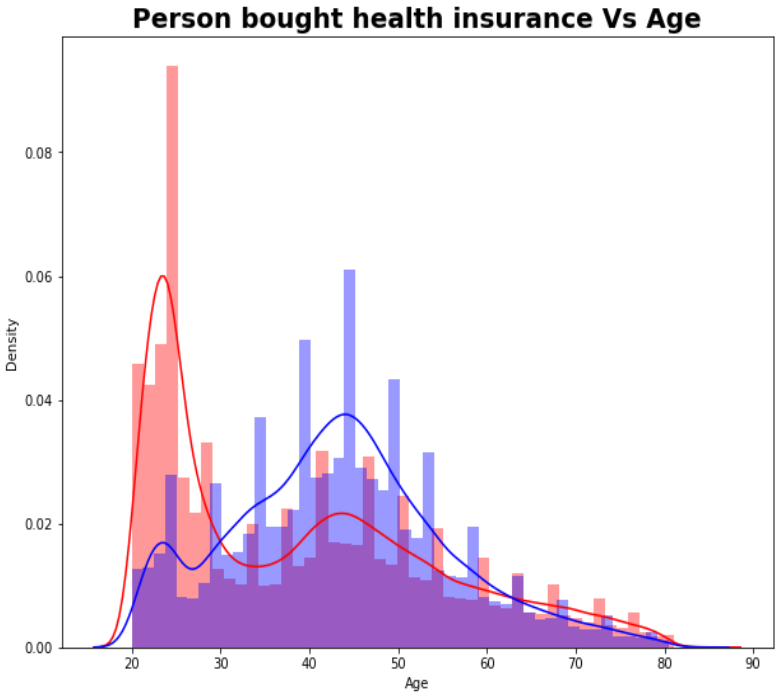


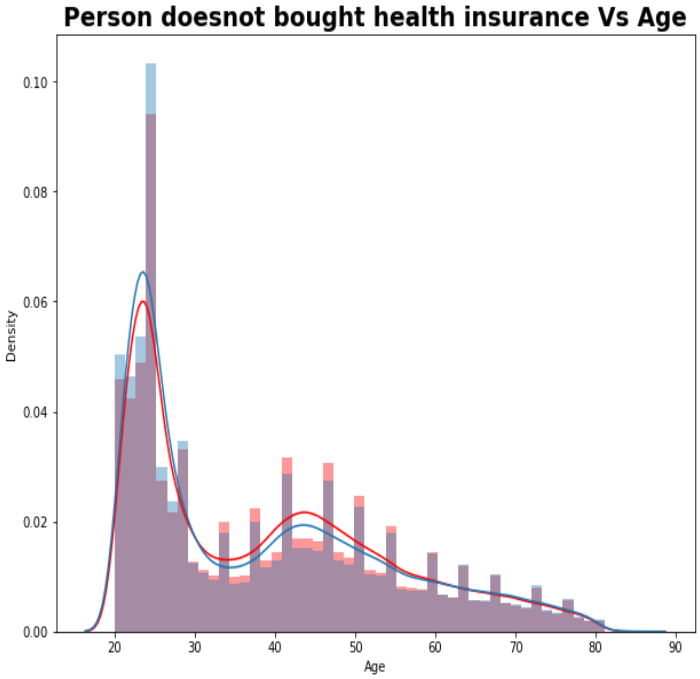


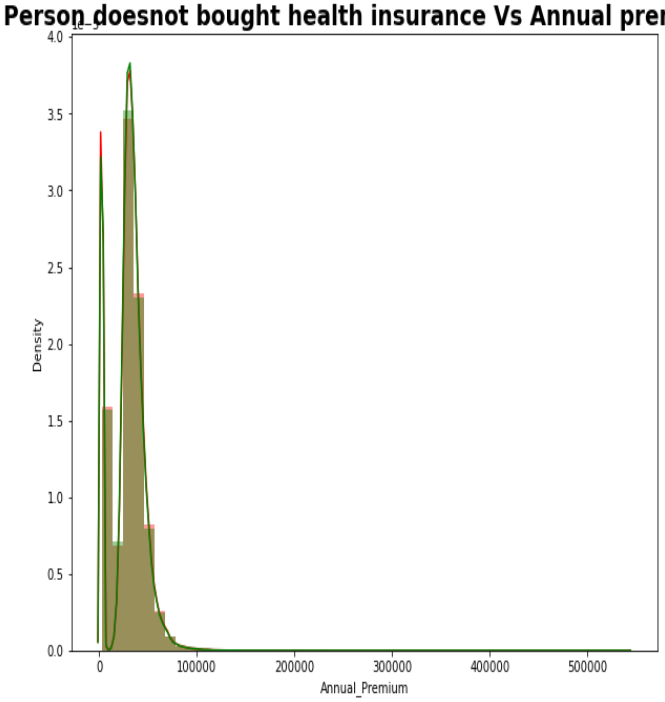
- Now we have univariate analysis of numerical features and find distribution plot of valuable feature Like Age and Annual premium corresponding Response value

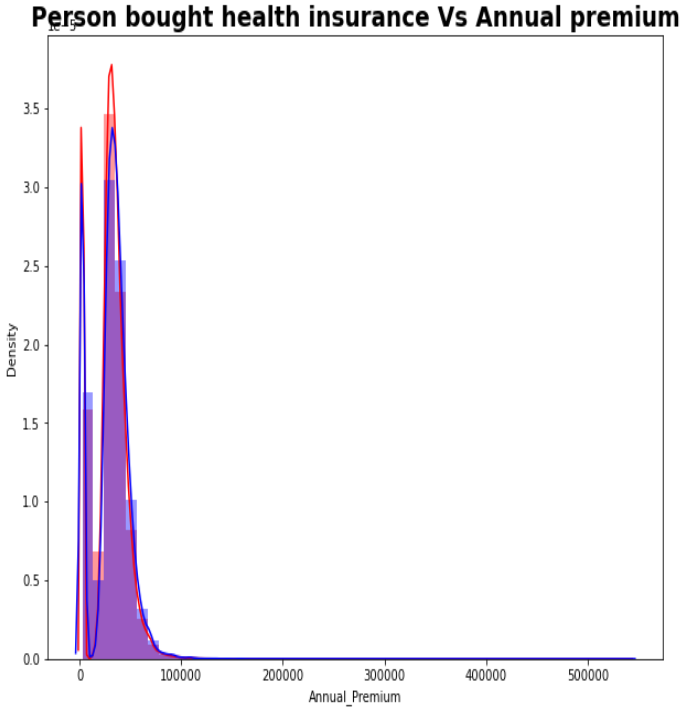
Age group of 40-50 have higher chance of buying the health insurance and also age distribution is skewed showing in red color and age corresponding to 1 response value means which is interested in buying insurance showing nearly normal distribution

- Annual Premium clearly show skewed distribution corresponding Response
- Scaling down the Features AGE and ANNUAL _PREMIUM(Standardization the data features)
- Do Label Encoding to categorical variable like ‘Gender’, ‘Vehicle_Age’, ‘Vehicle_Damage’

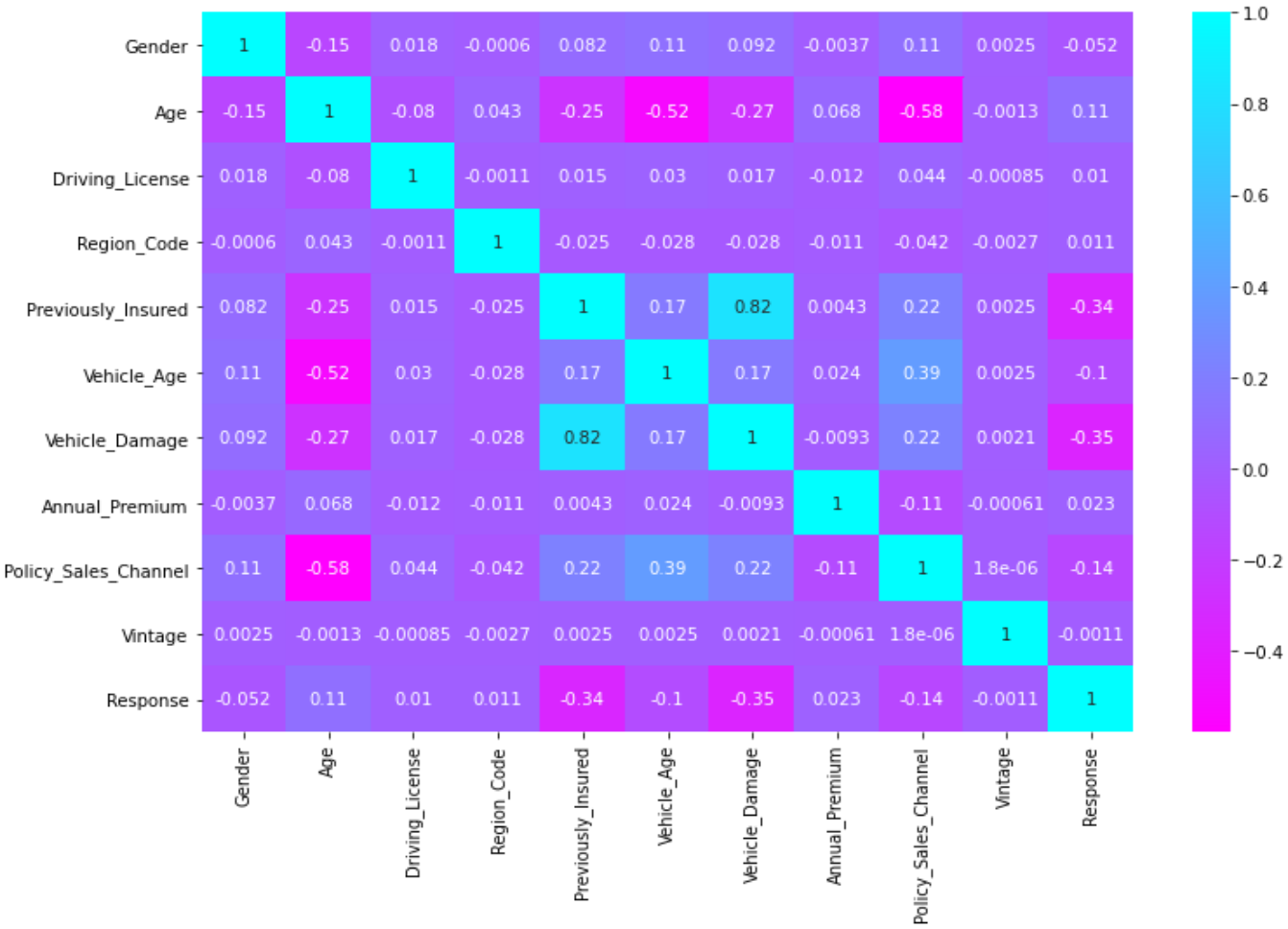








We analyze multivariate correlation factor to find correlation among different features and find only one Significant correlation factor 0.82 among Vehicle_Damage and Previously insured Also negative correlation Factor between of -0.52 Vehicle_Age and Age also -0.58 between policy channel and Age



Splitting the data

Before we start training the model, we'll need to split our dataset into a training and test portion.We'll use the training portion to train model and then evaluate it on the test portion to see how it performs on samples it hasn't seen before. It's also important to perform a stratified sampling which means that the probability of seeing a fraudulent transaction will be approximately the same in both the training data and the test data. Stratified sampling also ensures that our model metrics are as close as possible to what we'd see in a whole population

Modeling Building

1. **Classification Models**

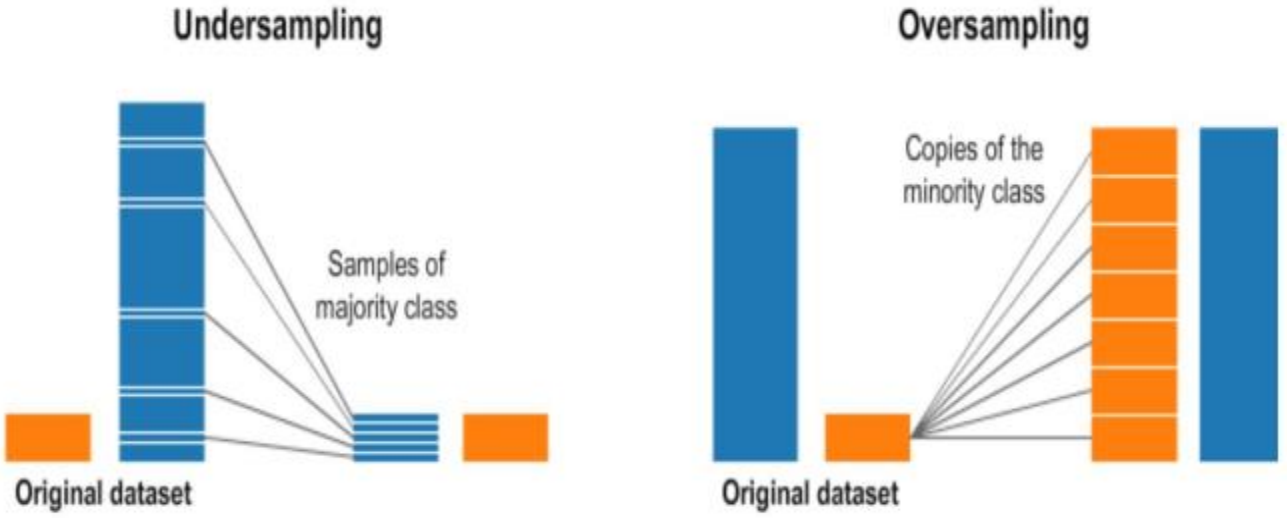
- Logistic Regression
- Decision Trees
- Random Forest
- Naive Bayes Classifier
- Xgbooster
- Gradient Booster
- KNN Classification
- Linear support vector machine

2 .**Class Imbalance Solutions**

- Under Sampling
- Over Sampling
- SMOTE

3. **Metrics(classification Report)**

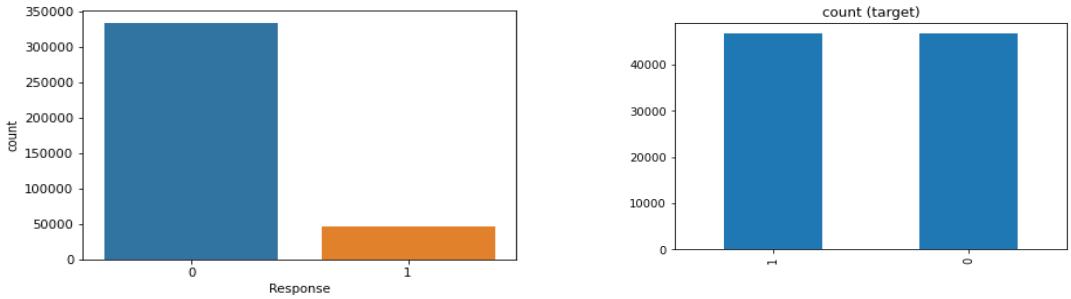
- Accuracy Score
- Confusion Matrix
- Precision Score
- Recall Score
- support value
- F1 Score



Under sampling and Oversampling: Under sampling techniques remove examples from the training dataset that belong to the majority class to better balance the class distribution, such as reducing the skew from a 1:100 to a 1:10, 1:2, or even a 1:1 class distribution. This is different from oversampling that involves adding examples to the minority class to reduce the skew in the class distribution.

For Under Sampled Data

- Original dataset shape Counter({0: 334399, 1: 46710})
- Resample dataset shape Counter({0: 46710, 1: 46710})



For Oversampled Data

- Original dataset shape Counter({0: 334399, 1: 46710})
- Resample dataset shape Counter({0: 334399, 1: 334399})

For SMOTE Data

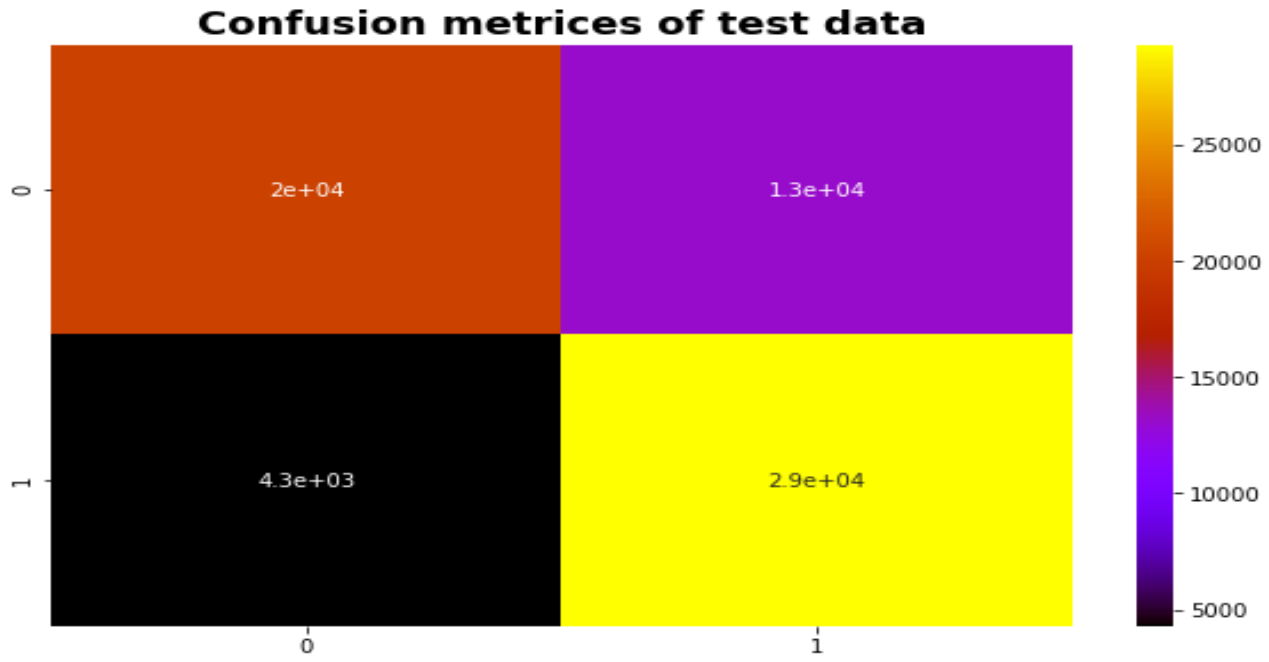
- Original dataset shape Counter({0: 334399, 1: 46710})
- Resample dataset shape Counter({0: 334399, 1: 334399})

Logistic Regression (LR)

Logistic regression mathematically means finding, $y = f(x)$, when y is a categorical variable. The use of this method is to find the label of categorical variable when know our predictors x. In our project the dependent variable y is a categorical variable, and the predictors are continuous variable. The input given to the model is the weighted average of attributes value and some bias. The input is processed by the logistic function called sigmoid function.

$$f(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x+\cdots)}}$$

where $\beta 0 + \beta 1x + \cdots$ is the weighted input given to the sigmoid function. The function processed the input data and result the output. The output is then compared with a threshold value. If the output is at least equal to the threshold value, then the output is 1 otherwise 0. Confusion Matrix is shown below for the Logistic Regression model on the Oversampling data, (which) turns out to give the best result out of the 4 sampling techniques employed.



Classification report of train data

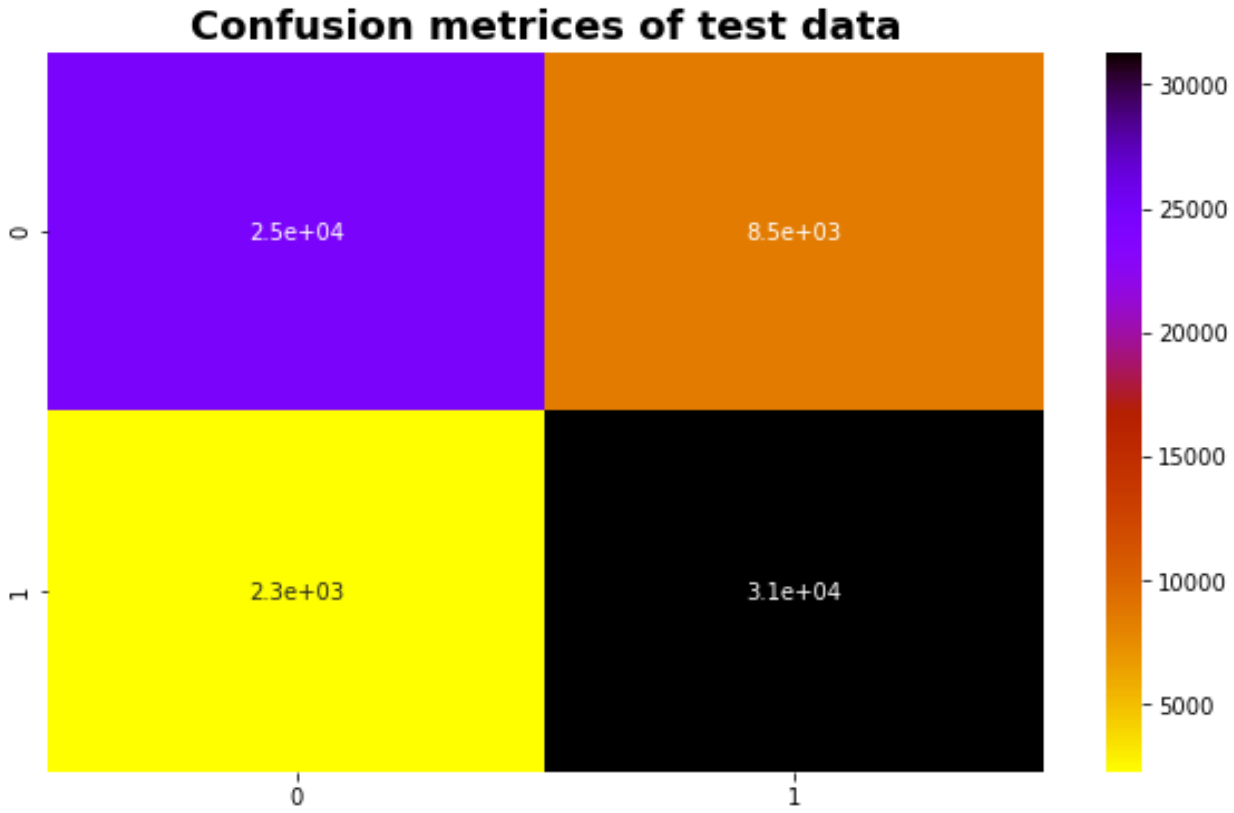
	precision	recall	f1-score	support
0	0.97	0.61	0.74	301112
1	0.71	0.98	0.82	300806
accuracy			0.79	601918
macro avg	0.84	0.79	0.78	601918
weighted avg	0.84	0.79	0.78	601918

Classification report of test data

	precision	recall	f1-score	support
0	0.96	0.61	0.75	33287
1	0.72	0.98	0.83	33593
accuracy			0.79	66880
macro avg	0.84	0.79	0.79	66880
weighted avg	0.84	0.79	0.79	66880

Decision Trees (DT)

Decision Trees makes use of Information Theory to apply classification procedure over the data set. Growing a tree involves deciding on **which features to choose** and **what conditions to use** for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, **you will need to trim it down** for it to be of use. Choosing of an attribute to split, lies at heart of the algorithm.



Classification report of train data

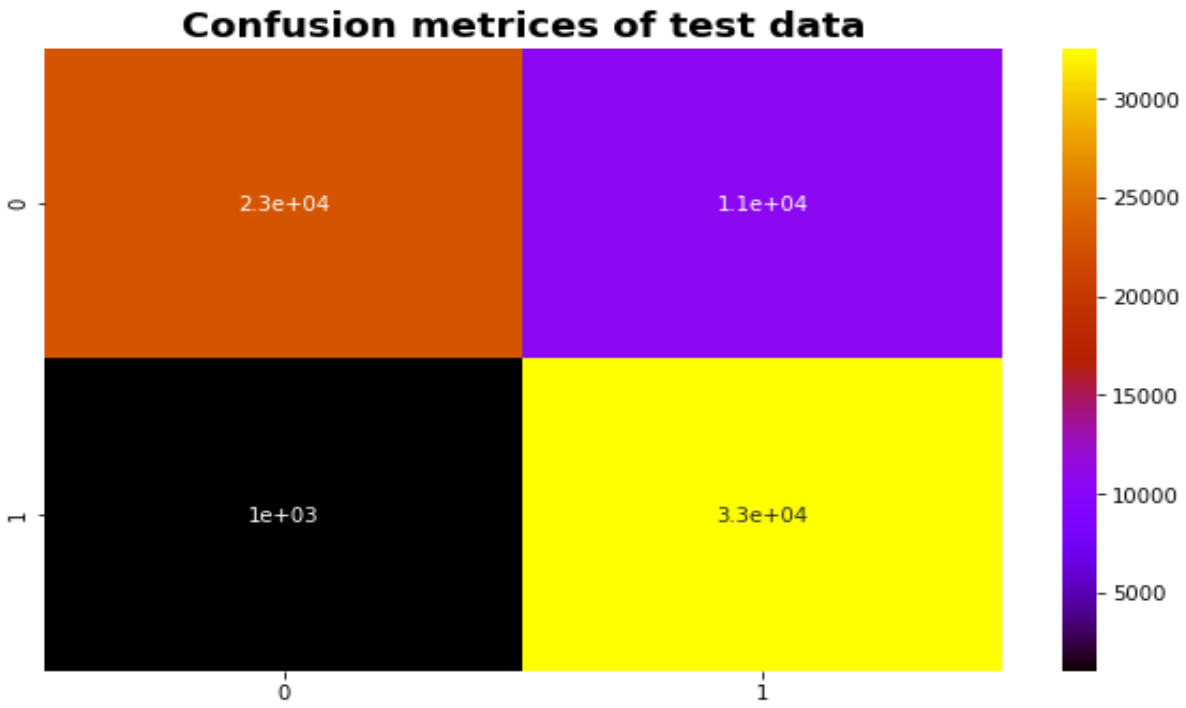
	precision	recall	f1-score	support
0	0.92	0.74	0.82	301112
1	0.78	0.93	0.85	300806
accuracy			0.84	601918
macro avg	0.85	0.84	0.84	601918
weighted avg	0.85	0.84	0.84	601918

Classification report of test data

	precision	recall	f1-score	support
0	0.92	0.74	0.82	33287
1	0.79	0.93	0.85	33593
accuracy			0.84	66880
macro avg	0.85	0.84	0.84	66880
weighted avg	0.85	0.84	0.84	66880

Random Forest (RF)

As its name suggests, Random Forest consists of many individual decision trees, that act as an ensemble. Every single tree in the Random Forest spits out a class prediction, and the class with most votes becomes the prediction of our model. Given below is the confusion matrix with the best result



Classification report of train data

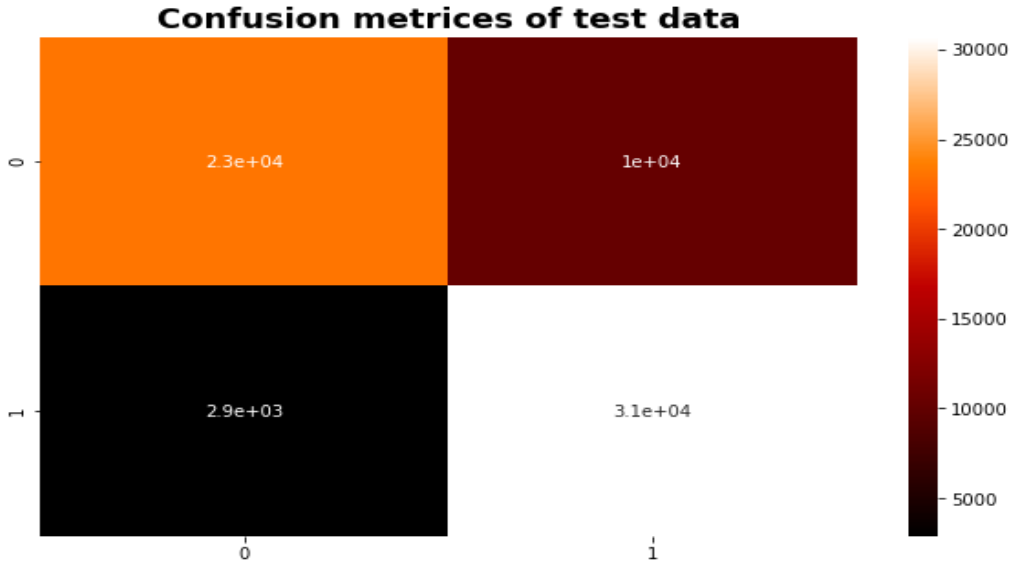
	precision	recall	f1-score	support
0	0.96	0.68	0.80	301112
1	0.75	0.97	0.85	300806
accuracy			0.83	601918
macro avg	0.85	0.83	0.82	601918
weighted avg	0.85	0.83	0.82	601918

Classification report of train data

	precision	recall	f1-score	support
0	0.96	0.68	0.80	33287
1	0.76	0.97	0.85	33593
accuracy			0.83	66880
macro avg	0.86	0.83	0.82	66880
weighted avg	0.86	0.83	0.82	66880

Naïve Bayes (NB)

Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive. While, on the other hand Decision Trees makes use of Information Theory to apply classification procedure over the data set. Choosing of an attribute to split, lies at heart of the algorithm.



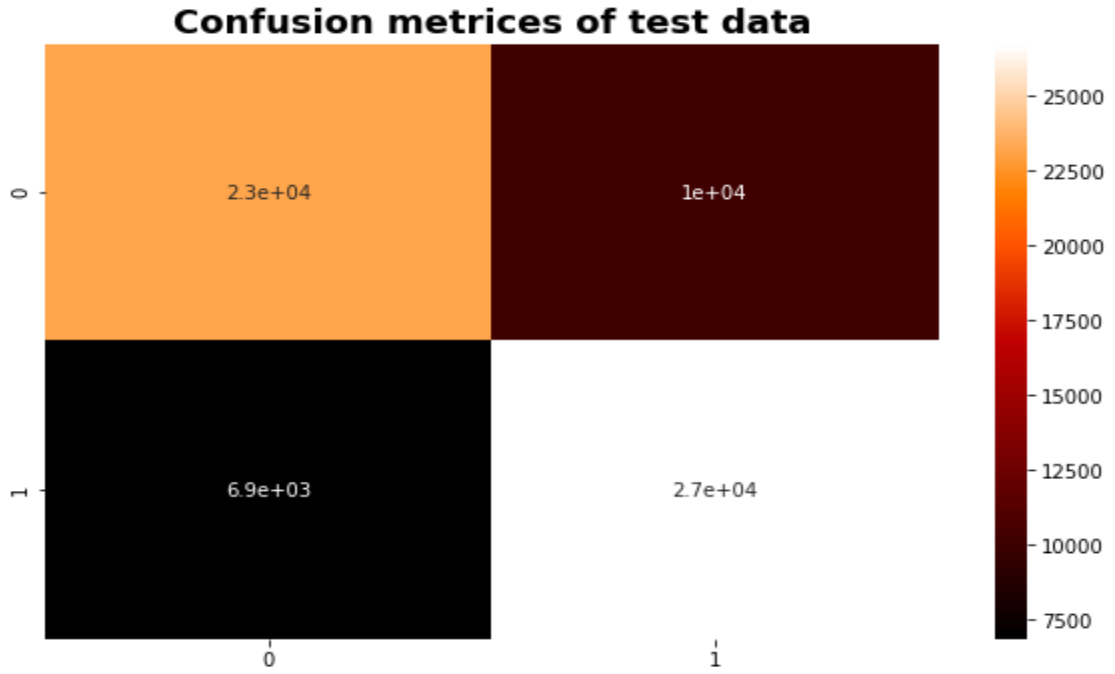
Classification report of train data

	precision	recall	f1-score	support
0	0.89	0.69	0.78	301112
1	0.75	0.91	0.82	300806
accuracy			0.80	601918
macro avg	0.82	0.80	0.80	601918
weighted avg	0.82	0.80	0.80	601918

Classification report of test data

	precision	recall	f1-score	support
0	0.89	0.69	0.78	33287
1	0.75	0.91	0.82	33593
accuracy			0.80	66880
macro avg	0.82	0.80	0.80	66880
weighted avg	0.82	0.80	0.80	66880

Baseline Algorithm **Linear SVC**



Classification report of train data

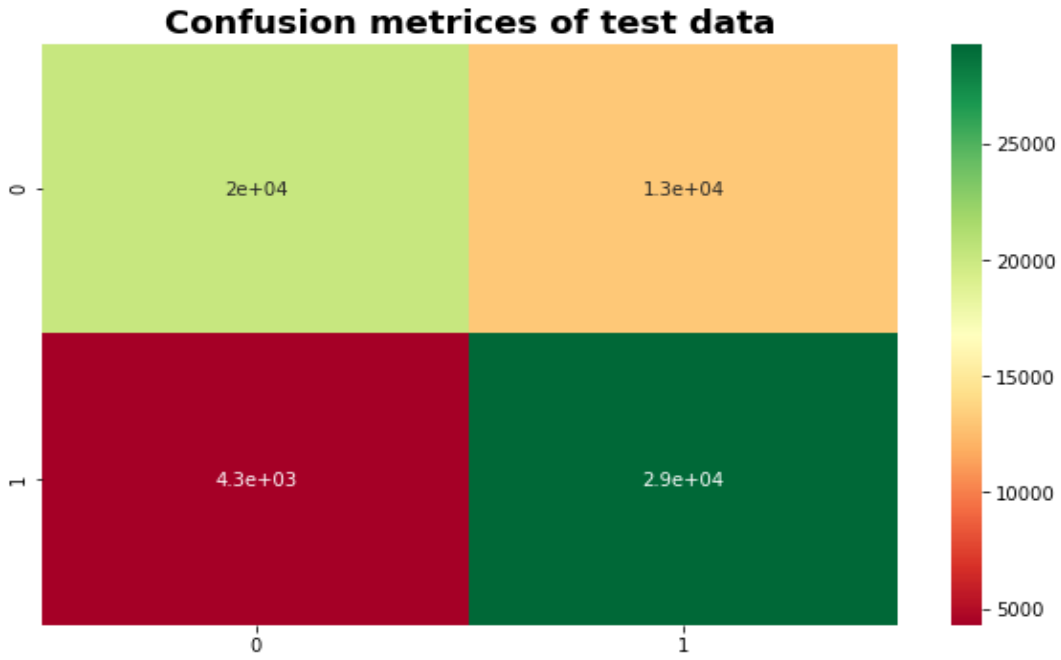
	precision	recall	f1-score	support
0	0.77	0.69	0.73	301112
1	0.72	0.80	0.76	300806
accuracy			0.74	601918
macro avg	0.75	0.74	0.74	601918
weighted avg	0.75	0.74	0.74	601918

Classification report of test data

	precision	recall	f1-score	support
0	0.77	0.70	0.73	33287
1	0.73	0.80	0.76	33593
accuracy			0.75	66880
macro avg	0.75	0.75	0.75	66880
weighted avg	0.75	0.75	0.75	66880

KNN Classifier

In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.



Classification report of train data

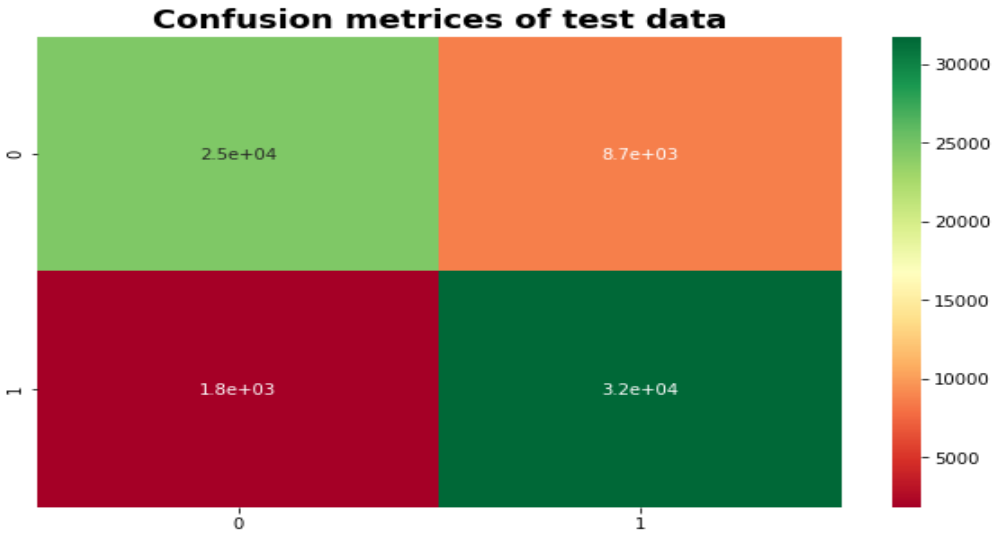
	precision	recall	f1-score	support
0	0.83	0.61	0.70	301112
1	0.69	0.87	0.77	300806
accuracy			0.74	601918
macro avg	0.76	0.74	0.74	601918
weighted avg	0.76	0.74	0.74	601918

Classification report of test data

	precision	recall	f1-score	support
0	0.82	0.61	0.70	33287
1	0.69	0.87	0.77	33593
accuracy			0.74	66880
macro avg	0.76	0.74	0.74	66880
weighted avg	0.76	0.74	0.74	66880

Gradient Boosting Classifier

Gradient boosting is a machine learning technique for classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms random forest. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.



Classification report of train data

	precision	recall	f1-score	support
0	0.93	0.74	0.82	301112
1	0.78	0.95	0.86	300806
accuracy			0.84	601918
macro avg	0.86	0.84	0.84	601918
weighted avg	0.86	0.84	0.84	601918

Classification report of test data

	precision	recall	f1-score	support
0	0.93	0.74	0.82	33287
1	0.79	0.95	0.86	33593
accuracy			0.84	66880
macro avg	0.86	0.84	0.84	66880
weighted avg	0.86	0.84	0.84	66880

XGBOOST Classification

Salient features of Xgbooster which make it different from other gradient boosting algorithms include:
•Clever penalization of trees
•A proportional shrinking of leaf nodes
•Newton Boosting
•Extra randomization parameter
•Implementation on single, distributed systems and out-of-core computation
•Automatic Feature selection

Classification report of train data

	precision	recall	f1-score	support
0	0.88	0.96	0.92	301112
1	0.96	0.87	0.91	300806
accuracy			0.92	601918
macro avg	0.92	0.92	0.92	601918
weighted avg	0.92	0.92	0.92	601918

Classification report of test data

	precision	recall	f1-score	support
0	0.88	0.96	0.92	33287
1	0.95	0.87	0.91	33593
accuracy			0.91	66880
macro avg	0.92	0.91	0.91	66880
weighted avg	0.92	0.91	0.91	66880

Comparing Models and Result

Since our Responses are highly imbalance , an algorithm that always predicts that the insurance buying is majority class would achieve an accuracy of nearly 86%. Nevertheless, that is the opposite of what we want. We do not want a 86% accuracy that is achieved by any dump model so in this case accuracy is not measure of right prediction so we use different technique precision ,recall and F1 score for evaluation and comparing the model

1. Recall: Recall answers the question: **out of the positive response (1), what percentage of these are correctly identified by our model?** In our best model has a recall ratio as 1 .

$$Recall = \frac{TP}{TP + FN}$$

2. Precision: Precision answers the question**: out of all the response predicted to insurance buying , what percentage were actually buying?** In our best model has a recall ratio as 0.96

$$Precision = \frac{TP}{TP + FP}$$

3. F-1 Score: The F1 score combines Recall and Precision into one metric as a weighted average of the two. Unlike Recall and Precision individually, **F1 takes both false positives and false negatives into consideration.** In imbalanced classes such as this, F1 is much more effective than accuracy at determining the performance of the model.

$$F1Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

Your data set is unbalanced since 46710 out of 334389 examples belong to class 0 (that is 86%). Therefore, your predictor almost always predicts any given sample as belonging to class 0 and thereby achieves very high scores like precision and recall for class 0 and very low scores for class 1.

n the case of weighted average the performance metrics are weighted accordingly:

Score weighted-avg = 0.86·score class 0 + 0.14·score class 1

Which turns out to be 1 due the class imbalances However, macro avg is not weighted and therefore

score macro-avg = 0.5·score class 0 + 0.5·score class 1

No	Model	Accuracy	Precision	Recall	F1 Score
1	Logistic regression(Smote oversampling)	0.80	0.71	0.98	0.82
2	Decision Tree (Smote Oversampling)	0.83	0.77	0.94	0.85
3	Linear support vector Machine (Smote oversampling)	0.58	0.75	0.24	0.37
4	XGBOOST (Smote oversampling)	0.91	0.95	0.87	0.91
5	Gradient Boosting (Smote Oversampling)	0.84	0.78	0.95	0.86
6	Naïve- Baise (smote oversampling)	0.80	0.75	0.91	0.82
7	KNN classification (smote)	0.74	0.69	0.87	0.77
8	Random forest(smote oversampling)	0.83	0.76	0.97	0.85

Conclusion

From the above analysis it is quite evident that Xgbooster Classifier works better than other algorithms. With test accuracy of about 91% precision 95% Recall 0.87 F1Score 0.91