Sri Lanka Institute of Information Technology

Fundamentals of Data Mining – IT3051

Group Project – Final Report

Group Number - 24

IT Number	Name
IT20035358	Jayarathna K.N.N.D.S
IT20017606	Wijerathna K.K.R.T
IT20097660	Warnasooriya S.D
IT20100698	Britto T.A
IT20133368	Wijayasena W.D.N.D.T

Table of Contents

Table of Figures

Table of Tables

1 Terms of Reference

The report is submitted in fulfilment of the requirements for the Fundamentals of Data Mining [IT3051] module, Faculty of Computing, Sri Lankan Institute of Information Technology (SLIIT).

2 Acknowledgment

First, we would like to thank our module leader Mr. Prasanna Sumathipala and lab instructors for the immense guidance and support provided to make this group project a success. Thank you, your input was extremely helpful for the successful completion of this project.

3 Abstract

The report is about a data mining web application based on Health insurance cross-sell prediction. It contains six sections which section one explains the introduction, problem statement, objectives, and development approach of the project. Selected datasets and preprocessing methods are thoroughly detailed in section two. section three explains the solution with model creation, application development, and testing. The complete user guide provides in section four, and section five discusses the project and team management aspects. Finally, an evaluation of the project provides in section six.

4 Introduction

4.1 Project Background

The real-world problem we used for this project is about Health Insurance Cross Sell Prediction based on information about customer's demographics (gender, age, region code type), Vehicles (Vehicle Age, Damage) and Policy (Premium, sourcing channel). This Insurance company provides Health Insurance to its customers, and they need to build a model to predict whether the policyholders (customers) from the 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.

4.2 Problem Statement

Our client is an insurance company that has offered Health Insurance to its customers. They need assistance in developing a model to forecast whether the policyholders from the previous year will be interested in the company's Vehicle insurance.

Vehicle insurance is similar to medical insurance in that the customer must pay a premium of a specified amount to the insurance provider business every year in order for the insurance provider company to compensate the client for an unfortunate accident.

Building a model to forecast a customer's interest in Vehicle Insurance is very beneficial to the company because it allows it to plan its communication approach to achieve out to those clients in the most effective way possible and maximize its business model and revenue.

4.3 Objectives

We consider following as objectives.

- Using vehicle insurance data, learn about cross-selling.
- Learn how to create a cross-sell prediction model.
- To forecast whether an insurance policyholder would be interested in purchasing vehicle insurance as well. Building a model to predict when a client is interested in Vehicle Insurance is incredibly advantageous to the company because it can then plan its communication strategy to reach out to those customers and improve its business model and revenue accordingly.
- The objective of this project is to use machine learning methods such as Logistic Regression and Random Forest to create a predictive model from the given data set using statistically relevant factors.
- Model accuracy will be assessed using different techniques such as ROC (Receiver operating characteristic), AUC (Area under the ROC curve) and Confusion Matrix.

5 Dataset Analysis & Preparation

5.1 Dataset – Health Insurance Cross Sell Prediction

Variable	Definition	Type
id	Unique ID for the customer	int
Gender	Gender of the customer	object
Age	Age of the customer	int
Driving_License	0 : Customer does not have DL, 1 : Customer already has DL	int
Region_Code	Unique code for the region of the customer	Float
Previously_Insured	1 : Customer already has Vehicle Insurance, 0 : Customer doesn't have Vehicle Insurance	int
Vehicle_Age	Age of the Vehicle	object
Vehicle_Damage	1 : Customer got his/her vehicle damaged in the past. 0 : Customer didn't get his/her vehicle damaged in the past.	object
Annual_ Premium	The amount customer needs to pay as premium in the year	Float
PolicySalesChannel	Anonymized Code for the channel of outreaching to the customer ie. Different Agents, Over Mail, Over Phone, In Person, etc.	Float
Vintage	Number of Days, Customer has been associated with the company	int

Table 1: Description about the Attributes

Original Source : [Health Insurance Cross Sell Prediction]

5.2 Preprocessing

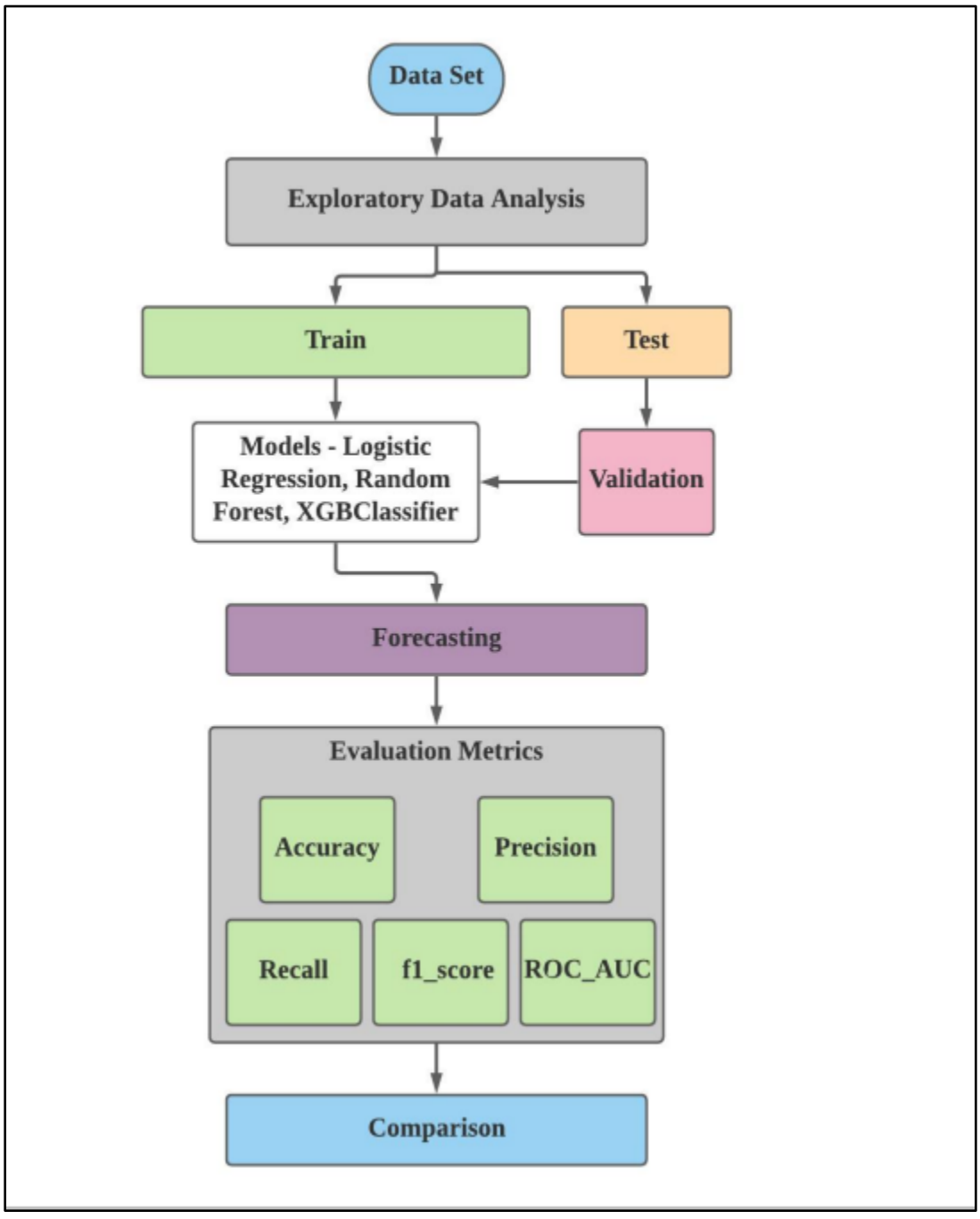

Figure 1: Preprocessing and Model Implementation

After a brief understanding of the dataset, necessary techniques were applied to understand the nature of the dataset further, also to preprocess to ensure high quality dataset is generated.

Step 01 : Calculate the size of the dataset with number of features/attributes and generate column names.

```
[7]  # How many rows and columns are there in this data set
     HIC.shape

     (381109, 12)

[8]  # columns name shows
     HIC.columns

     Index(['id', 'Gender', 'Age', 'Driving_License', 'Region_Code',
            'Previously_Insured', 'Vehicle_Age', 'Vehicle_Damage', 'Annual_Premium',
            'Policy_Sales_Channel', 'Vintage', 'Response'],
           dtype='object')
```

Figure 2: Code for Calculate the Size and Get the Columns

Step 02 : Check null and missing values in the dataset.

```
[15] HIC.isnull().sum().sort_values(ascending=False)

     id                    0
     Gender                0
     Age                   0
     Driving_License       0
     Region_Code           0
     Previously_Insured    0
     Vehicle_Age           0
     Vehicle_Damage        0
     Annual_Premium        0
     Policy_Sales_Channel  0
     Vintage               0
     Response              0
     dtype: int64
```

Figure 3: Code for Checking NULL Values

Step 03 : Based on the observation of the dataset, plot a diagram to identify the correlation between the features/fields.

Correlation Matrix

```
[23] correlation = HIC.corr()

     f, ax = plt.subplots(figsize=(10,10))

     sns.heatmap(correlation, ax=ax, annot=True,linewidths=3,cmap='YlGn')

     plt.title("Pearson correlation of Features", y=1.05, size=15)
```

Figure 4: Code for Correlation Matrix

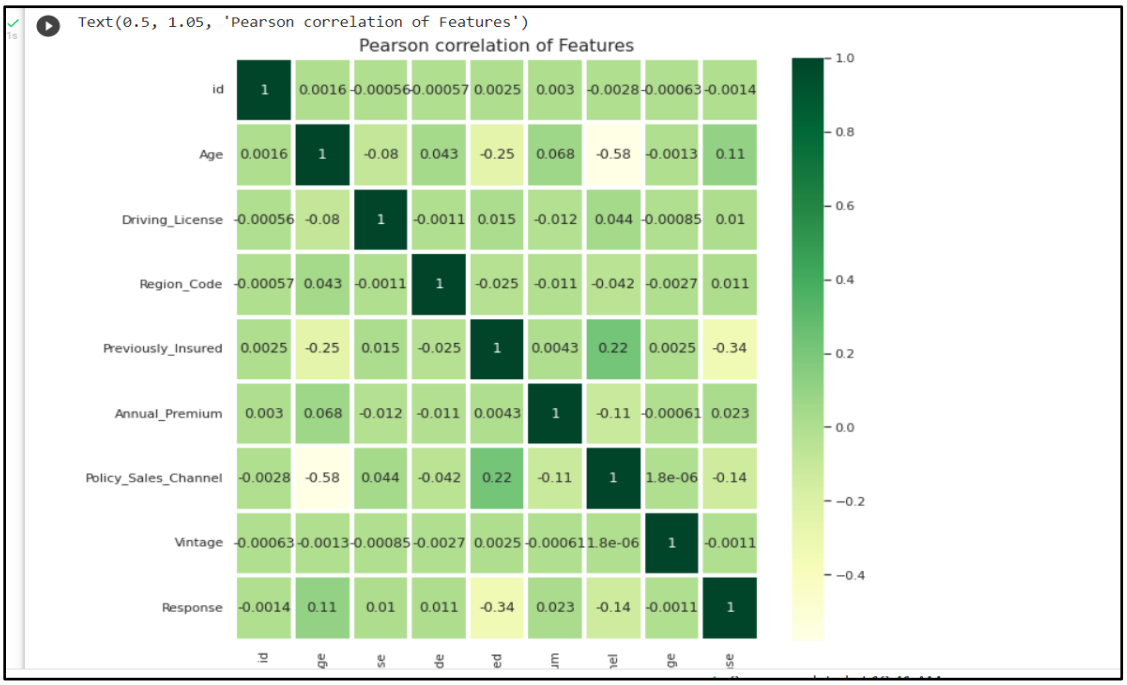

Figure 5: Pearson Correlation of Features

```
HIC= HIC.drop(columns=['id','Policy_Sales_Channel','Vintage'])
```

Figure 6: Code for Dropping Columns

```
HIC.columns
```

```
Index(['Gender', 'Age', 'Driving_License', 'Region_Code', 'Previously_Insured',
       'Vehicle_Age', 'Vehicle_Damage', 'Annual_Premium', 'Response'],
      dtype='object')
```

Figure 7: Code for Existing Columns After Dropping

Step 3: Label encoding

Label encoding

```
[27] #converting categorical value to numerical values
     labelEncoder= LabelEncoder()
     HIC['Gender'] = labelEncoder.fit_transform(HIC['Gender'])
     HIC['Vehicle_Age'] = labelEncoder.fit_transform(HIC['Vehicle_Age'])
     HIC['Vehicle_Damage'] = labelEncoder.fit_transform(HIC['Vehicle_Damage'])
```

Figure 8: Code for Label Encoding

Step 4: Separating dependent and independent variables.

Seprating dependent & independent variables

```
[28] # independent variable
     x=HIC.drop(['Response'],axis=1)

     #dependent variable
     y=HIC['Response']
```

Figure 9: Code for Separate Dependent & Independent Variable

Step 5 : Feature Selection - Drop columns that has less importance or impact towards the classification model, mainly the user generated fields are being omitted along with field that consist of facts.

Feature Selection

```
[29] from sklearn.ensemble import ExtraTreesClassifier
     Model = ExtraTreesClassifier()
     Model.fit(x,y)
     #use inbuilt class feature_importances of tree based class
     print(Model.feature_importances_)

     #plot of feature importances for best visualization
     feat_importances = pd.Series(Model.feature_importances_, index=x.columns)
     feat_importances.nlargest(11).plot(kind='barh')
     plt.show()
```

```
[1.81672906e-03 1.18515674e-01 4.26769830e-04 7.30012617e-02
 6.38134684e-02 2.02556204e-02 8.58453243e-02 6.36325152e-01]
```

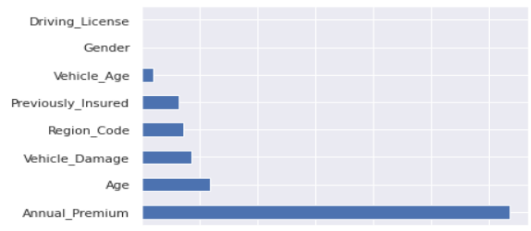

Figure 10: Code for Feature Selection

Step 6: Handling imbalanced data - When observation in one class is higher than the observation in other classes then there exists a class imbalance. We can clearly see that there is a huge difference between the data set. Solving this issue, we use resampling technique.

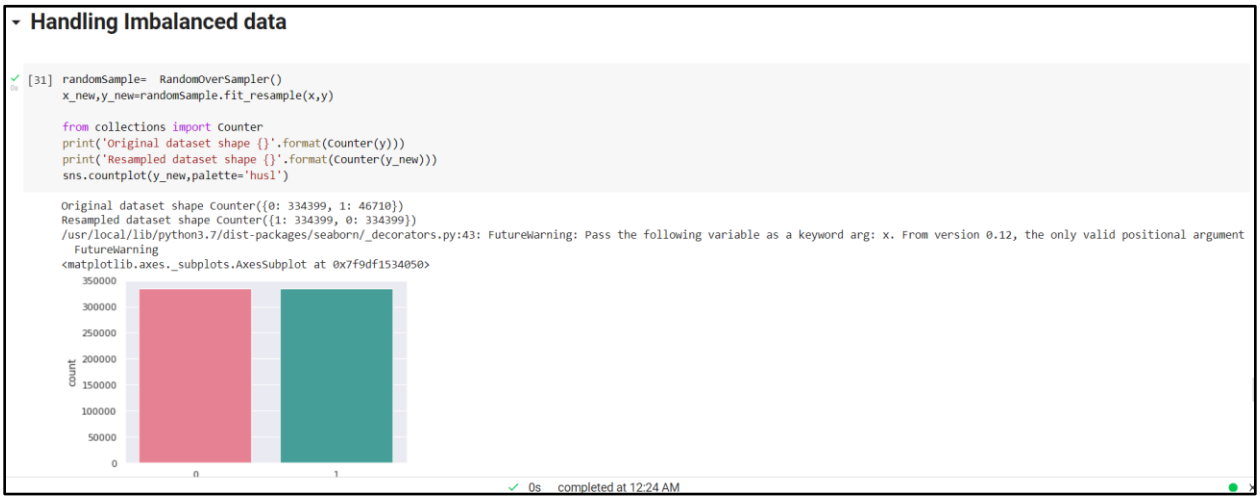

Figure 11: Code for Handling Imbalanced Data

Step 7: Splitting data set into training and testing

```
#dividing the dataset into training and testing
xtrain,xtest,ytrain,ytest=train_test_split(x_new,y_new,test_size=.30,random_state=0)
print(xtrain.shape,xtest.shape,ytrain.shape,ytest.shape)

#feature scaling
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
xtrain=scaler.fit_transform(xtrain)
xtest=scaler.transform(xtest)
```

```
(468158, 6) (200640, 6) (468158,) (200640,)
```

Figure 12: Code for Splitting dataset into training and testing

5.3 Data Visualization

Target Variable

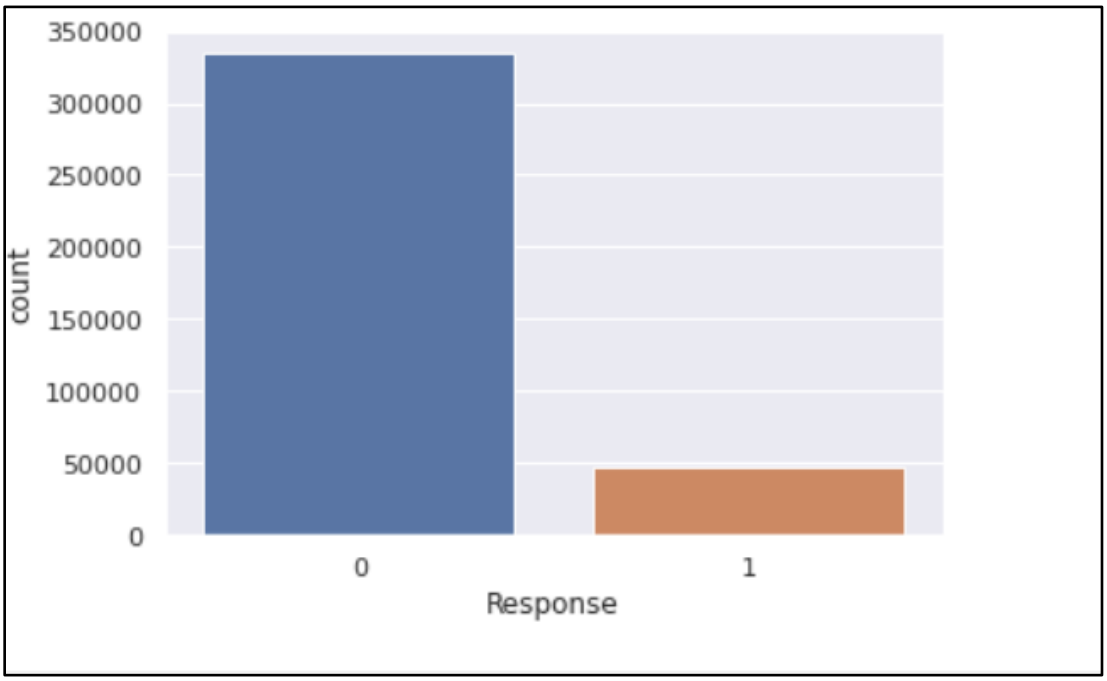

Figure 13: Target Variable Graph

- 0: Indicates a disinterested customer
- 1: Express customer interest
- As you can see from the graph, there are relatively few consumers with stats below 50000 who are interested and many more with stats above 300000 who are not.

Gender

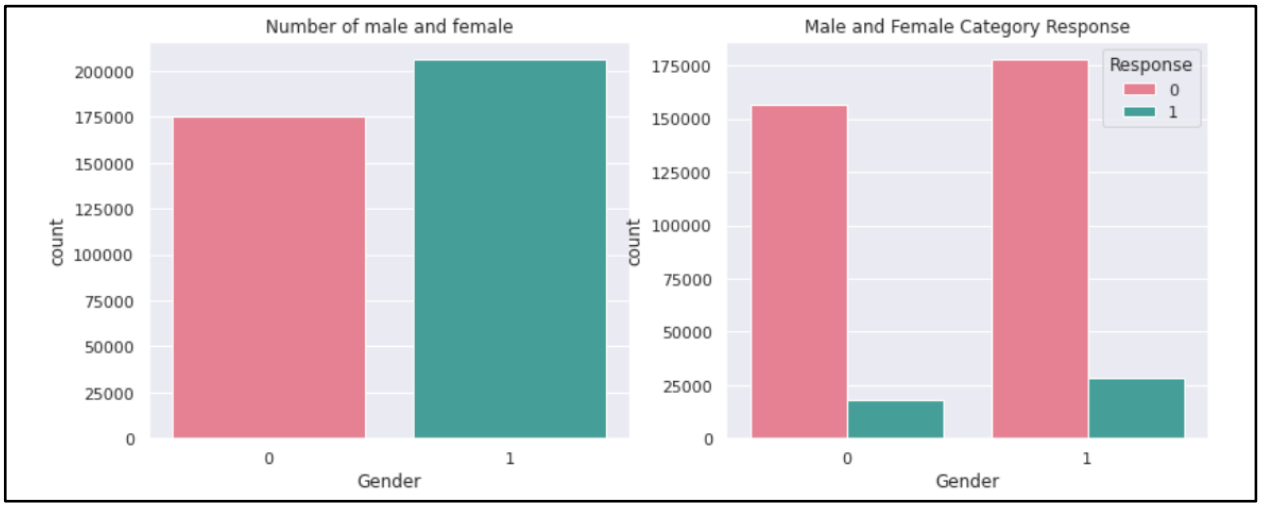

Figure 14: Graph for Gender

- As shown in the graph above, the number of males exceeds 200000, while the number of females is close to 175000.
- The number of males interested is larger than 25000, whereas the number of females interested is less than 25000.
- The male category is significantly larger than the female group, and the chances of purchasing insurance are likewise slightly higher.

Age Vs Response

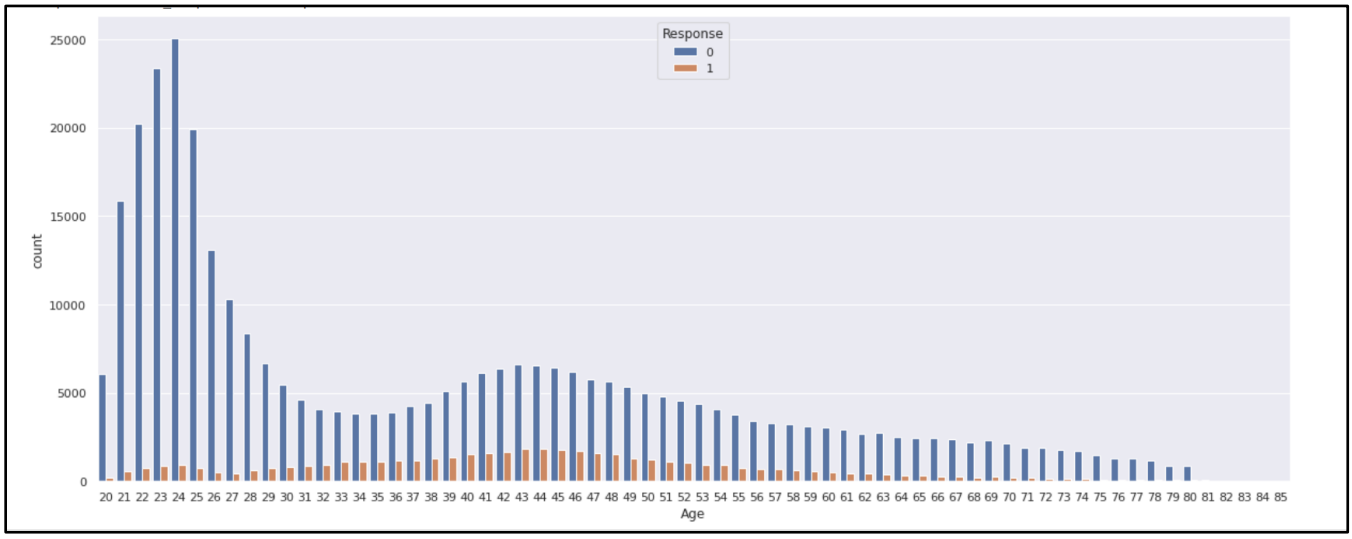

Figure 15: Age vs Response Bar Graph

- Young people under the age of 30 are not interested in auto insurance.
- Reasons could include a lack of experience, a lack of maturity, and a lack of pricey vehicles.

- People between the ages of 30 and 60 are more likely to be interested. The boxplot shows that there are no outliers in the data.

Driving License

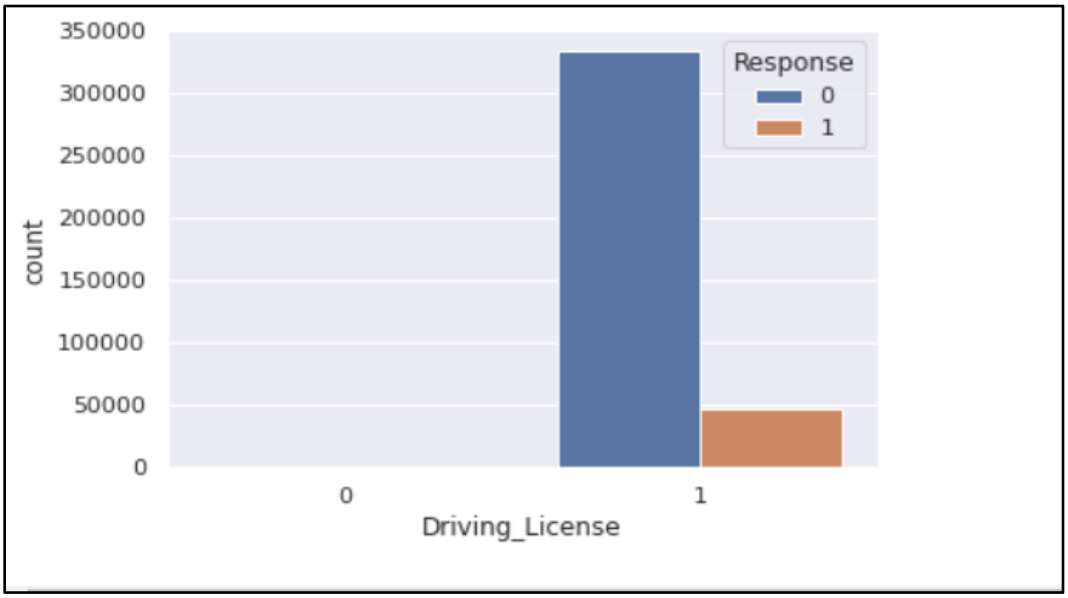

Figure 16: Bar Graph for Driving License

- 0: indicate Customer is not interested
- 1: indicate Customer is interested Customers who are interested in Vehicle Insurance almost all have driving license

Previously vs response

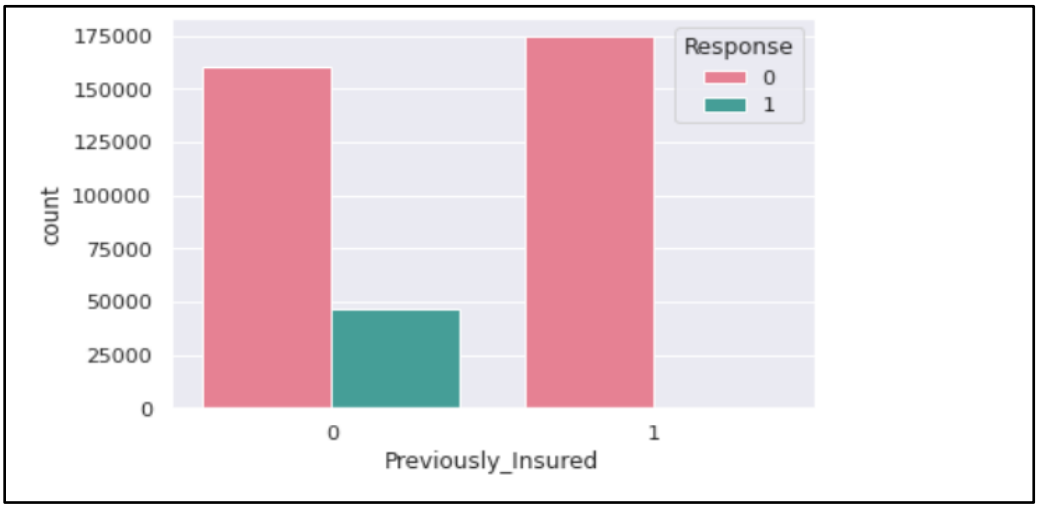

Figure 17: Bar Graph for Previously Insured Attribute

- Customer who are not previously insured are likely to be interested.1 indicate no one interested and the value of this is null.

Vehicle Age

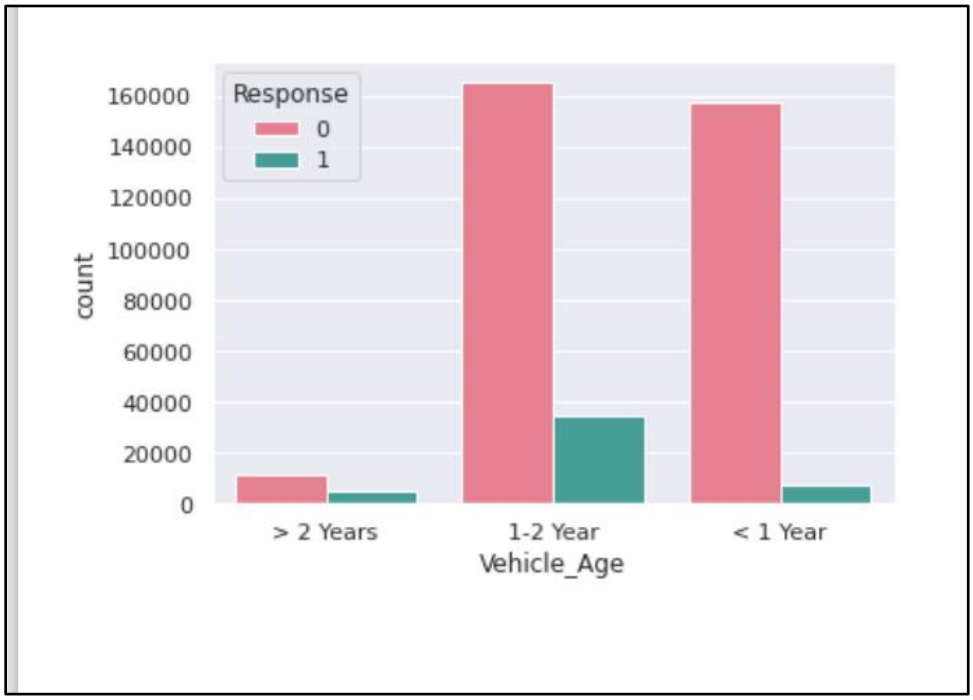

Figure 18: Bar Graph for Vehicle Age Attribute

- Customers with vehicle age 1- 2 years are more likely to interested as compared to the other two. Customers with Vehicle_Age < 1 years have very less chance of buying Insurance.

Vehicle Damage

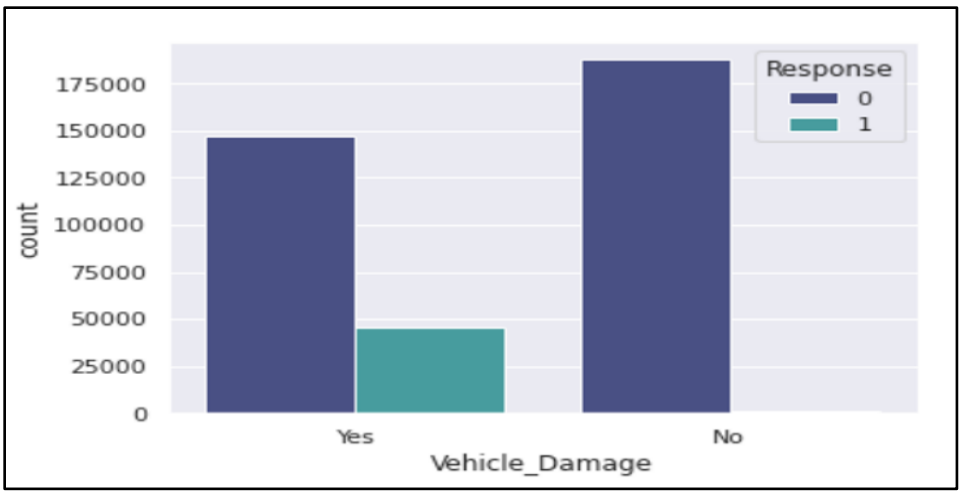

Figure 19: Bar Graph for Vehicle Damage

- From the above plot, we can infer that if the vehicle has been damaged previously then the customer will be more interested in buying the insurance as they know the cost.
- It is also important to look at the target column, as it will tell us whether the problem is a balanced problem or an imbalanced problem. This will define our approach further.
- The given problem is an imbalance problem as the Response variable with the value 1 is significantly lower than the value zero.

Annual

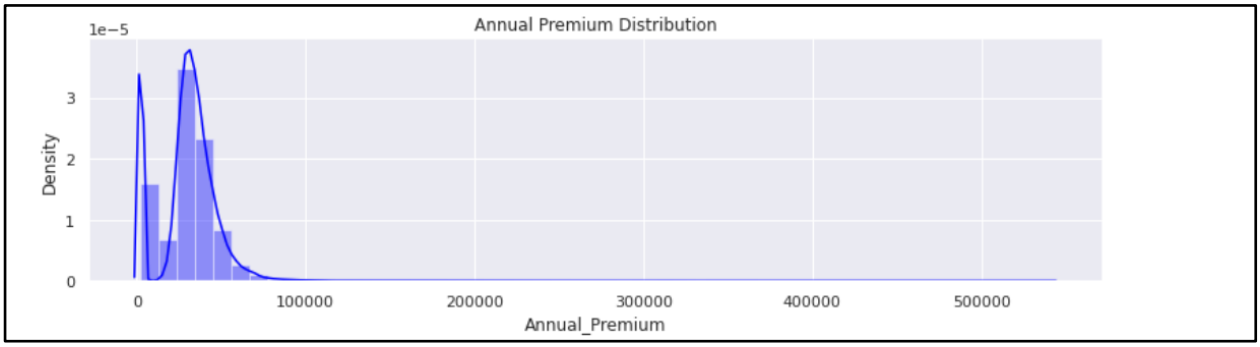

Figure 20: Graph for Annual Premium

- From the distribution plot we can identify that the annual premium variable is right skewed

Correlation Plot

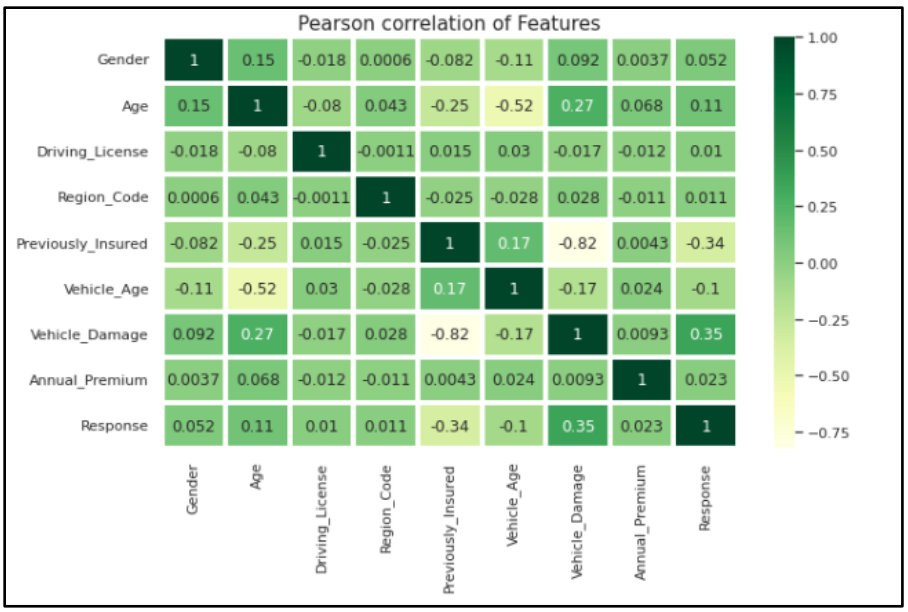

Figure 21: Pearson Correlation of Features

- Heat maps are a great tool for visualizing complex statistical data. By this graph we will find out correlation all about the data. Which column is useful, and which is not useful for us. We can easily find out by this graph.
- Target variable is not much affected by id, Vintage variable, Policy sales channel. we can drop least correlated variable. As we can see the heatmap graph Vehicle Damage column is more correlated with target variable.

Feature Selection

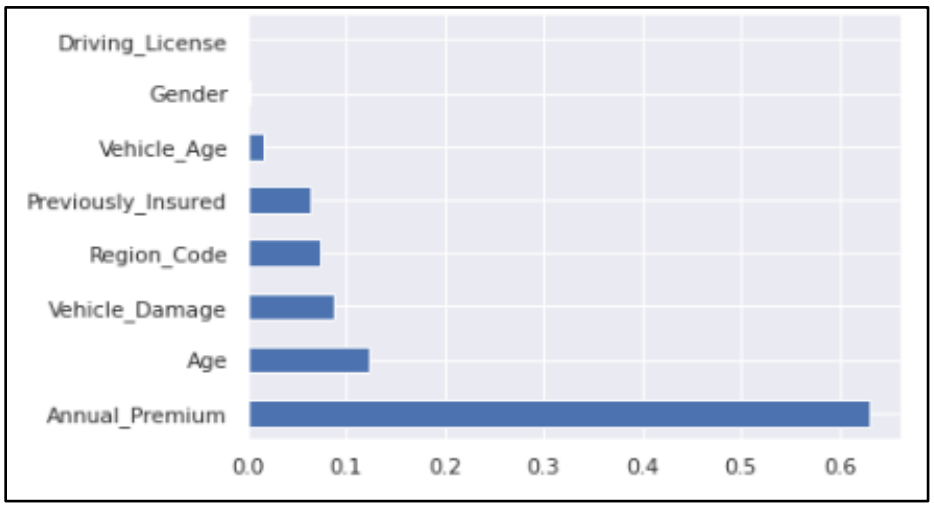

Figure 22: Feature Selection Graph

- We can remove less important features from the data set like Driving license, Gender.

Handling Imbalanced Data

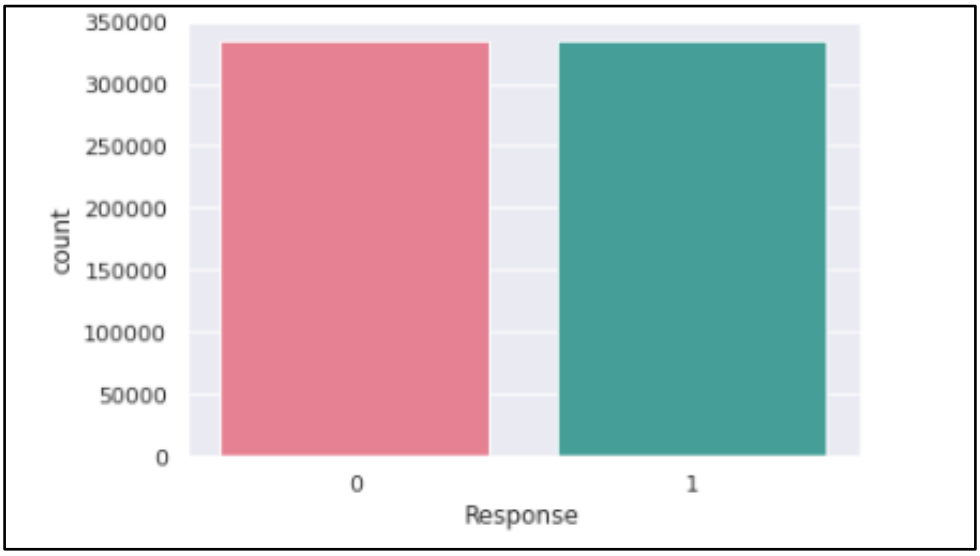

Figure 23: Graph for Handling Imbalanced Data

- As from the distribution of target variables in the exploratory data analysis section, we know it is an imbalance problem. The imbalance datasets could have their own challenge.
- For example, a disease prediction model may have an accuracy of 99% but it is of no use if it cannot classify a patient successfully. So, to handle such a problem, we can resample the data. In the following code, we will be using under sampling. Under sampling is the method where we will be reducing the occurrence of the majority class up to a given point

6 Proposed Solution

6.1 Tools and Technologies.

The tools that used for the project were describe from the below table.

Language	Python: Python offers a robust technology stack and a large number of machine learning libraries.
IDE	Jupyter/Colab: can examine both the code and the results, has all of the necessary Python libraries installed, and can execute codes more quickly than local machines.
Framework	Flask: Flask is a web framework, it's a Python module that let develop web applications easily. It's having a small and easy-to-extend core

Server/Hosting	Heroku: This is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud

Table 2: Tools and Technologies

6.2 Model Evaluation

6.2.1 Classification Model

A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data. The team proposed to test the dataset using two models' logistic regression, and random forest under binary classification.

Logistic Regression

The process of modeling the probability of a discrete outcome given an input variable is known as logistic regression. The most common logistic regression models have a binary outcome, which can take two values like true/false, yes/no, and so on.

```
Model=LogisticRegression()
Model=Model.fit(xtrain,ytrain)
pred=Model.predict(xtest)
lr_probability =Model.predict_proba(xtest)[:,1]


accuracyLr=accuracy_score(ytest,pred)
recallLr=recall_score(ytest,pred)
precisionLr=precision_score(ytest,pred)
f1scoreLr=f1_score(ytest,pred)
AUC_LR=roc_auc_score(pred,ytest)
#print accuracy and Auc values of model
print("Accuracy : ", accuracy_score(ytest,pred))
print("Precision:",precision_score(ytest,pred))
print("Recall:",recall_score(ytest,pred))
print("F1-Score:",f1_score(ytest,pred))
print("ROC_AUC Score:",AUC_LR)
```

Figure 24: Code to Get the Accuracy of Logistic Regression

```
Accuracy :  0.7836722488038278
Precision: 0.8584849142609073
Recall: 0.9856846638487917
F1-Score: 0.917698051390571
ROC_AUC Score: 0.9208413572650116
```

Figure 25: Accuracy Level of Logistic Regression

```
print(classification_report(pred,ytest))
```

Figure 26: Code for Getting Classification Report

	precision	recall	f1-score	support
0	0.59	0.96	0.73	61464
1	0.98	0.70	0.82	139176
accuracy			0.78	200640
macro avg	0.78	0.83	0.78	200640
weighted avg	0.86	0.78	0.79	200640

Figure 27: Classification Report for Logistic Regression

```
from sklearn.metrics import roc_curve
fpr, tpr, _ = roc_curve(ytest, lr_probability)

plt.title('Logistic Regression ROC curve')
plt.xlabel('FPR (Precision)')
plt.ylabel('TPR (Recall)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='red')
plt.show()
```

Figure 28: Code for Linear Regression ROC Curve

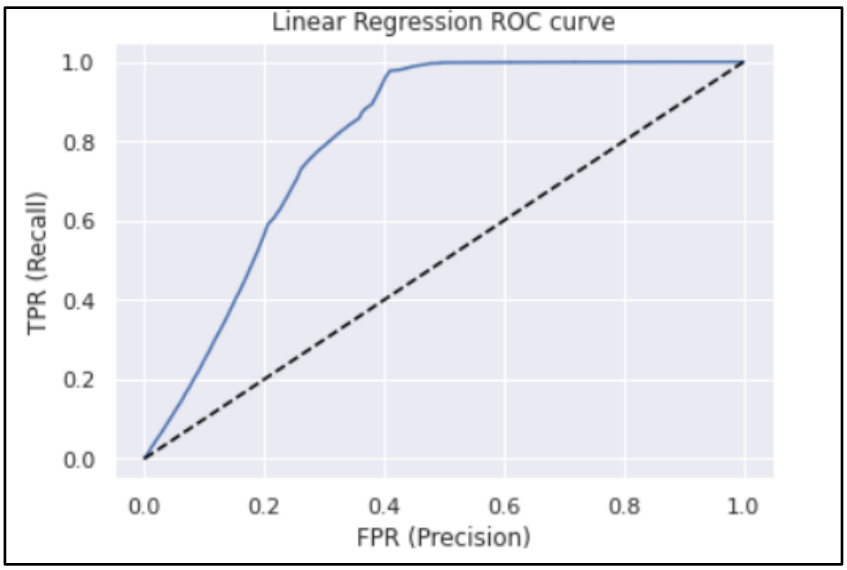

Figure 29: Linear Regression ROC Curve

```
cm=confusion_matrix(ytest,pred)
print(cm)
sns.heatmap(cm,annot=True,cmap='BuPu')
```

Figure 30: Code for Confusion Matrix for Logistic Regression

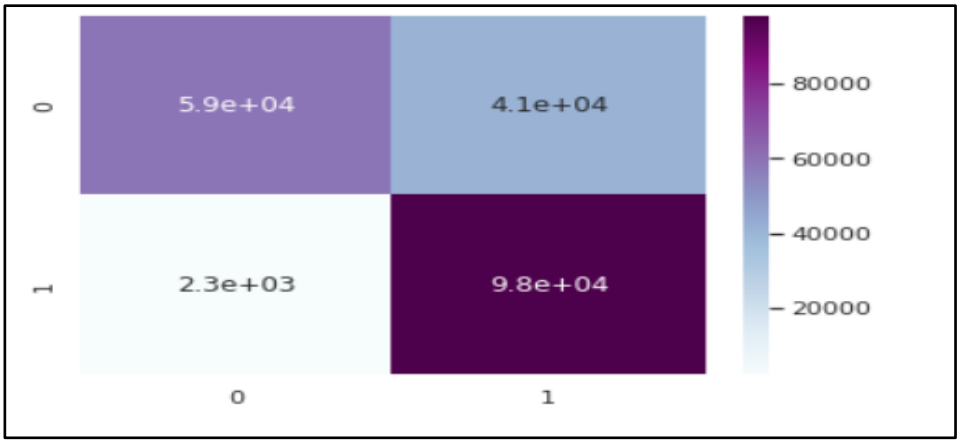

Figure 31: Conclusion Matrix for Logistic Regression

Random Forest

The random forest algorithm is a classification algorithm made up of many decision trees. When building each individual tree, it uses bagging and feature randomness to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

```
from sklearn.ensemble import RandomForestClassifier

randomForest = RandomForestClassifier()
randomForest=randomForest.fit(xtrain, ytrain)
y_pred = randomForest.predict(xtest)
RF_probability = randomForest.predict_proba(xtest)[:,1]



AUC_RF=roc_auc_score(y_pred,ytest)
acc_rf=accuracy_score(ytest,y_pred)
recall_rf=recall_score(ytest,y_pred)
precision_rf=precision_score(ytest,y_pred)
f1score_rf=f1_score(ytest,y_pred)

print("Accuracy : ", accuracy_score(ytest,y_pred))
print("Precision:",precision_score(ytest,y_pred))
print("Recall:",recall_score(ytest,y_pred))
print("F1-Score:",f1_score(ytest,y_pred))
print("ROC_AUC Score:",AUC_RF)
```

Figure 32: Code to Get the Accuracy of Random Forest

```
Accuracy :  0.911782296650717
Precision: 0.8586461987288577
Recall: 0.9858441662014514
F1-Score: 0.9178593306231554
ROC_AUC Score: 0.9210158966686831
```

Figure 33: Accuracy of the Random Forest Model

```
print(classification_report(y_pred,ytest))
```

Figure 34: Code for Get the Classification Report for Random Forest

	precision	recall	f1-score	support
0	0.84	0.98	0.90	85468
1	0.99	0.86	0.92	115172
accuracy			0.91	200640
macro avg	0.91	0.92	0.91	200640
weighted avg	0.92	0.91	0.91	200640

Figure 35: Classification Report for the Random Forest

```
from sklearn.metrics import roc_curve
fpr, tpr, _ = roc_curve(ytest, RF_probability)

plt.title('Linear Regression ROC curve')
plt.xlabel('FPR (Precision)')
plt.ylabel('TPR (Recall)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='red')
plt.show()
```

Figure 36: Code for the Linear Regression ROC Curve for Random Forest

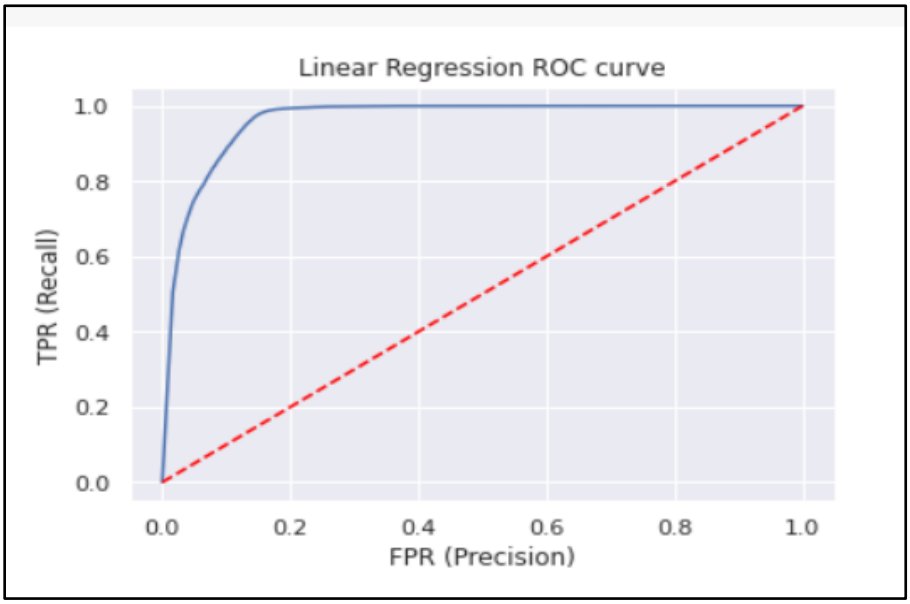

Figure 37: Linear Regression ROC Curve for Random Forest

```
cm=confusion_matrix(y_pred,ytest)
print(cm)
sns.heatmap(cm,annot=True,cmap='GnBu')
```

Figure 38: Code for Get the Confusion Matrix for Random Forest

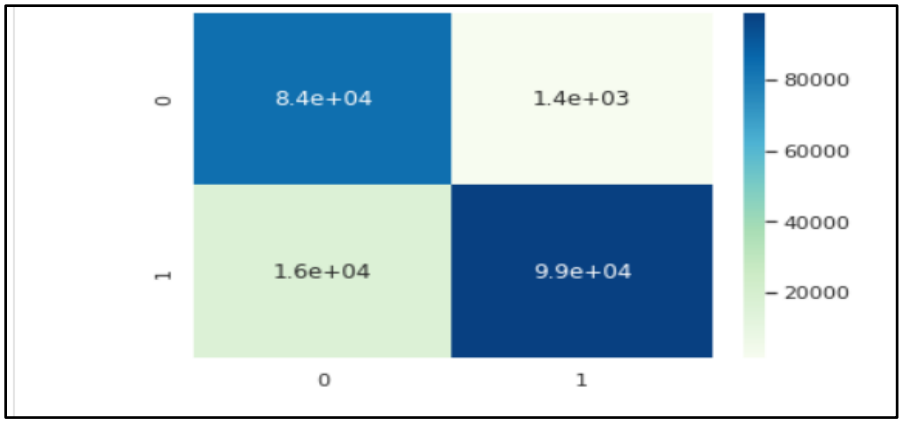

Figure 39: Confusion Matrix for Random Forest

XGB Classifier

The Extreme Gradient Boosting (XGBoost) machine learning library is scalable and distributed gradient-boosted decision tree (GBDT). It is the leading machine learning library for regression, classification, and ranking problems, and it offers parallel tree boosting.

```
from xgboost import XGBClassifier
xgb=XGBClassifier()
XGB_fit=xgb.fit(xtrain, ytrain)
y_predict = XGB_fit.predict(xtest)
XGB_probability = XGB_fit.predict_proba(xtest)[:,1]

acc_xgb=accuracy_score(ytest,y_predict)
recall_xgb=recall_score(ytest,y_predict)
precision_xgb=precision_score(ytest,y_predict)
f1score_xgb=f1_score(ytest,y_predict)

AUC_xgb=roc_auc_score(y_predict,ytest)


print("Accuracy:",acc_xgb)
print("ROC_AUC Score:",AUC_xgb)
```

Figure 40: Code for Get the Accuracy of XGB Classifier

```
Accuracy: 0.7960177432216906
ROC_AUC Score: 0.8221635281654662
```

Figure 41: Accuracy of the XGB Classifier

```
print(classification_report(y_predict,ytest))
```

	precision	recall	f1-score	support
0	0.65	0.91	0.76	71747
1	0.94	0.73	0.82	128893
accuracy			0.80	200640
macro avg	0.80	0.82	0.79	200640
weighted avg	0.84	0.80	0.80	200640

Figure 42: Classification Report for XGB Classifier

```
from sklearn.metrics import roc_curve
fpr, tpr, _ = roc_curve(ytest, XGB_probability)

plt.title('XGBoost ROC curve')
plt.xlabel('FPR (Precision)')
plt.ylabel('TPR (Recall)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='red')
plt.show()
```

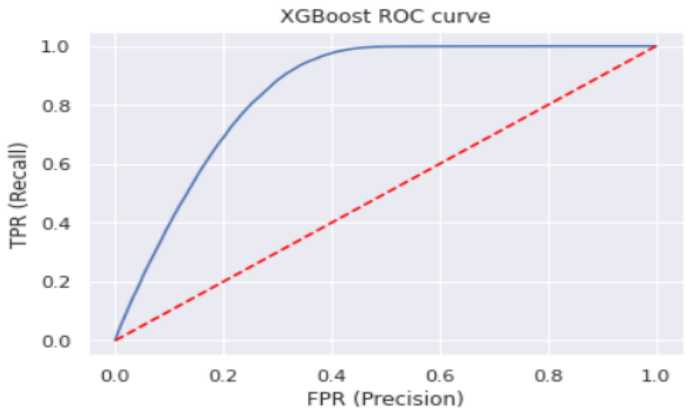

Figure 43: Linear Regression ROC Curve for Random Forest

```
#it helps to identify how many values are classified correctly
cm=confusion_matrix(ytest,y_predict)
print(cm)
sns.heatmap(cm,annot=True,cmap='RdPu')
```

```
[[65574 34754]
 [ 6173 94139]]
<matplotlib.axes._subplots.AxesSubplot at 0x7f5145e1e550>
```

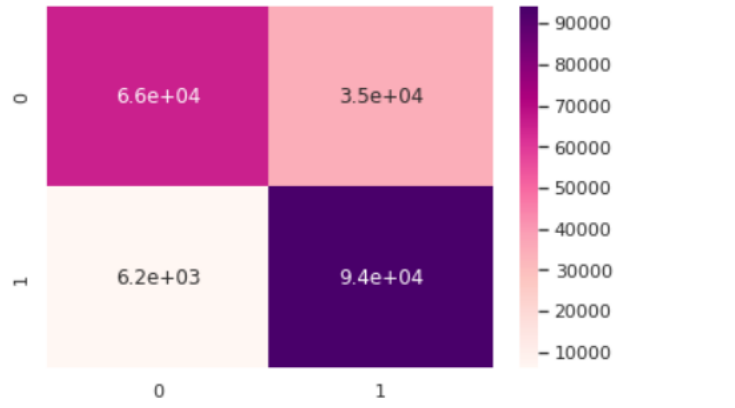

Figure 44: Confusion Matrix for XGB Classifier

Comparing Logistic Regression, random Forest and XGB Classifier Models

```
ind=['Logistic regression','Randomforest','XGBClassifier']
data={"Accuracy":[accuracyLr,acc_rf,acc_xgb],"Recall":[recallLr,recall_rf,recall_xgb],"Precision":[precisionLr,precision_rf,precision_xgb],
    'f1_score':[f1scoreLr,f1score_rf,f1score_xgb],"ROC_AUC":[AUC_LR,AUC_RF,AUC_xgb]}
result=pd.DataFrame(data=data,index=ind)
result
```

	Accuracy	Recall	Precision	f1_score	ROC_AUC
Logistic regression	0.783887	0.977799	0.704537	0.818975	0.834140
Randomforest	0.911782	0.985844	0.858646	0.917859	0.921016
XGBClassifier	0.796018	0.938462	0.730365	0.821439	0.822164

Figure 45: Comparison of Models

We developed three models for this problem: logistic regression, random forest, and XGB classifier. The ML model for the problem statement was created in Python using the dataset, and the ML model created with the Random Forest and XGBClassifier models surpassed the Logistics Regression model. When we compare the ROC curves, we can see that the Random Forest model performs better. Curves that are closer to the top-left corner indicate better performance.

Conclusion

Customers of age between 30 to 60 are more likely to buy insurance.

Customers with Driving License have higher chance of buying Insurance.

Customers with Vehicle Damage are likely to buy insurance.

The variable such as Age, previously insured, Annual premium are more affecting the target variable.

Comparing ROC curve, we can see that Random Forest model perform better. Because curves closer to the top-left corner, it indicates a better performance.

6.3 Application Development

The development of the application was different compared to previous projects, as this involves an integration of machine learning model to the real-world scenario. The development focuses on aspects such as designing the user interfaces, planning on integration of models to the application. The team has used flask, which is advanced and user-friendly framework to build machine learning web applications, therefore integration of models was not a tedious task.

The following screenshots illustrates the user interfaces of the final application deployed to the server.

Home page display a small description about the application and user can navigate to the prediction page by clicking 'Prediction' Button.

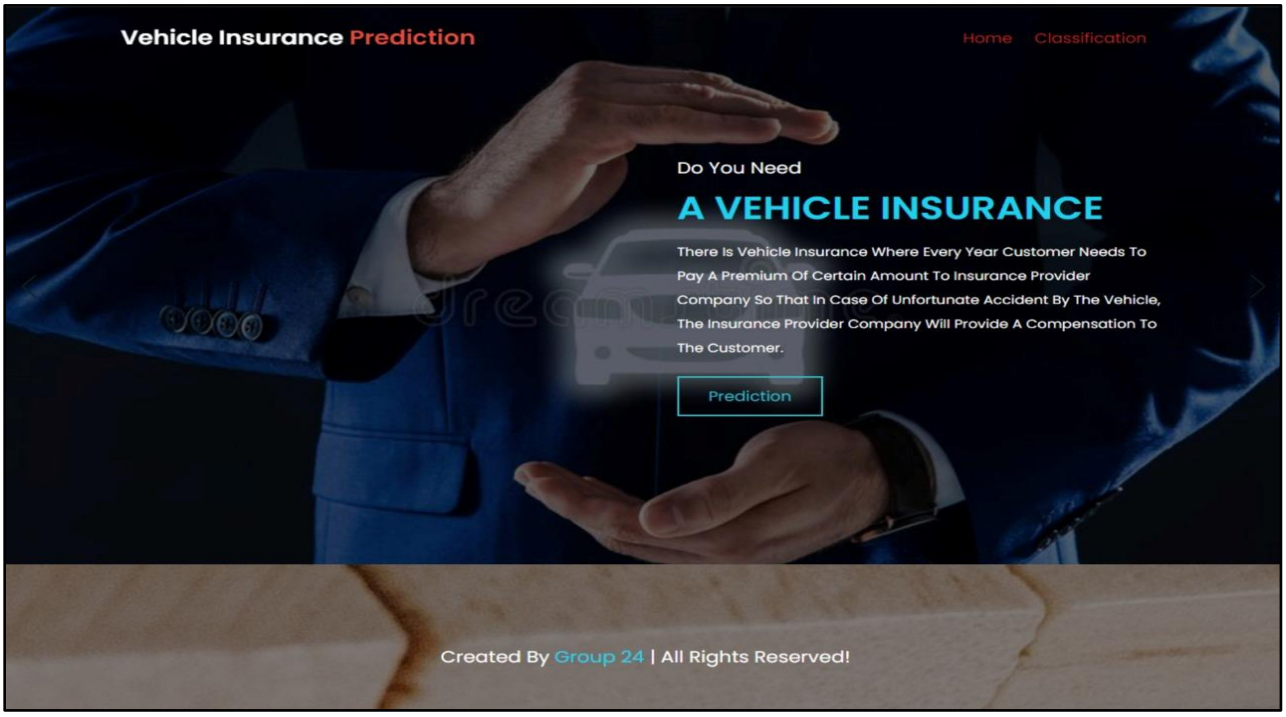

Figure 46: Home Page of the Final Application

In the prediction page, there is a form to input the details that wants to predict, and after click ‘Predict’ button in there, it will display prediction status.

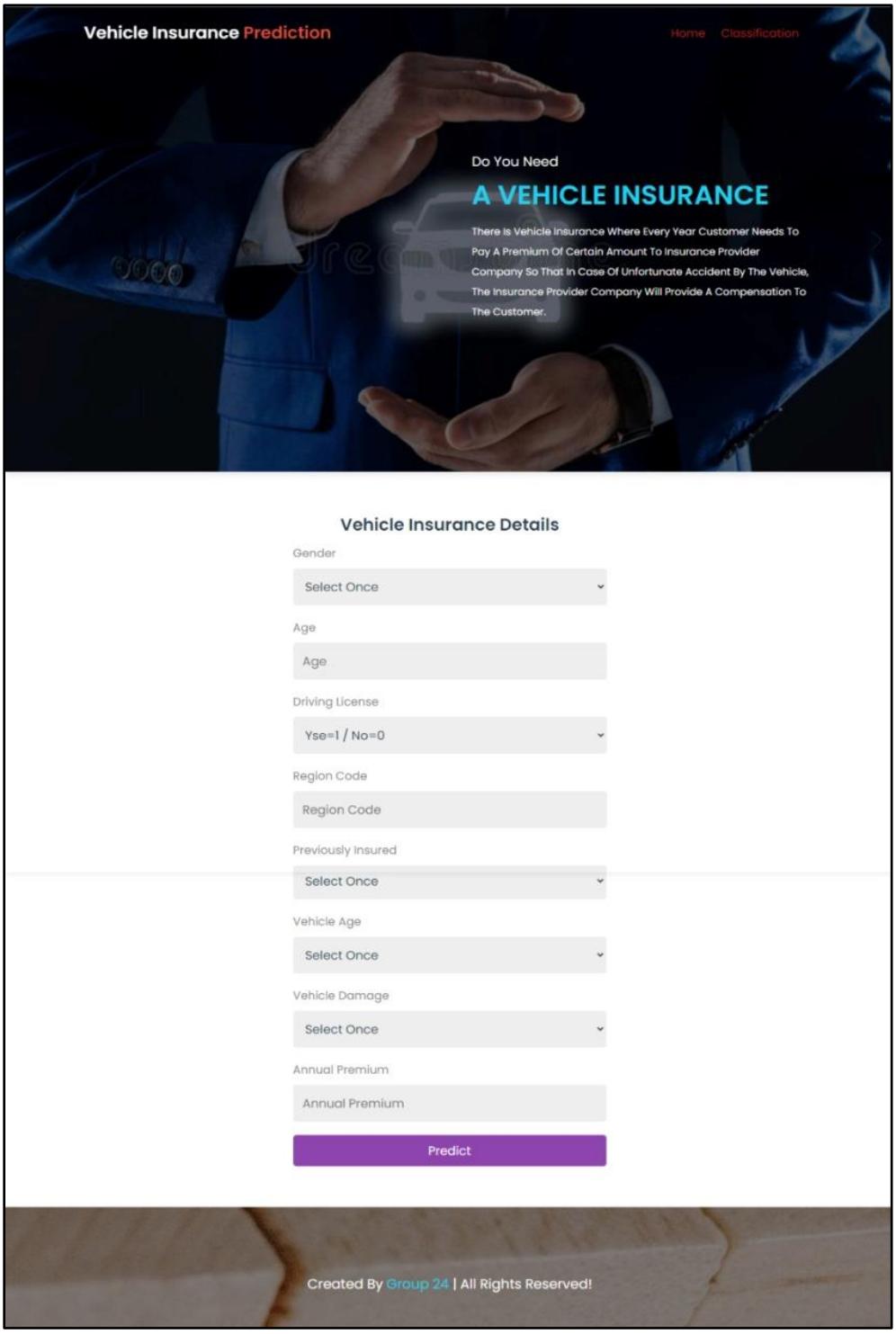

Figure 47: Prediction Page of the Final Application

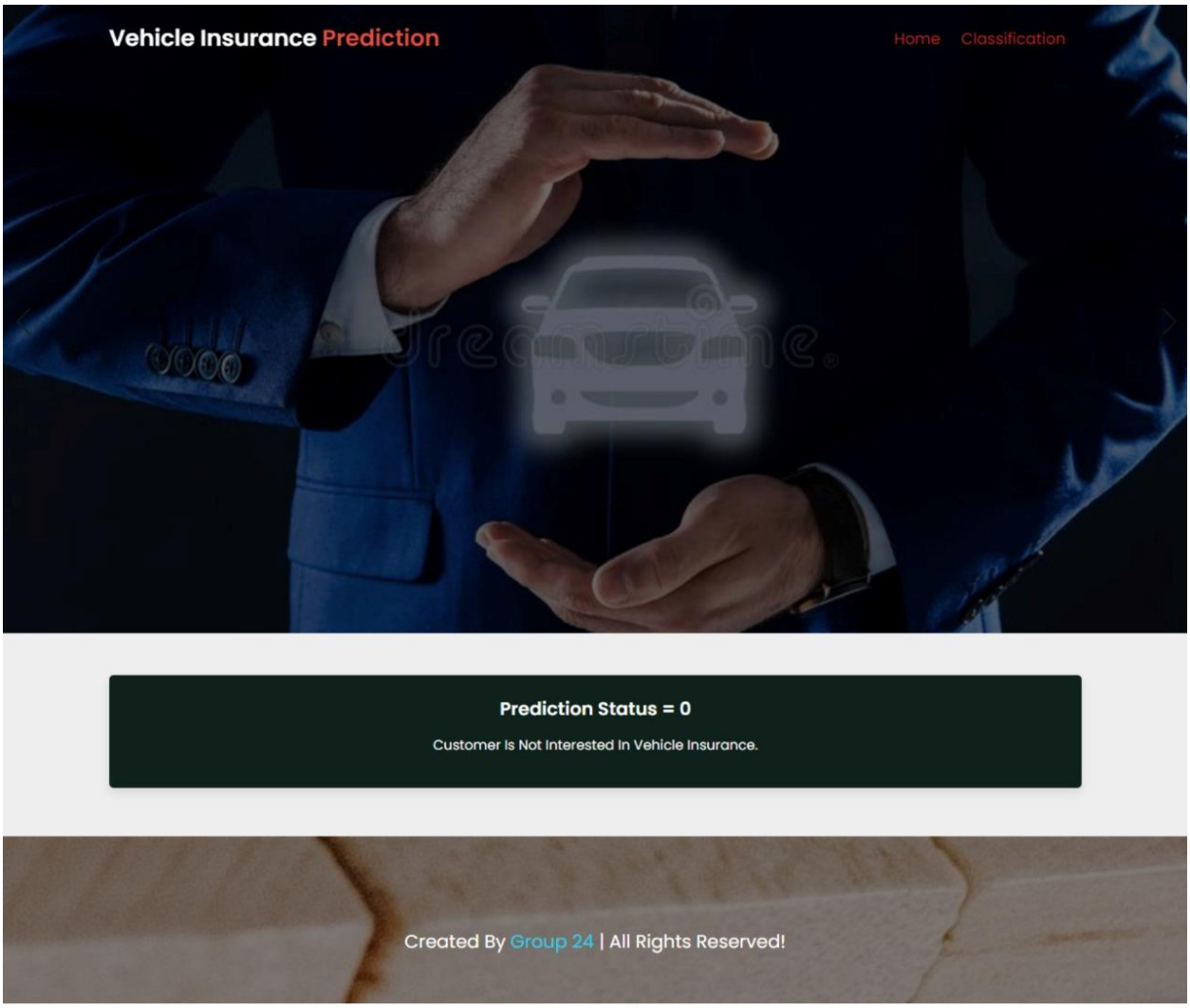

Figure 48: Prediction Status

6.4 Test Cases

Test Case ID	TC001		Test Case Description	
Pre-Requisite(s)	Dataset for classification is preprocessed and prepared. Application is hosted. Single User inputs which are used to predict whether the customer is interest to claim the vehicle insurance, normalized according to the pretrained parameters.		The user enters the vehicle details, required for a prediction. Once the details are entered, the application will generate the customer interest for the vehicle insurance.	
User Action	User Inputs	Expected Output	Actual Output	Test Result
Direct to Vehicle Insurance Prediction Screen	Enters required parameters, along with other basic details of the vehicle	Predict customer is interested to the vehicle insurance.	Based on the parameters entered, the application has predicted the customer's interest status.	Pass
Test Case ID	TC002		Test Case Description	
Pre-Requisite(s)	Single User inputs which are used to predict whether the customer is not interested to claim the vehicle insurance, normalized according to the pretrained parameters.		The user enters the vehicle details, required for a prediction. Once the details are entered, the application will generate the customer is not interested in the vehicle insurance.	

User Action	User Inputs	Expected Output	Actual Output	Test Result
Direct to Vehicle Insurance Prediction Screen	Enters required parameters, along with other basic details of the vehicle	Predict customer is not interested to the vehicle insurance.	Based on the parameters entered, the application has predicted the customer's interest status.	Pass

Table 3: Test Cases

7 . Project Management & Methodology

7.1 Methodology

Agile was adopted as project management methodology. According to the selected methodology, the project has been segregated into key phases like plan, design, develop, test, release, and feedback.

Plan	**Design**
• Identify Requirements/ Define Scope • Identify & Preprocess Datasets • Selection of tools and technologies	• Plan Model Integration •
Develop	**Test**
• Develop Machine Learning Models • Develop Web Application • Integrate Models to Application	• Test Accuracy of Data Model • Test Application with Test Cases
Release	**Feedback**

• Host Application(s) of different versions	• Obtain Feedback from Testing Team • Review Feedback on Model Accuracy and Web Application

Table 4 : Stages of Project

After commencing the project and based on the initial development there were changes identified and implemented user interfaces, model accuracy and selection of framework(s). However, adaptation of agile ensured that there are not any pauses in the progress of the project and enable the team to manage the changes effectively and deliver the project on time.

7.2 Deliverables

Delivery	Description	Submitted On
Statement of Work (SOW) Report	The report consists of key information on the project background, problem definition, scope of work, tools and techniques utilized, activities planned along with timeline and deliverables, also the tasks performed by each member.	15/10/2022
Data Mining Models	The model that would perform the tasks (Predictive or classification) for the selected data sets. This may be a file consisting only the code relevant to the models developed.	19/10/2021
Web Application	Hosted web application that could be assessed by the users to perform the operations. The model(s) are integrated to the web application.	24/10/2021
Video	Demonstration of the features of the application.	28/10/2021
Final Report	The report consists of information on project domain, solution developed and test cases.	02/11/2021
Git Hub Repository	Contains application source codes with version management.	02/11/2021

Table 4: Deliverables

8 References.

Classification Model

Random Forest

Logistic Regression

XGBoost