HEALTH INSURANCE CROSS SELL PREDICTION

Introduction:

Insurance companies invest the money securely and pay when we claim. We have different types of insurance health insurance, vehicle insurance, life insurance, pet insurance, and many more. Vehicle insurance is mandatory when you drive the vehicle because if anything goes wrong with the vehicle or any accident happens it covers all the repairs and it also covers all damages which occur due to cyclone, fire, theft, flood, etc. Vehicle insurance protects us from legal and financial liabilities towards the third party. Driving without insurance may fine you a huge amount and also cancel the driver's license. Health insurance provides financial help when you have an accident or medical problem.

This data mainly deals with health and vehicle insurance. Here our client will be an insurance company that provides health insurance to the customers and want to build a model to predict whether how many of their insurance holders from the past year are interested to take vehicle insurance from them. It will be really helpful if we build a model that predicts whether how many are interested to take vehicle insurance because they can plan accordingly and reach the customers.

Problem statement:

Vehicle insurance is mandatory as it protects us from legal and financial liabilities. The purpose of this project is to build a model to predict whether the customers are willing to take the vehicle insurance from the same insurance company. If we predict the willing customers, we can plan accordingly communicate with customers and improve the business and revenue

Objective:

The main objective of this project is to develop a prediction model in which the insurance company to predict the customers who are willing to take the vehicle insurance. The models developed in this project are Random forest, Logistic Regression, KNN model and some visualizations. The trained data helps us to predict the information about the customers.

Literature review:

Health insurance is important for every person which covers a whole or part of risk of the person including medical expenses, even the vehicle insurance is mandatory as it helps us from legal and financial liabilities. Taking the proper health and vehicle insurance will lower the risk. According to the Health Insurance Association of America, health insurance is defined as coverage that provides for the payments of benefits as a result of sickness or injury. It includes insurance for losses from accident, medical expense, disability, or accidental death and dismemberment.

Data Collection (Give the link to the files, or upload your files if they are not accessible online):

For this project, I am working on a dataset called health insurance cross-sell prediction which is a second-hand data taken from online resources called Kaggle. This dataset contains train data and test data. The variables in this dataset are

Id: Unique id of the customer

Gender: Gender of the customer

Age: Age of the customer

Driving_License: 1-customer have DL,0-customer doesn't have DL

Region_Code: Unique region code of the customer.

Previously_Insured:1-customer have vehicle insurance,0-customer doesn't have vehicle insurance

Vehicle_Age: Age of the vehicle

Vehicle_Damage: 1-customer got his vehicle damaged in past,0-customer doesn't get his vehicle damaged in past

Annual_Premium: Amount customers need to pay for the company for a premium.

Vintage: No.of days customers are associated with the company

Response: 1-customer is interested,0-customer is not interested

https://www.kaggle.com/anmolkumar/health-insurance-cross-sell-prediction

Research design and methodology:

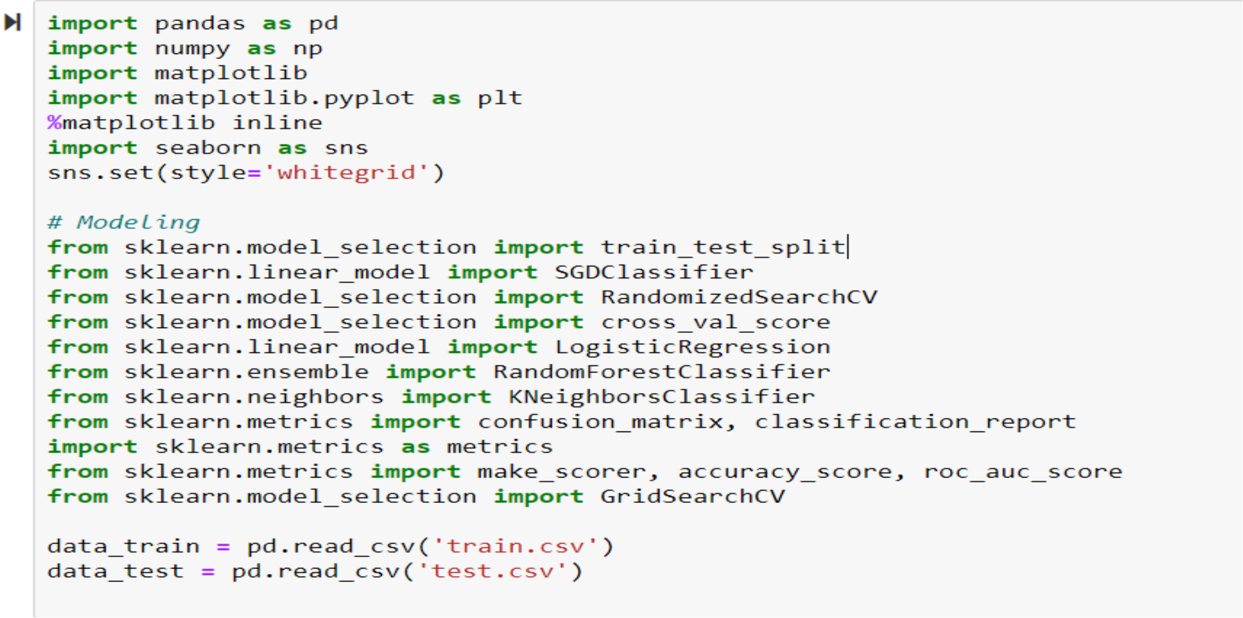
Initially, we study the data, and then data cleaning is very important for data processing as the data contains many missing variables and null variables. Now I will process the data and predict whether the customers are willing to take the vehicle insurance or not using the previous data by making it as the trained data and using this trained data. I have used Random forest, Logistic Regression and KNN models to compare the accuracy which best fits the model and performed some visualizations and perform sentimental analysis for removing the stop words and determine common customers.

Exploratory data analysis (EDA) and Hypotheses for the Study:

Initially imported the data, and observed the data and made necessary changes, removed null values, deleted the unwanted columns from the data and changed the categorical values to numerical values which helped in building the model.

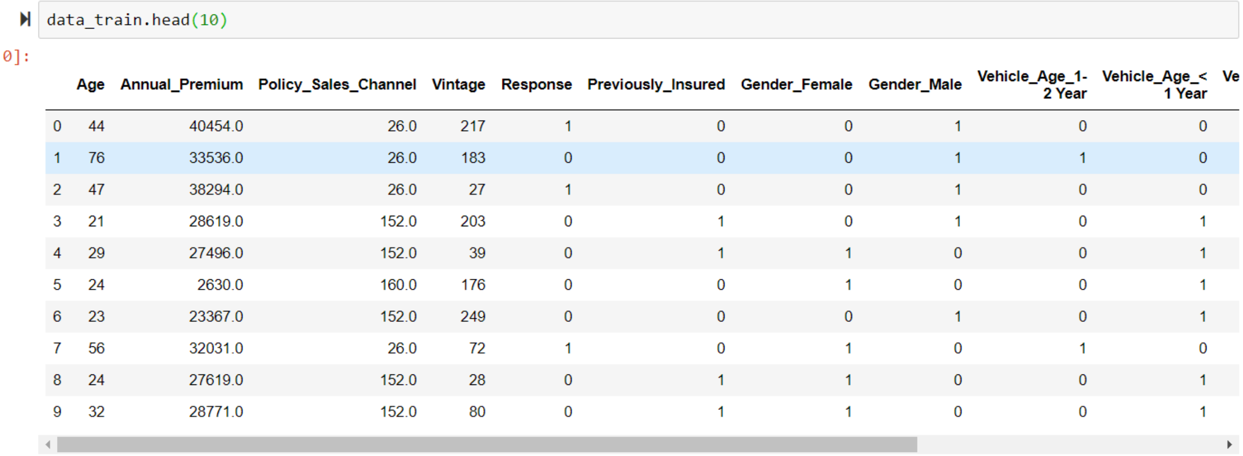
Data Analytics:

Below is the brief summary of my data, I have imported all the necessary libraries which helps in visualizations and building the models and also read the train data and test data.



```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(style='whitegrid')

# Modeling
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report
import sklearn.metrics as metrics
from sklearn.metrics import make_scorer, accuracy_score, roc_auc_score
from sklearn.model_selection import GridSearchCV

data_train = pd.read_csv('train.csv')
data_test = pd.read_csv('test.csv')
```

Here are the first 10 rows of data which we are using for building a model.

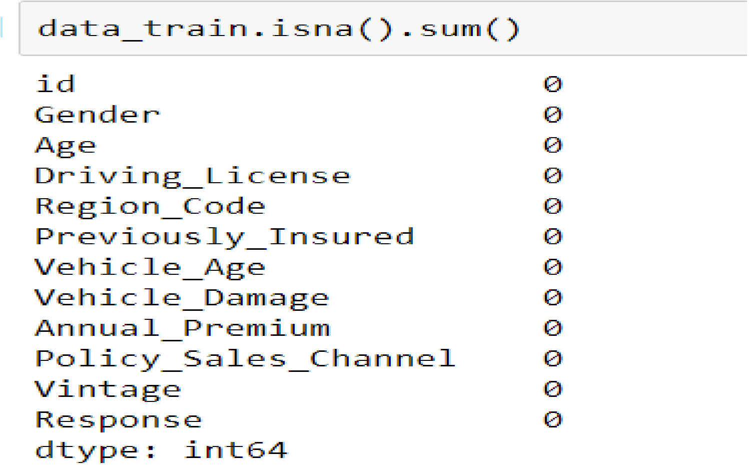


```
data_train.head(10)
```

0]:

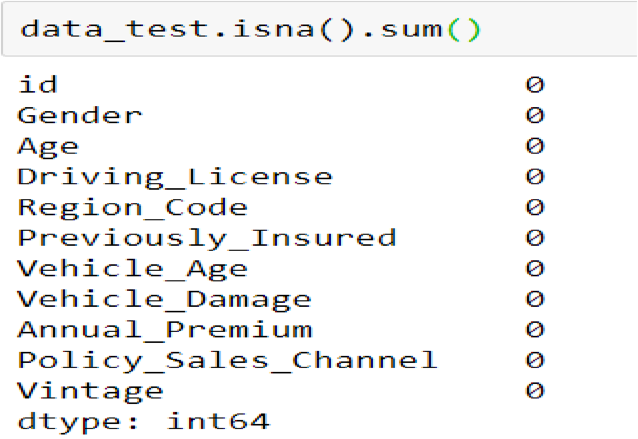
	Age	Annual_Premium	Policy_Sales_Channel	Vintage	Response	Previously_Insured	Gender_Female	Gender_Male	Vehicle_Age_1-2 Year	Vehicle_Age_< 1 Year	Ve
0	44	40454.0	26.0	217	1	0	0	1	0	0	
1	76	33536.0	26.0	183	0	0	0	1	1	0	
2	47	38294.0	26.0	27	1	0	0	1	0	0	
3	21	28619.0	152.0	203	0	1	0	1	0	1	
4	29	27496.0	152.0	39	0	1	1	0	0	1	
5	24	2630.0	160.0	176	0	0	1	0	0	1	
6	23	23367.0	152.0	249	0	0	0	1	0	1	
7	56	32031.0	26.0	72	1	0	1	0	1	0	
8	24	27619.0	152.0	28	0	1	1	0	0	1	
9	32	28771.0	152.0	80	0	1	1	0	0	1	

Here I have checked whether the train data and test data contain any null values, but this dataset doesn't contain any values.



```
data_train.isna().sum()
```

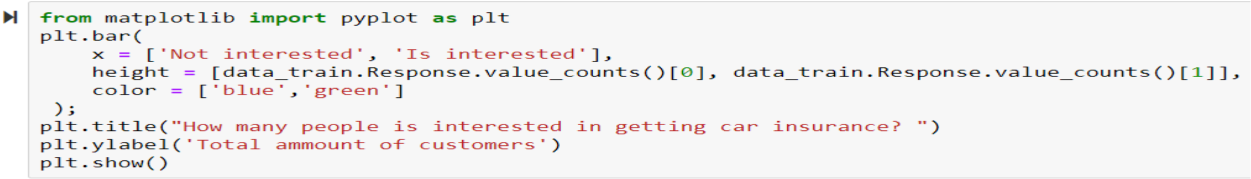
```
id                      0
Gender                  0
Age                     0
Driving_License         0
Region_Code             0
Previously_Insured      0
Vehicle_Age             0
Vehicle_Damage          0
Annual_Premium          0
Policy_Sales_Channel    0
Vintage                 0
Response  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



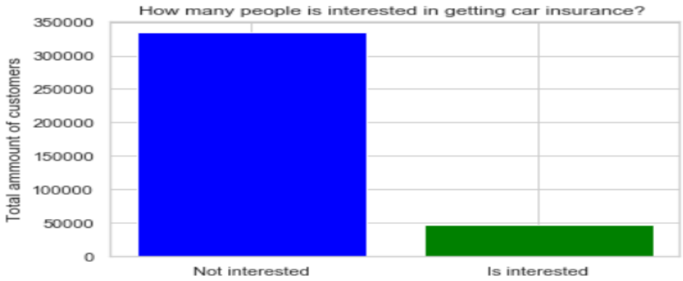
```
data_test.isna().sum()
```

```
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
dtype: int64
```

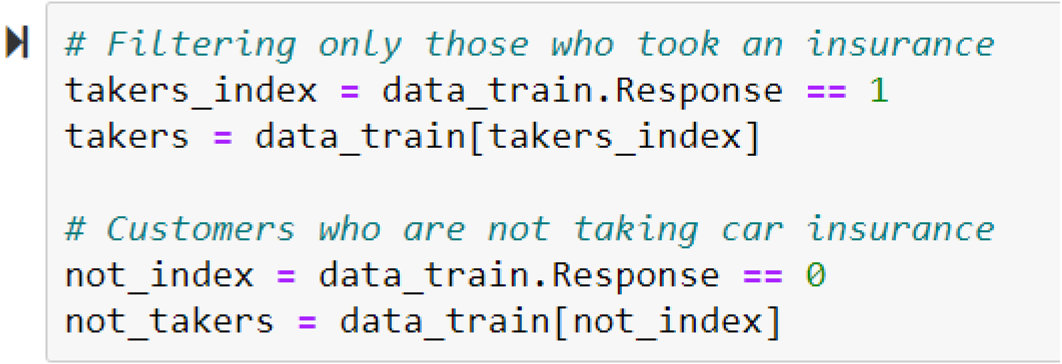
Data Visualization and Results Report:



```
from matplotlib import pyplot as plt
plt.bar(
    x = ['Not interested', 'Is interested'],
    height = [data_train.Response.value_counts()[0], data_train.Response.value_counts()[1]],
    color = ['blue','green']
 );
plt.title("How many people is interested in getting car insurance? ")
plt.ylabel('Total ammount of customers')
plt.show()
```

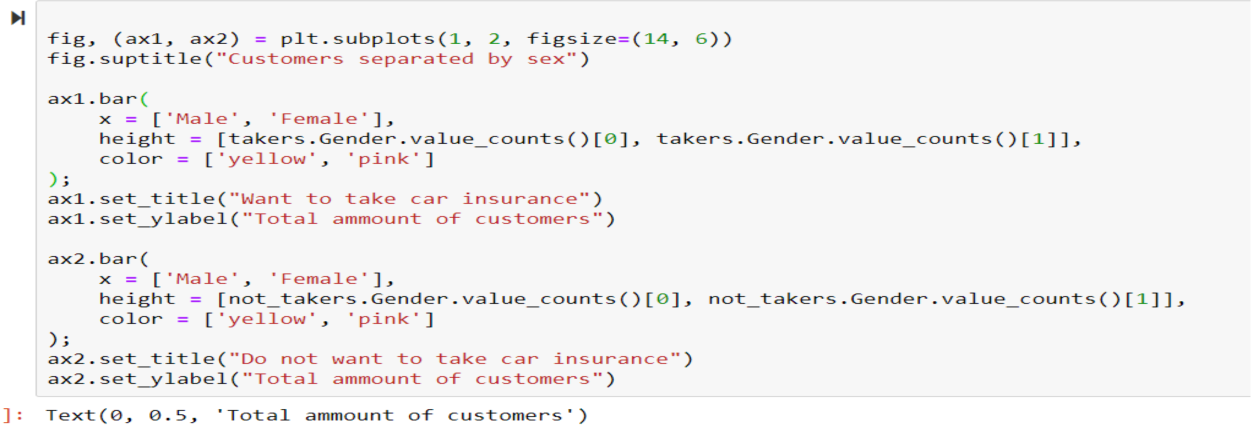


From the above graph, I want to represent who are interested in taking the car insurance, so from the graph we can say that maximum people are not interested in taking the car insurance than taking the insurance from the same company.



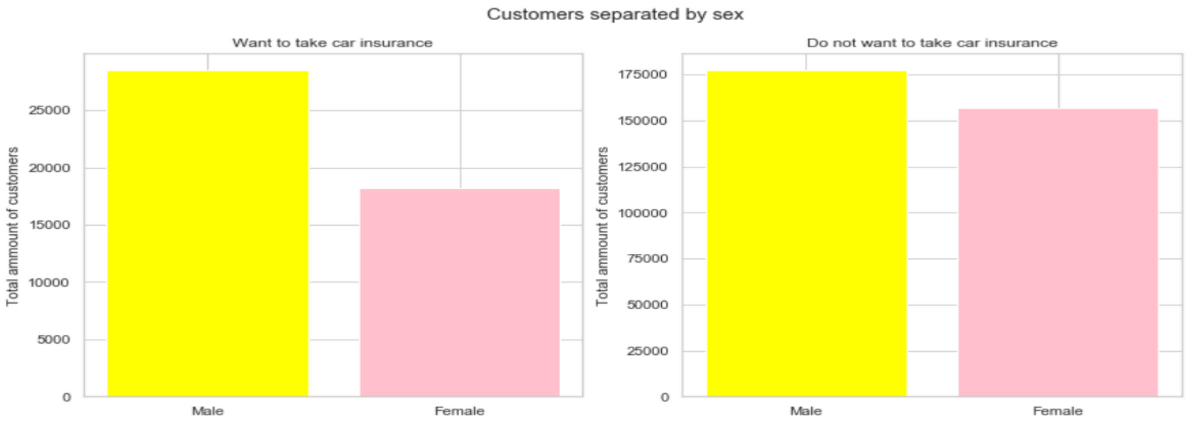
```
# Filtering only those who took an insurance
takers_index = data_train.Response == 1
takers = data_train[takers_index]

# Customers who are not taking car insurance
not_index = data_train.Response == 0
not_takers = data_train[not_index]
```

Here I have filtered the customers according to their priority in taking the insurance.

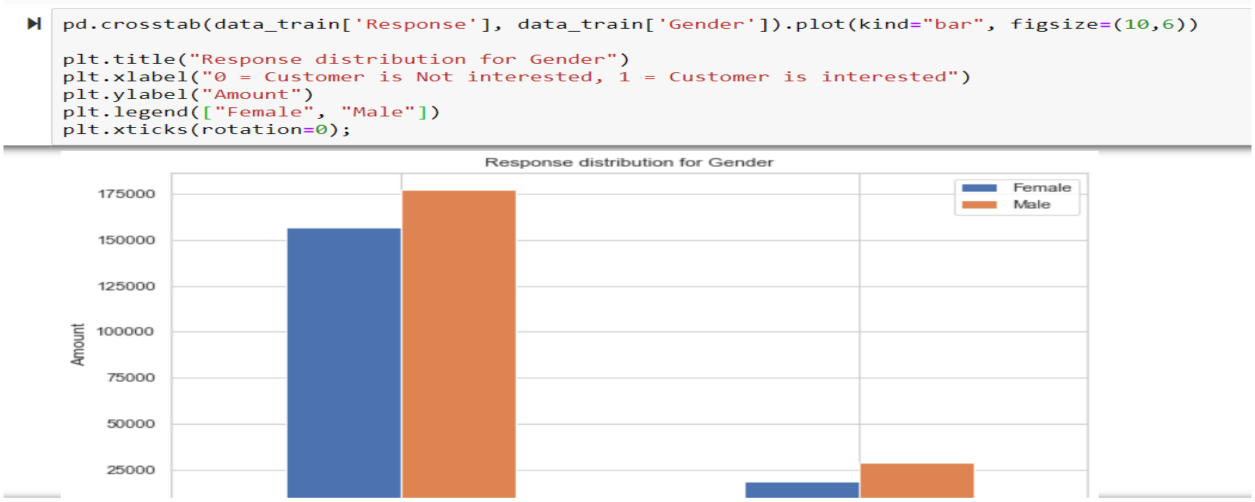


```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))
fig.suptitle("Customers separated by sex")

ax1.bar(
    x = ['Male', 'Female'],
    height = [takers.Gender.value_counts()[0], takers.Gender.value_counts()[1]],
    color = ['yellow', 'pink']
);
ax1.set_title("Want to take car insurance")
ax1.set_ylabel("Total ammount of customers")

ax2.bar(
    x = ['Male', 'Female'],
    height = [not_takers.Gender.value_counts()[0], not_takers.Gender.value_counts()[1]],
    color = ['yellow', 'pink']
);
ax2.set_title("Do not want to take car insurance")
ax2.set_ylabel("Total ammount of customers")
```

```
]: Text(0, 0.5, 'Total ammount of customers')
```

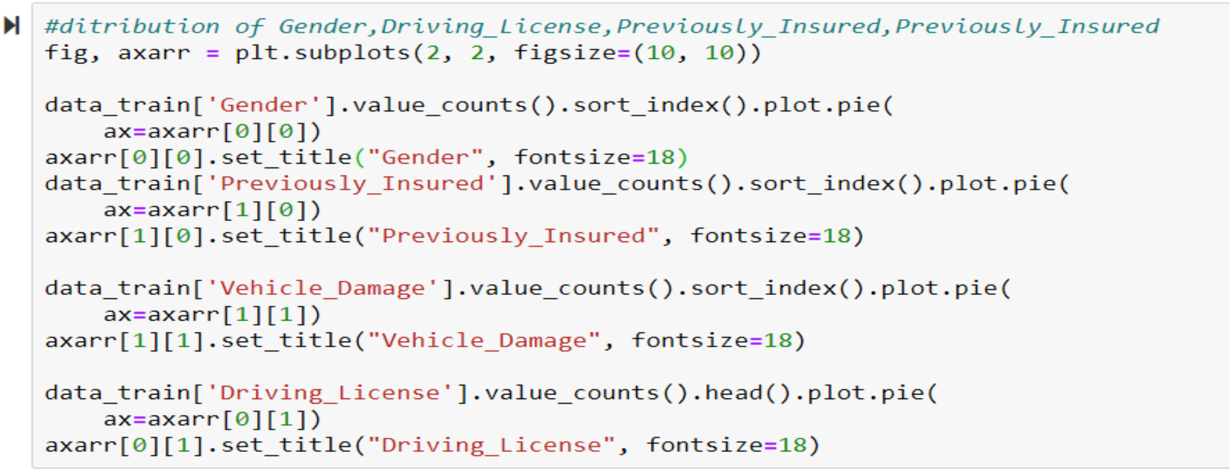


From the above graph , I want to represent their willingness in taking the insurance according to gender, so by observing the above graph we can say that male is more likely to take the insurance, and even in the other case also male is more likely to not take the insurance.

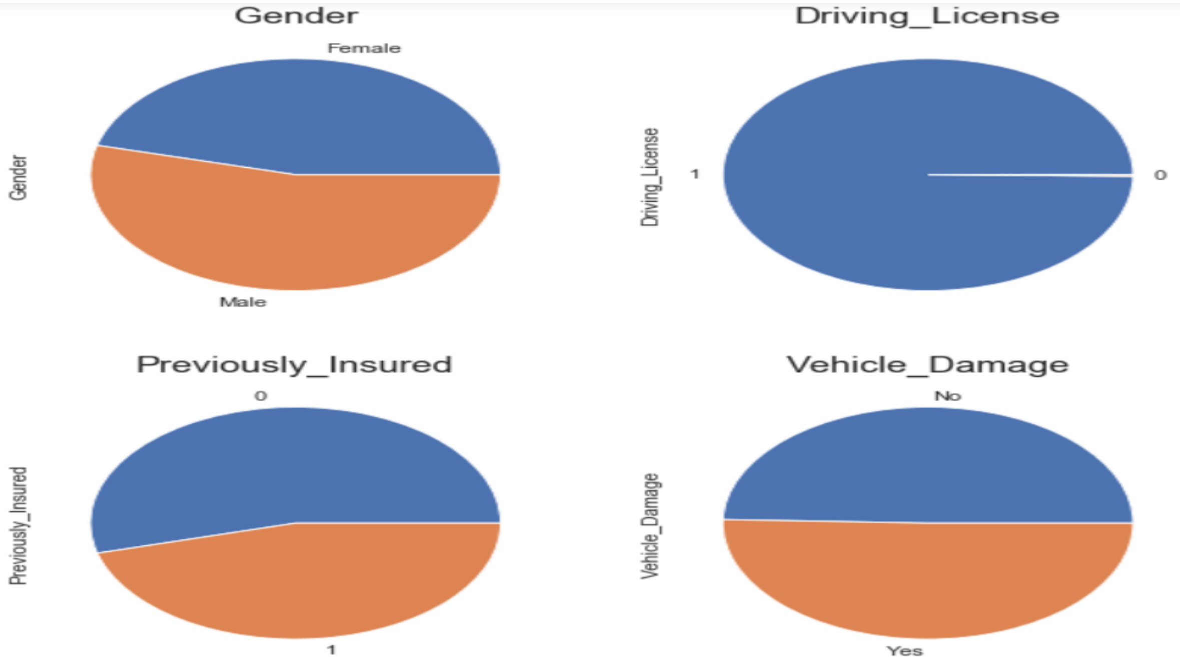


```
pd.crosstab(data_train['Response'], data_train['Gender']).plot(kind="bar", figsize=(10,6))

plt.title("Response distribution for Gender")
plt.xlabel("0 = Customer is Not interested, 1 = Customer is interested")
plt.ylabel("Amount")
plt.legend(["Female", "Male"])
plt.xticks(rotation=0);
```

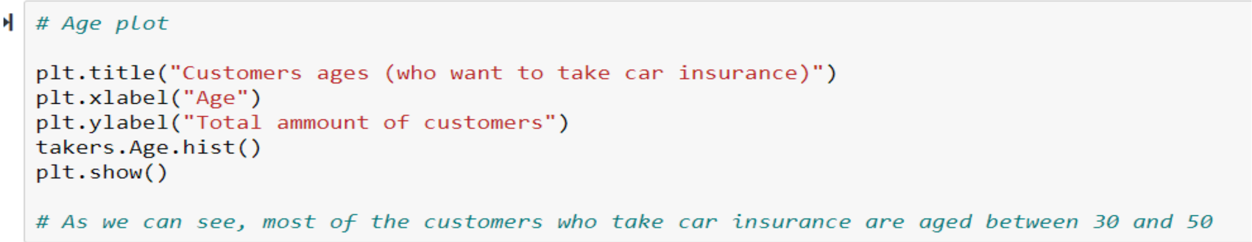
From the above graph , I want to represent the response distribution for gender based on the amount, from the bar graph we can say that male is more interested in taking the insurance than female.



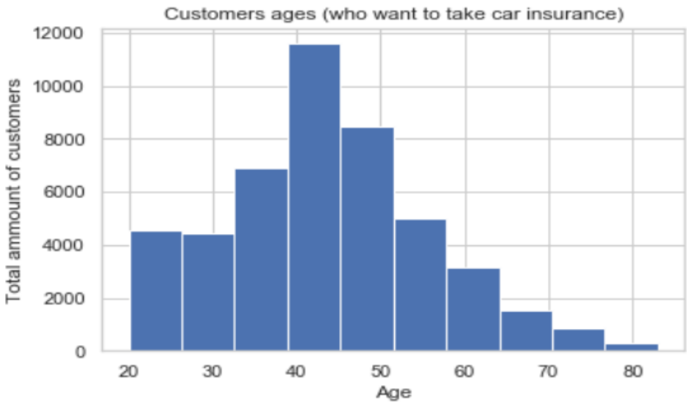
```
#ditribution of Gender,Driving_License,Previously_Insured,Previously_Insured
fig, axarr = plt.subplots(2, 2, figsize=(10, 10))

data_train['Gender'].value_counts().sort_index().plot.pie(
    ax=axarr[0][0])
axarr[0][0].set_title("Gender", fontsize=18)
data_train['Previously_Insured'].value_counts().sort_index().plot.pie(
    ax=axarr[1][0])
axarr[1][0].set_title("Previously_Insured", fontsize=18)

data_train['Vehicle_Damage'].value_counts().sort_index().plot.pie(
    ax=axarr[1][1])
axarr[1][1].set_title("Vehicle_Damage", fontsize=18)

data_train['Driving_License'].value_counts().head().plot.pie(
    ax=axarr[0][1])
axarr[0][1].set_title("Driving_License", fontsize=18)
```



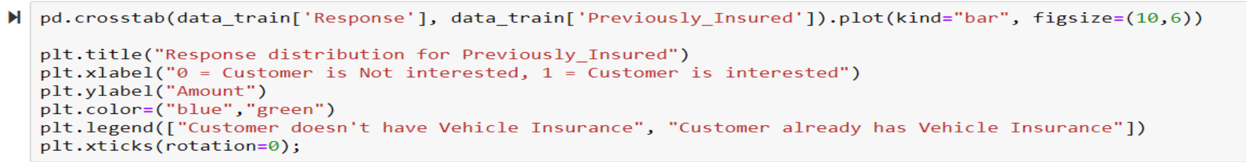
Here in the above pie-chart, I want to represent the distribution of gender , driving license, previous insured, vehicle damage. We can say that male is more likely to take insurance.



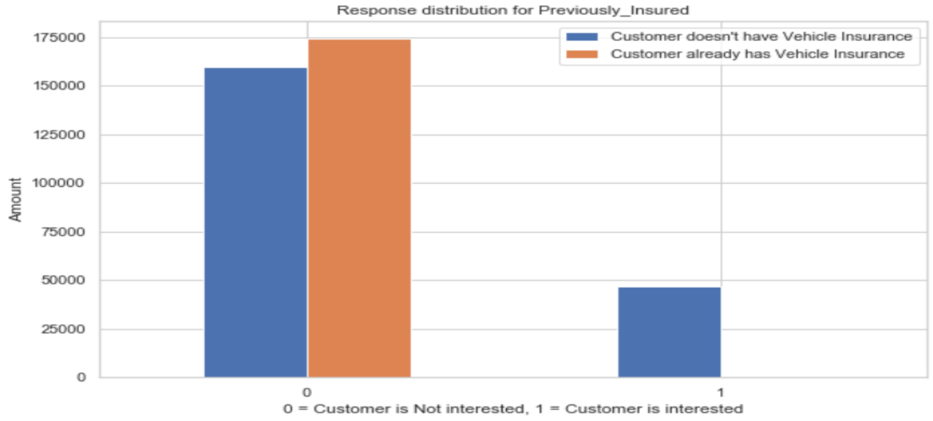
```
# Age plot

plt.title("Customers ages (who want to take car insurance)")
plt.xlabel("Age")
plt.ylabel("Total ammount of customers")
takers.Age.hist()
plt.show()

# As we can see, most of the customers who take car insurance are aged between 30 and 50
```



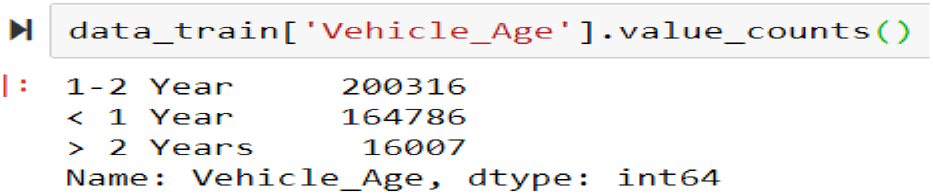
From the above histogram, I have represented the customer ages who are taking the car insurance, from the graph we can observe that customers with age 30 and 50 are more willing to take the car insurance.



```
pd.crosstab(data_train['Response'], data_train['Previously_Insured']).plot(kind="bar", figsize=(10,6))

plt.title("Response distribution for Previously_Insured")
plt.xlabel("0 = Customer is Not interested, 1 = Customer is interested")
plt.ylabel("Amount")
plt.color=("blue","green")
plt.legend(["Customer doesn't have Vehicle Insurance", "Customer already has Vehicle Insurance"])
plt.xticks(rotation=0);
```



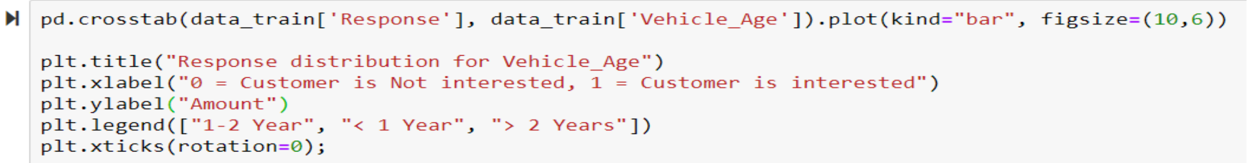
From the above bar graph, I have represented response distribution for previously insurance customers, so from above graph we can say that customer who are not interested already have vehicle insurance, so they are not willing to take vehicle insurance.



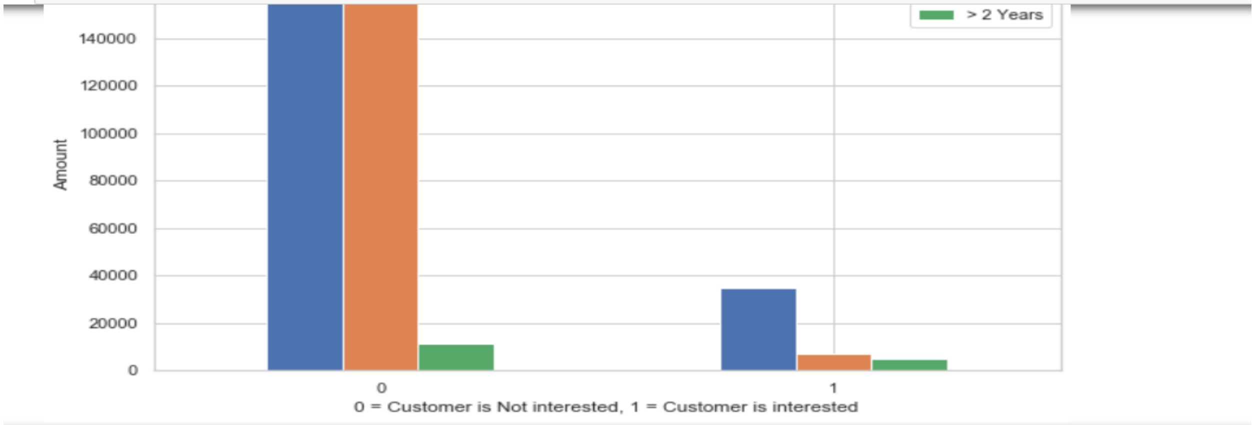
```
data_train['Vehicle_Age'].value_counts()
```

```
1-2 Year     200316
< 1 Year     164786
> 2 Years     16007
Name: Vehicle_Age, dtype: int64
```

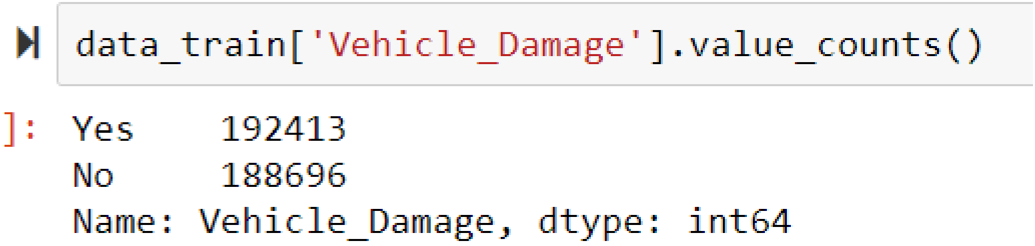
Here there is count of vehicles with regarding their vehicle age.



```
pd.crosstab(data_train['Response'], data_train['Vehicle_Age']).plot(kind="bar", figsize=(10,6))

plt.title("Response distribution for Vehicle_Age")
plt.xlabel("0 = Customer is Not interested, 1 = Customer is interested")
plt.ylabel("Amount")
plt.legend(["1-2 Year", "< 1 Year", "> 2 Years"])
plt.xticks(rotation=0);
```



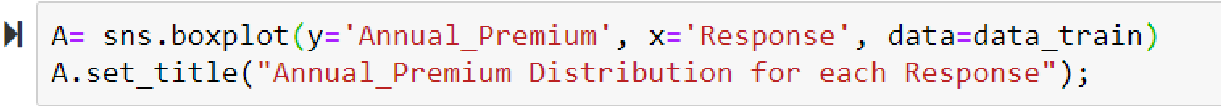
From the above bar graph, I have represented the customer vehicle ages who are interested and not interested in taking insurance ,customers who have vehicle with age < 1 year are not likely to take insurance.



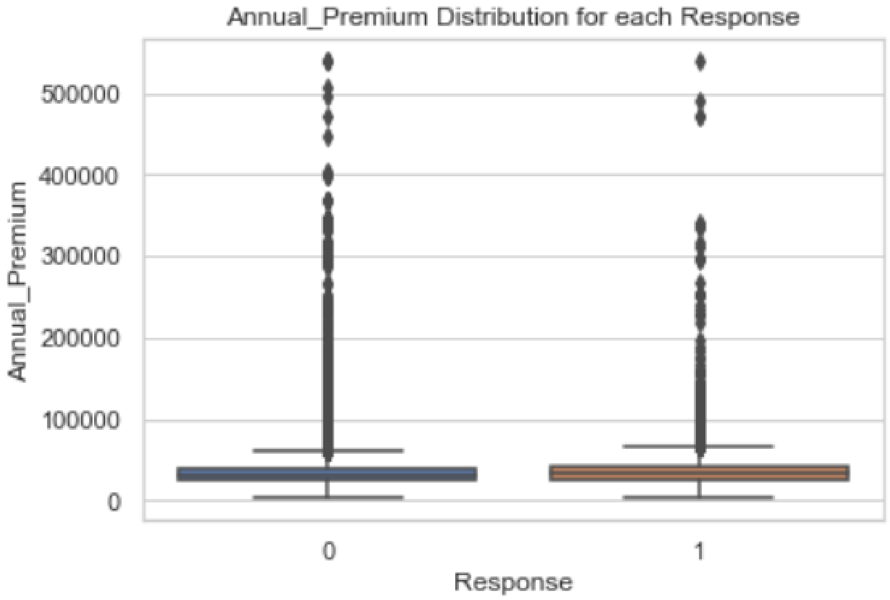
```
data_train['Vehicle_Damage'].value_counts()
```

```
]:  Yes     192413
    No      188696
    Name: Vehicle_Damage, dtype: int64
```

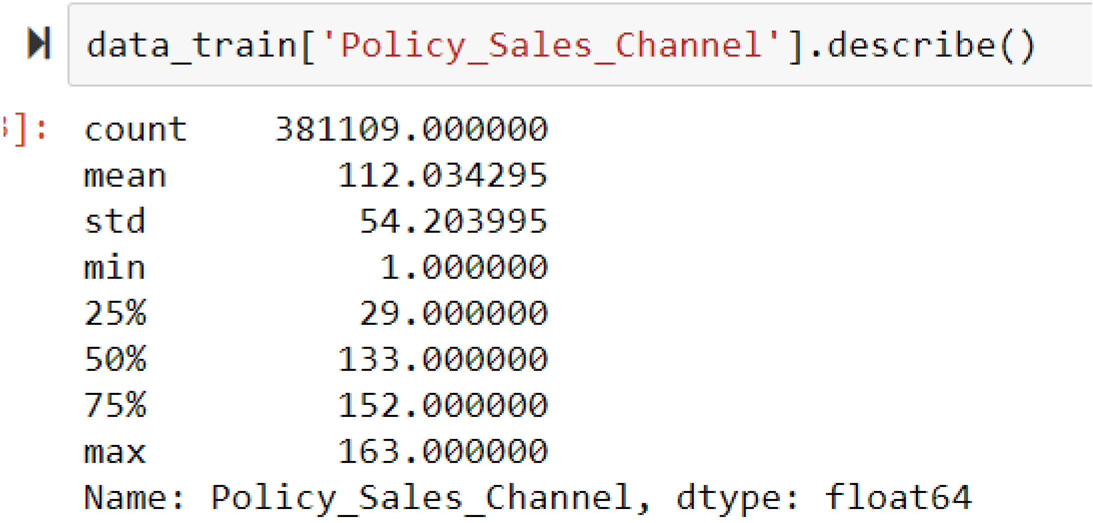
Here is the count of all the vehicle damage who have damage and don’t have damage.



```
A= sns.boxplot(y='Annual_Premium', x='Response', data=data_train)
A.set_title("Annual_Premium Distribution for each Response");
```



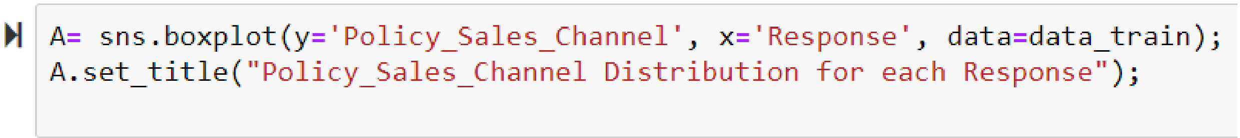
From the above boxplot, I have represented the annual premium distribution for each response, from the above graph we can say that we have many outliers.



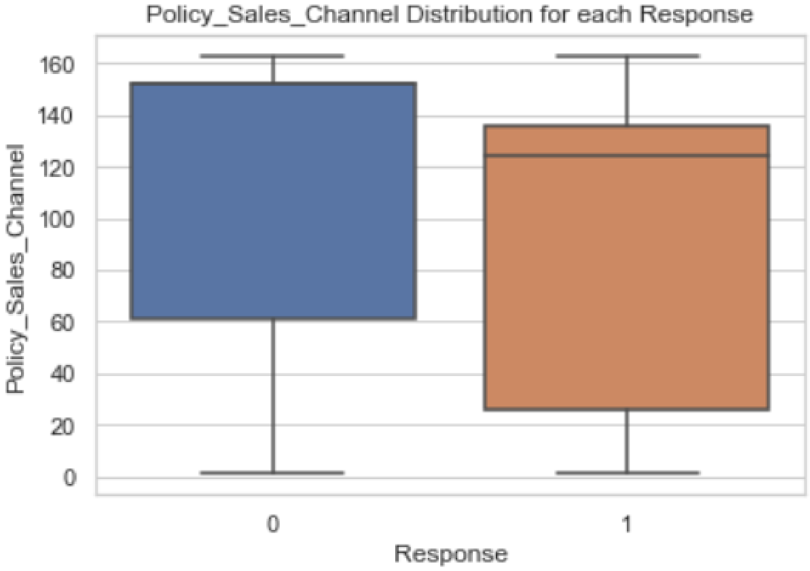
```
data_train['Policy_Sales_Channel'].describe()
```

```
]:  count    381109.000000
    mean        112.034295
    std          54.203995
    min           1.000000
    25%          29.000000
    50%         133.000000
    75%         152.000000
    max         163.000000
    Name: Policy_Sales_Channel, dtype: float64
```

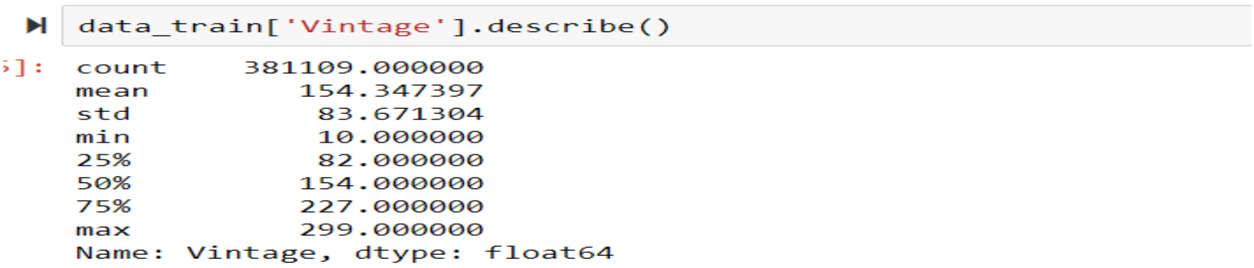
Here it described the count, mean, min, std of policy sales channel.



```
A= sns.boxplot(y='Policy_Sales_Channel', x='Response', data=data_train);
A.set_title("Policy_Sales_Channel Distribution for each Response");
```

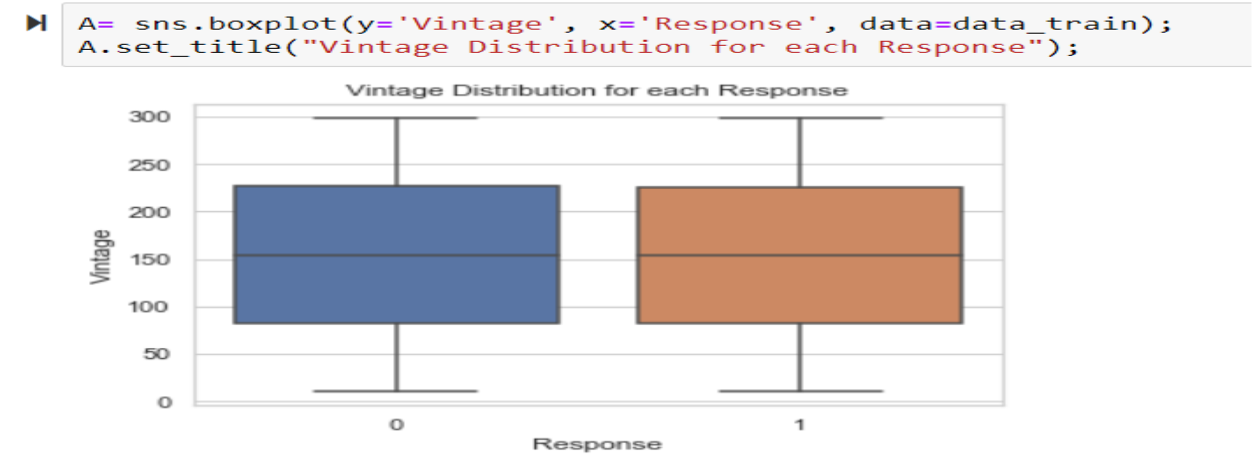


Here is the box plot for policy sales channel distribution for each response.



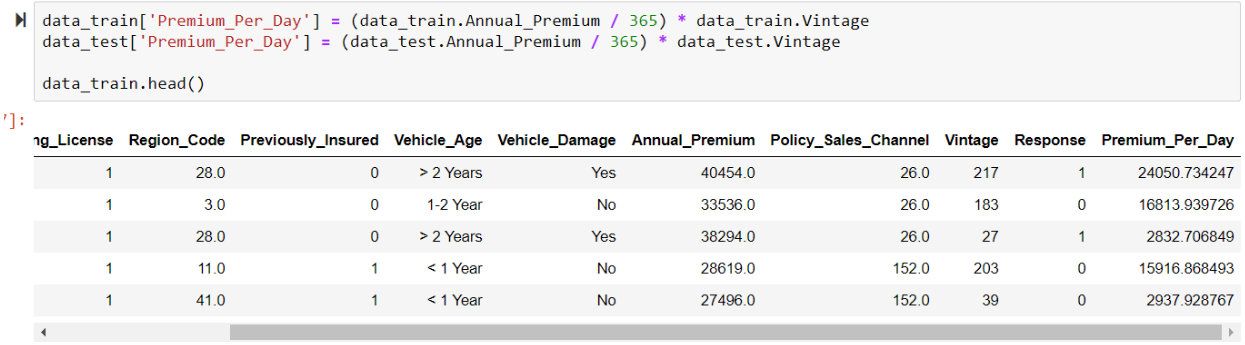
```
data_train['Vintage'].describe()
```

```
5]:  count     381109.000000
     mean         154.347397
     std           83.671304
     min           10.000000
     25%           82.000000
     50%          154.000000
     75%          227.000000
     max          299.000000
     Name: Vintage, dtype: float64
```



```
A= sns.boxplot(y='Vintage', x='Response', data=data_train);
A.set_title("Vintage Distribution for each Response");
```

Here is the box plot about vintage distribution for each response.

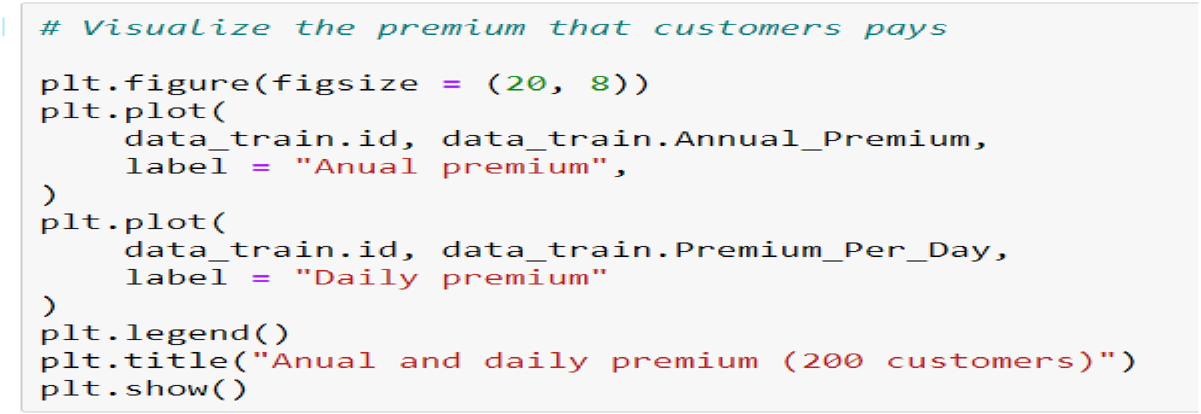


```
data_train['Premium_Per_Day'] = (data_train.Annual_Premium / 365) * data_train.Vintage
data_test['Premium_Per_Day'] = (data_test.Annual_Premium / 365) * data_test.Vintage

data_train.head()
```

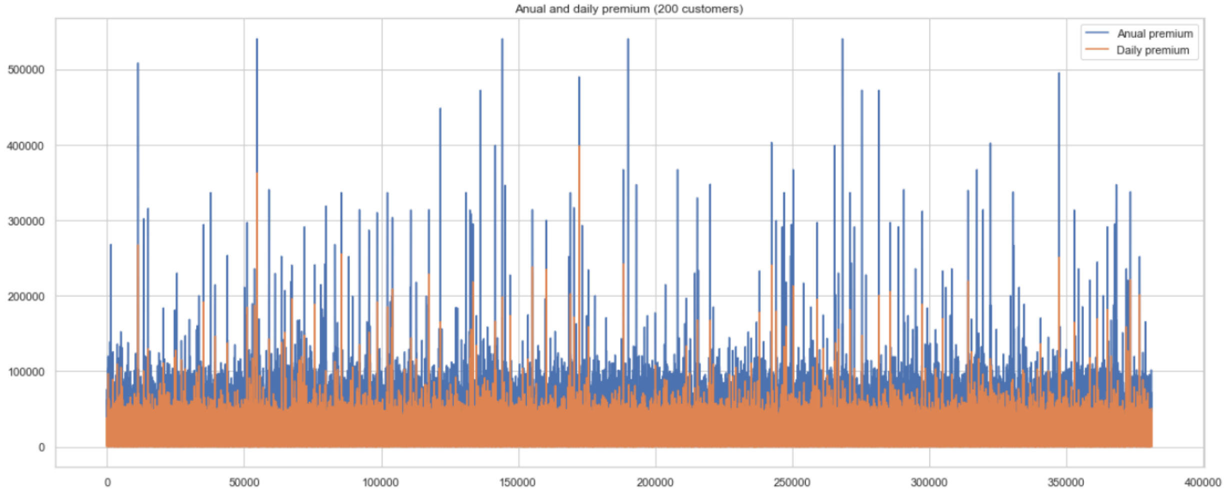
']:

ng_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel	Vintage	Response	Premium_Per_Day
1	28.0	0	> 2 Years	Yes	40454.0	26.0	217	1	24050.734247
1	3.0	0	1-2 Year	No	33536.0	26.0	183	0	16813.939726
1	28.0	0	> 2 Years	Yes	38294.0	26.0	27	1	2832.706849
1	11.0	1	< 1 Year	No	28619.0	152.0	203	0	15916.868493
1	41.0	1	< 1 Year	No	27496.0	152.0	39	0	2937.928767

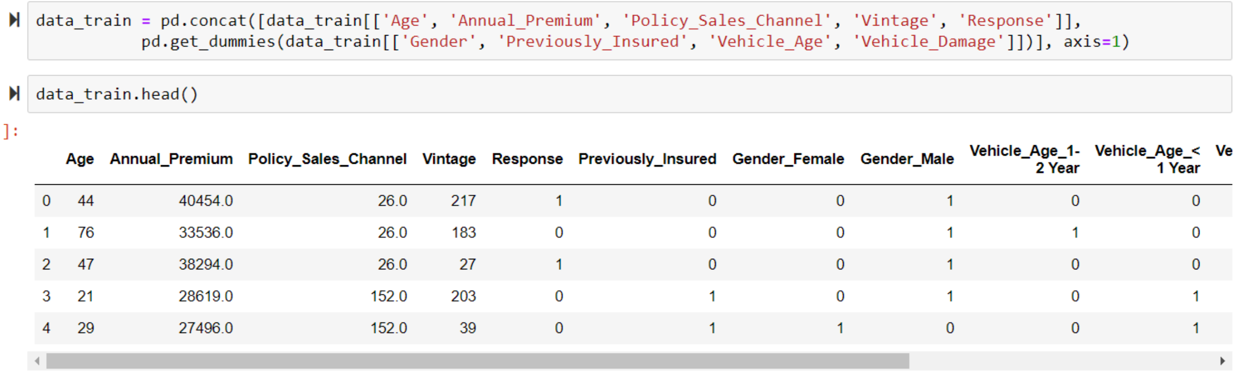
Here we have created a new column called premium per day using annual premium by diving it with 365 days * data train vintage, which may be used to calculate the other responses.



```
# Visualize the premium that customers pays

plt.figure(figsize = (20, 8))
plt.plot(
    data_train.id, data_train.Annual_Premium,
    label = "Anual premium",
)
plt.plot(
    data_train.id, data_train.Premium_Per_Day,
    label = "Daily premium"
)
plt.legend()
plt.title("Anual and daily premium (200 customers)")
plt.show()
```



Here the graph, for annual and daily premium of the 200 customers who pays insurance with regard with the amount.



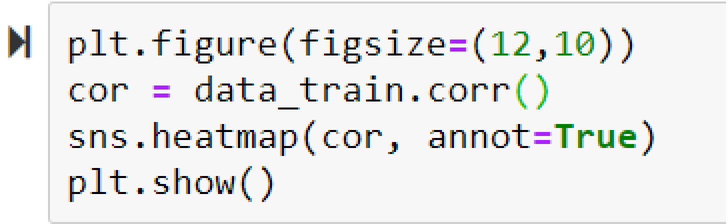
```
data_train = pd.concat([data_train[['Age', 'Annual_Premium', 'Policy_Sales_Channel', 'Vintage', 'Response']],
             pd.get_dummies(data_train[['Gender', 'Previously_Insured', 'Vehicle_Age', 'Vehicle_Damage']])], axis=1)
```

```
data_train.head()
```

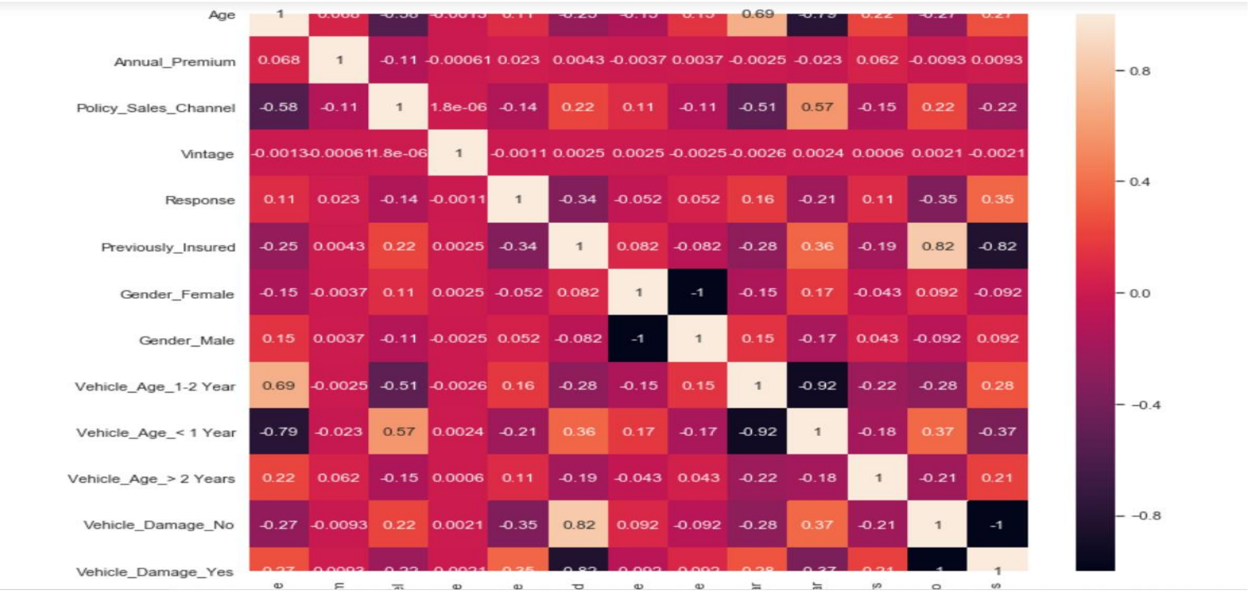
]:

	Age	Annual_Premium	Policy_Sales_Channel	Vintage	Response	Previously_Insured	Gender_Female	Gender_Male	Vehicle_Age_1-2 Year	Vehicle_Age_< 1 Year	Ve
0	44	40454.0	26.0	217	1	0	0	1	0	0	
1	76	33536.0	26.0	183	0	0	0	1	1	0	
2	47	38294.0	26.0	27	1	0	0	1	0	0	
3	21	28619.0	152.0	203	0	1	0	1	0	1	
4	29	27496.0	152.0	39	0	1	1	0	0	1	

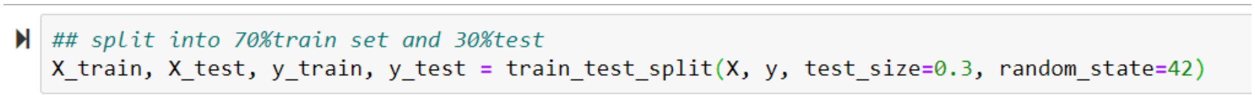
Here I have concatenated age, annual premium, policy sales channel, vintage, response together, and also changed the categorical values to numerical values using dummies. I have changed gender, previously insured, vehicle age and vehicle damage into numerical values.



```
plt.figure(figsize=(12,10))
cor = data_train.corr()
sns.heatmap(cor, annot=True)
plt.show()
```



Here is the heatmap which helps in understanding the co-relation between different variables, it is a co-relation matrix, where previously insured and vehicle damage age are more corelated with 0.82.



```
## split into 70%train set and 30%test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

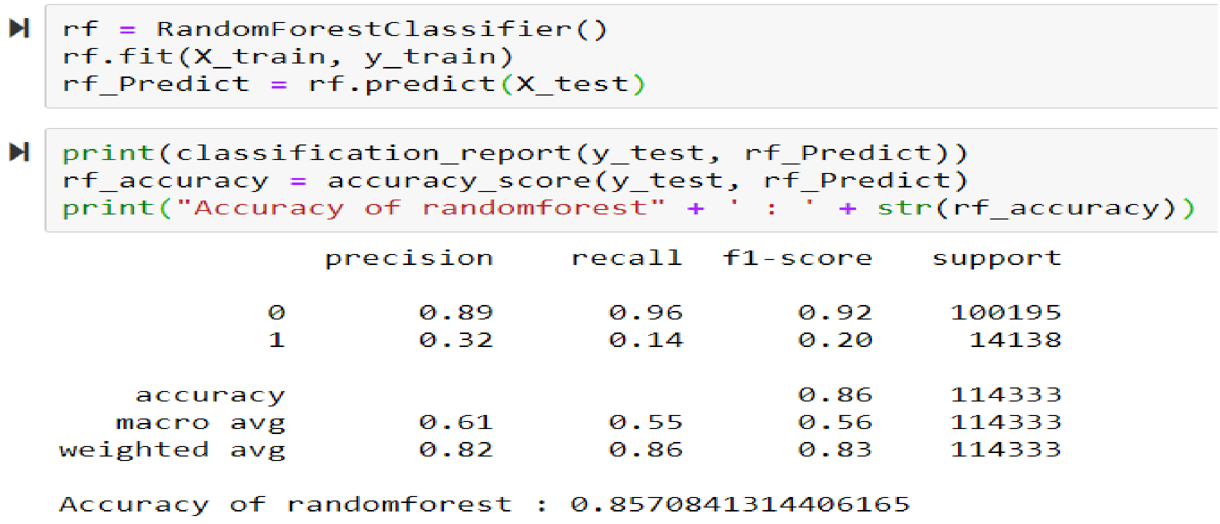
Here we have divided 70% data into train data and 30% into test.

Building the model.

RANDOM FOREST:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing at training time a multitude of decision trees and generating the class that is the class mode or the individual trees' mean/average prediction. It creates many classification trees, and a bootstrap sample technique is used to train each tree from the set of training data.

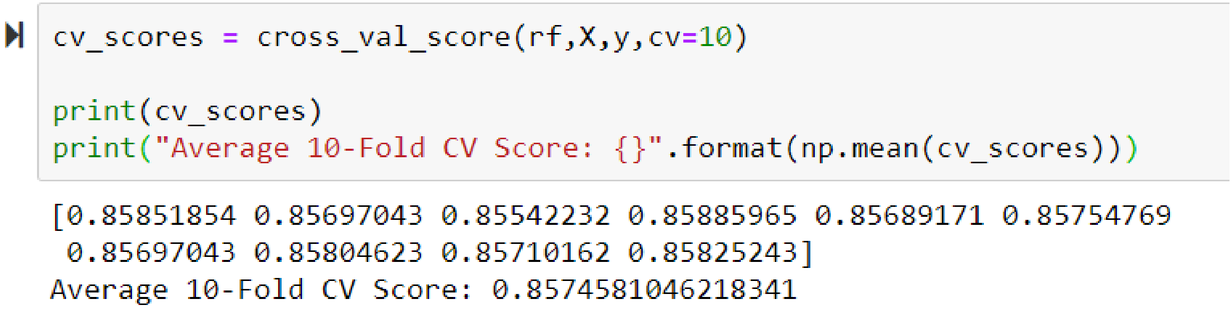
Random forest is an algorithm for classification consisting of several decision tress. When constructing each individual tree, it uses bagging and feature randomness to try to build an uncorrelated forest of trees whose prediction by committee is more precise than that of any individual tree. I have imported random forest classifier from sklearn package and predicted the accuracy of the model.



```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf_Predict = rf.predict(X_test)
```

```
print(classification_report(y_test, rf_Predict))
rf_accuracy = accuracy_score(y_test, rf_Predict)
print("Accuracy of randomforest" + ' : ' + str(rf_accuracy))
```

```
              precision    recall  f1-score   support

           0       0.89      0.96      0.92    100195
           1       0.32      0.14      0.20     14138

    accuracy                           0.86    114333
   macro avg       0.61      0.55      0.56    114333
weighted avg       0.82      0.86      0.83    114333

Accuracy of randomforest : 0.8570841314406165
```

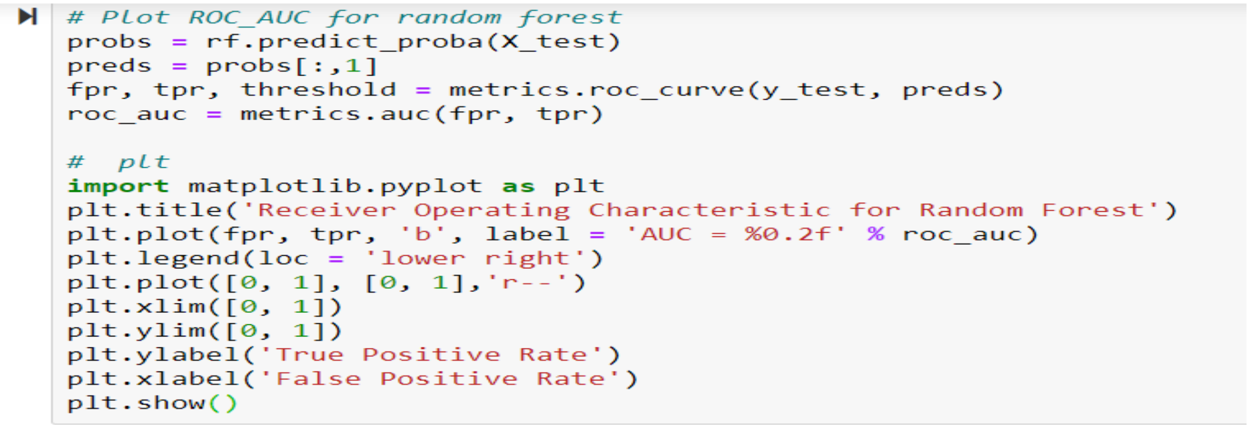


```
cv_scores = cross_val_score(rf,X,y,cv=10)

print(cv_scores)
print("Average 10-Fold CV Score: {}".format(np.mean(cv_scores)))
```

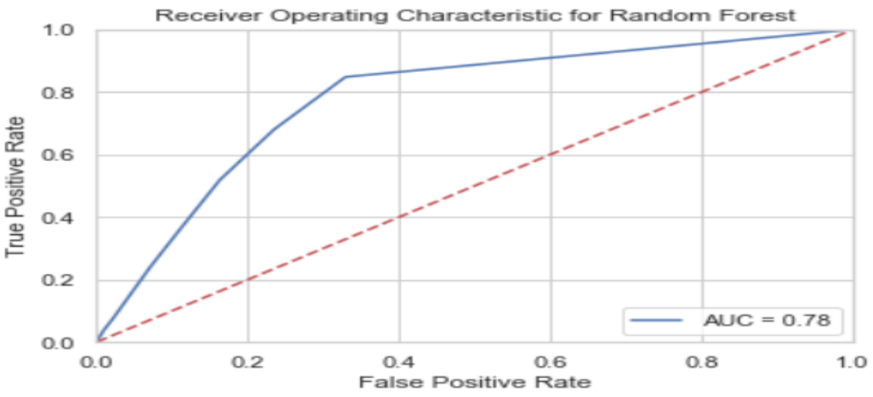
```
[0.85851854 0.85697043 0.85542232 0.85885965 0.85689171 0.85754769
 0.85697043 0.85804623 0.85710162 0.85825243]
Average 10-Fold CV Score: 0.8574581046218341
```

Average 10-fold CV score is 0.8574581046218341.

Generating ROC and AUC score for random forest



```
# Plot ROC_AUC for random forest
probs = rf.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

#  plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic for Random Forest')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

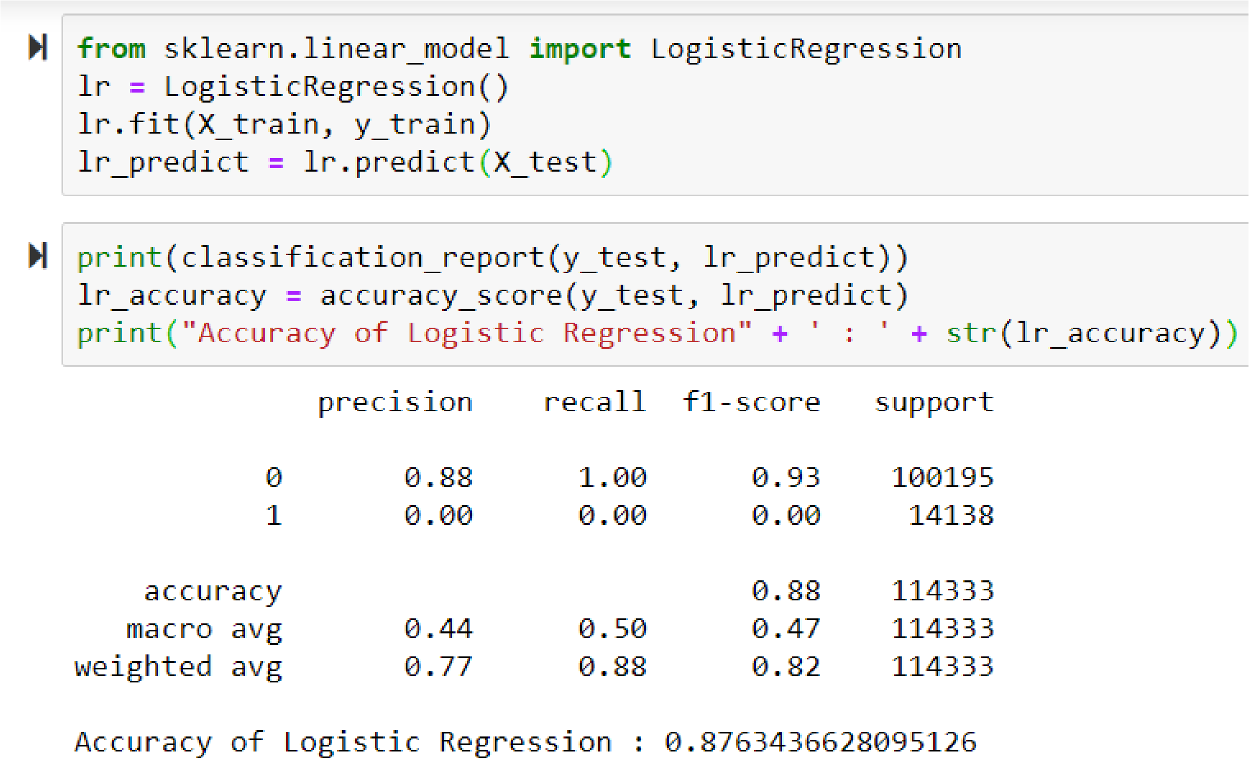


AUC score for random forest is 0.78.

Logistic Regression:

Logistic regression is a statistical model , while several more complex extensions exist, uses a logistic function to model a binary dependent variable in its basic form. Logistic regression in regression analysis estimates the parameters of a logistic model (a form of binary regression).

In statistics, the logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion. I have imported logistic regression from sklearn package and build a model.

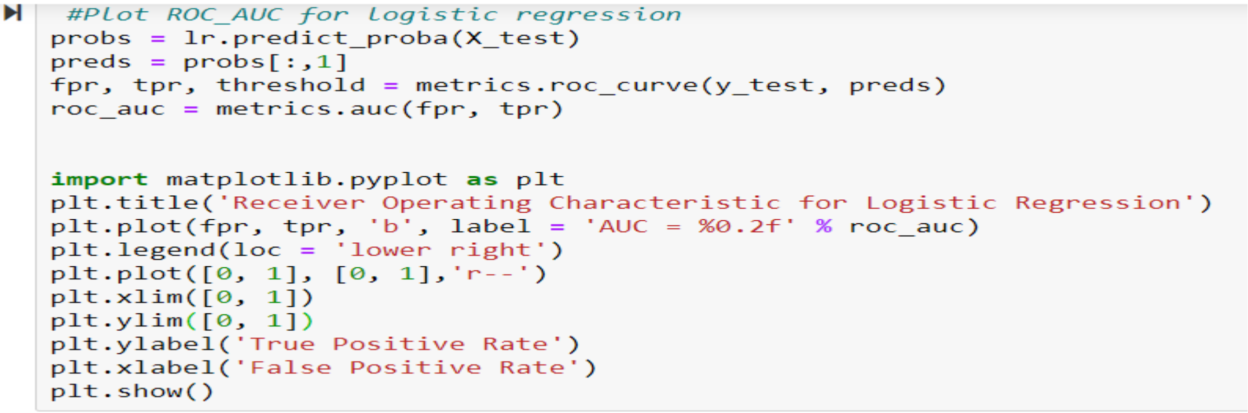


```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr_predict = lr.predict(X_test)
```

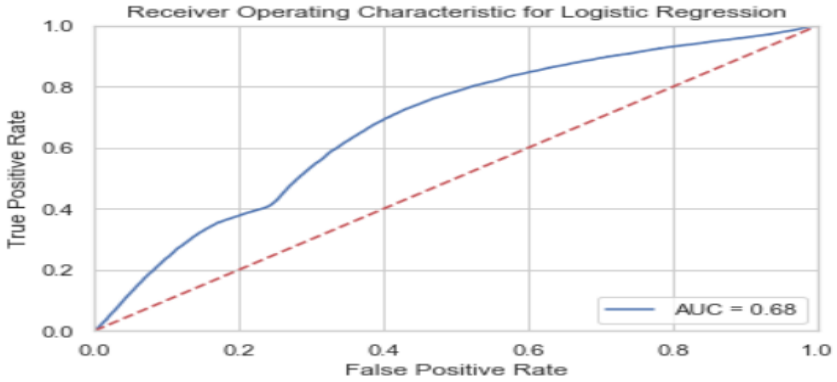
```
print(classification_report(y_test, lr_predict))
lr_accuracy = accuracy_score(y_test, lr_predict)
print("Accuracy of Logistic Regression" + ' : ' + str(lr_accuracy))
```

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.93    100195
           1       0.00      0.00      0.00     14138

    accuracy                           0.88    114333
   macro avg       0.44      0.50      0.47    114333
weighted avg       0.77      0.88      0.82    114333

Accuracy of Logistic Regression : 0.8763436628095126
```

Accuracy of logistic regression is 0.8763436628095126



```
#Plot ROC_AUC for logistic regression
probs = lr.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic for Logistic Regression')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

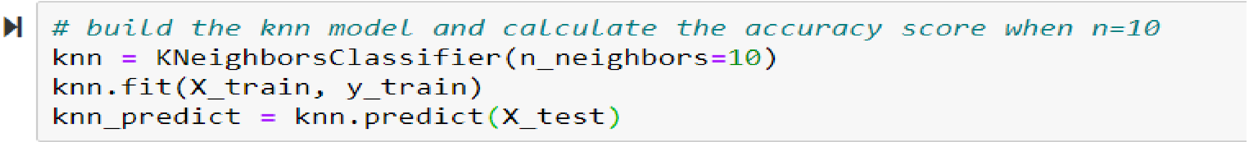


AUC score of logistic regression is 0.68

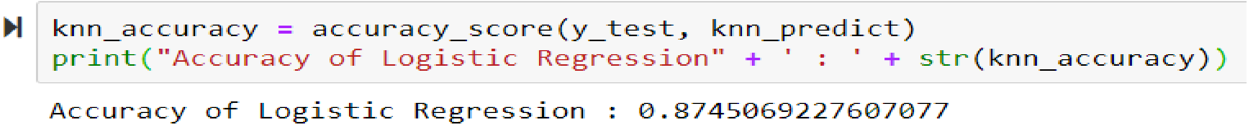
KNN MODEL:

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

In statistics, the k-nearest neighbors algorithm is a non-parametric method proposed by Thomas Cover used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. It is a supervised machine learning algorithm used for both regression and classification problems.

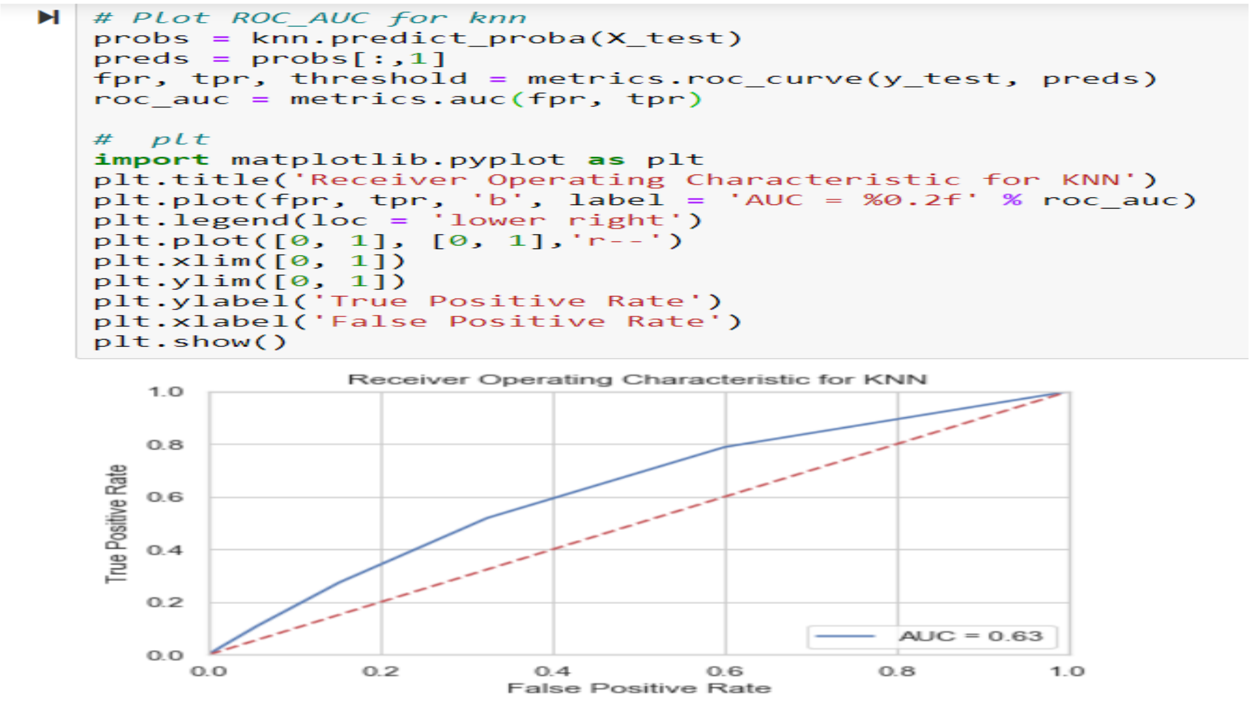


```
# build the knn model and calculate the accuracy score when n=10
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train, y_train)
knn_predict = knn.predict(X_test)
```



```
knn_accuracy = accuracy_score(y_test, knn_predict)
print("Accuracy of Logistic Regression" + ' : ' + str(knn_accuracy))
```

```
Accuracy of Logistic Regression : 0.8745069227607077
```



```
# Plot ROC_AUC for knn
probs = knn.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

#  plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic for KNN')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

AUC score of KNN is 0.63

Conclusion:

Through this data, I have predicted the customers who are willing to take the insurance by using previous data as trained data, implemented logistic regression, random forest, KNN algorithms to understand the design of the data which helps the company to increase its revenue. Our prediction shows that Random forest is a best model with AUC score of 0.78 , which is more accurate, so random forest is the best model to predict the willingness of the customers.

Bibliography:

https://prezi.com/feykni_8hcx_/automobile-insurance-project/

https://www.iproject.com.ng/insurance/assessing-the-role-of-insurance-companies-in-the-economy-of-insurance-industry/index.html

https://afribary.com/works/vehicular-insurance-impacts-on-road-safety-management-nigeria-as-case-study

https://ijhpr.biomedcentral.com/articles/10.1186/s13584-017-0163-2

https://davidcard.berkeley.edu/papers/healthinsur.pdf

https://www.annualreviews.org/doi/full/10.1146/annurev.publhealth.28.021406.144042