K L E F

KONERU LAKSHMAIAH EDUCATION FOUNDATION

(Deemed to be university estd, u/s, 3 of the UGC Act, 1956)

(NAAC Accredited "A" Grade University)

PROJECT REPORT

On

HEALTH INSURANCE ON

CROSS SELL PREDICTION

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

Bachelor of Technology

In

Computer science and Engineering

Under the esteemed guidance of

Mrs. T. SAJANA Asst. Prof.

By

TEJASWI REDDY. K
(180030537)

(DST-FIST Sponsored Department)

K L EDUCATION FOUNDATION

Green Fields, Vaddeswaram, Guntur District-522 502

2020-2021

K L EDUCATION FOUNDATION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DST-FIST Sponsored Department)

CERTIFICATE

This is to certify that this project based lab report entitled **"Health Insurance Cross Sell Prediction"** is a bonafide work done by **TEJASWI REDDY. K (180030537)** in the course **18CS3065S BIG DATA ANALYTICS** in partial fulfillment of the requirements for the award of Degree in Bachelor of Technology in **COMPUTER SCIENCE & ENGNEERING** during the Even Semester of Academic year 2020-2021.

Mrs. T. Sajana Asst. Prof.
Faculty in Charge

Hari Kiram. Vege
Head of the Department

K L EDUCATION FOUNDATION

DEPT OF COMPUTER SCINCE AND ENGINEERING

(DST-FIST Sponsored Department)

DECLARATION

We hereby declare that this project based lab report entitled **"Health Insurance Cross Sell Prediction"** has been prepared by us in the course **18CS3065S BIG DATA ANALYTICS** in partial fulfillment of the requirement for the award of degree bachelor of technology in **COMPUTER SCIENCE & ENGINEERING** during the Even Semester of the academic year 2020-2021. We also declare that this project-based lab report is of our own effort and it has not been submitted to any other university for the award of any degree.

Date:

23.04.2021

Place:

K L University

ACKNOWLEDGEMENT

Our sincere thanks to **Mrs. T. Sajana Asst. Prof.** in the Project for her outstanding support throughout the project for the successful completion of the work.

We express our gratitude to **Dr P. VIDYULLATHA** Course Co-coordinator for **18CS3065S , BIG DATA ANALYTICS** course in the Computer Science and Engineering Department for providing us with adequate planning and support and means by which we can complete this project-based Lab.

We express our gratitude to **Mr. V. HARIKIRAN,** Head of the Department for computer science and Engineering for providing us with adequate facilities, ways and means by which we can complete this project-based Lab.

We would like to place on record the deep sense of gratitude to the honorable Vice Chancellor, K L University for providing the necessary facilities to carry the project-based Lab.

Last but not the least, we thank all Teaching and Non-Teaching Staff of our department and especially our classmates and our friends for their support in the completion of our project-based Lab.

Tejaswi Reddy. K
(180030537)
Name of the student

TABLE OF CONTENTS

ABSTRACT

In this project work, we apply the modern machine learning techniques on the insurance policyholders' data to analyze and predict their behavior to help the insurance companies in modeling their businesses. Here is an insurance company that has provided health insurance to its policyholders. It wants to predict whether the policyholders from the past year will also be interested in the insurance company's vehicle insurance. This project aims to build a model to predict the policyholders' response to vehicle insurance. This is necessary for the insurance company to plan how to reach out to its customers and optimize its business model and revenue. In this model, the key is to maximize recall, and in this way, the insurance company could send advertisements to all possible customers later.

INTRODUCTION

An Insurance company that has provided Health Insurance to its customers now they need to build a model to predict whether the policyholders (customers) from past years will also be interested in Vehicle Insurance provided by the company. 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.

PROBLEM DEFINITION:

Your 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 customers from past year will also be interested in Vehicle Insurance provided by the company.

SCOPE:

We could try boosting models such as, XGBoost, to reduce the over-fitting problems. We can apply our models to time series data and analyze how they perform over the years since the data was limited to policyholders' information from the previous year. By comparing our result to the insurance company's work in the future to see how building a model and executing in a real-life environment has a similarity and difference.

PURPOSE:

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.

PROBLEM AND EXISTING TECHNOLOGY:

Our primary goal is to estimate whether the policyholders from the past year will be interested in the insurance company's vehicle insurance, thus our optimization will focus on how to increase the precision of the models. After data preprocessing, we found there is no missing value, but we found that the original data has a response of interest vs otherwise. This imbalanced data will negatively impact algorithms such as Logistic Regression that optimizes across the entire training set. As a result, after using pure random sampling to get training data and test data set, we use oversampling and under sampling to balance the training data to have a 50/50 split. Finally, we get four data files to be used across our models building and analysis.

PROPOSED SYSTEM:

For solving the problem with the health insurance cross sell prediction, we make use of some of the machine learning techniques such as logistic regression, decision tree and random forest. Logistic regression is a linear model using a sigmoid function for classification, being used to predict binary outcome from a linear combination of predictor variables. A decision tree is a tree-like collection of nodes intended to create a decision on values affiliation to a class or an estimate of a numerical target value. Random forest ensemble model made of many decision trees using bootstrapping, random subsets of features, and average voting to make predictions.

REQUIREMENTS AND ANALYSIS

PLATFORM REQUIREMENTS:

Operating system: WINDOWS 10

Tools: R Studio or

Python Jupyter Notebook

Language: R SPARK or

Python

RAM: 2 GB

Hard-Disk: 6 GB

Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz

MODULE DESCRIPTION:

This module helps us in prediction of health insurance cross sell for the given dataset through the kaggle link- https://www.kaggle.com/anmolkumar/health-insurance-cross-sell-prediction/ . This dataset is analyzed and visualized with the help of some of the machine learning techniques likes logistic regression, decision tree and random forest in R, to determine the cross sell prediction of insurance of vehicles.

DESIGN AND IMPLEMENTATION

PSEUDO CODE (in PYTHON):

```
import pandas as pd

import numpy as np

import matplotlib as mpl

import matplotlib.pyplot as plt

import seaborn as sns

import sklearn

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.tree import DecisionTreeClassifier

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

from sklearn.model_selection import train_test_split

df=pd.read_csv("C:/Users/tejaswi/Downloads/BDA/PROJECT/train.csv")

df.head()

df.info()

pd.isnull(df).sum()

df.nunique()

#ditribution of Response

fig_dims = (5, 5)

fig, ax = plt.subplots()

sns.countplot('Response',

        data = df,

        order = df['Response'].value_counts().index,

        ax = ax)

ax.set(xlabel='Response', ylabel='Count')

plt.show()

#ditribution of Gender,Driving_License,Previously_Insured,Previously_Insured
```

```
fig, axarr = plt.subplots(2, 2, figsize=(10, 10))



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



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



df['Driving_License'].value_counts().head().plot.pie(
    ax=axarr[0][1])
axarr[0][1].set_title("Driving_License", fontsize=18)
fig=plt.figure(figsize=(5, 5))
```

```
sns.countplot(x="Gender", hue="Vehicle_Damage", data=df)

plt.title("Vehicle Damage by Gender")

#ditribution of Age

fig_dims = (15, 8)

fig, ax = plt.subplots(figsize=fig_dims)

sns.countplot('Age',

        data = df,

        ax = ax)

ax.set(xlabel='Age', ylabel='Count')

plt.show()

df.head()

# represent binary variable as 1and 0

df['Gender'].replace(to_replace={'Male':0,'Female':1},

        inplace=True)

df['Vehicle_Damage'].replace(to_replace={'No':0,'Yes':1},

        inplace=True)

df['Vehicle_Age'].replace(to_replace={'< 1 Year':0,'1-2 Year':1,'> 2 Years':2},
```

```
        inplace=True)
df.info()
df.head()
plt.figure(figsize=(10,10))
cor=df.corr()
sns.heatmap(cor,annot=True,cmap=plt.cm.Blues)
plt.show()
df.describe()
df=df.drop(columns=['id'])
y=df.Response
X=df.drop(columns=['Response'])
# 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)
```

#Decision Tree

```
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
dt_predict = dt.predict(X_test)
```

```
print(classification_report(y_test, dt_predict))

dt_accuracy = accuracy_score(y_test, dt_predict)

print("Accuracy of decision tree" + ' : ' + str(dt_accuracy))

# Compute 10-fold cross-validation scores: cv_scores

from sklearn.model_selection import cross_val_score

cv_scores = cross_val_score(dt,X,y,cv=10)

print(cv_scores)

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

# use  GridSearchCV to test all accuracy, and choose the combinations of the highest accuracy

from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth': np.arange(3, 10),

        'criterion' : ['gini','entropy'],

        'max_leaf_nodes': [5,10,50,100],

        'min_samples_split': [2, 5, 10, 20]}

grid_tree = GridSearchCV(DecisionTreeClassifier(), param_grid, cv = 5, scoring= 'accuracy')

grid_tree.fit(X_train, y_train)

np.abs(grid_tree.best_score_)
```

```
#test the accuracy of all the combination of the parameters, then output the highest parameter.
print(grid_tree.best_estimator_)
# use the best performance combinations  to test
Tree = DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                              max_depth=9, max_features=None, max_leaf_nodes=50,
                              min_impurity_decrease=0.0, min_impurity_split=None,
                              min_samples_leaf=1, min_samples_split=2,
                              min_weight_fraction_leaf=0.0, presort='deprecated',
                              random_state=None, splitter='best')
Tree.fit(X_train, y_train)
predictions = Tree.predict(X_test)
accuracy_score(y_true = y_test, y_pred = predictions)
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = dt.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 Decision Tree')

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()
```

#Random Forest

```
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 rf" + ' : ' + str(rf_accuracy))

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

print(cv_scores)

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

# 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()
```

#Logistic Regression

```
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))

# 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)
```

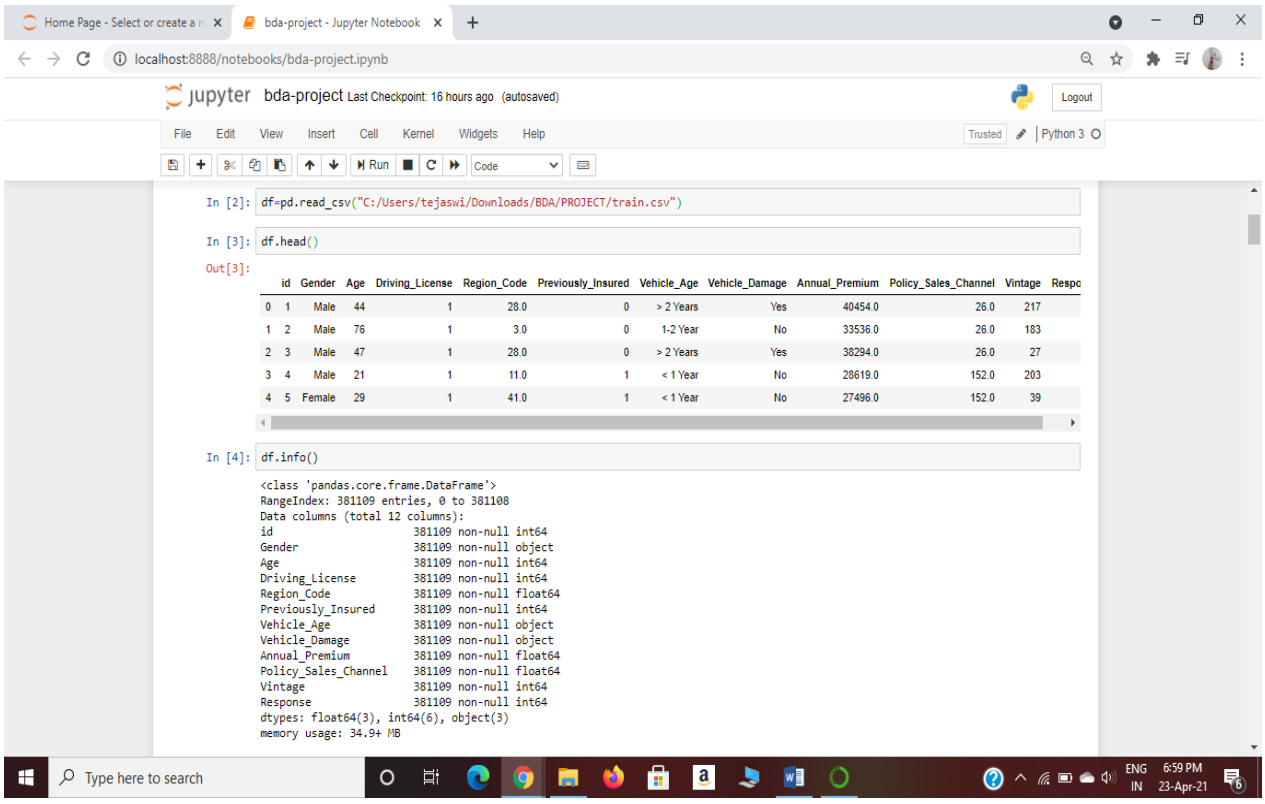
```
import matplotlib.pyplot as plt

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()
```

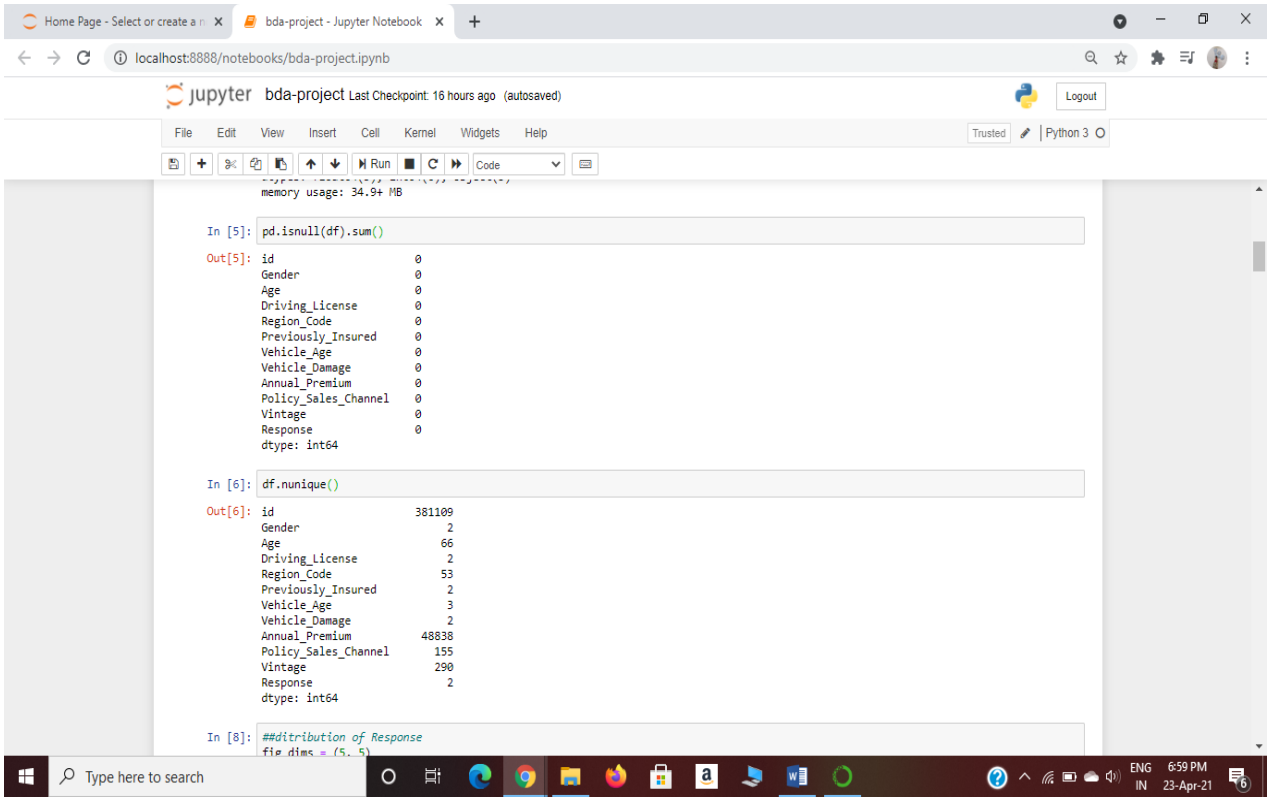
#KNN

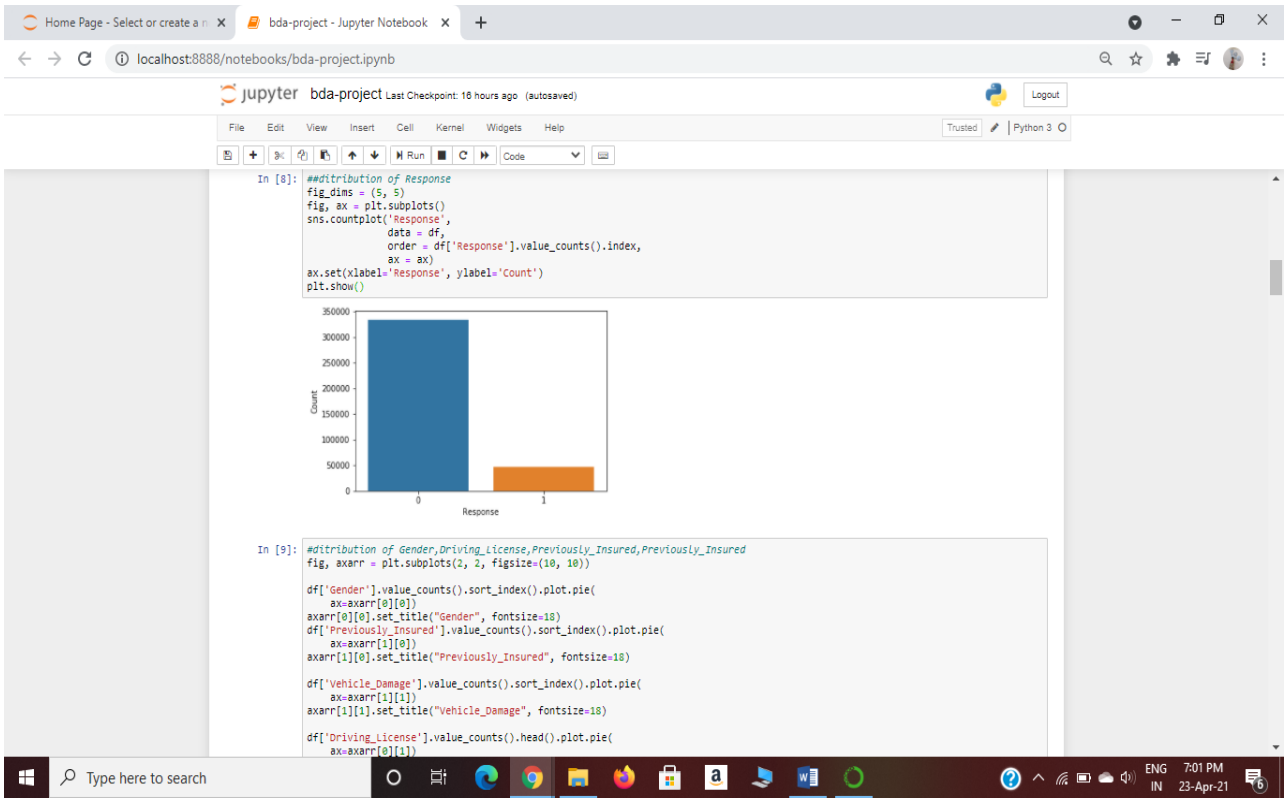
```
# build the knn model and calculate the accuracy score when n=10

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))
```

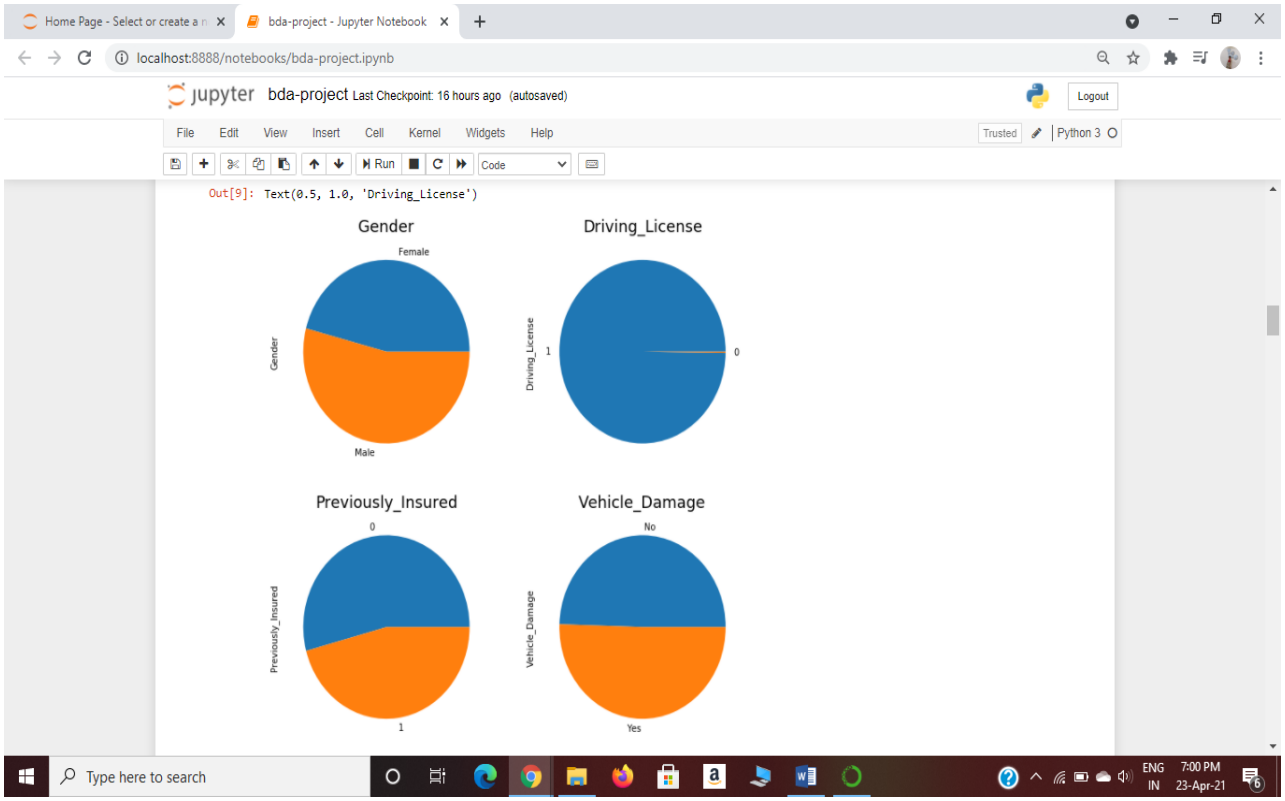
```
# Plot ROC_AUC for knn

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()
```

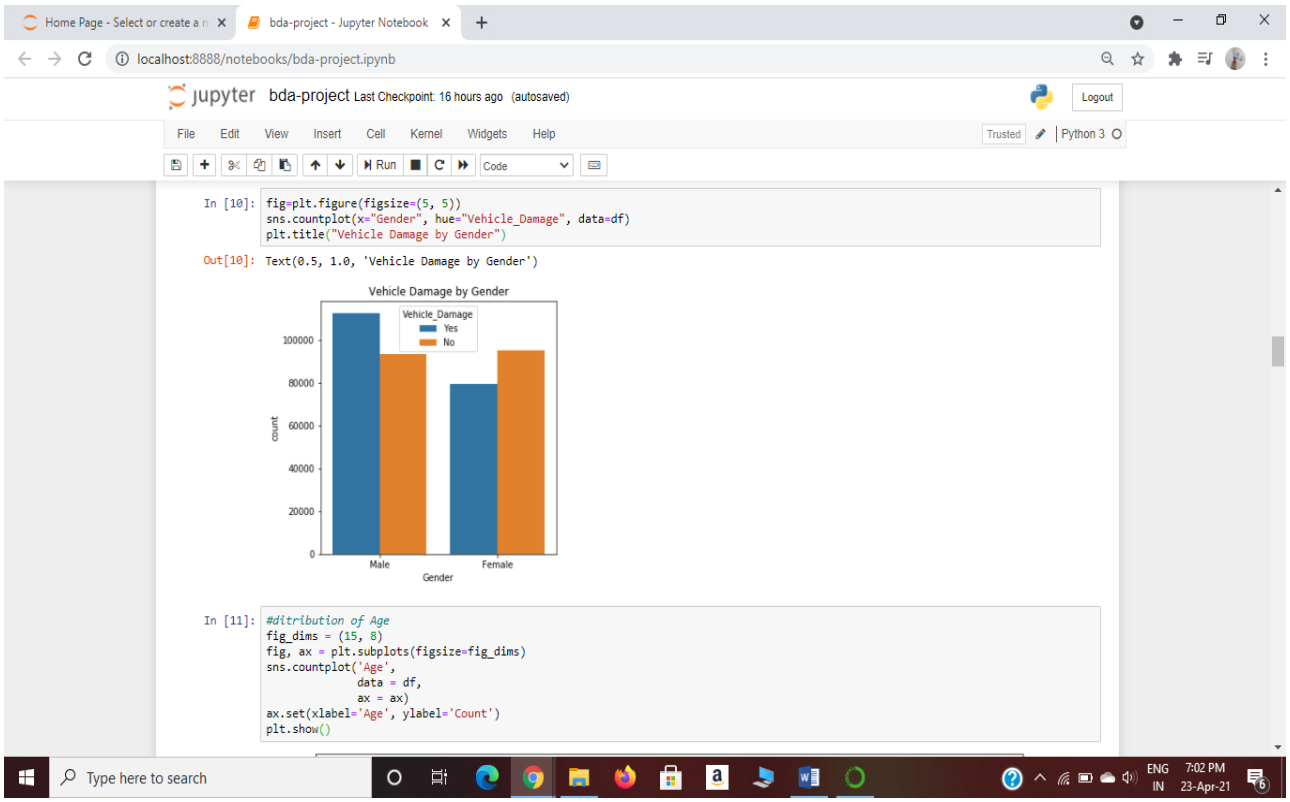
SCREENSHOTS

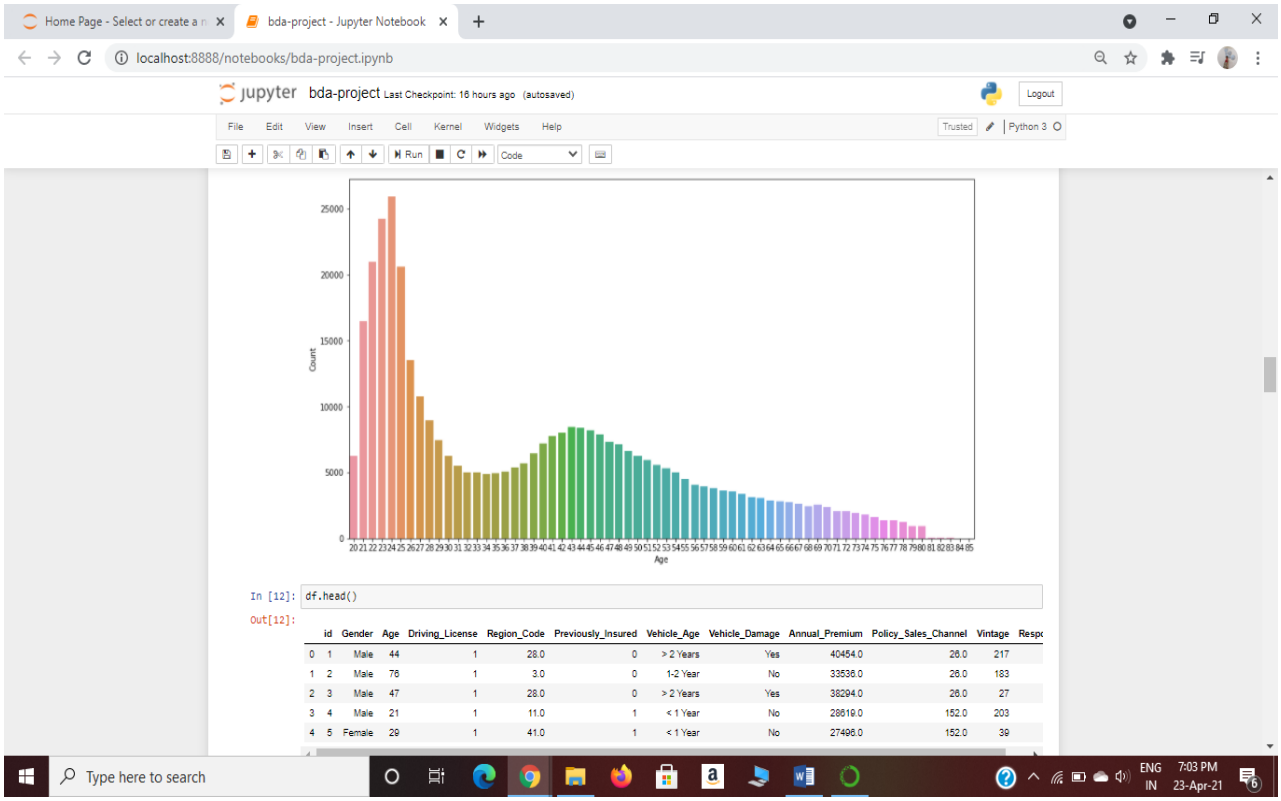


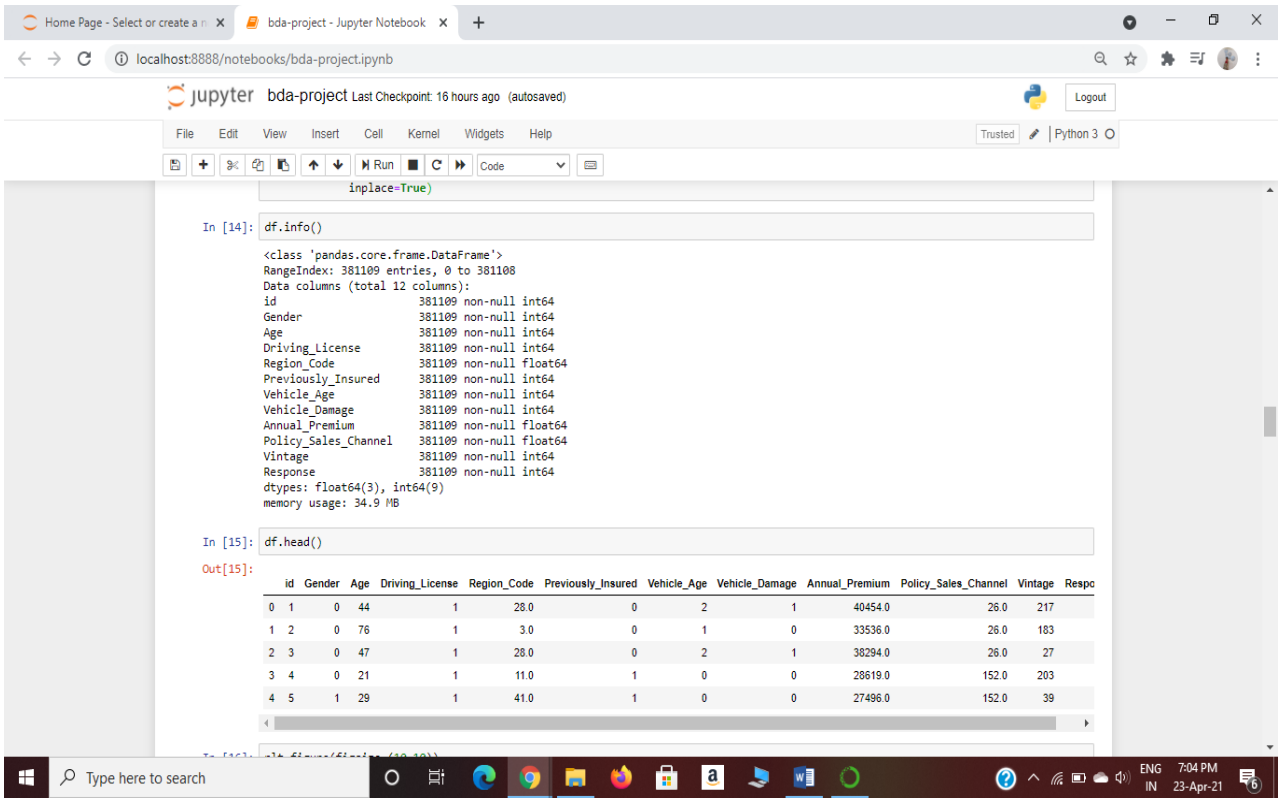

Home Page - Select or create a n
bda-project - Jupyter Notebook
localhost:8888/notebooks/bda-project.ipynb
jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
memory usage: 34.9+ MB
In [5]: pd.isnull(df).sum()
Out[5]: 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
In [6]: df.nunique()
Out[6]: id 381109
Gender 2
Age 66
Driving_License 2
Region_Code 53
Previously_Insured 2
Vehicle_Age 3
Vehicle_Damage 2
Annual_Premium 48838
Policy_Sales_Channel 155
Vintage 290
Response 2
dtype: int64
In [8]: ##ditribution of Response
fig_dims = (5, 5)
Type here to search
ENG IN 6:59 PM 23-Apr-21


Home Page - Select or create a n
bda-project - Jupyter Notebook
localhost:8888/notebooks/bda-project.ipynb
jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
In [8]: ##ditribution of Response
fig_dims = (5, 5)
fig, ax = plt.subplots()
sns.countplot('Response',
data = df,
order = df['Response'].value_counts().index,
ax = ax)
ax.set(xlabel='Response', ylabel='Count')
plt.show()
350000
300000
250000
200000
150000
100000
50000
0
Count
0
1
Response
In [9]: #ditribution of Gender,Driving_License,Previously_Insured,Previously_Insured
fig, axarr = plt.subplots(2, 2, figsize=(10, 10))
df['Gender'].value_counts().sort_index().plot.pie(
ax=axarr[0][0])
axarr[0][0].set_title("Gender", fontsize=18)
df['Previously_Insured'].value_counts().sort_index().plot.pie(
ax=axarr[1][0])
axarr[1][0].set_title("Previously_Insured", fontsize=18)
df['Vehicle_Damage'].value_counts().sort_index().plot.pie(
ax=axarr[1][1])
axarr[1][1].set_title("Vehicle_Damage", fontsize=18)
df['Driving_License'].value_counts().head().plot.pie(
ax=axarr[0][1])
Type here to search
ENG IN 7:01 PM 23-Apr-21


Home Page - Select or create a n
bda-project - Jupyter Notebook
localhost:8888/notebooks/bda-project.ipynb
jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
Out[9]: Text(0.5, 1.0, 'Driving_License')
Gender
Female
Male
Gender
Driving_License
1
0
Driving_License
Previously_Insured
0
1
Previously_Insured
Vehicle_Damage
No
Yes
Vehicle_Damage
Type here to search
ENG IN
7:00 PM 23-Apr-21


Home Page - Select or create a n
bda-project - Jupyter Notebook
localhost:8888/notebooks/bda-project.ipynb
jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
In [10]:
fig=plt.figure(figsize=(5, 5))
sns.countplot(x="Gender", hue="Vehicle_Damage", data=df)
plt.title("Vehicle Damage by Gender")
Out[10]: Text(0.5, 1.0, 'Vehicle Damage by Gender')
Vehicle Damage by Gender
Vehicle_Damage
Yes
No
100000
80000
60000
40000
20000
0
count
Male
Female
Gender
In [11]:
#ditribution of Age
fig_dims = (15, 8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.countplot('Age',
data = df,
ax = ax)
ax.set(xlabel='Age', ylabel='Count')
plt.show()
Type here to search
ENG IN
7:02 PM 23-Apr-21


Home Page - Select or create a n
bda-project - Jupyter Notebook
localhost:8888/notebooks/bda-project.ipynb
jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
Count
Age
In [12]: df.head()
Out[12]:
id Gender Age Driving_License Region_Code Previously_Insured Vehicle_Age Vehicle_Damage Annual_Premium Policy_Sales_Channel Vintage Respo
0 1 Male 44 1 28.0 0 > 2 Years Yes 40454.0 26.0 217
1 2 Male 76 1 3.0 0 1-2 Year No 33536.0 26.0 183
2 3 Male 47 1 28.0 0 > 2 Years Yes 38294.0 26.0 27
3 4 Male 21 1 11.0 1 < 1 Year No 28619.0 152.0 203
4 5 Female 29 1 41.0 1 < 1 Year No 27496.0 152.0 39
Type here to search
ENG IN 7:03 PM 23-Apr-21


Home Page - Select or create a n
bda-project - Jupyter Notebook
localhost:8888/notebooks/bda-project.ipynb
jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3
Run Code
inplace=True)
In [14]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 381109 entries, 0 to 381108
Data columns (total 12 columns):
id 381109 non-null int64
Gender 381109 non-null int64
Age 381109 non-null int64
Driving_License 381109 non-null int64
Region_Code 381109 non-null float64
Previously_Insured 381109 non-null int64
Vehicle_Age 381109 non-null int64
Vehicle_Damage 381109 non-null int64
Annual_Premium 381109 non-null float64
Policy_Sales_Channel 381109 non-null float64
Vintage 381109 non-null int64
Response 381109 non-null int64
dtypes: float64(3), int64(9)
memory usage: 34.9 MB
In [15]: df.head()
Out[15]:
id Gender Age Driving_License Region_Code Previously_Insured Vehicle_Age Vehicle_Damage Annual_Premium Policy_Sales_Channel Vintage Respo
0 1 0 44 1 28.0 0 2 1 40454.0 26.0 217
1 2 0 76 1 3.0 0 1 0 33536.0 26.0 183
2 3 0 47 1 28.0 0 2 1 38294.0 26.0 27
3 4 0 21 1 11.0 1 0 0 28619.0 152.0 203
4 5 1 29 1 41.0 1 0 0 27496.0 152.0 39
Type here to search
ENG IN 7:04 PM 23-Apr-21

Home Page - Select or create a n | bda-project - Jupyter Notebook

localhost:8888/notebooks/bda-project.ipynb

jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)

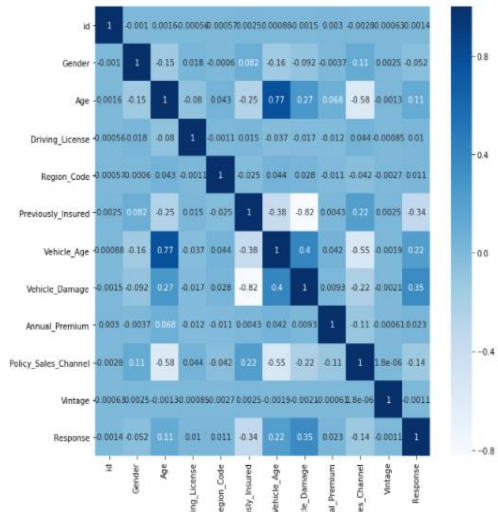
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [16]:
```
plt.figure(figsize=(10,10))
cor=df.corr()
sns.heatmap(cor,annot=True,cmap=plt.cm.Blues)
plt.show()
```



Type here to search

ENG IN 7:04 PM 23-Apr-21

Home Page - Select or create a n | bda-project - Jupyter Notebook

localhost:8888/notebooks/bda-project.ipynb

jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [22]:
```
print(classification_report(y_test, dt_predict))
dt_accuracy = accuracy_score(y_test, dt_predict)
print("Accuracy of decision tree" + ' : ' + str(dt_accuracy))
```

```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90    100195
           1       0.29      0.31      0.30     14138

    accuracy                           0.82    114333
   macro avg       0.60      0.60      0.60    114333
weighted avg       0.83      0.82      0.82    114333

Accuracy of decision tree : 0.8232706217802385
```

In [23]:
```
 # Compute 10-fold cross-validation scores: cv_scores
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(dt,X,y,cv=10)

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

```
[0.82453885 0.8210228  0.8228333  0.82422398 0.82070793 0.82348928
 0.82401406 0.82684789 0.8241715  0.82277617]
Average 10-Fold CV Score: 0.8234625771461632
```

In [24]:
```
# use  GridSearchCV to test all accuracy, and choose the combinations of the highest accuracy
from sklearn.model_selection import GridSearchCV
param_grid = {'max_depth': np.arange(3, 10),
              'criterion' : ['gini','entropy'],
              'max_leaf_nodes': [5,10,50,100],
              'min_samples_split': [2, 5, 10, 20]}
grid_tree = GridSearchCV(DecisionTreeClassifier(), param_grid, cv = 5, scoring= 'accuracy')
grid_tree.fit(X_train, y_train)
np.abs(grid_tree.best_score_)
#test the accuracy of all the combination of the parameters, then output the highest parameter.
print(grid_tree.best_estimator_)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=9,
                       max_features=None, max_leaf_nodes=50,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
```

Type here to search

ENG IN 7:05 PM 23-Apr-21

Home Page - Select or create a n | bda-project - Jupyter Notebook

localhost:8888/notebooks/bda-project.ipynb

jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)

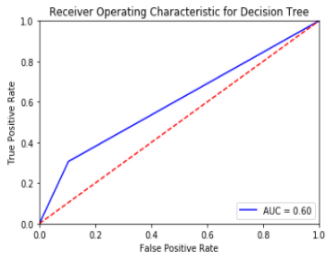
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [28]:
```
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = dt.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 Decision Tree')
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()
```



Type here to search

ENG IN 7:06 PM 23-Apr-21

Home Page - Select or create a n | bda-project - Jupyter Notebook

localhost:8888/notebooks/bda-project.ipynb

jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [29]:
```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf_Predict = rf.predict(X_test)
```

```
C:\Users\tejaswi\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estimators w
ill change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

In [30]:
```
print(classification_report(y_test, rf_Predict))
rf_accuracy = accuracy_score(y_test, rf_Predict)
print("Accuracy of rf" + ' : ' + str(rf_accuracy))
```

```
              precision    recall  f1-score   support

           0       0.89      0.97      0.93    100195
           1       0.36      0.14      0.20     14138

    accuracy                           0.86    114333
   macro avg       0.63      0.55      0.56    114333
weighted avg       0.82      0.86      0.84    114333

Accuracy of rf : 0.8636176781856507
```

In [31]:
```
cv_scores = cross_val_score(rf,X,y,cv=10)

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

```
[0.86250689 0.86290047 0.860434   0.86434363 0.86387132 0.86208706
 0.86179843 0.86290047 0.86166724 0.86525846]
Average 10-Fold CV Score: 0.8627767974035381
```

In [32]:
```
# Plot ROC_AUC for random forest
probs = rf.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
```

Type here to search

ENG IN 7:07 PM 23-Apr-21

Home Page - Select or create a n | bda-project - Jupyter Notebook

localhost:8888/notebooks/bda-project.ipynb

jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)

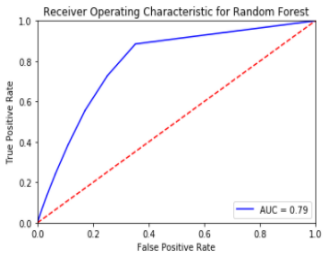
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [32]:
```
# 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()
```



Type here to search

ENG IN 7:07 PM 23-Apr-21

Home Page - Select or create a n | bda-project - Jupyter Notebook

localhost:8888/notebooks/bda-project.ipynb

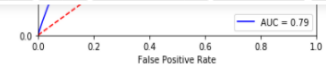
jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code



In [33]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr_predict = lr.predict(X_test)
```

```
C:\Users\tejaswi\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be change
d to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

In [34]:
```
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
```

```
C:\Users\tejaswi\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1437: UndefinedMetricWarning: Precision and F-sc
ore are ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [35]:
```
# Plot ROC_AUC for logistic regression
probs = lr.predict_proba(X_test)
preds = probs[:,1]
```

Type here to search

ENG IN 7:08 PM 23-Apr-21

Home Page - Select or create a n | bda-project - Jupyter Notebook

localhost:8888/notebooks/bda-project.ipynb

jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)

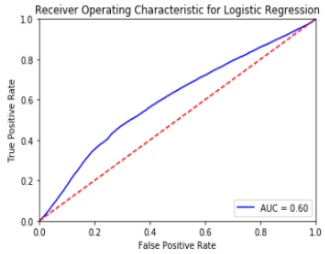
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [35]:
```
# 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()
```



Type here to search

ENG IN 7:08 PM 23-Apr-21

Home Page - Select or create a n | bda-project - Jupyter Notebook

localhost:8888/notebooks/bda-project.ipynb

jupyter bda-project Last Checkpoint: 16 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

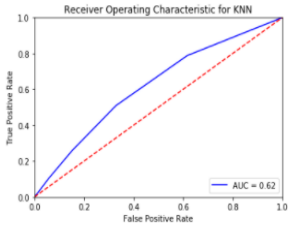
Run | Code

```
knn_predict = knn.predict(X_test)
```

In [37]:
```
knn_accuracy = accuracy_score(y_test, knn_predict)
print("Accuracy of Logistic Regression" + ' : ' + str(knn_accuracy))
```

```
Accuracy of Logistic Regression : 0.874253277706349
```

In [38]:
```
# 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()
```



Type here to search

ENG IN 7:09 PM 23-Apr-21

CONCLUSION

Through this project we maximize the recall score which is used to measure the number of interested customers. With the help of this method, the insurance company can plan for the most efficient way of approaching the future vehicle insurance policyholders. In the given dataset, we have high imbalance of data, so, to avoid the over-fitting issues, we used the under sampling data, for which, we use some of the machine learning techniques like logistic regression, decision tree and random forest.

REFERENCES

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

https://github.com/anjalysam/Health-Insurance-Cross-Sell-Prediction

https://gutentagworld.wordpress.com/2020/12/13/health-insurance-cross-sell-prediction/