Project Report

|  |  |
| --- | --- |
| Course Name (NICF) | *Bundle Program-Artificial Intelligence* |
| Product Name (Marketing & Sales) | *Bundle Program-Artificial Intelligence* |
| Module Name (NICF) | NICF-Applied Machine Learning (SF) |
| Product Name (Marketing & Sales) | NICF-Applied Machine Learning (SF) |

|  |  |  |  |
| --- | --- | --- | --- |
| Student name | | Assessor name | |
| Saminathan Renganayagi | | Rajendra kissan | |
| Date issued | Completion date | | Submitted on |
|  |  | |  |
|  | |  | |
| Project title | Prediction model using Scikit Learn | | |

|  |
| --- |
| Learner declaration |
| I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.  No need sign  Student signature: Date: date only |

The project aims to demonstrate the exploration of data, specifically customer data, and AI’s capability to classify/predict the value of labels, specifically bike buyer and average monthly spending, using supervised learning (training and testing models using features and known labels).

To achieve this, we are supposed to write the following scripts:

1. **Script 1.** Script does the following:

* Reads three csv files – customer, bike buyer and average monthly spending, and merges them into one data frame.
* Cleans the data by detecting and removing duplicates and missing values.
* Identifies and assigns the age group of each customer.
* Explores the data by displaying statistics such as minimum, maximum, mean, median, standard deviation and distribution.
* Explores the date by plotting features using boxplot and bar chart.

1. **Script 2**. Script does the following:

* Reads three csv files – customer, bike buyer and average monthly spending, and merges them into one data frame.
* Cleans the data by detecting and removing duplicates and missing values.
* Identifies and assigns the age group of each customer.
* Encodes and scales the data.
* Splits the data into training and testing data.
* Creates and trains a logistic regression model using the training data. Specifically, the model is supposed to classify the customer as a bike buyer.
* Tests the model using the test data and different thresholds.
* Evaluates the results of testing the model through metrics and ROC curve.
* As part of a challenge, Script 2 classifies the customers in another test data with no known label, as bike buyers. Save predicted bike buyers in a csv file.

1. **Script 3.** Script does the following:

* Reads three csv files – customer, bike buyer and average monthly spending, and merges them into one data frame.
* Cleans the data by detecting and removing duplicates and missing values.
* Identifies and assigns the age group of each customer
* Encodes and scales the data.
* Splits the data into training and testing data.
* Creates and trains a linear regression model using the training data. Specifically, the model is supposed to predict the customer’s average monthly spending.
* Tests the model using the test data.
* Evaluates the results of testing the model through metrics, histogram, QQ plot and reg plot.
* As part of a challenge, Script 3 predicts average monthly spending of customers in another test data with no known label. Save predicted average monthly spending in a csv file.

The scripts mentioned above will be coded and run using Azure Data Studio. The code will be written in Python language.

**1 . Project TechnicalEenvironment(describe the Architecture with tools used**

The scripts mentioned above will be coded and run using Azure Data Studio. The code will be written in Python language.

Architecture for the activities to demonstrate exploration of customer data

Bike Buyer.csv

Ave Mo Spending.csv

ustomer.csv

C

|  |
| --- |
| Script1   * Reads three csv files – customer, bike buyer and average monthly spending, and merges them into one data frame. * Cleans the data by detecting and removing duplicates and missing values. * Identifies and assigns the age group of each customer. * Explores the data by displaying statistics such as minimum, maximum, mean, median, standard deviation and distribution. * Explores the date by plotting features using boxplot and bar chart. |

Display statistics, boxplot, bar chart

2. Architecture for the activities to demonstrate AI’s capability to classify customer as bike buyer.

Bike Buyer.csv

Ave Mo Spending.csv

Ave Mo Spending.csv

|  |
| --- |
| Script 2   * Reads three csv files – customer, bike buyer and average monthly spending, and merges them into one data frame. * Cleans the data by detecting and removing duplicates and missing values. * Identifies and assigns the age group of each customer. * Encodes and scales the data. * Splits the data into training and testing data. * Creates and trains a logistic regression model using the training data. Specifically, the model is supposed to classify the customer as a bike buyer. * As part of a challenge, Script 2 classifies the customers in another test data with no known label, as bike buyers. Save predicted bike buyers in a csv file. |

Display metrics, ROC curve

IIB.

Architecture for the activities to demonstrate AI’s capability to classify customer as bike buyer. (Challenge)

Bike Buyer.csv

stomer.csv

Tes data no Bike Buyer.csv

Ave Mo Spending.csv

|  |
| --- |
| script 3   * Reads three csv files – customer, bike buyer and average monthly spending, and merges them into one data frame. * Cleans the data by detecting and removing duplicates and missing values. * Identifies and assigns the age group of each customer * Encodes and scales the data. * Splits the data into training and testing data. * Creates and trains a linear regression model using the training data. Specifically, the model is supposed to predict the customer’s average monthly spending. * Tests the model using the test data. * Evaluates the results of testing the model through metrics, histogram, QQ plot and reg plot.   . |

s

PredictedBike Buyer.csv



Architecture for the activities to demonstrate AI’s capability to predict customer’s average monthly spending.

Customer.csv

Bike Buyer.csv

Ave Mo Spending.csv

Script 4

|  |
| --- |
| Script 4   * Reads three csv files – customer, bike buyer and average monthly spending, and merges them into one data frame. * Cleans the data by detecting and removing duplicates and missing values. * Identifies and assigns the age group of each customer * Encodes and scales the data. * Splits the data into training and testing data. * Creates and trains a linear regression model using the training data. Specifically, the model is supposed to predict the customer’s average monthly spending. * Tests the model using the test data. * Evaluates the results of testing the model through metrics, histogram, QQ plot and reg plot. |

Display metrics, histogram,QQ plot, reg plot

IIIB.

Architecture for the activities to demonstrate AI’s capability to predict customer’s average monthly spending. (Challenge)

Test data no Ave Mo Spending.csv

Ave Mo Spending.csv

Customer.csv

Bike Buyer.csv

Script5

|  |
| --- |
| script 5   * Reads three csv files – customer, bike buyer and average monthly spending, and merges them into one data frame. * Cleans the data by detecting and removing duplicates and missing values. * Identifies and assigns the age group of each customer * Encodes and scales the data. * Splits the data into training and testing data. * Creates and trains a linear regression model using the training data. Specifically, the model is supposed to predict the customer’s average monthly spending. * As part of a challenge, Script 3 predicts average monthly spending of customers in another test data with no known label. Save predicted average monthly spending in a csv file. |

Predicted Ave Mo Spending.csv

Predicted Ave Mo Spending.csv

3.Design the Model: (Explain the training model (Classification and Regression which you are designing

1. Script 1 – clean and explore customer data
2. Import required packages.
3. Read customer csv file, bike buyer csv file, and average monthly spending csv file.
4. Merge the three csv files into one data frame.
5. Drop duplicate rows by customer id and keep the last row.
6. Detect and drop columns with null values.
7. Check imbalance by bike buyer.
8. Draw a box plot of occupation against yearly income.
9. Compute the age of each customer using birthdate as the start date and 01 Jan 1998 as end date.
10. Create a new column age group and assign each customer the following age groups – ‘under 25’, ‘between 24 and 45’, ‘between 45 and 55’, ‘over 55’ – according to their age.
11. Draw a bar plot of age group against average monthly spending. Specify gender as hue.
12. Draw box plots for each of the following:
    1. Marital status against average monthly spending.
    2. Number of cars owned against average monthly spending.
    3. Gender against average monthly spending.
    4. Number of children at home against average monthly spending.
13. Draw box plots for bike buyer against yearly income, and bike buyer against number of cars owned.
14. Draw cat plots for the each of the following:
    1. Occupation by bike buyer.
    2. Gender by bike buyer.
    3. Marital Status by bike buyer.
15. Script 2 – clean customer data; build, train and test logistic regression model to classify customers as bike buyers; evaluate model.
16. Import required packages.
17. Read customer csv file, bike buyer csv file, and average monthly spending csv file.
18. Merge the three csv files into one data frame.
19. Drop duplicate rows by customer id and keep the last row.
20. Detect and drop columns with null values.
21. Check imbalance by bike buyer.
22. Compute the age of each customer using birthdate as the start date and 01 Jan 1998 as end date.
23. Create a new column age group and assign each customer the following age groups – ‘under 25’, ‘between 24 and 45’, ‘between 45 and 55’, ‘over 55’ – according to their age.
24. Drop columns such as customer id, first name, etc. that are unlikely to influence bike buyer.
25. Encode categorical columns such as country region name, education, etc.
26. Copy bike buyer (the label) in an array.
27. Copy the remaining features in another array.
28. Scale the features.
29. Split the bike buyer array and features array into training and testing arrays. Split them using a 70/30 ratio.
30. Create a logistic regression model.
31. Train the model using the training array.
32. Using the model and features testing array, predict the probabilities of bike buyer 0 and bike buyer 1 for each customer. Copy the probabilities in an array.
33. For each customer, use a threshold to predict if bike buyer should be 1 or 0. Copy the predicted bike buyers in an array.
34. Use the bike buyer test array from (14) and the probabilities array from (17) to determine AUC. Display AUC.
35. Use the bike buyer test array from (14) and the predicted bike buyer array from (18) to determine accuracy, the confusion matrix, and metrics such as precision, recall, etc. Display confusion matrix, and metrics.
36. Draw the ROC curve using the bike buyer test array from (14) and the probabilities array from (17).
37. Repeat steps (18) and (20) using different thresholds to see changes in accuracy, the confusion matrix and metrics.

Challenge

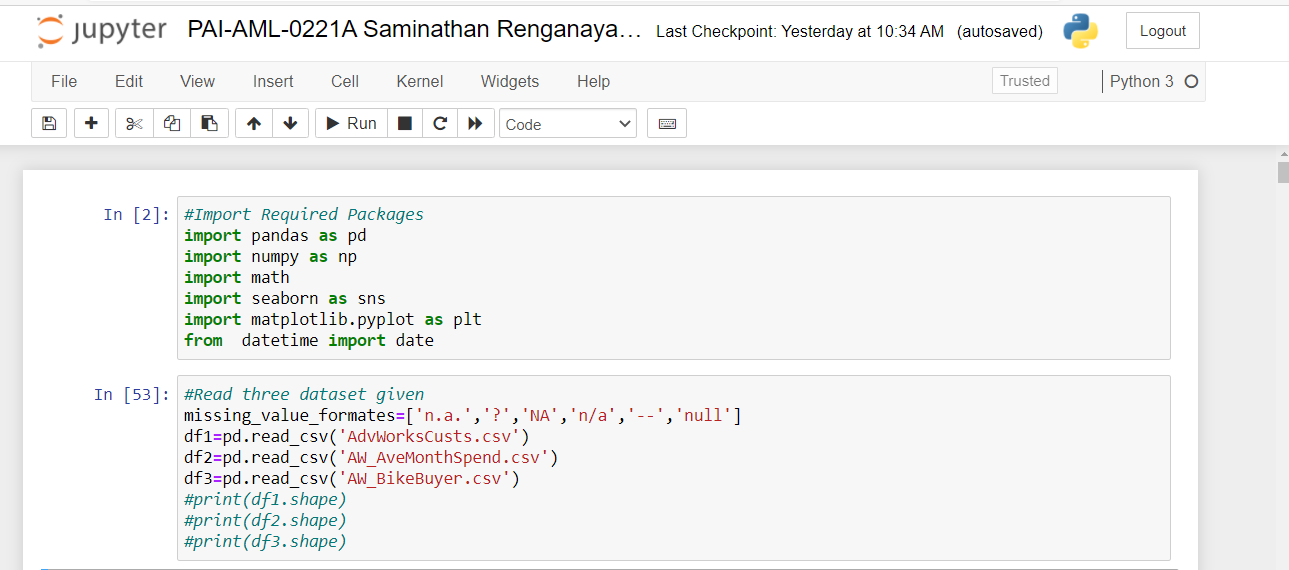
1. Read customer csv for challenge into a data frame.
2. Drop duplicate rows by customer id and keep the last row.
3. Detect and drop columns with null values.
4. Compute the age of each customer using birthdate as the start date and 01 Jan 1998 as end date.
5. Create a new column age group and assign each customer the following age groups – ‘under 25’, ‘between 24 and 45’, ‘between 45 and 55’, ‘over 55’ – according to their age.
6. Drop columns such as customer id, first name, etc. that are unlikely to influence bike buyer.
7. Encode categorical columns such as country region name, education, etc.
8. Scale the features.
9. Using the model and cleaned customer data for the challenge, classify each customer as a bike buyer or not.
10. Save the predicted bike buyers in a csv file.
11. Script 3 – clean customer data; build, train and test linear regression model to predict the average monthly spending; evaluate model.
12. Import required packages.
13. Read customer csv file, bike buyer csv file, and average monthly spending csv file.
14. Merge the three csv files into one data frame.
15. Drop duplicate rows by customer id and keep the last row.
16. Detect and drop columns with null values.
17. Check imbalance by bike buyer.
18. Compute the age of each customer using birthdate as the start date and 01 Jan 1998 as end date.
19. Create a new column age group and assign each customer the following age groups – ‘under 25’, ‘between 24 and 45’, ‘between 45 and 55’, ‘over 55’ – according to their age.
20. Drop columns such as customer id, first name, etc. that are unlikely to influence bike buyer.
21. Encode categorical columns such as country region name, education, etc.
22. Copy average monthly spending (the label) in an array.
23. Copy the remaining features in another array.
24. Scale the features.
25. Split the average monthly spending array and features array into training and testing arrays. Split them using a 70/30 ratio.
26. Create a linear regression model.
27. Train the model using the training array.
28. Using the model and features testing array, predict the average monthly spending of each customer. Copy the predicted average monthly spending in an array.
29. Use the average monthly spending test array and the predicted average monthly spending array to determine metrics such as MSE, R2, etc. Display metrics.
30. Compute residuals using the average monthly spending test array and the predicted average monthly spending array.
31. Draw a histogram of the residuals. Draw QQ plot and reg plot using the residuals and the average monthly spending test array.

Challenge

1. Read customer csv for challenge into a data frame.
2. Drop duplicate rows by customer id and keep the last row.
3. Detect and drop columns with null values.
4. Compute the age of each customer using birthdate as the start date and 01 Jan 1998 as end date.
5. Create a new column age group and assign each customer the following age groups – ‘under 25’, ‘between 24 and 45’, ‘between 45 and 55’, ‘over 55’ – according to their age.
6. Drop columns such as customer id, first name, etc. that are unlikely to influence bike buyer.
7. Encode categorical columns such as country region name, education, etc.
8. Scale the features.
9. Using the model and cleaned customer data for the challenge, predict the average monthly spending of each customer.
10. Save the predicted average monthly spending in a csv file.

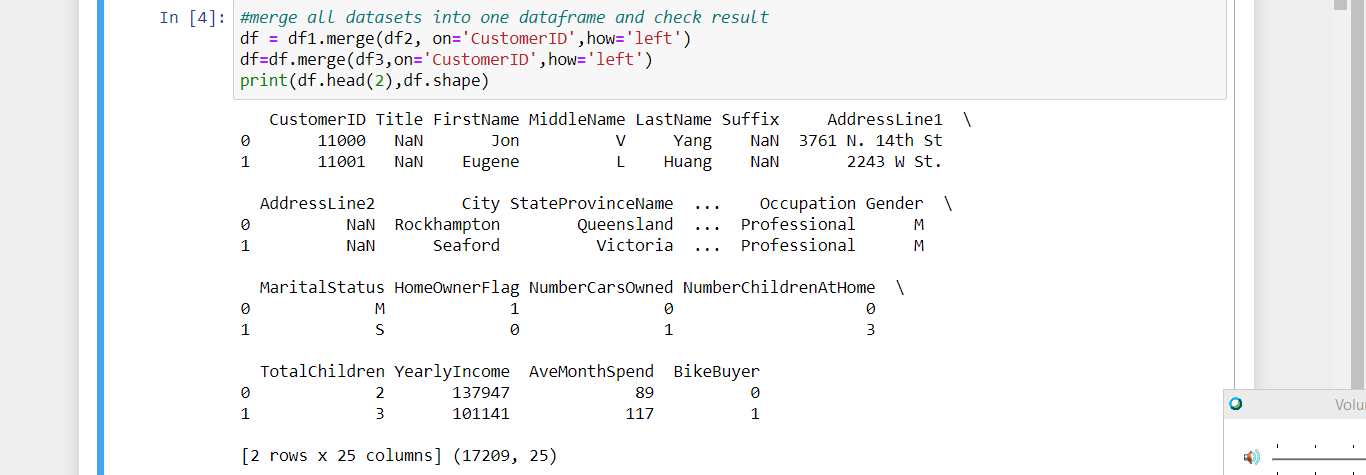
Activity 1

Task1: Import Required packages and read three data given

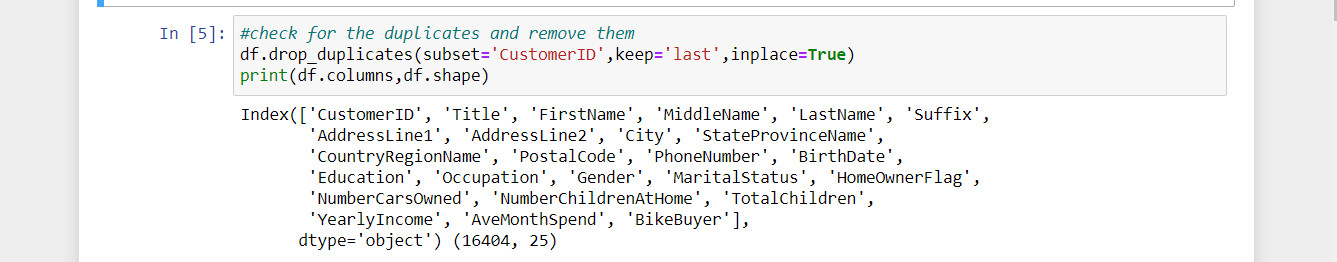


Setting up the EDA Model

Task 2: Merge all Datasets into one Data frame and check the result

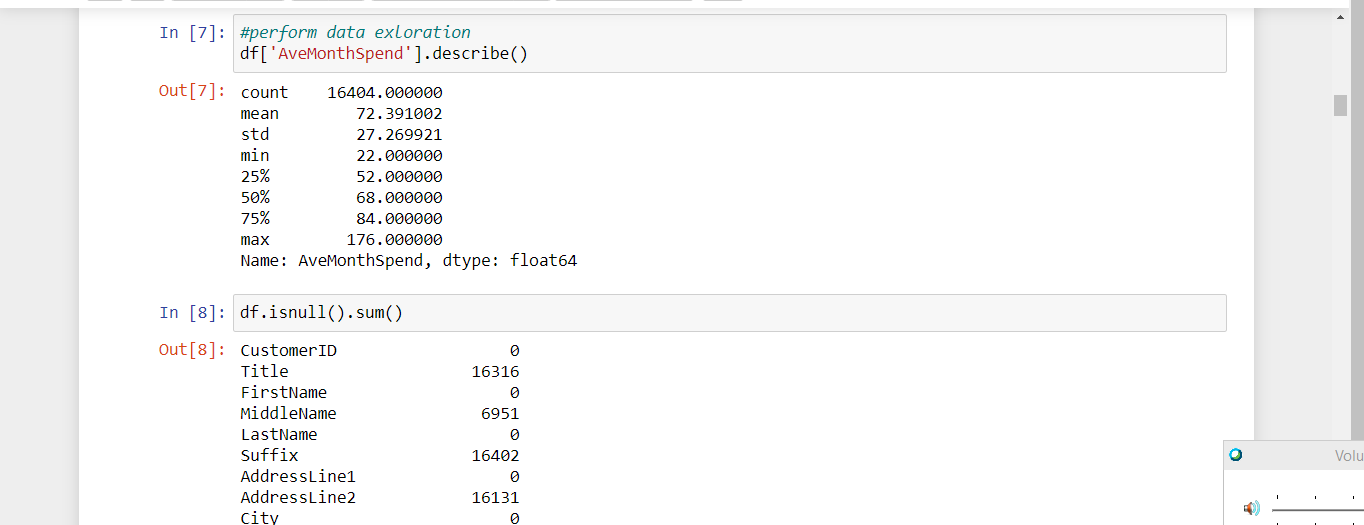
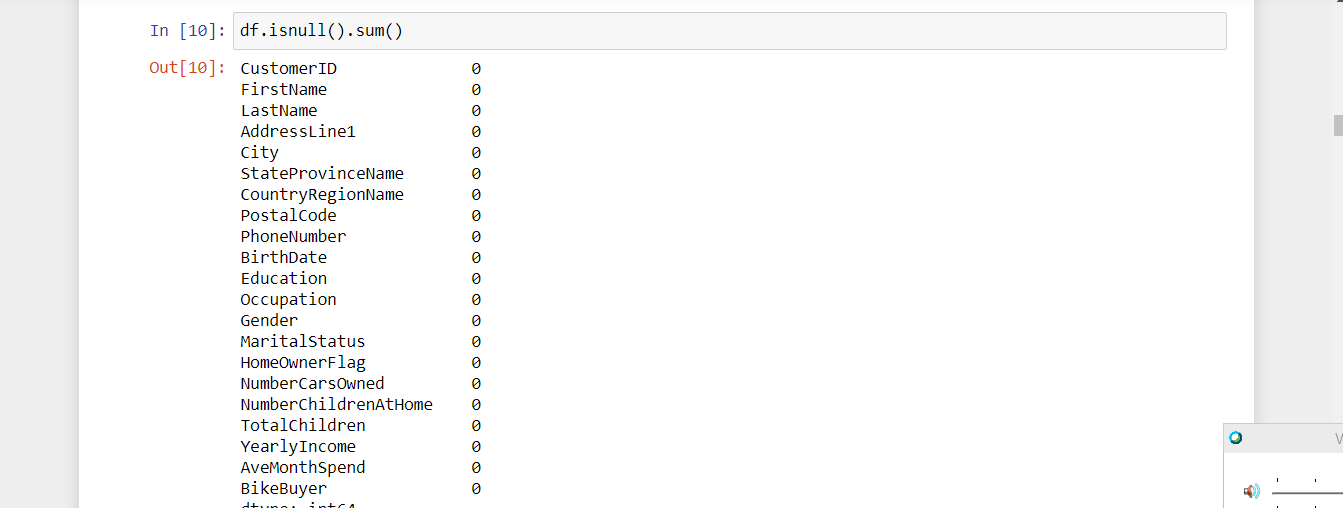


Task3: check the duplicates and remove them



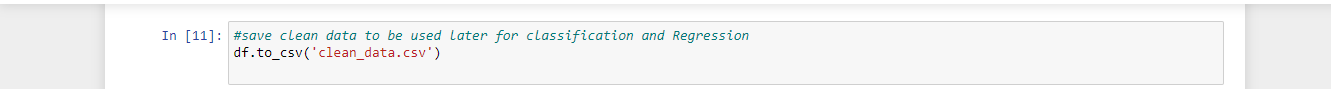
**Task 4: Perform Data Exploration and answer q1 to q6 in LMS**

**Task 5: Check the columns with missing values and drop the columns which has more number of values missing and check for missing values to ensure all columns return 0**

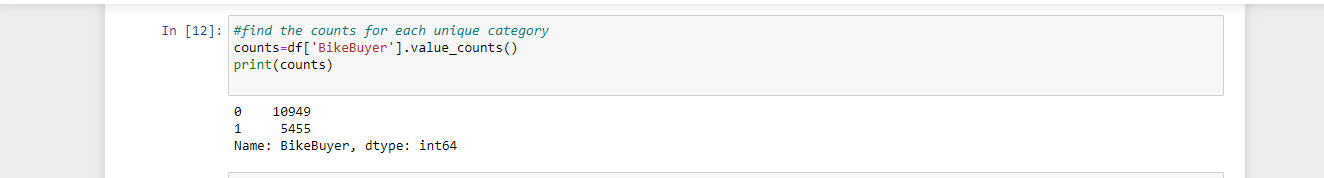
 C

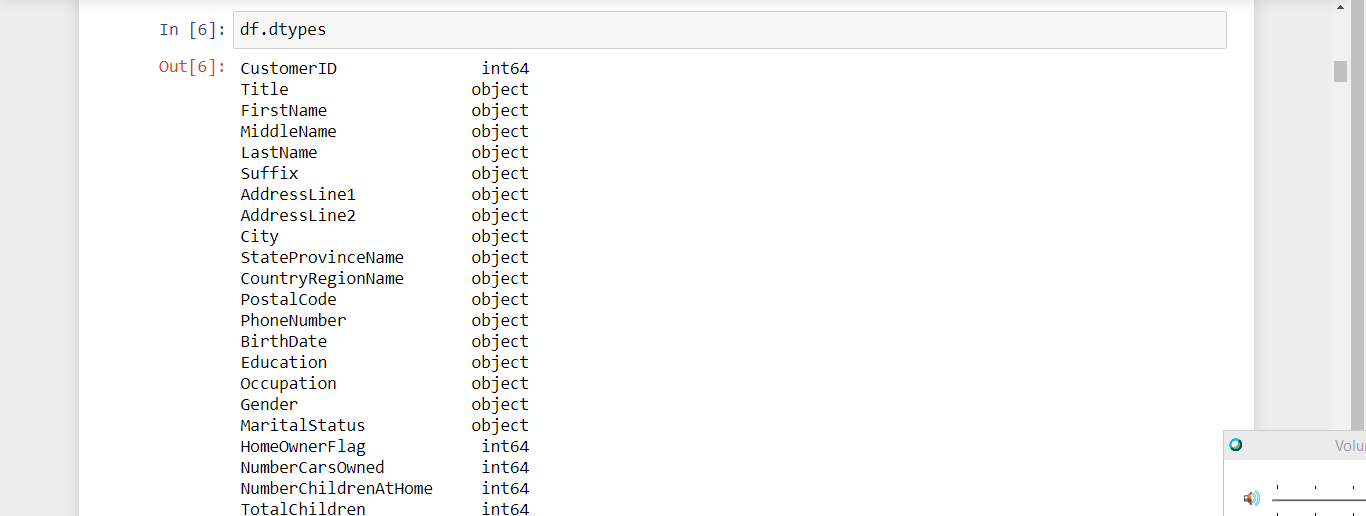
Fi

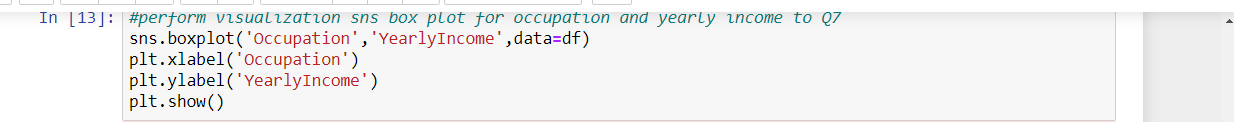
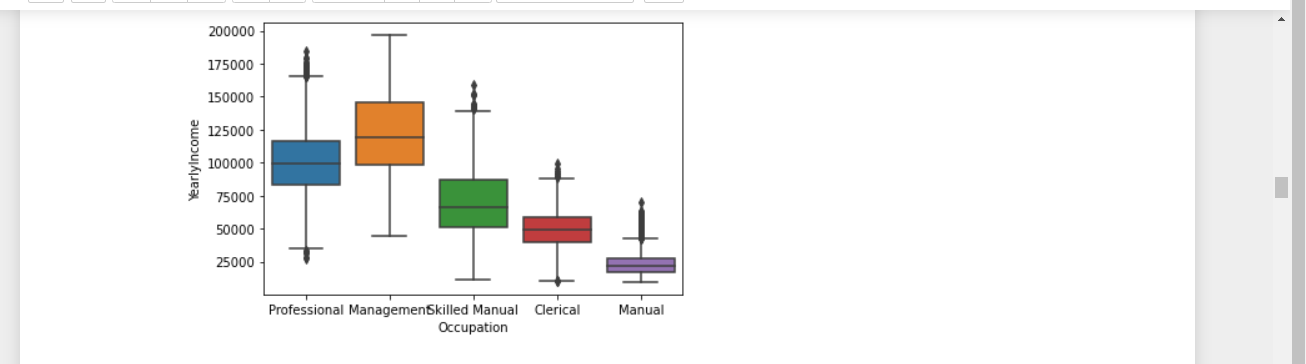
T f



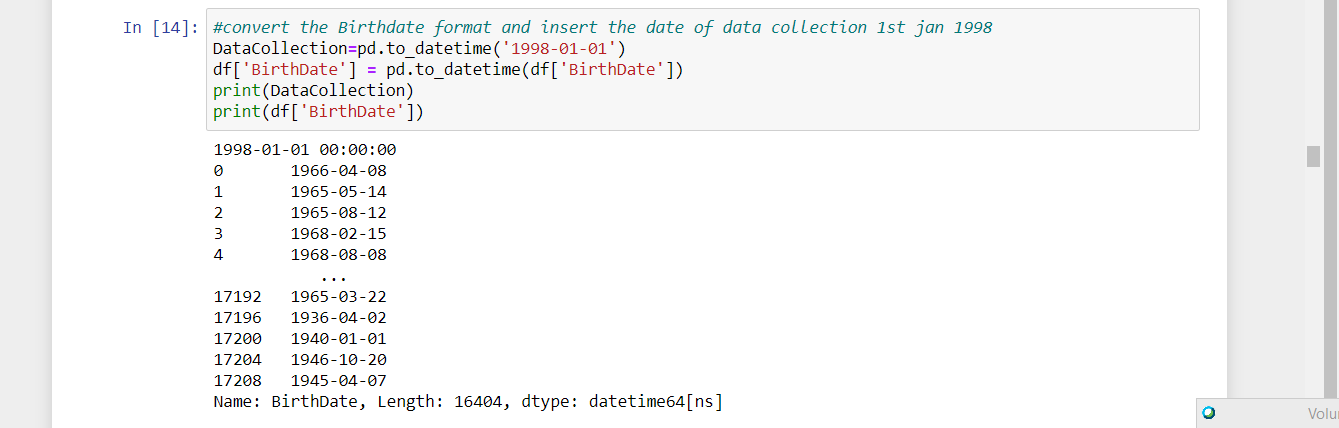
**Task 6: Find the Bike Buyers and Non Bike Buyers**



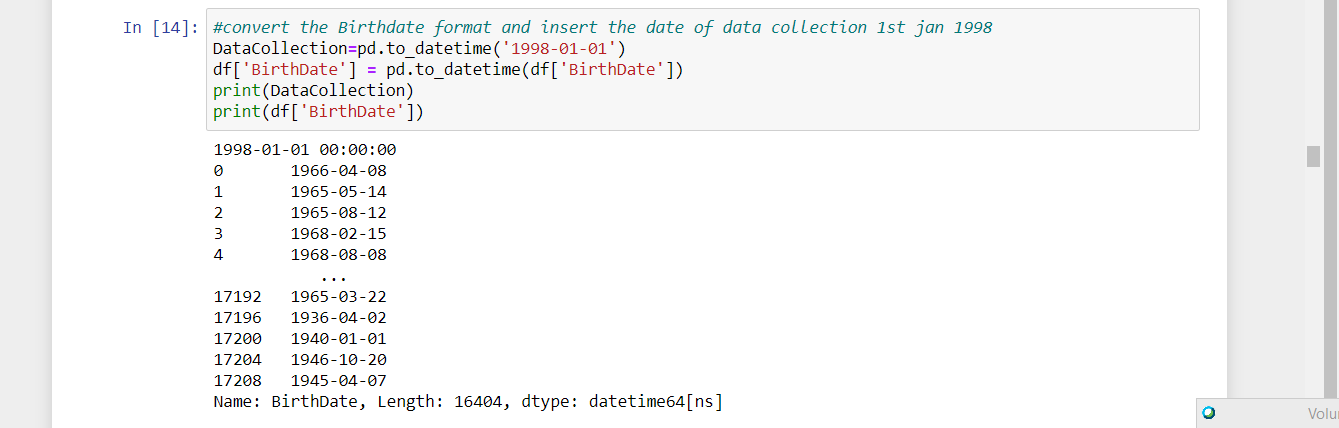


**Task 7: Perform visualization Sns box plot for occupation and yearly in**  **come to Q7:**   ****

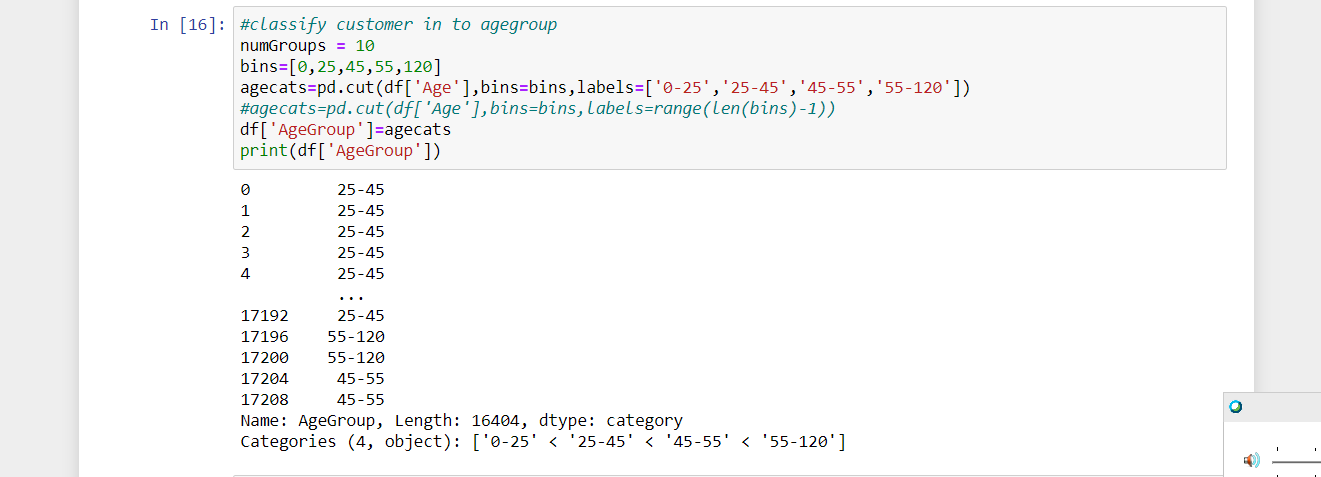
Task 8: convert Birth date formate and insert the date of data collection 1st Jan 1998



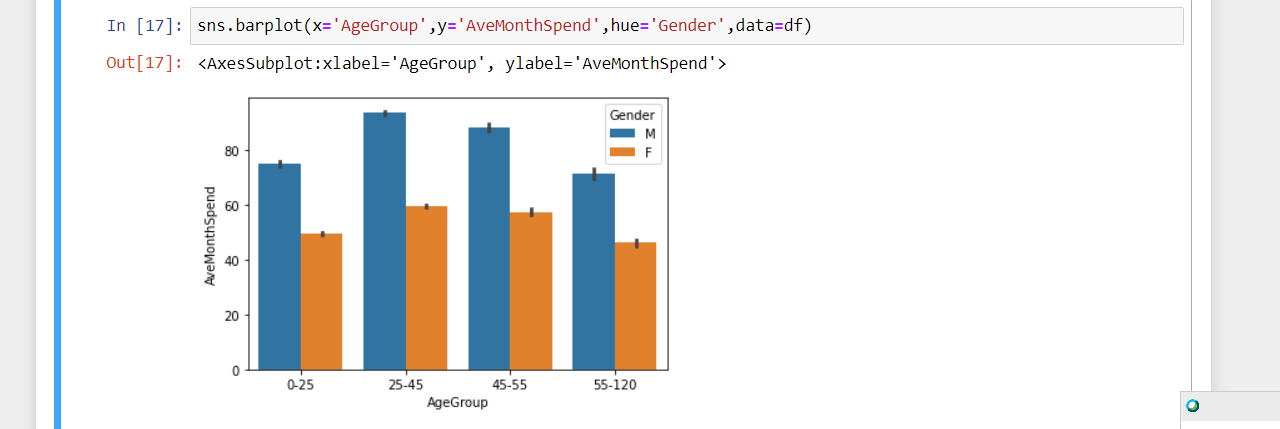
**Task 9: convert the Birth date into at the collection data using function and return age**



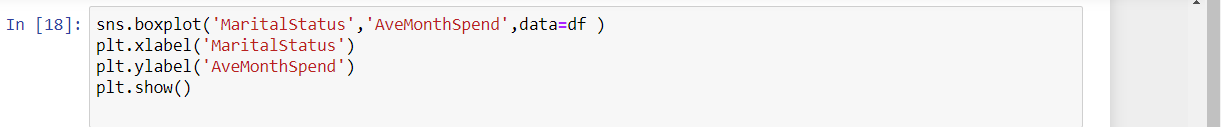
**Task 10 : classify customer into Age groups**

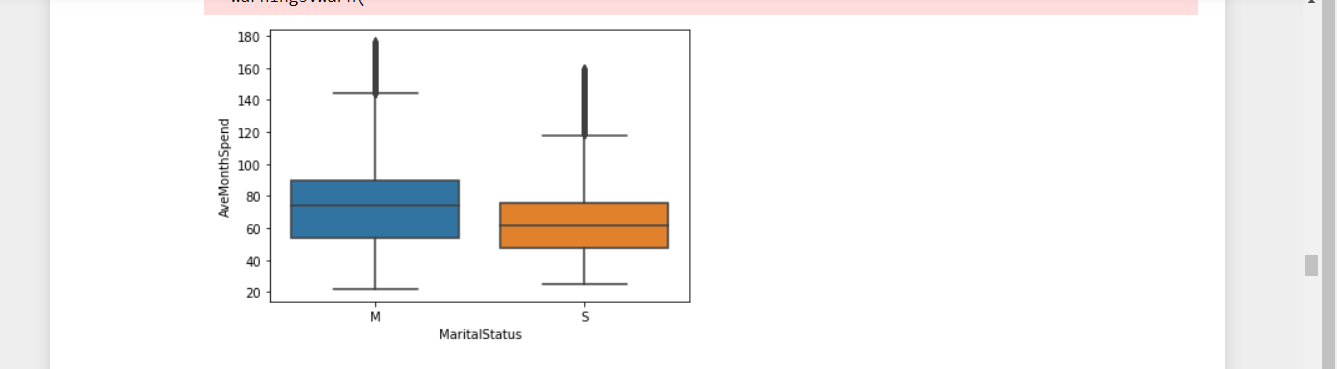


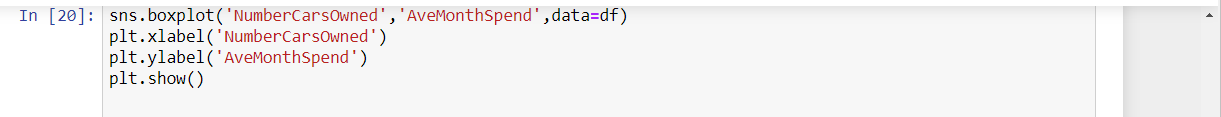
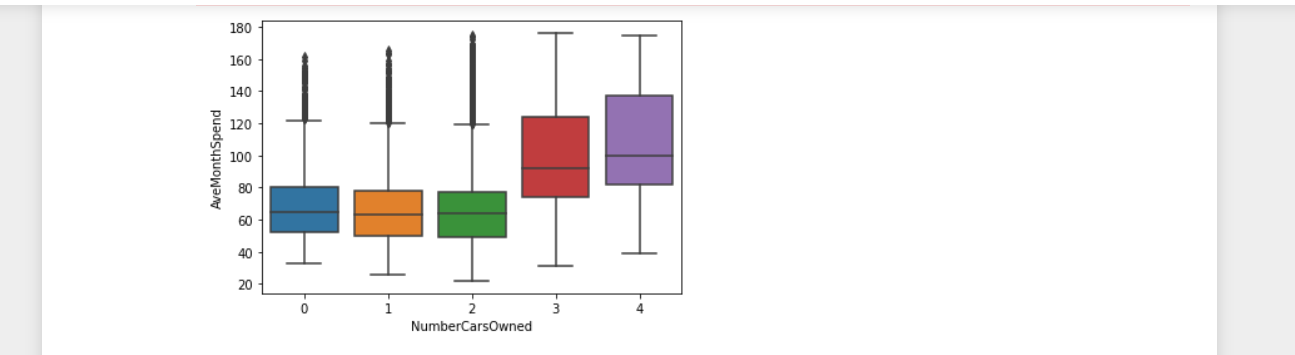
**Task 11: Plot barchart to demonstrate the distibution of AvemonthSpend among differennt Agegroups and Genders**

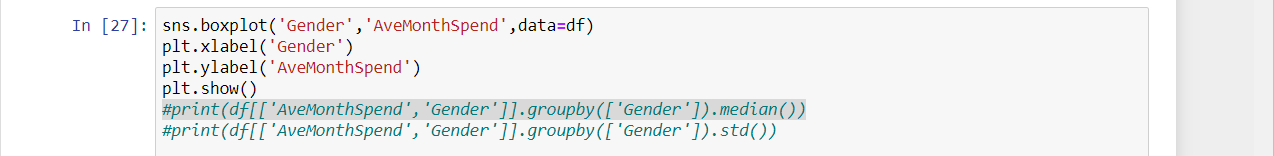
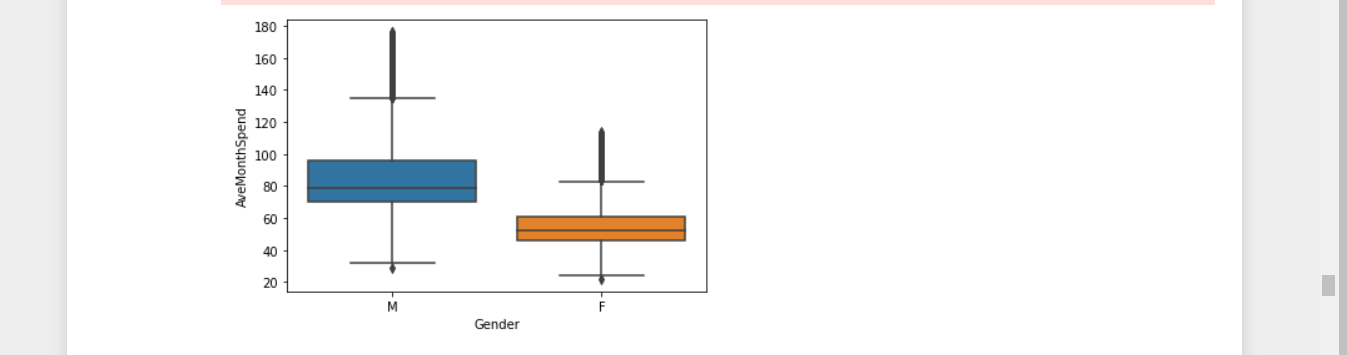


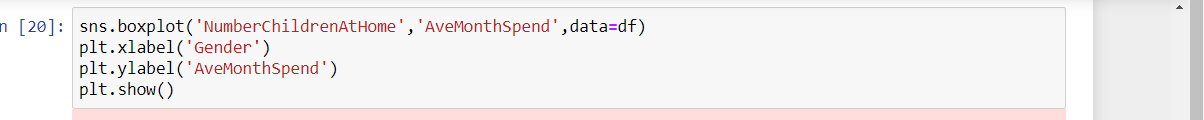
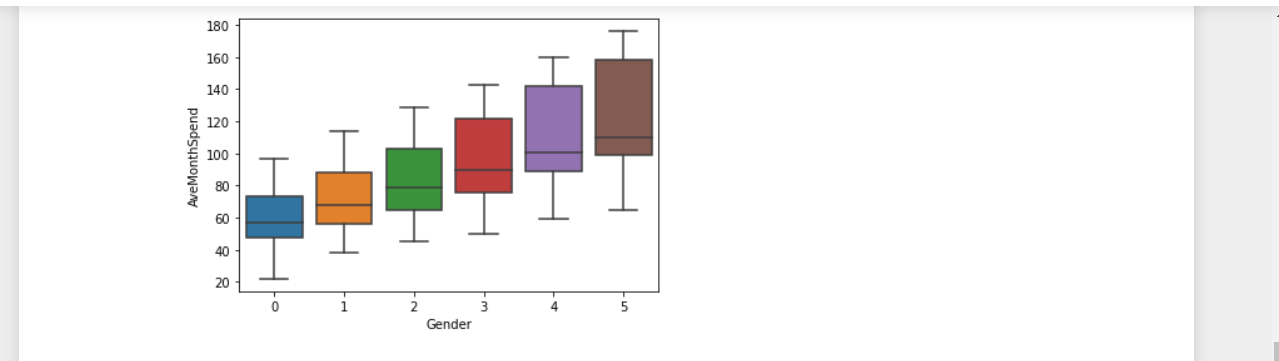
**Task 12: Compare AveMonthSpend among different demographic values MaritalStatus NumberCarsOwned**



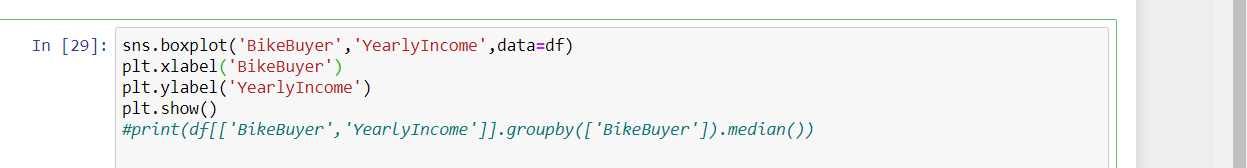
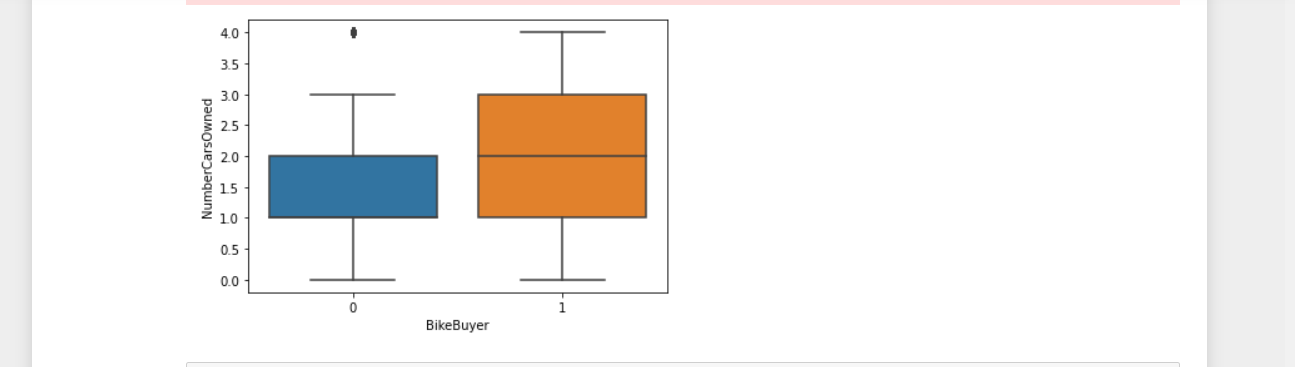


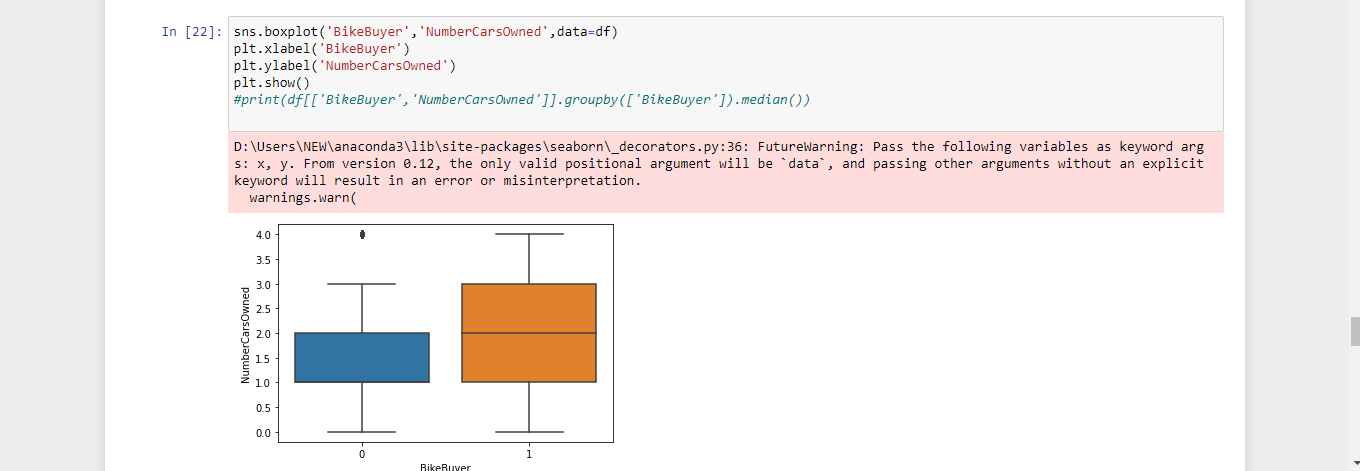
Creates and trains a logistic regression model using the training data. Spe  with no kn 

• ****

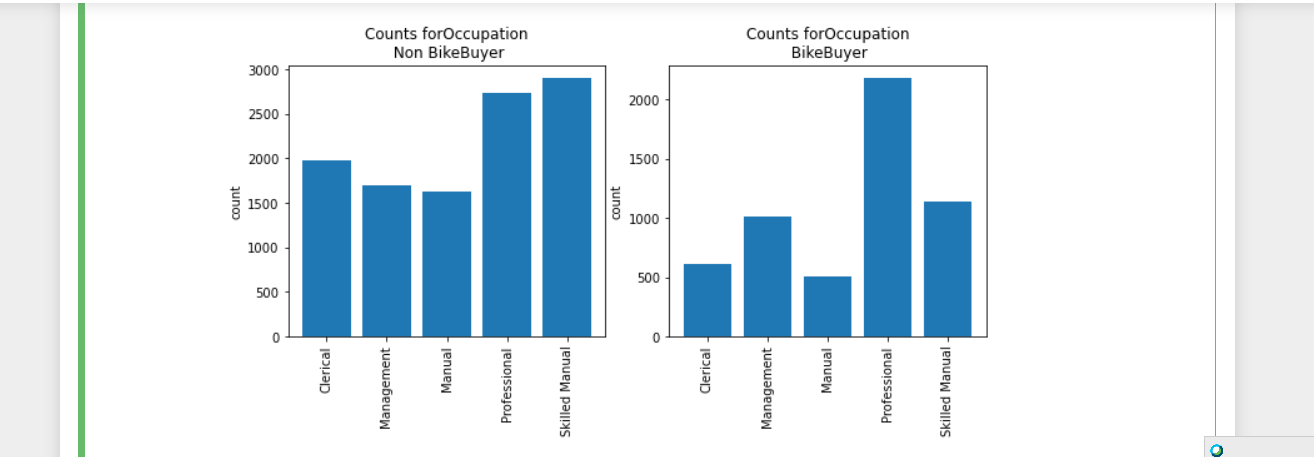
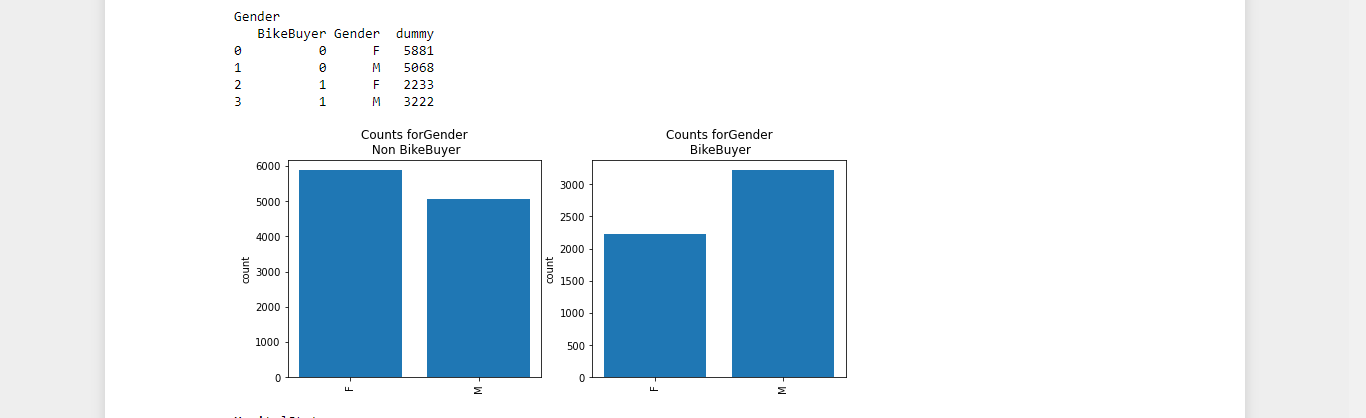
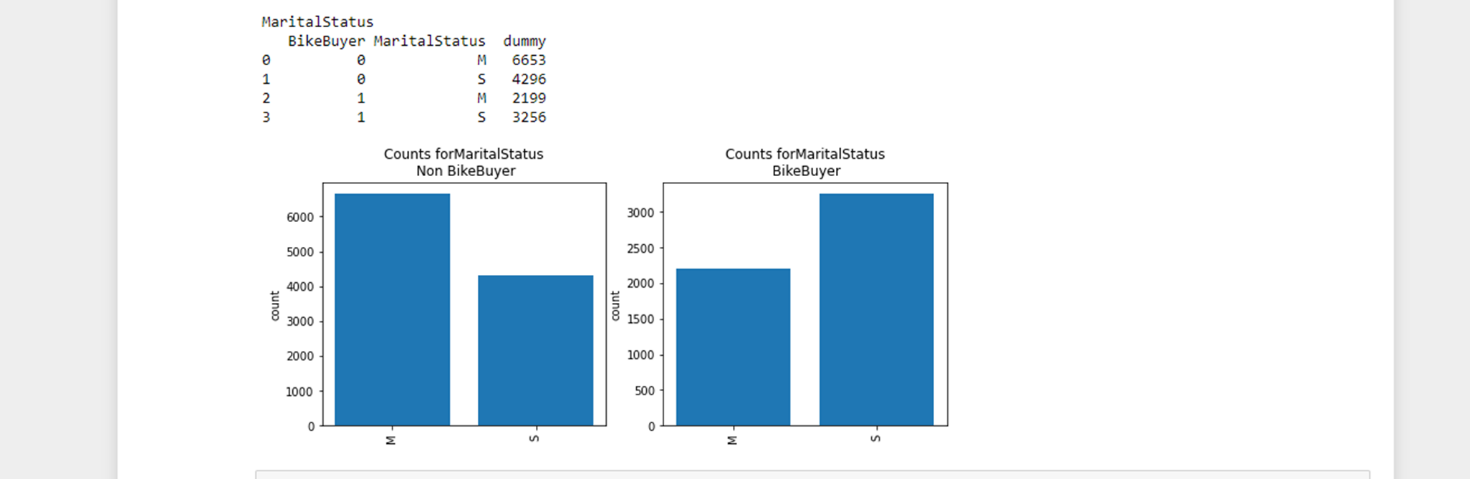
****.**** • f

**Task 13: Compare attributes for BikeBuyers and non BikeBuyers YearlyIncomes,NumberCarsOwned**

 ****

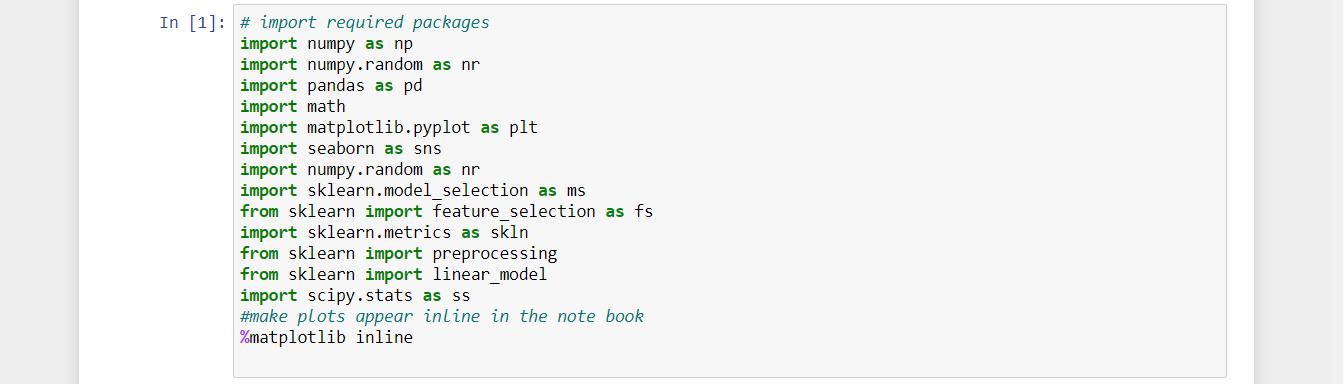


****

. Tests t

**Activity 2 : Build a classification model to predict customer servicing behaviour**

**Task 1: Setting up the classification model**

****

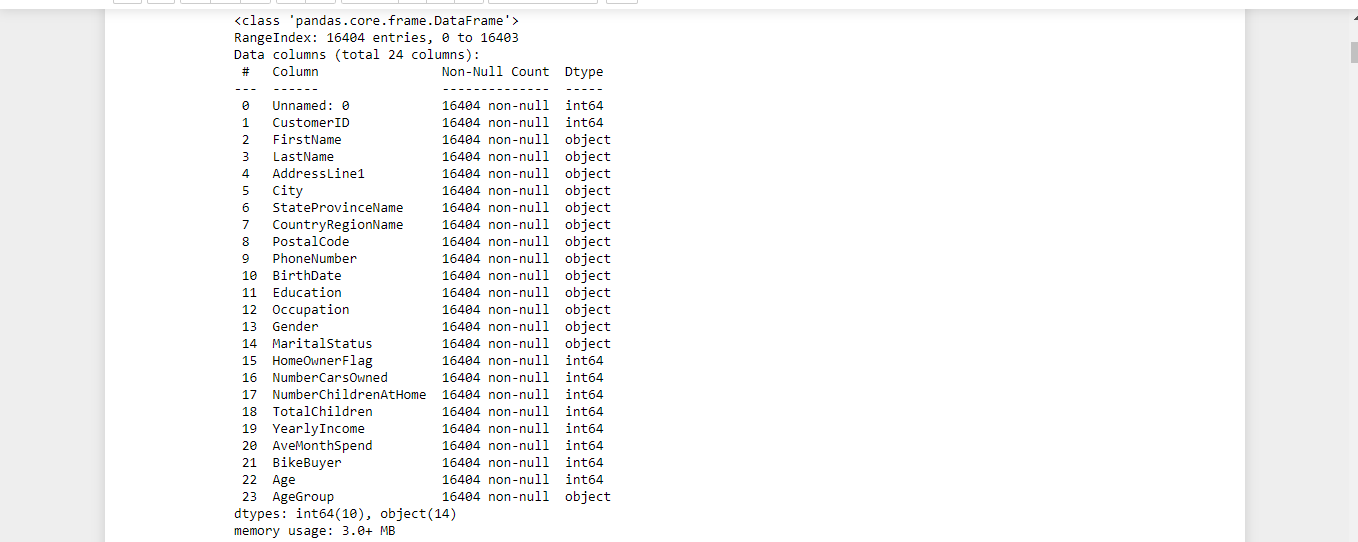
**T**

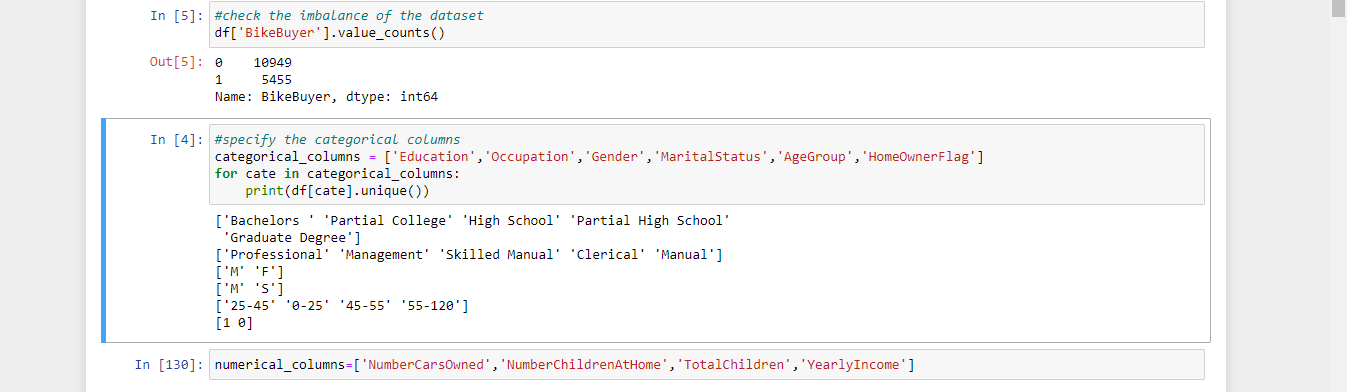
**Task 2: Check for duplicate rows missing values abd find out the imbalance of data swt**

**C:\Users\NEW\Pictures\dataread.png**

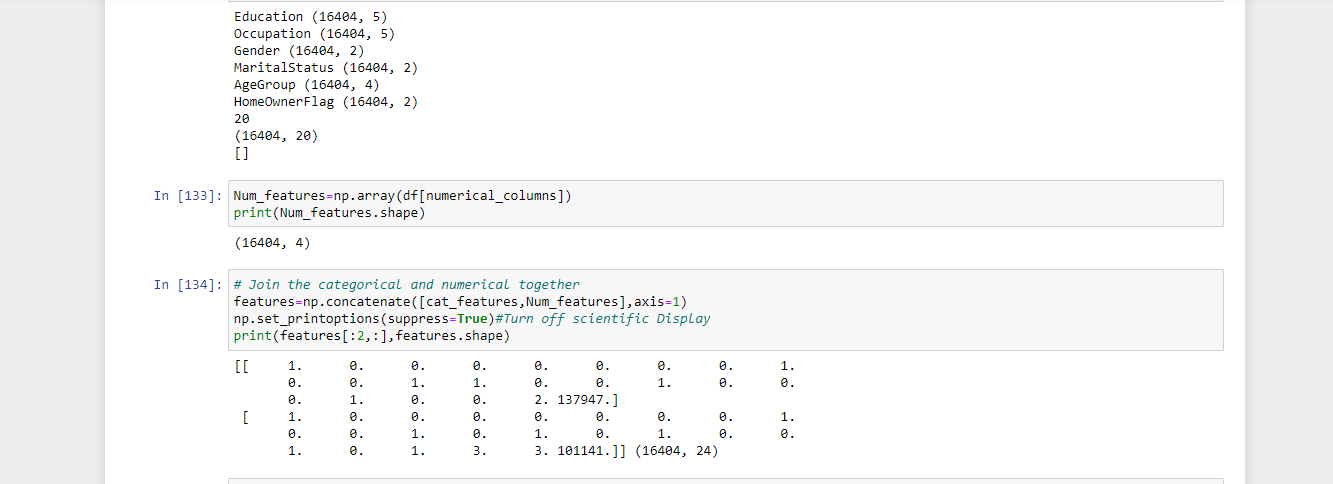
**Task 3: Encode the categorical columns after the checing for the info for the dataset**

****

1. 

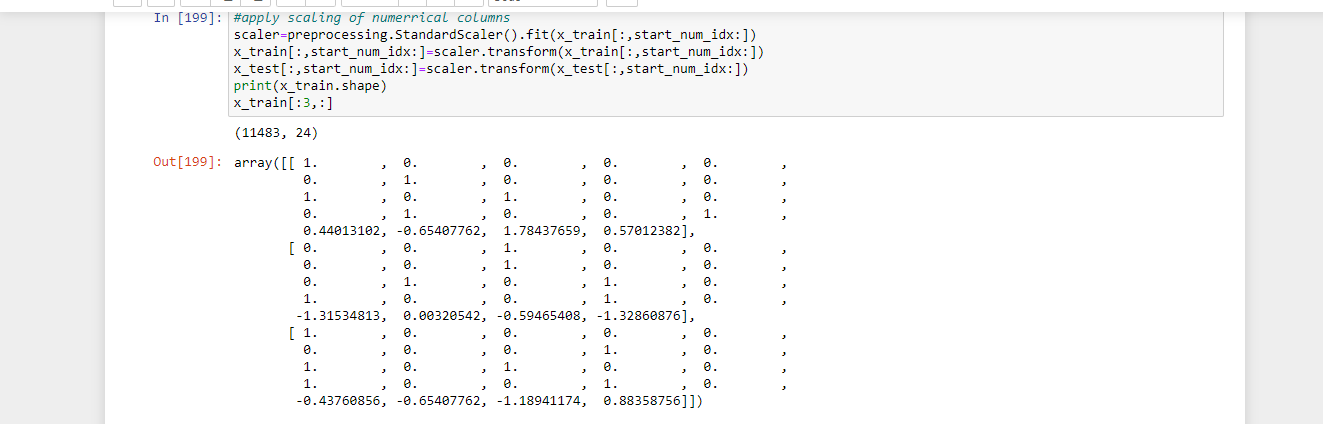




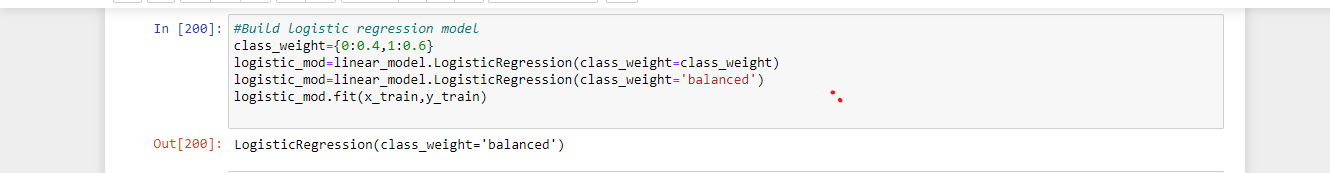


**Task 4: Split the dataset with 70:30 Based on BikeBuyer and create some random samples if needed**

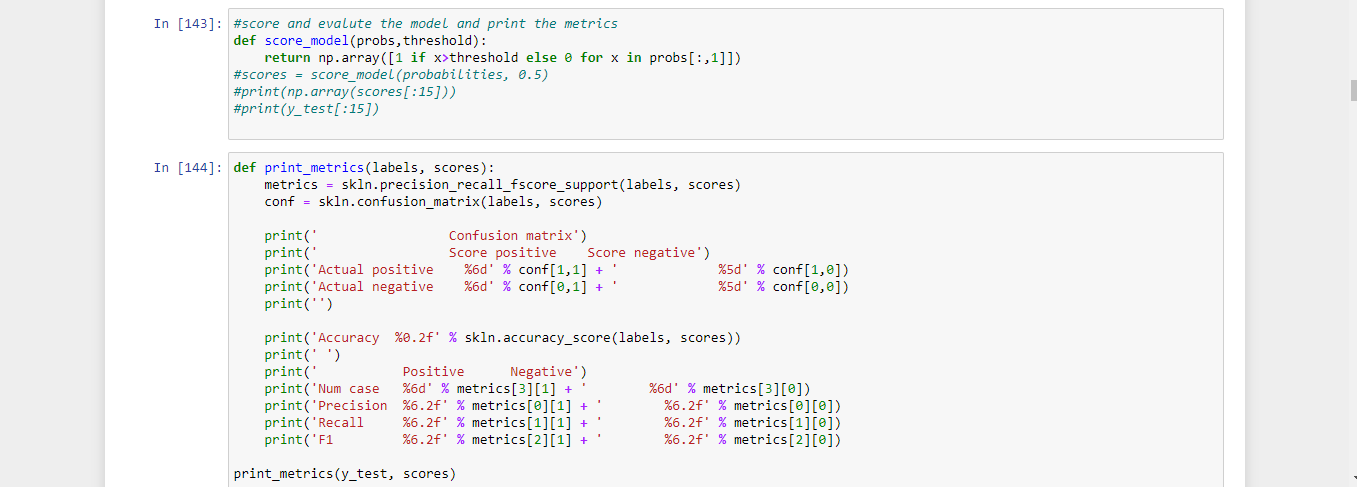


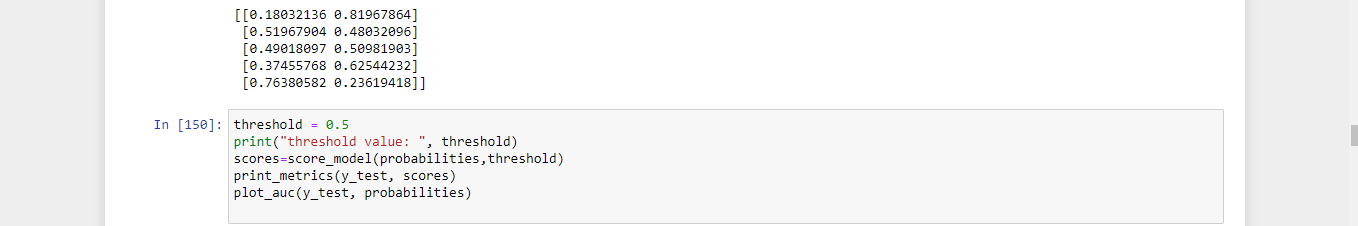
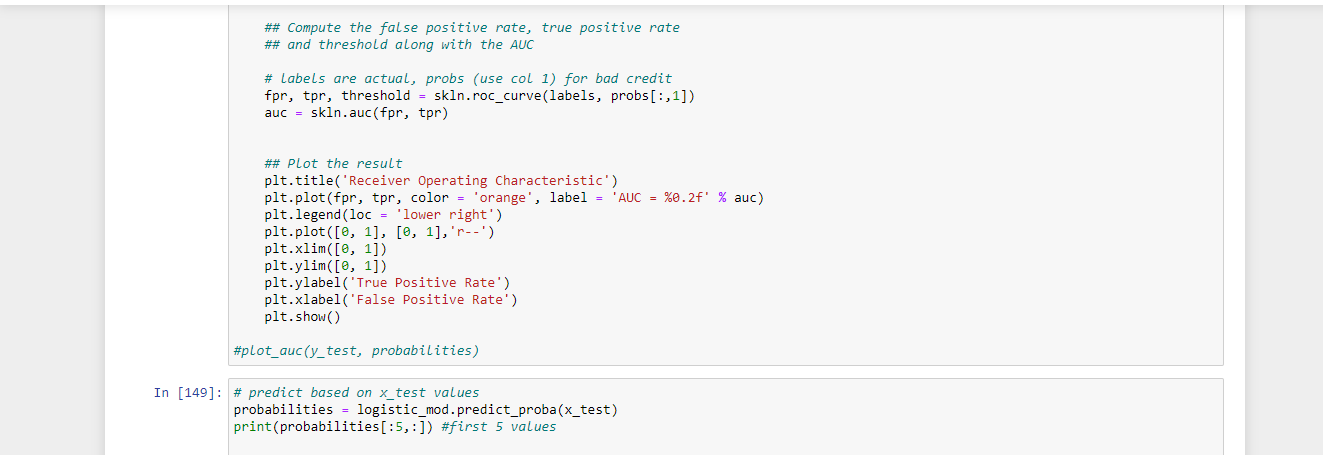


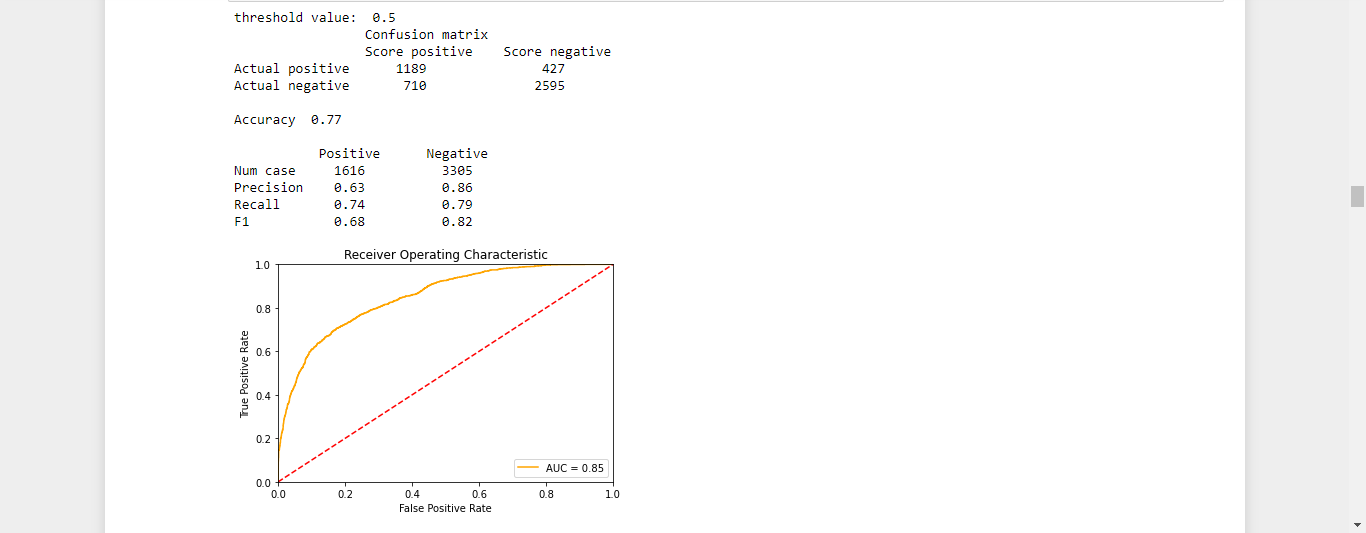
Task 5: Setting up the Regression Model: (Explain the Process for setting up Regression using Python ) screenshot of all the above steps



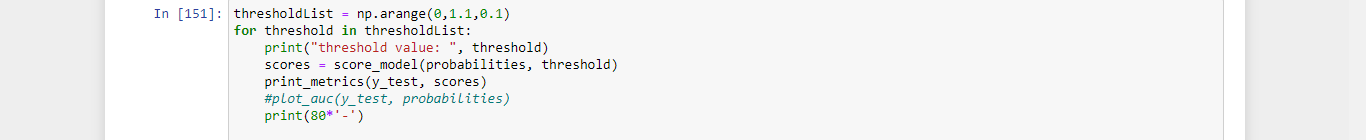
Task 6: score and evaluate the model and print the metrics

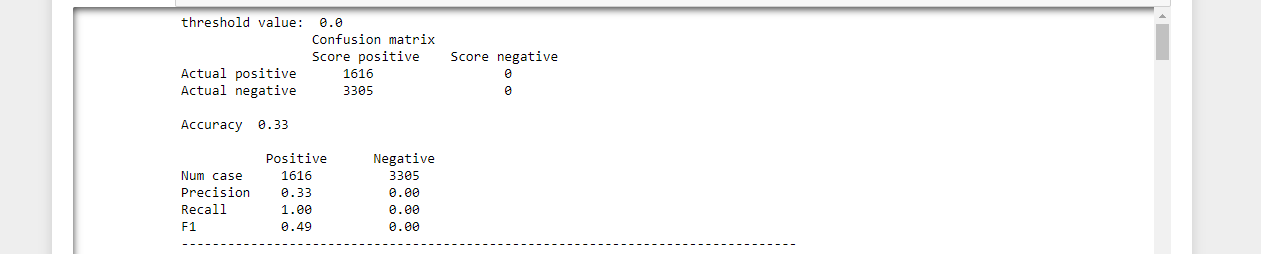


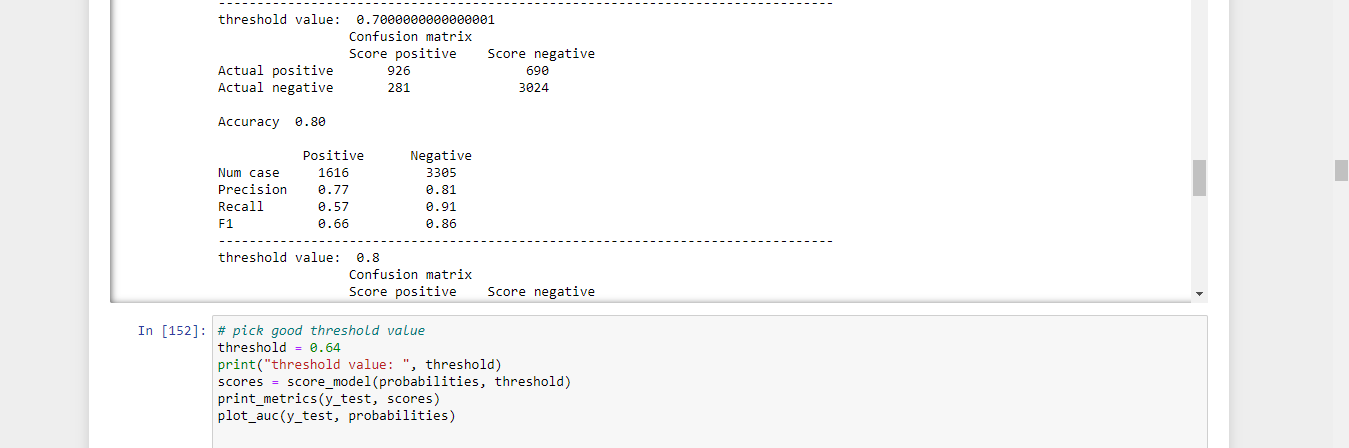


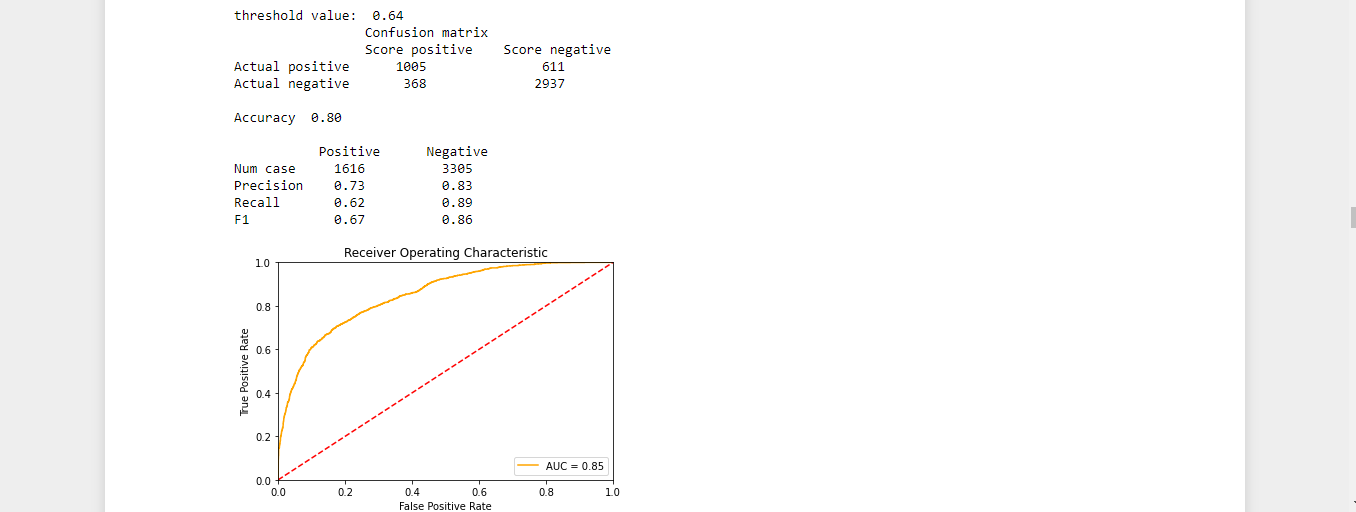


Task7: Check the AUC curve and play will few threshold values to see the change of TP,FN etc





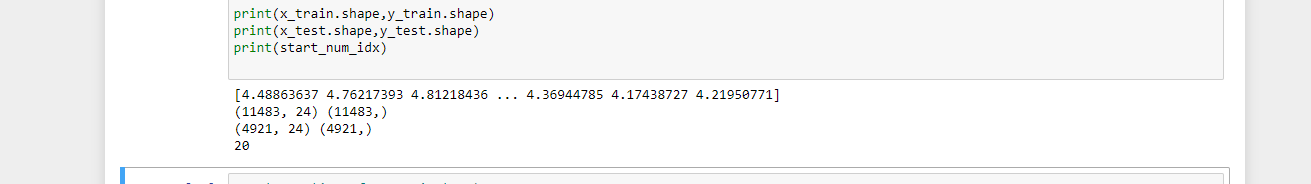


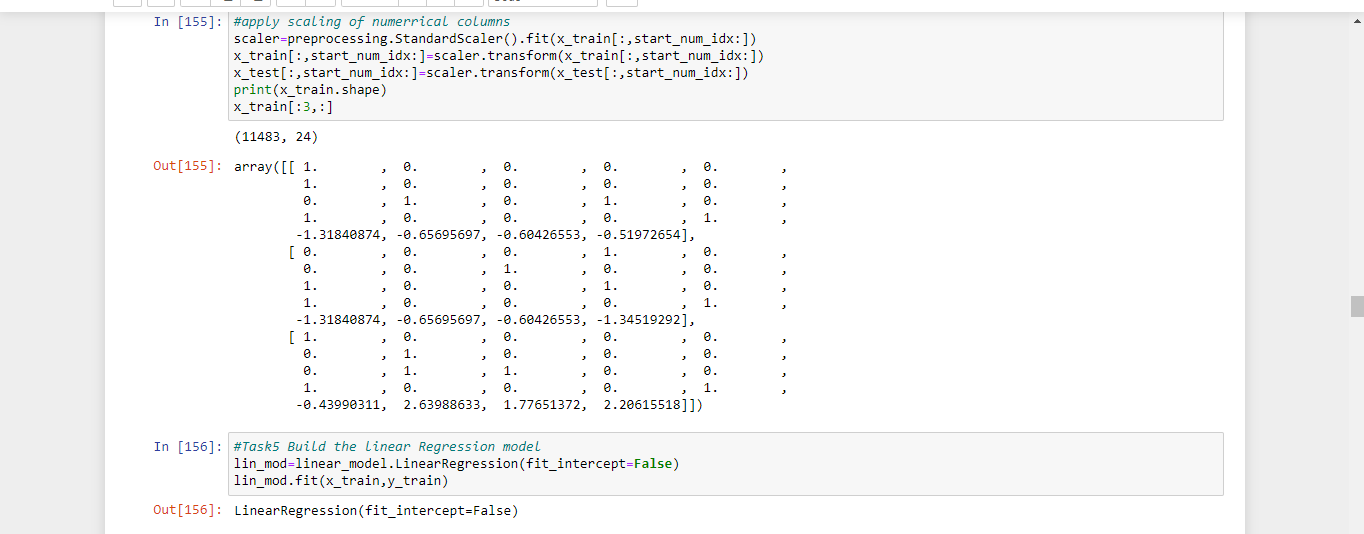


Setting up the Regression moel

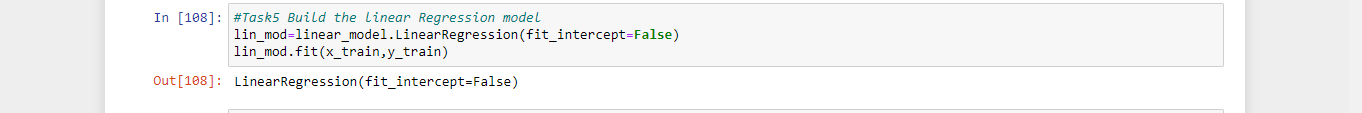
Task 4: Split the dataset with 70:30 Based on AveMonthSpend and create some random samples if needed



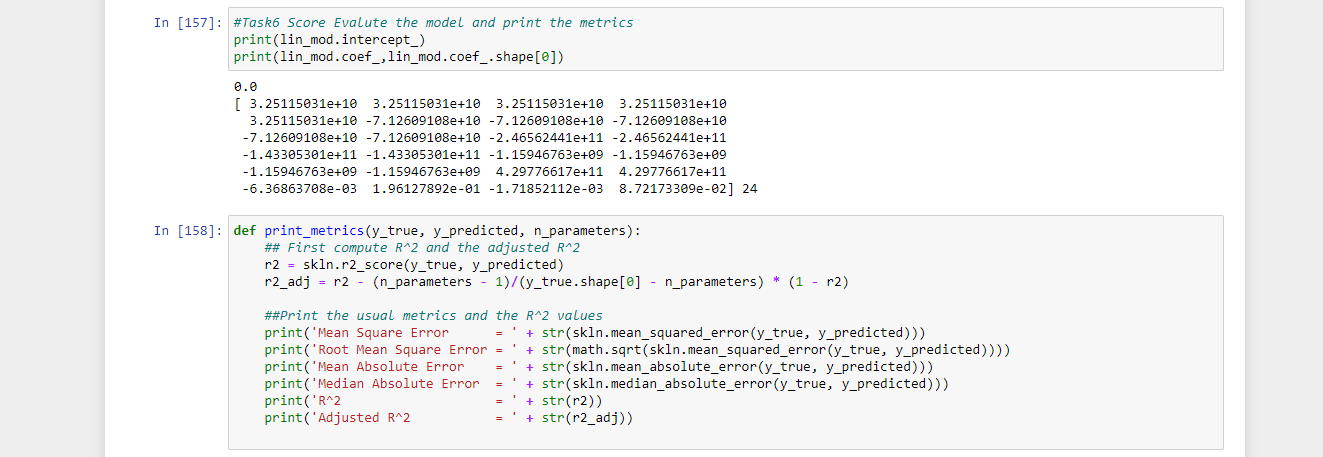


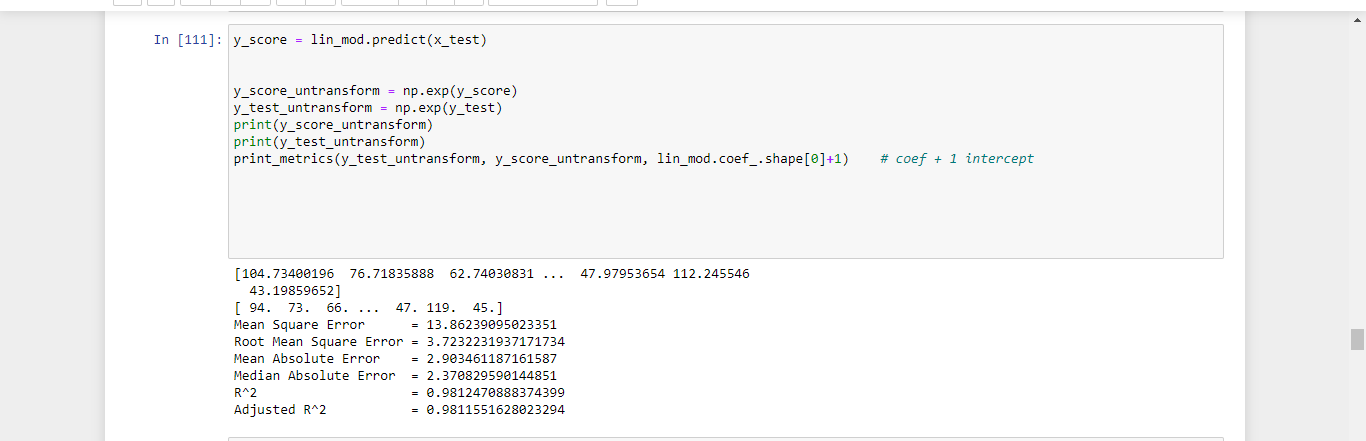


Task5: Build linear Regression model

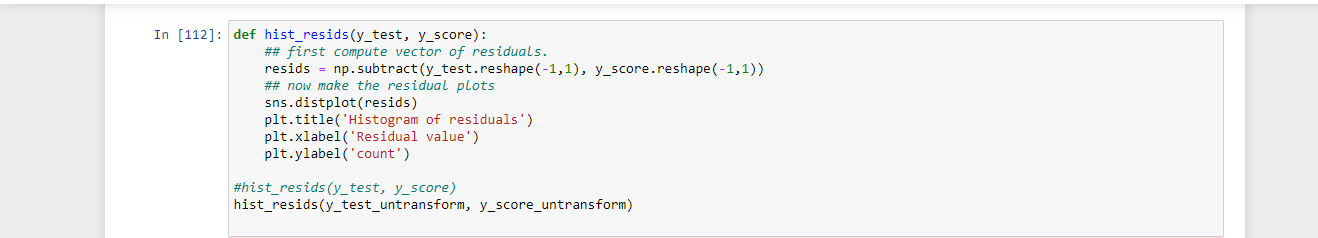


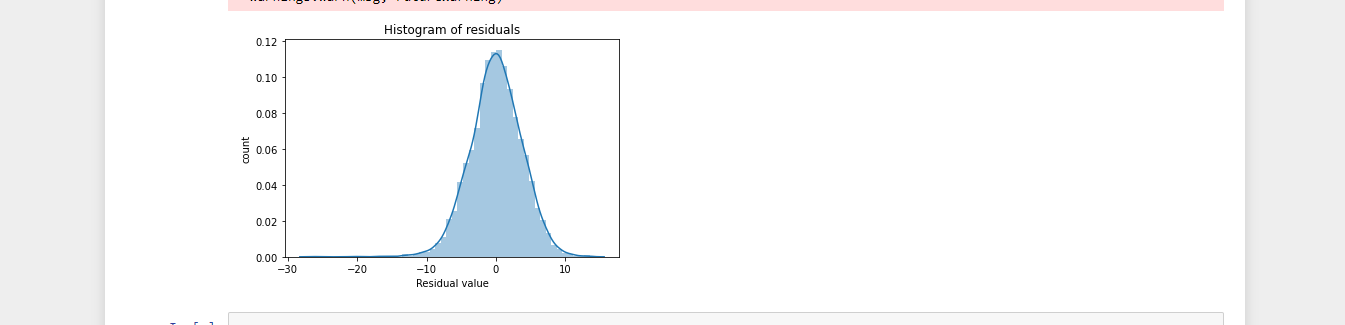
Task6: score and evaluate the model and print the metrics

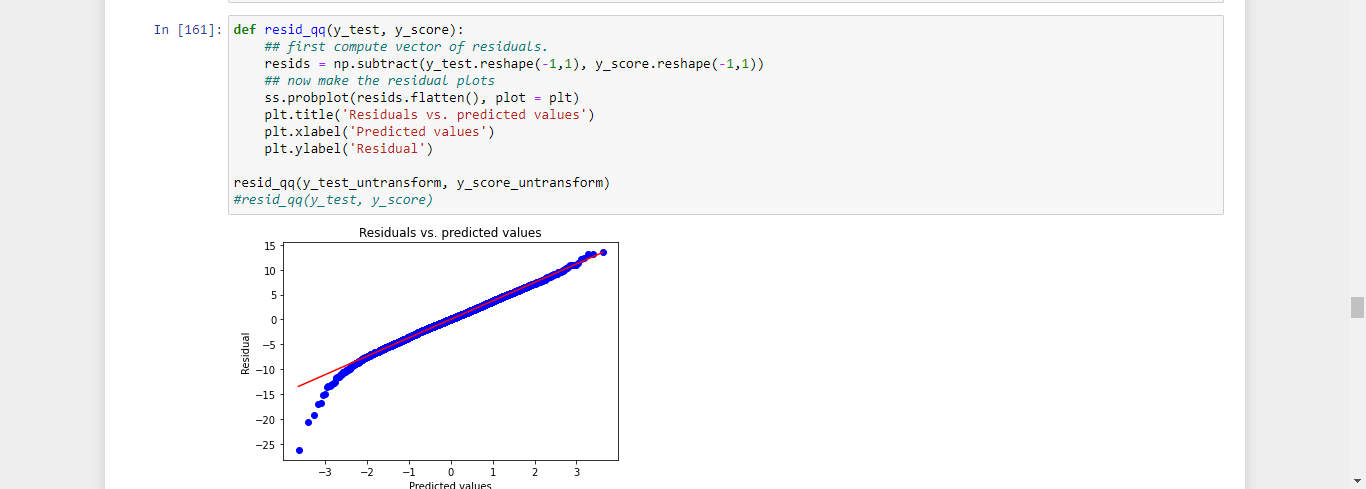


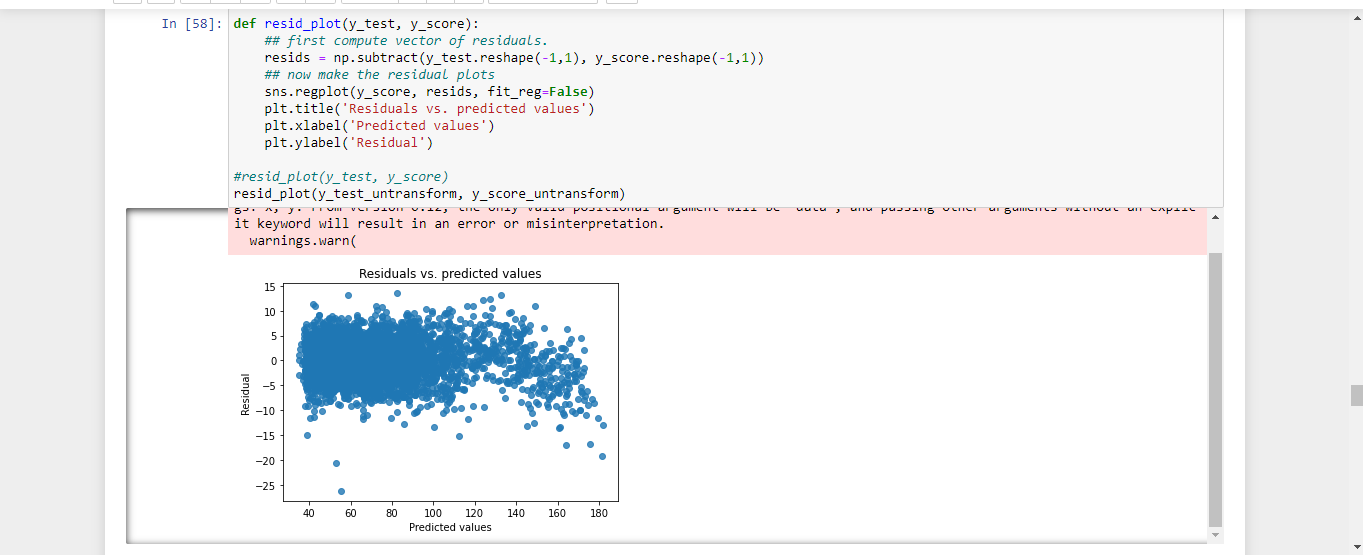


Task7: plots





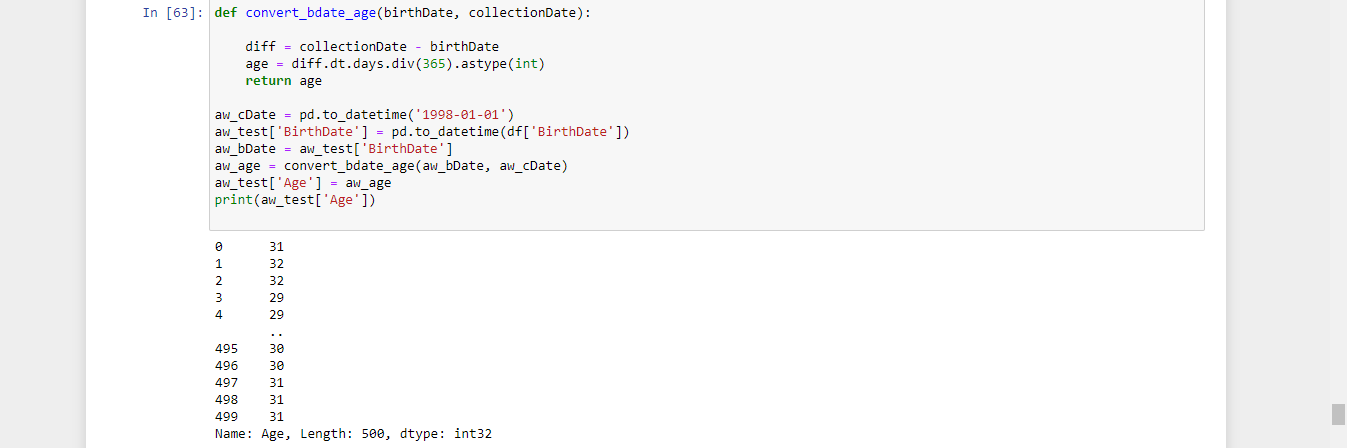






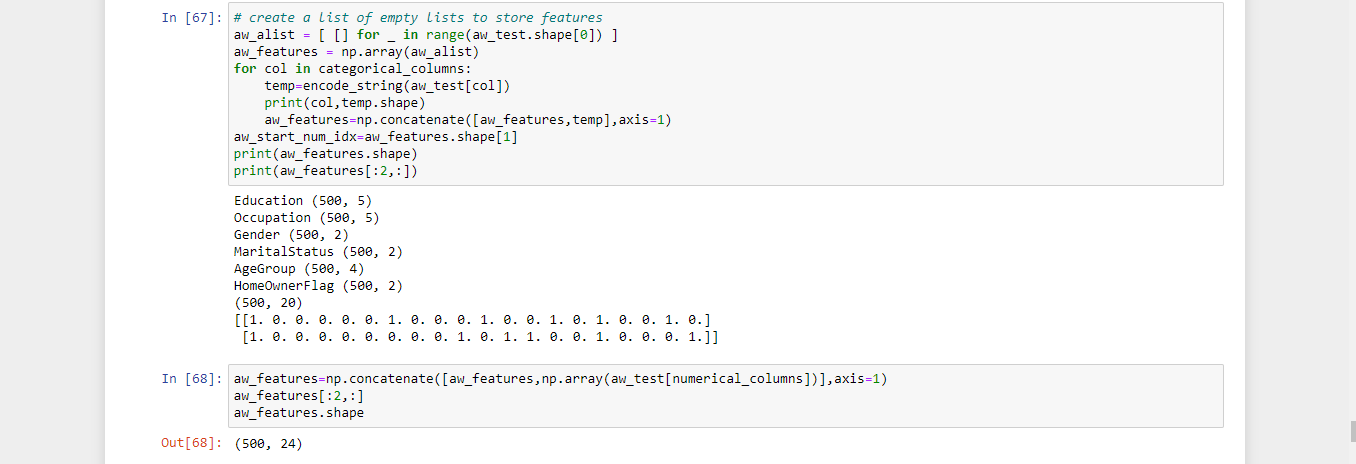


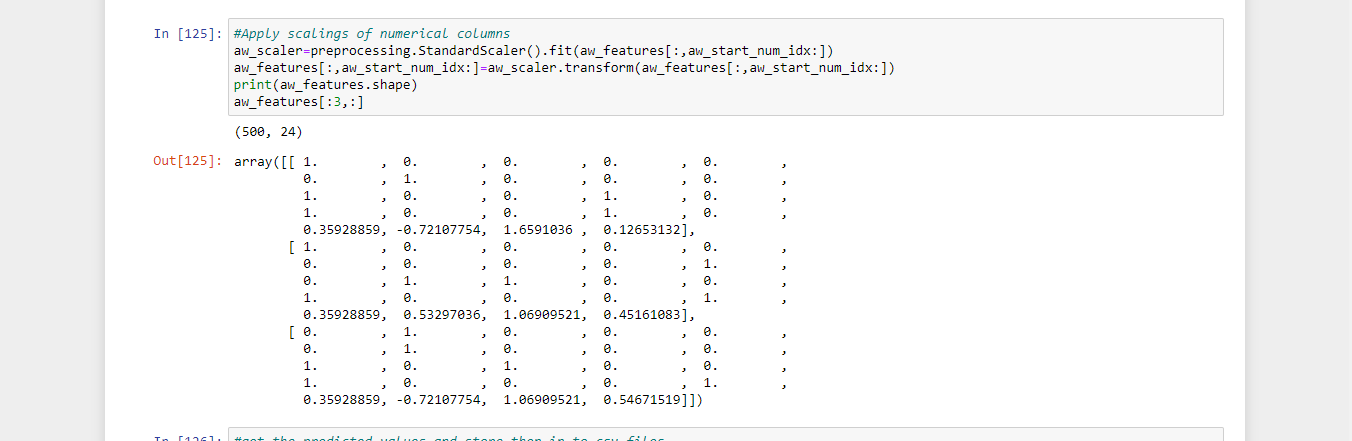




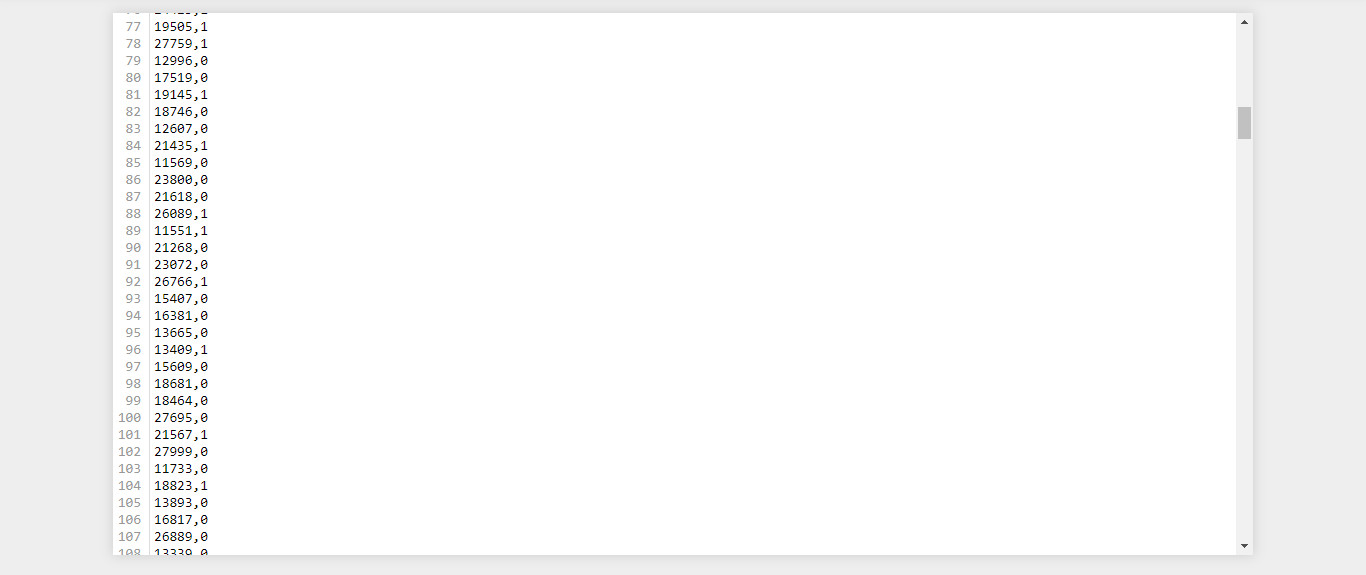
Read clean AW\_test file

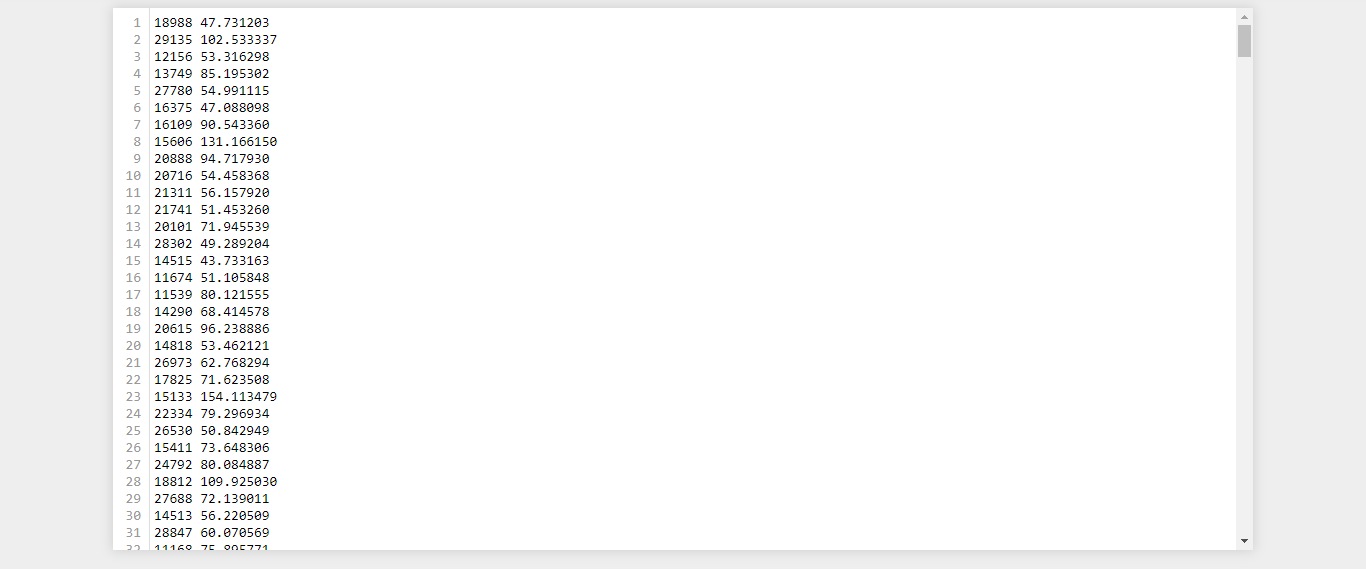










****

**Annexure**  **Annex 1 (Python code for EDA):**

# Import required packages

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import numpy.random as nr

import math

Annex 1b (Python code for Classification):

#Read three dataset given

missing\_value\_formates=['n.a.','?','NA','n/a','--','null']

df1=pd.read\_csv('AdvWorksCusts.csv')

df2=pd.read\_csv('AW\_AveMonthSpend.csv')

df3=pd.read\_csv('AW\_BikeBuyer.csv')

#print(df1.shape)

#print(df2.shape)

#print(df3.shape)

merge all datasets into one dataframe and check result

df = df1.merge(df2, on='CustomerID',how='left')

df=df.merge(df3,on='CustomerID',how='left')

print(df.head(2),df.shape)

#check for the duplicates and remove them

df.drop\_duplicates(subset='CustomerID',keep='last',inplace=True)

print(df.columns,df.shape

#perform data exploration

df['AveMonthSpend']. df describe().

isnull().sum()

df. drop(columns=['Title','MiddleName','Suffix','AddressLine2'],inplace=True)

df.i #save clean data to be used later for classification and Regression

df.to\_csv('clean\_data.csv')snull().sum()

#find the counts for each unique category

counts=df['BikeBuyer'].value\_counts()

#perform visualization sns box plot for occupation and yearly income to Q7

sns.boxplot('Occupation','YearlyIncome',data=df)

plt.xlabel('Occupation')

plt.ylabel('YearlyIncome')

plt.show()prin

#convert the Birthdate format and insert the date of data collection 1st jan 1998

DataCollection=pd.to\_datetime('1998-01-01')

df['BirthDate'] = pd.to\_datetime(df['BirthDate'])

print(DataCollection)

print(df['BirthDate'])t(counts)

#convert birthdate in to age at the collection date using function and return age

def convert\_bdate\_age(cdate,bdate):

diff=cdate-bdate

age=diff.dt.days.div(365).astype(int)

return age

bdate=df['BirthDate']

age=convert\_bdate\_age(DataCollection,bdate)

df['Age']=age

print(df['Age'])

#classify customer in to agegroup

numGroups = 10

bins=[0,25,45,55,120]

agecats=pd.cut(df['Age'],bins=bins,labels=['0-25','25-45','45-55','55-120'])

#agecats=pd.cut(df['Age'],bins=bins,labels=range(len(bins)-1))

df['AgeGroup']=agecats

print(df[‘'AgeGroup'’] )

sns.boxplot('MaritalStatus','AveMonthSpend',data=df )

plt.xlabel('MaritalStatus')

plt.ylabel('AveMonthSpend')

plt.show().

sns.boxplot('NumberCarsOwned','AveMonthSpend',data=df)

plt.xlabel('NumberCarsOwned')

plt.ylabel('AveMonthSpend')

plt.show()

sns.boxplot('Gender','AveMonthSpend',data=df)

plt.xlabel('Gender')

plt.ylabel('AveMonthSpend')

plt.show()

#print(df[['AveMonthSpend','Gender']].groupby(['Gender']).median())

#print(df[['AveMonthSpend','Gender']].groupby(['Gender']).std())

sns.boxplot('NumberChildrenAtHome','AveMonthSpend',data=df)

plt.xlabel('Gender')

plt.ylabel('AveMonthSpend')

plt.show()

sns.boxplot('BikeBuyer','YearlyIncome',data=df)

plt.xlabel('BikeBuyer')

plt.ylabel('YearlyIncome')

plt.show()

#print(df[['BikeBuyer','YearlyIncome']].groupby(['BikeBuyer']).median())

sns.boxplot('BikeBuyer','NumberCarsOwned',data=df)

plt.xlabel('BikeBuyer')

plt.ylabel('NumberCarsOwned')

plt.show()

#print(df[['BikeBuyer','NumberCarsOwned']].groupby(['BikeBuyer']).median())

cat\_cols=['Occupation','Gender','MaritalStatus']

df['dummy']=np.ones(shape=df.shape[0])

for col in cat\_cols:

print(col)

counts=df[['dummy','BikeBuyer',col]].groupby(['BikeBuyer',col],as\_index=False).count()

print(counts)

\_=plt.figure(figsize=(10,4))

plt.subplot(1,2,1)

temp=counts[counts['BikeBuyer']==0][[col,'dummy']]

plt.bar(temp[col],temp.dummy)

plt.xticks(rotation=90)

plt.title('Counts for'+col+'\n Non BikeBuyer')

plt.ylabel('count')

plt.subplot(1,2,2)

temp=counts[counts['BikeBuyer']==1][[col,'dummy']]

plt.bar(temp[col],temp.dummy)

plt.xticks(rotation=90)

plt.title('Counts for'+col+'\n BikeBuyer')

plt.ylabel('count')

plt.show()

df.to\_csv('clean\_data\_with\_Agegroup'

**Annexure 2:**

Script 2

# import required packages

import numpy as np

import numpy.random as nr

import pandas as pd

import math

import matplotlib.pyplot as plt

import seaborn as sns

import numpy.random as nr

import sklearn.model\_selection as ms

from sklearn import feature\_selection as fs

import sklearn.metrics as skln

from sklearn import preprocessing

from sklearn import linear\_model

import scipy.stats as ss

#make plots appear inline in the note book

%matplotlib inline

#check for duplicate rows,missing values and find out the imbalance of dataset

df=pd.read\_csv('clean\_data\_with\_Agegroup')

df.i #check the imbalance of the dataset

df['BikeBuyer'].value\_counts()nfo(#specify the categorical columns

categorical\_columns = ['Education','Occupation','Gender','MaritalStatus','AgeGroup','HomeOwnerFlag']

for cate in categorical\_columns:

p numerical\_columns=['NumberCarsOwned','NumberChildrenAtHome','TotalChildren','Year def encode\_string(cat\_feature):

#first encode the strings to numeric categories

enc=preprocessing.LabelEncoder()

enc.fit(cat\_feature)

enc\_cat\_feature=enc.transform(cat\_feature)

#now apply one hot encodig

ohe=preprocessing.OneHotEncoder()

encoded=ohe.fit(enc\_cat\_feature.reshape(-1,1))

return encoded.transform(enc\_cat\_feature.reshape(-1,1)).toarray()

lyIncome']rint(df #create a list of empty lists to categorical features

alist=[ [] for \_ in range(df.shape[0])]

cat\_features=np.array(alist)

for col in categorical\_columns:

temp=encode\_string(df[col])

print(col,temp.shape)

cat\_features=np.concatenate([cat\_features,temp],axis=1)

np.set\_printoptions(suppress=True)#Turn off scientific Display

print(features[:2,:],features.shape)

nr.seed(77)

labels = np.array(df['BikeBuyer'])

indx = range(features.shape[0])

test\_size = int(0.3\*features.shape[0])

indx = ms.train\_test\_split(indx, test\_size = test\_size)

x\_train = features[indx[0],:]

y\_train = np.ravel(labels[indx[0]])

x\_test = features[indx[1],:]

y\_test = np.ravel(labels[indx[1]])

print(x\_train.shape, y\_train.shape)

print(x\_test.shape, y\_test.shape)

pr #apply scaling of numerrical columns

scaler=preprocessing.StandardScaler().fit(x\_train[:,start\_num\_idx:])

x\_train[:,start\_num\_idx:]=scaler.transform(x\_train[:,start\_num\_idx:])

x\_test[:,start\_num\_idx:]=scaler.transform(x\_test[:,start\_num\_idx:])

print(x\_train.shape)

x\_train[:3,:]int(start\_num\_idx) Build logistic regression model

class\_weight={0:0.4,1:0.6}

logistic\_mod=linear\_model.LogisticRegression(class\_weight=class\_weight)

logistic\_mod=linear\_model.LogisticRegression(class\_weight='balanced')

logistic\_mod.fit(x\_train,y\_train) #score and evalute the model and print the metrics

def score\_model(probs,threshold):

return np.array([1 if x>threshold else 0 for x in probs[:,1]])

#scores = score\_model(probabilities, 0.5)

#print(np.array(scores[:15]))

#print(y\_test[:15]) def print\_metrics(labels, scores):

metrics = skln.precision\_recall\_fscore\_support(labels, scores)

conf = skln.confusion\_matrix(labels, scores)

print(' Confusion matrix')

print(' Score positive Score negative')

print('Actual positive %6d' % conf[1,1] + ' %5d' % conf[1,0])

print('Actual negative %6d' % conf[0,1] + ' %5d' % conf[0,0])

print('')

print('Accuracy %0.2f' % skln.accuracy\_score(labels, scores))

print(' ')

print(' Positive Negative')

print('Num case %6d' % metrics[3][1] + ' %6d' % metrics[3][0])

print('Precision %6.2f' % metrics[0][1] + ' %6.2f' % metrics[0][0])

print('Recall %6.2f' % metrics[1][1] + ' %6.2f' % metrics[1][0])

print('F1 def plot\_auc(labels, probs):

## Compute the false positive rate, true positive rate

## and threshold along with the AUC

# labels are actual, probs (use col 1) for bad credit

fpr, tpr, threshold = skln.roc\_curve(labels, probs[:,1])

auc = skln.auc(fpr, tpr)

## Plot the result

plt.title('Receiver Operating Characteristic')

plt.plot(fpr, tpr, color = 'orange', label = 'AUC = %0.2f' % 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()

#plot\_auc(y\_test, probabilities) %6.2f' % metrics[2][1] + ' %6.2f' % metrics # predict based on x\_test values

probabilities = logistic\_mod.predict\_proba(x\_test)

print(probabilities[:5,:]) #first 5 values

[2][0])

threshold = 0.5

print("threshold value: ", threshold)

scores=score\_model(probabilities,threshold)

print\_metrics(y\_test, scores)

plot\_auc(y\_test, probabilities) hresholdList = np.arange(0,1.1,0.1)

for threshold in thresholdList:

print("threshold value: ", threshold)

scores = score\_model(probabilities, threshold)

print\_metrics(y\_test, scores)

#plot\_auc(y\_test, probabilities)

print(80\*'-') # pick good threshold value

threshold = 0.64

print("threshold value: ", threshold)

scores = score\_model(probabilities, threshold)

print\_metrics(y\_test, scores)

plot\_auc(y\_test, probabilities)

Annexure3

Script 3:

Task4

#split the dataset with 70:30 based Bike buyer and create some random sample if needed

nr.seed(777)

labels=np.array(np.log(df['AveMonthSpend']))

print(labels)

indx=range(features.shape[0])

test\_size=int(0.3\*features.shape[0])

indx=ms.train\_test\_split(indx,test\_size=test\_size)

x\_train=features[indx[0],:]

y\_train=np.ravel(labels[indx[0]])

x\_test=features[indx[1],:]

y\_test=np.ravel(labels[indx[1]])

print(x\_train.shape,y\_train.shape)

print(x\_test.shape,y\_test.shape)

print(start\_num\_idx)

apply scaling of numerrical columns

scaler=preprocessing.StandardScaler().fit(x\_train[:,start\_num\_idx:])

x\_train[:,start\_num\_idx:]=scaler.transform(x\_train[:,start\_num\_idx:])

x\_test[:,start\_num\_idx:]=scaler.transform(x\_test[:,start\_num\_idx:])

print(x\_train.shape)

x\_train[:3,:]

#Task5 Build the linear Regression model

lin\_mod=linear\_model.LinearRegression(fit\_intercept=False)

lin\_mod.fit(x\_train,y\_train)

#Task6 Score Evalute the model and print the metrics

print(lin\_mod.intercept\_)

print(lin\_mod.coef\_,lin\_mod.coef\_.shape[0])

def print\_metrics(y\_true, y\_predicted, n\_parameters):

## First compute R^2 and the adjusted R^2

r2 = skln.r2\_score(y\_true, y\_predicted)

r2\_adj = r2 - (n\_parameters - 1)/(y\_true.shape[0] - n\_parameters) \* (1 - r2)

##Print the usual metrics and the R^2 values

print('Mean Square Error = ' + str(skln.mean\_squared\_error(y\_true, y\_predicted)))

print('Root Mean Square Error = ' + str(math.sqrt(skln.mean\_squared\_error(y\_true, y\_predicted))))

print('Mean Absolute Error = ' + str(skln.mean\_absolute\_error(y\_true, y\_predicted)))

print('Median Absolute Error = ' + str(skln.median\_absolute\_error(y\_true, y\_predicted)))

print('R^2 = ' + str(r2))

print('Adjusted R^2 = ' + str(r2\_adj))

y\_score = lin\_mod.predict(x\_test)

y\_score\_untransform = np.exp(y\_score)

y\_test\_untransform = np.exp(y\_test)

print(y\_score\_untransform)

print(y\_test\_untransform)

print\_metrics(y\_test\_untransform, y\_score\_untransform, lin\_mod.coef\_.shape[0]+1) # coef + 1 intercept

def hist\_resids(y\_test, y\_score):

## first compute vector of residuals.

resids = np.subtract(y\_test.reshape(-1,1), y\_score.reshape(-1,1))

## now make the residual plots

sns.distplot(resids)

plt.title('Histogram of residuals')

plt.xlabel('Residual value')

plt.ylabel('count')

#hist\_resids(y\_test, y\_score)

hist\_resids(y\_test def resid\_qq(y\_test, y\_score):

## first compute vector of residuals.

resids = np.subtract(y\_test.reshape(-1,1), y\_score.reshape(-1,1))

## now make the residual plots

ss.probplot(resids.flatten(), plot = plt)

plt.title('Residuals vs. predicted values')

plt.xlabel('Predicted values')

plt.ylabel('Residual')

resid\_qq(y\_test\_untransform, y\_score\_untransform)

#resid\_qq(y\_test, y\_score) \_untransform, y\_score\_untransform)

def resid\_plot(y\_test, y\_score):

## first compute vector of residuals.

resids = np.subtract(y\_test.reshape(-1,1), y\_score.reshape(-1,1))

## now make the residual plots

sns.regplot(y\_score, resids, fit\_reg=False)

plt.title('Residuals vs. predicted values')

plt.xlabel('Predicted values')

plt.ylabel('Residual')

#resid\_plot(y\_test, y\_score)

resid\_plot(y\_test\_untransform, y\_score\_untransform) # this portion to read AW\_test.csv, clean and te st

missing\_value\_formats=['n.a.','?','NA','n/a','na','--', 'null']

#df.to\_csv('AW\_test')

aw\_test = pd.read\_csv('AW\_test.csv', na\_values=missing\_value\_formats)

aw\_test.drop\_duplicates(subset='CustomerID', keep='last', inplace=True)

print(aw\_test.shape, aw\_test.columns)

aw\_test.drop(columns=['Title','MiddleName','Suffix','AddressLine2'], inplace=True)

aw\_tes def convert\_bdate\_age(birthDate, collectionDate):

diff = collectionDate - birthDate

age = diff.dt.days.div(365).astype(int)

return age

aw\_cDate = pd.to\_datetime('1998-01-01')

aw\_test['BirthDate'] = pd.to\_datetime(df['BirthDate'])

aw\_bDate = aw\_test['BirthDate']

aw\_age = convert\_bdate\_age(aw\_bDate, aw\_cDate)

aw\_test['Age'] = aw\_age

print(aw\_test['Age'])t.isnull().sum()umGroups = 10

bins = [0,25,45,55,120]

aw\_ageCats = pd.cut(aw\_test['Age'], bins=bins, labels=['0-25','25-45','45-55','55-120'])

aw\_test['AgeGroup'] = aw\_ageCats

aw\_test.to\_csv('clean\_aw\_test.csv') aw\_test = pd.read\_csv('clean\_aw\_test.csv')

aw\_test.columns #there are NO BikeBu

yer and AveMonthSpend # create a list of empty lists to store features

aw\_alist = [ [] for \_ in range(aw\_test.shape[0]) ]

aw\_features = np.array(aw\_alist)

for col in categorical\_columns:

temp=encode\_string(aw\_test[col])

print(col,temp.shape)

aw\_features=np.concatenate([aw\_features,temp],axis=1)

aw\_start\_num\_idx=aw\_features.shape[1]

print(aw\_features.shape)

print(aw\_features[:2,:])

aw\_features=np.concatenate([aw\_features,np.array(aw\_test[numerical\_columns])],axis=1)

aw\_features[:2,:]

aw\_features.shape

#Apply scalings of numerical columns

aw\_scaler=preprocessing.StandardScaler().fit(aw\_features[:,aw\_start\_num\_idx:])

aw\_features[:,aw\_start\_num\_idx:]=aw\_scaler.transform(aw\_features[:,aw\_start\_num\_idx:])

print(aw\_features.shape)

aw\_features[:3,:]

#get the predicted values and store then in to csv files

aw\_probabilities=logistic\_mod.predict\_proba(aw\_features)

aw\_scores=score\_model(aw\_probabilities,threshold)

pd\_scores=pd.Series(aw\_scores)

pd\_id=aw\_test['CustomerID']

newdf=pd.concat([pd\_id,pd\_scores],axis=1)

newdf.reset\_index()

np.savetxt('ClassificationResults\_anitha.csv',newdf,delimiter=',',fmt='%d')

aw\_scores = lin\_mod.predict(aw\_features)

aw\_scores\_untransform = np.exp(aw\_scores) #get back due to initial log

pd\_scores = pd.Series(aw\_scores\_untransform)

pd\_id = aw\_test['CustomerID']

newdf = pd.concat([pd\_id, pd\_scores], axis=0)

newdf.reset\_index()

np.savetxt('RegressionResults.csv', newdf, delimiter = ',', fmt='%d ')