**Name : ATCHE BHUVANA CHANDRA**

**CSU ID : 2754725**

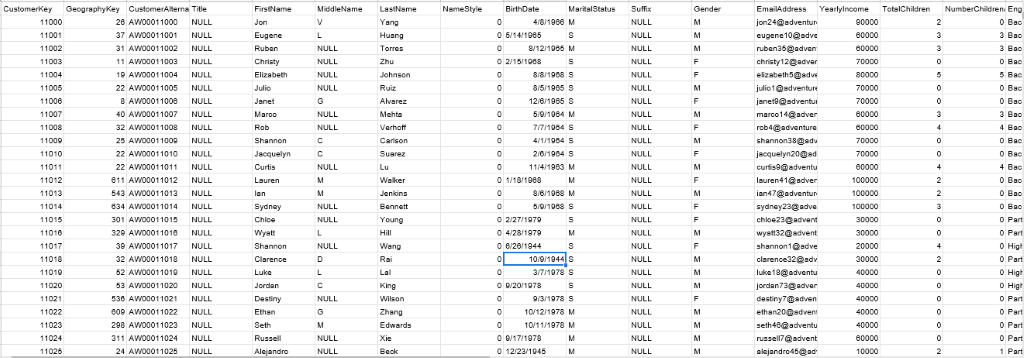
**LAB ASSIGNMENT 1 (CIS 660,Fall’19)**

**Part 1:**

i),ii)

In this part we are supposed to go through all the attributes and their values to decide on the data we need to use in the process of deciding the bike buyers. Selecting a set of attributes only that would affect to predict future customers is the primary part.

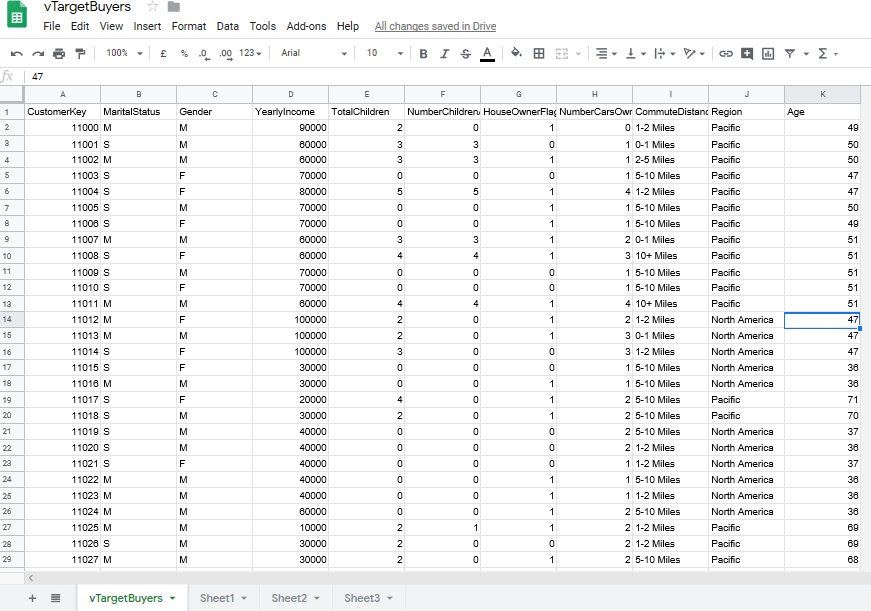
The below shown figure is the data set with complete data, from this we have to set useful data to predict based on the attribute’s real-life influence on a person’s decision to buy a bike.



Reasons for selecting the attributes to be in the dataset:

* CustomerKey - This is the primary key in the data base it helps us to uniquely identify each objects and when we work on the data and finally try to link them together it will be easy to link by using primary key.
* MaritalStatus – Having a knowledge on marital status will provide an idea about his/her financial and decision-making power.
* Gender – Gender helps us to find a relation between mentality, physical and how likely it is for them to buy.
* YearlyIncome – Income is an important attribute for us to decide either certain customer can or cannot buy a bike.
* Total Children – Decisions can be different based on the Income and customers because either they can think that their children might use or just think that this is unnecessary to spend so I am going to use this attribute and Number of children at home attribute to decide.
* NumberChildrenathome – As specified in the above field it is going to be a helpful piece of information when combined with the Total Children.
* Houseownerflag- A person who owns a house has enough place to park the bike and also it is easy for him to operate.
* NumberCarsOwned- Based on the number of cars owned we can say how luxurious a person is and how needful it is for the person to own a bike.
* CommuteDistance – When used with the age attribute it is more likely to develop a relation between the person capacity by age.
* Age – Going to be one of the major roles to identify if a person can ride a bike.

After deciding on the attributes ,new dataset has been created using duplicate of the original dataset which consists of the above mentioned attributes.



iii)

Identification of data value types (Discrete, or Continuous , then Nominal,Ordinal,Interval,Ratio)

Defining the data types so that we can select which attribute falls under which data type

* Discrete : The data that is distinct, which cannot be measured but it can be counted that is representing information is Discrete.

1. Nominal : These are labels which can be represented as numerical values but that doesn't have any meaning.

For example let us consider Gender - F =0 & M=1 they can be just counted no need to be in an order changing F=1 and M = 0 will be the same.

1. Ordinal : These data type shares the same properties as the Nominal except the fact that the ordering of values matters in this in the main dataset we had their educational qualifications like Degree,High School and Elementary so if we want to assign Degree as 3 High School as 2 and Elementary as 1 the order of 3 2 1 matters because they have order of expression.

* Continuous : Continuous data represents measurements and therefore their values can’t be counted but they can be measured.

1. Interval : Interval values represent ordered units that have the same difference. For example commuting distance in the above data set.
2. Ratio : These are also ordered units that have the same difference .Ratio values are the same as interval values ,with the difference that they do have an absolute zero.

Now naming each of the attribute with datatype

Column Content Type Data Type

Age Continuous,Interval Long

Commute Distance Discrete,Ordinal Text

CustomerKey Key Long

Gender Discrete,Nominal Text

House Owner Flag Discrete,Nominal Text

Marital Status Discrete,Nominal Text

Number Cars Owned Discrete,Ordinal Long

Number Children At Home Discrete,Ordinal Long

Region Discrete,Nominal Text

Total Children Discrete,Ordinal Long

Yearly Income Continuous,Ratio Double

Part 2.

i)

Handling null values can be done in a whole finding number of null values in each of the column.

file\_handler = open("vTargetBuyers.csv","r")

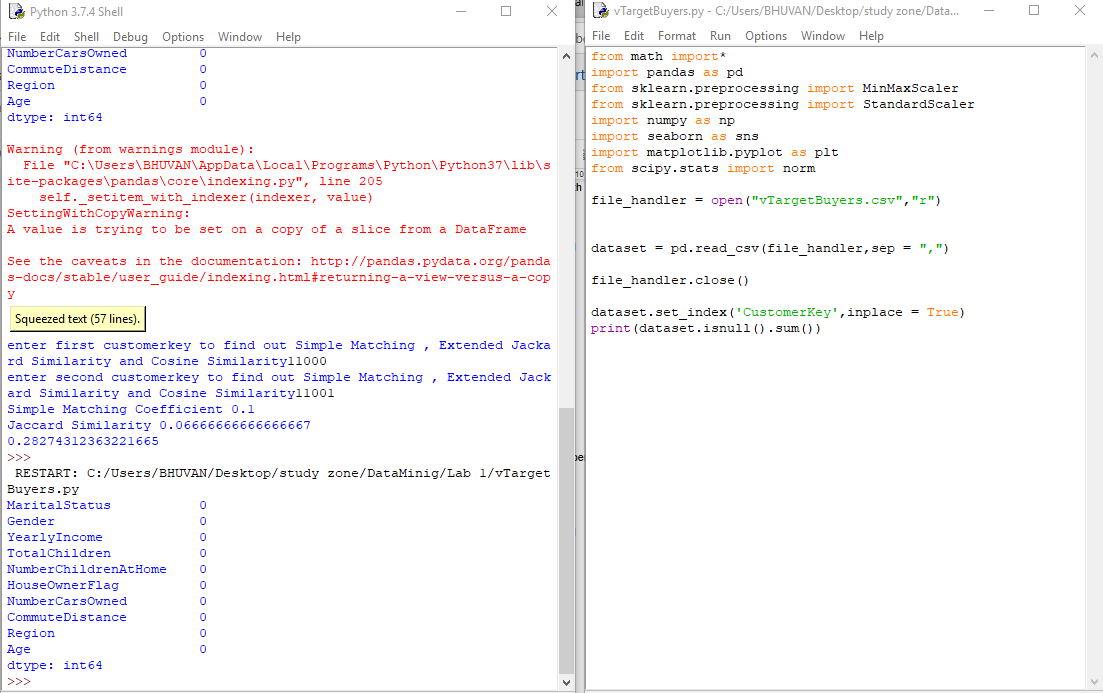
dataset = pd.read\_csv(file\_handler,sep = ",")

file\_handler.close()

dataset.set\_index('CustomerKey',inplace = True)

print(dataset.isnull().sum())

The above lines are used to find if there are any NULL values in the data set.



Handling Discrete Nominal data by banarization replacing them by 0 and 1

'''Converting Gender to binary values replacing Male attribute by 1 and Female by 0 '''

dataset.Gender.loc[dataset.Gender == 'M'] = 1

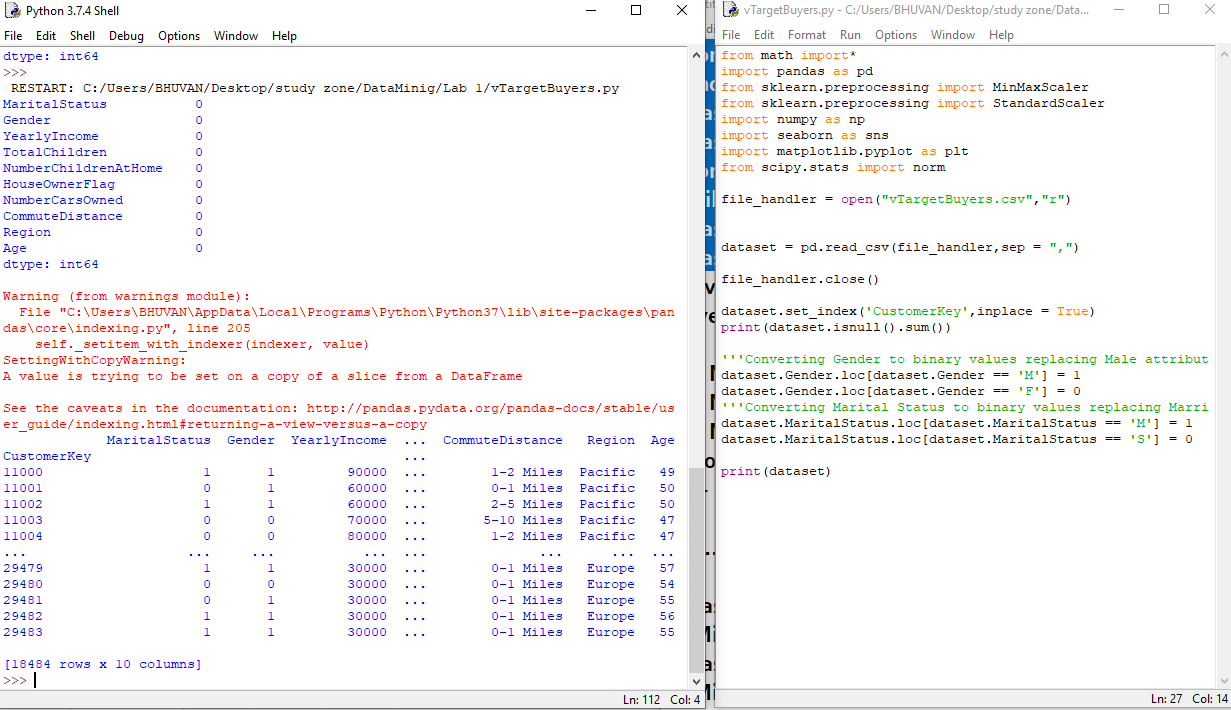
dataset.Gender.loc[dataset.Gender == 'F'] = 0

'''Converting Marital Status to binary values replacing Married attribute by 1 and Single by 0 '''

dataset.MaritalStatus.loc[dataset.MaritalStatus == 'M'] = 1

dataset.MaritalStatus.loc[dataset.MaritalStatus == 'S'] = 0

The above code sets all the Gender and MaritalStatus columns to binary values.



HouseOwnerFlag column is already in binary representation.

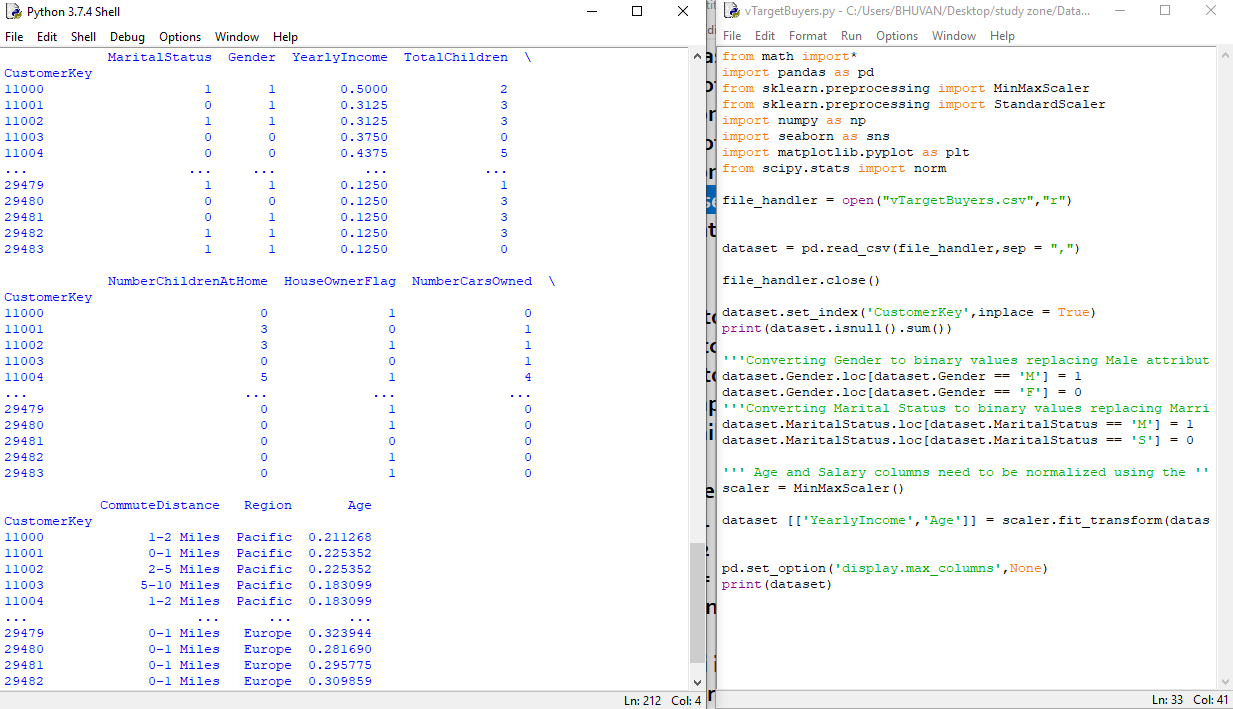
Continuous Values are Normalized and brought down to similar scale of measurement using minmax rescaling in python. The columns are Age and YearlyIncome

''' Age and Salary columns need to be normalized using the '''

scaler = MinMaxScaler()

dataset [['YearlyIncome','Age']] = scaler.fit\_transform(dataset[['YearlyIncome','Age']])

\*This piece of code set all the values of Age and YearlyIncome between 0-1.



Conversion of CommuteDistance column to numeric data for transformation

'''Giving Commute distance ranks based on the distance they travel

0-1 Miles as 1

1-2 Miles as 2

2-5 Miles as 3

5-10 Miles as 4

10+ Miles as 5

1,2.. 5 are in ascending order '''

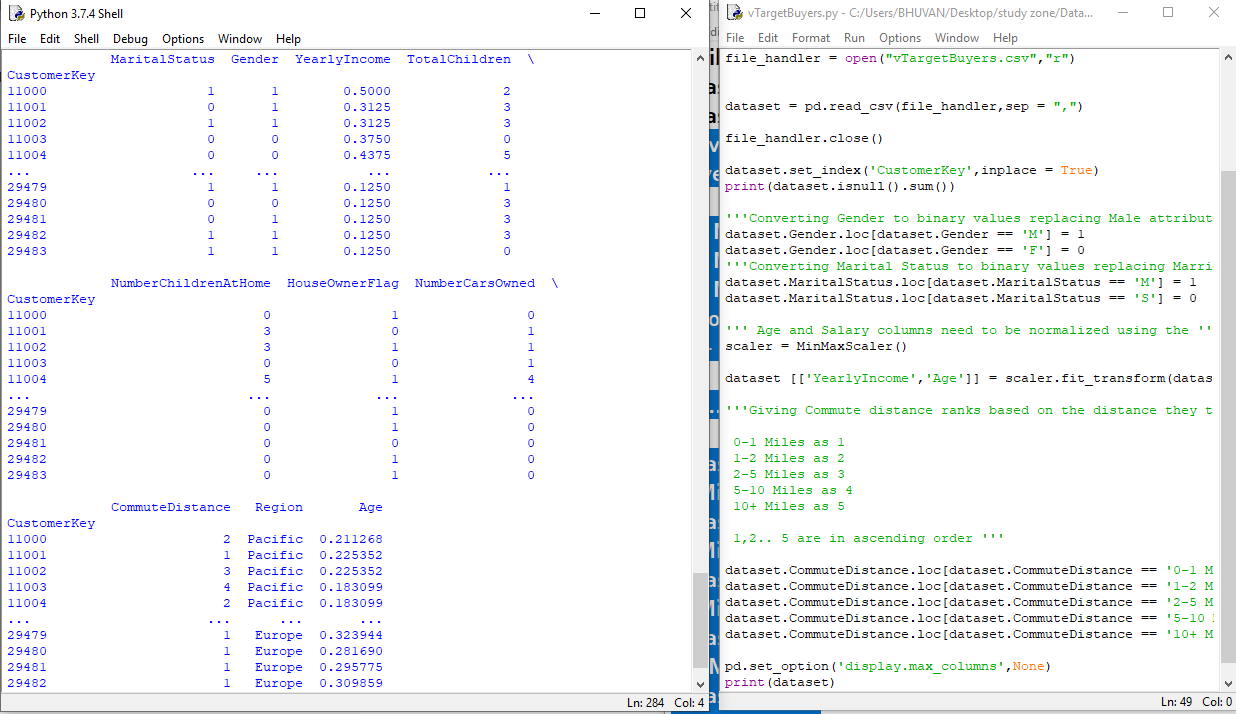
dataset.CommuteDistance.loc[dataset.CommuteDistance == '0-1 Miles'] = 1

dataset.CommuteDistance.loc[dataset.CommuteDistance == '1-2 Miles'] = 2

dataset.CommuteDistance.loc[dataset.CommuteDistance == '2-5 Miles'] = 3

dataset.CommuteDistance.loc[dataset.CommuteDistance == '5-10 Miles'] = 4

dataset.CommuteDistance.loc[dataset.CommuteDistance == '10+ Miles'] = 5



Now we have ordinal numeric data in the column CommuteDistance we can apply data transformation by mean median to set that to range between zero and one

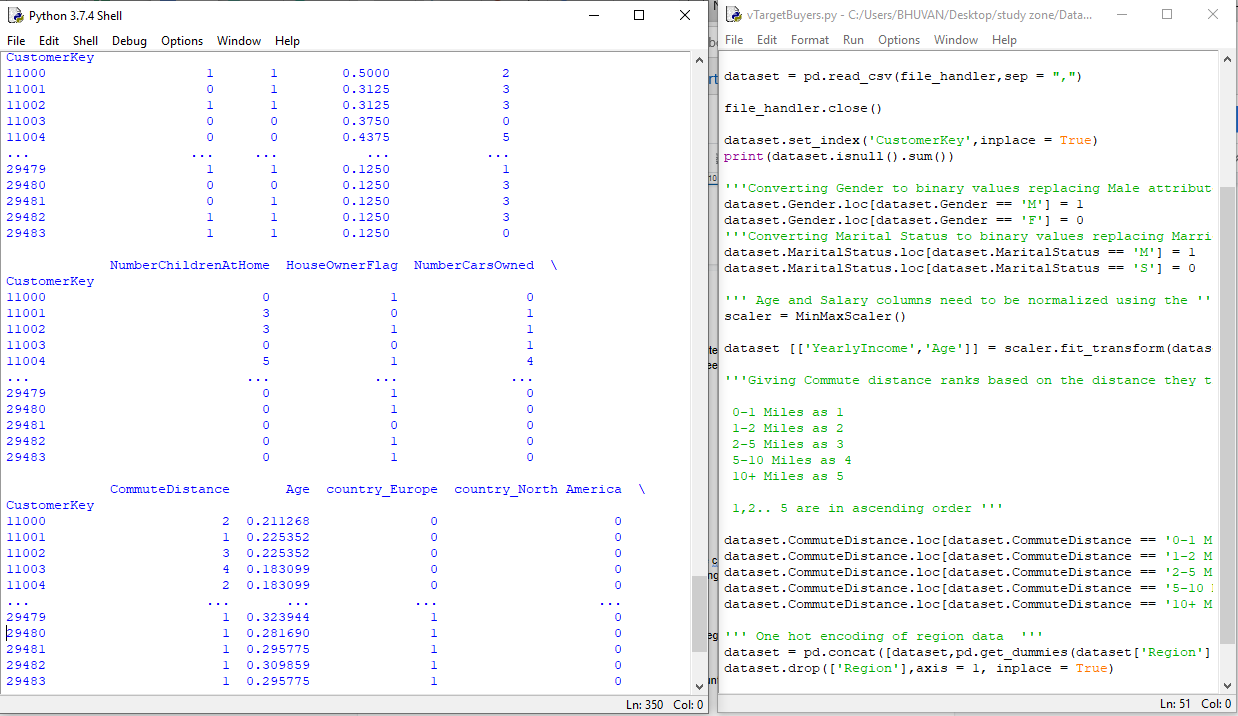
Region data need to be converted to number form but we cant just assign 1,2.. Because w don’t have priority based on region I used one hot encoding to Binarization to the lowest form.

''' One hot encoding of region data '''

dataset = pd.concat([dataset,pd.get\_dummies(dataset['Region'], prefix = 'country')],axis =1)

dataset.drop(['Region'],axis = 1, inplace = True)

This will create three new columns with flags for each country



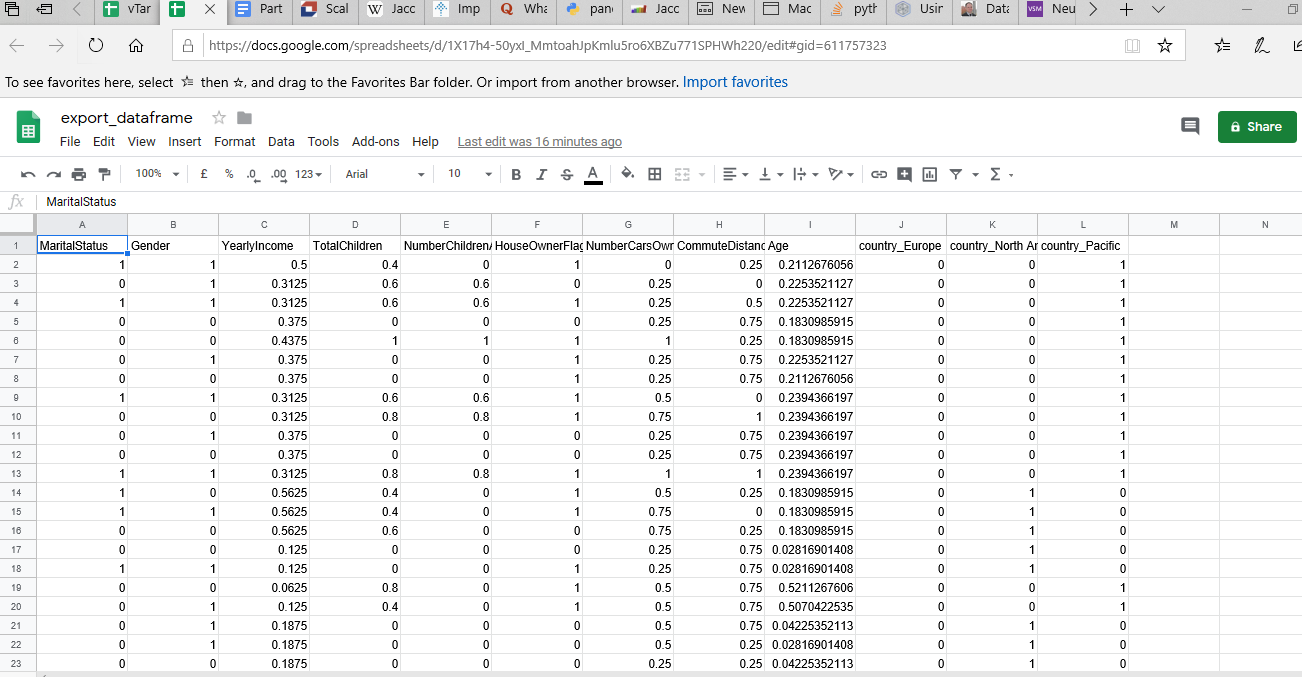
Standardization of Numeric Ordinal data columns that are CommuteDistance,Number,TotalChildren,NumberChildrenAtHome,NumberCarsOwned

Using the mean method and the scaler function in python they are brought into same dimension as the other values.

''Transfoming TotalChildren ,NumberCarsOwned, CommuteDistance'''

dataset[['TotalChildren','NumberChildrenAtHome','NumberCarsOwned','CommuteDistance']]=scaler.fit\_transform(dataset[['TotalChildren','NumberChildrenAtHome','NumberCarsOwned',

'CommuteeDistance']])



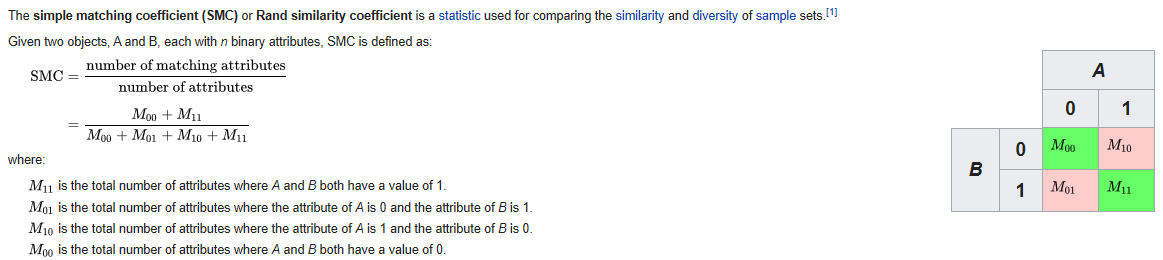
After the second part the transformation of all the columns is done and the output looks like above diagram ready to find out similarity.

The CustomerKey column is not present because of converting it to index in the python program.

Part 3 :

Now we have all the columns in the comparable form all the numeric attributes on the same scale like age and yearlyincome , Nominal attributes in binary format and discretized numeric attributes like number of cars owned , number of children , commute distance to an order ,where then can be measured.

Simple Matching :



sm1 = dataset.loc[int(customerkey1)]

sm2 = dataset.loc[int(customerkey2)]

res= sm1 == sm2

count = 0

i =0

print(res)

for i in range(0,8):

if res[i] == True:

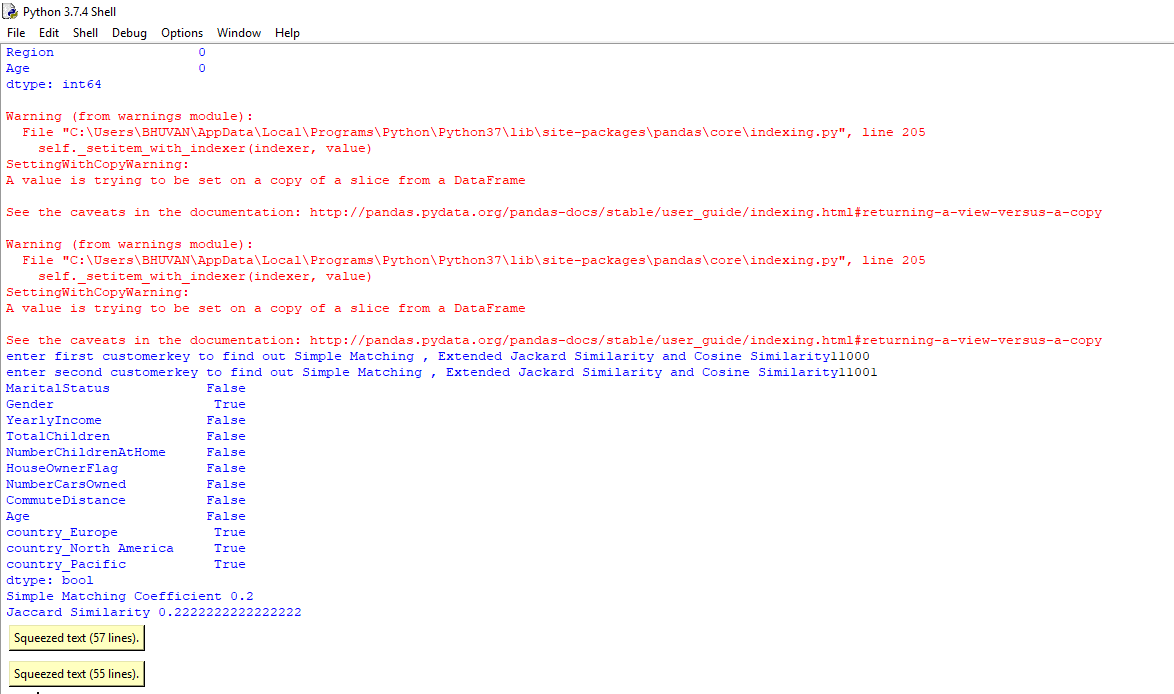
count =count+1

if (res[9] == True | res[10] == True | res[11] == True):

count = count + 1

print('Simple Matching Coefficient',count/10)

The above lines count the true values for each similarity/matching attributes and for the region if one of the column value matches then count is increased because the rest are dummy variables.



So it will ask us to enter two CustomerKeys to find out the similarity and after the input is provided it gives out Simple Matching Coefficient, Jaccard Similarity and Cosine Similarity.

Jaccard Similarity :

'''Result for Jaccard Similarity'''

intersection =0

union =0

for i in range(0,8):

if res[i] == True:

intersection = intersection + 1

union = union + 1

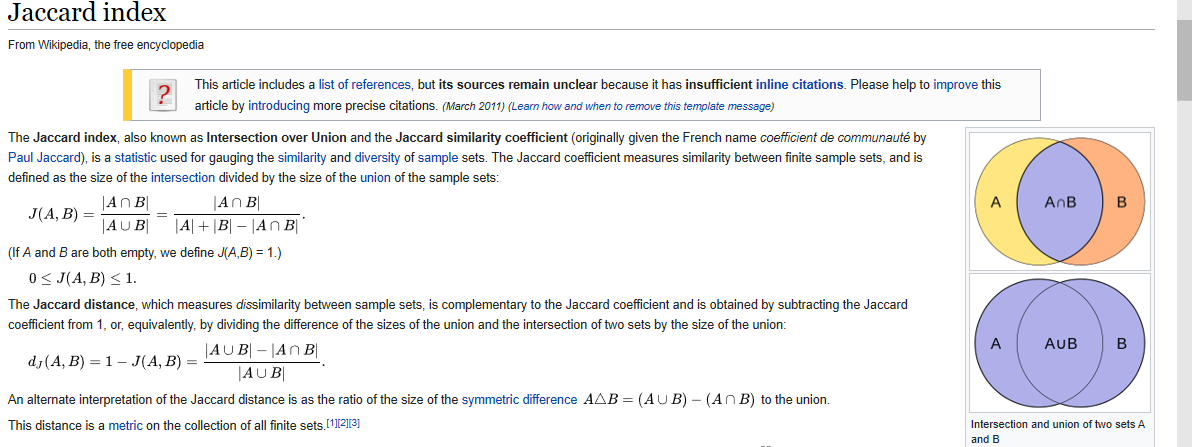
elif res[i]== False:

union = union +1

if (res[9] == True|res[10] == True |res[11] == True):

intersection = intersection +1

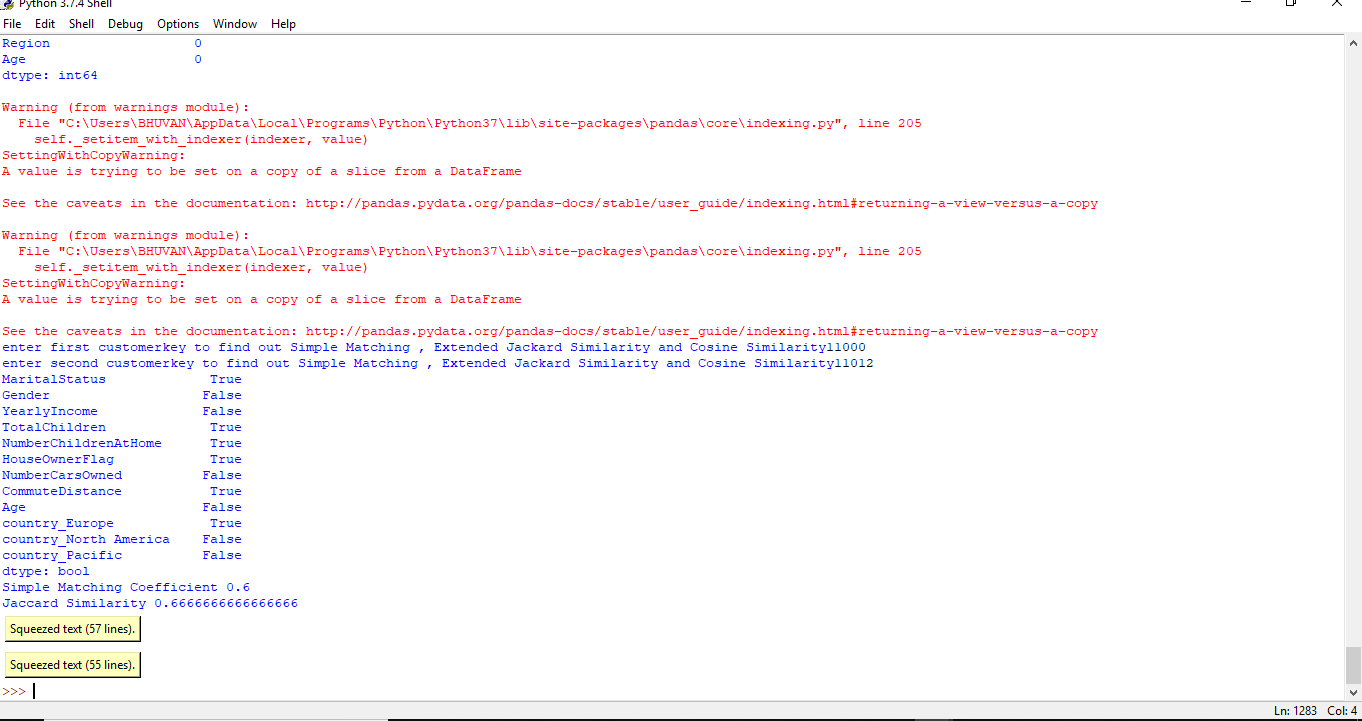
union = union +1

print('Jaccard Similarity',(intersection/union))

We compare each and every attribute of the row and if it happens in both then we are making it A intersection B and , if it happens in either one out of both then union.

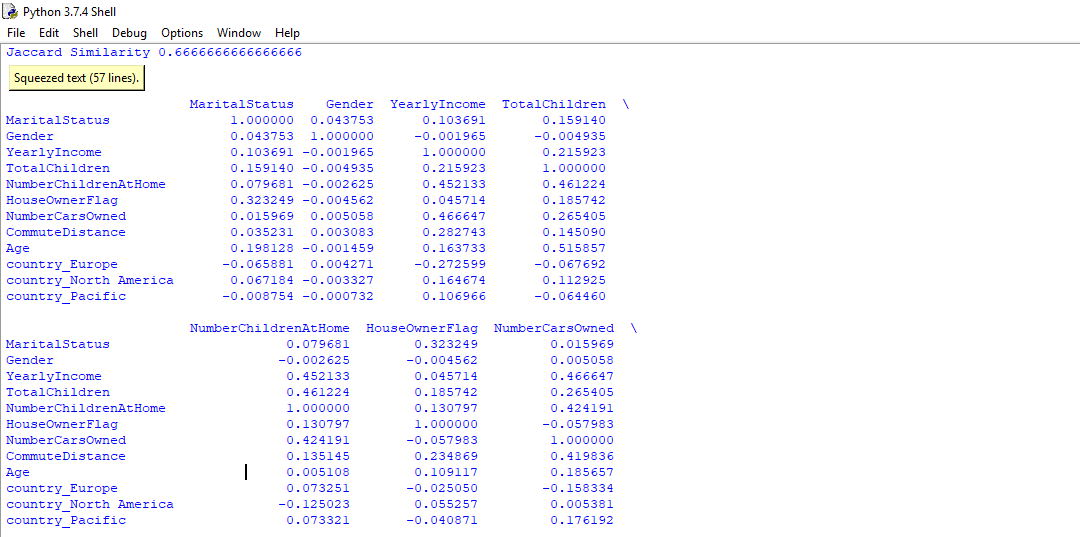
So if the value is similar include in intersection and union

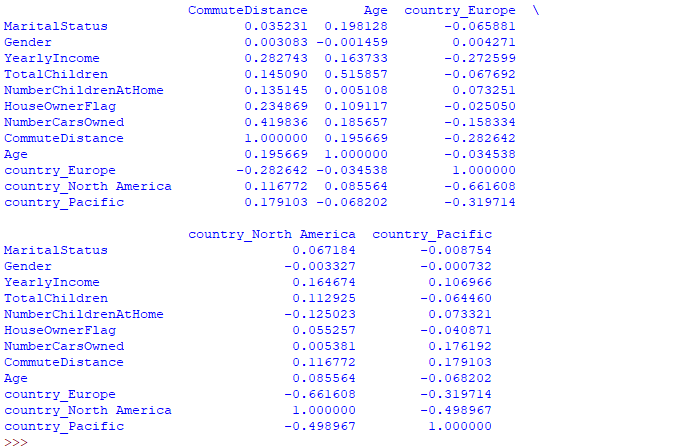
If the value is false or then only include in union.



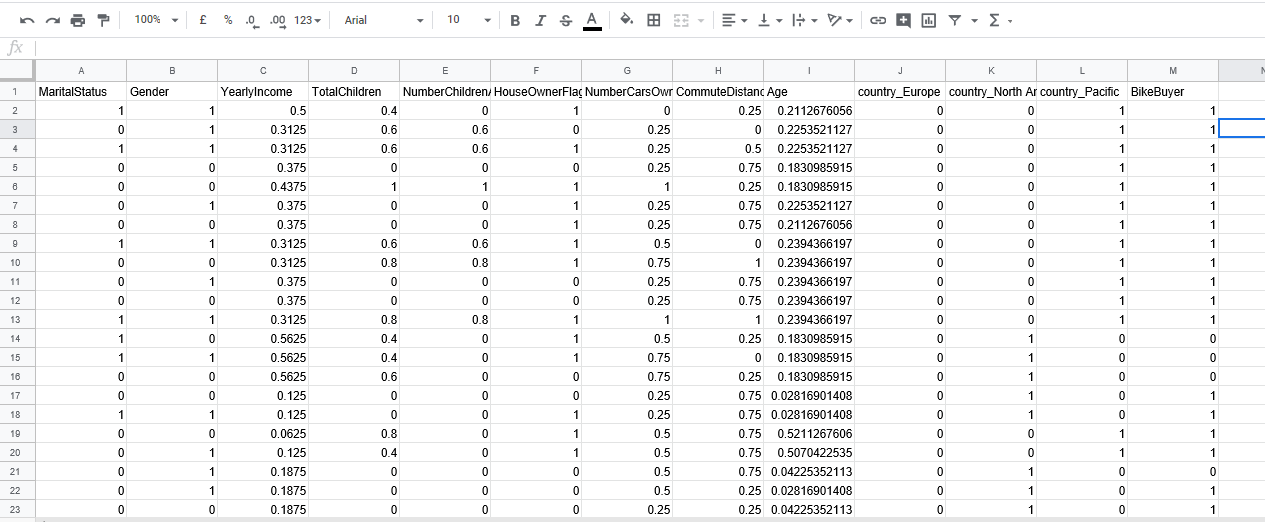
Part 4 :

i)After having the dataset with everything transformed then correlation matrix using all the features for the entire data set is build using dataset.corr()





After this bike buyer column is added to the output of the part 2



Now the whole dataset in moved again into the program and then two datasets are created based on BikeBuyer is 0 or 1

file\_handler = open("corelation.csv","r")

dataset1 = pd.read\_csv(file\_handler,sep = ",")

file\_handler.close()

dataset\_bike1 = dataset1[dataset1['BikeBuyer'] ==1]

dataset\_bike0= dataset1[dataset1['BikeBuyer'] ==0]

print(dataset\_bike1)

print(dataset\_bike0)

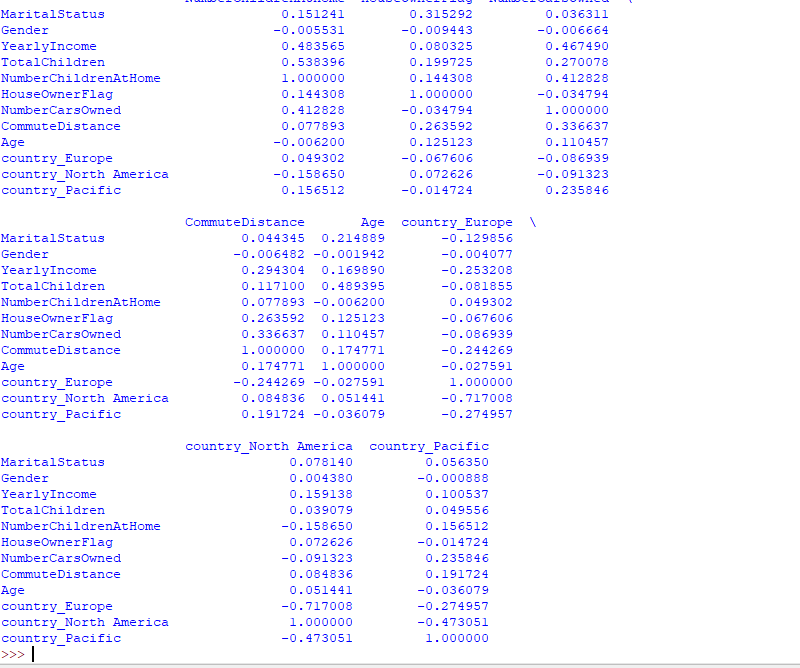
dataset\_bike1.drop(['BikeBuyer'],axis = 1, inplace = True)

dataset\_bike0.drop(['BikeBuyer'],axis = 1, inplace = True)

print(dataset\_bike1.corr())

print(dataset\_bike0.corr())

Dropping the bikebuyer column in the newly created data sets to avoid conflicts and generating correlation matrix for both the data sets when bike buyer is 0 and 1



from math import\*

import pandas as pd

from pandas import DataFrame

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import StandardScaler

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from scipy.stats import norm

file\_handler = open("vTargetBuyers.csv","r")

dataset = pd.read\_csv(file\_handler,sep = ",")

file\_handler.close()

dataset.set\_index('CustomerKey',inplace = True)

print(dataset.isnull().sum())

'''Converting Gender to binary values replacing Male attribute by 1 and Female by 0 '''

dataset.Gender.loc[dataset.Gender == 'M'] = 1

dataset.Gender.loc[dataset.Gender == 'F'] = 0

'''Converting Marital Status to binary values replacing Married attribute by 1 and Single by 0 '''

dataset.MaritalStatus.loc[dataset.MaritalStatus == 'M'] = 1

dataset.MaritalStatus.loc[dataset.MaritalStatus == 'S'] = 0

''' Age and Salary columns need to be normalized using the '''

scaler = MinMaxScaler()

dataset [['YearlyIncome','Age']] = scaler.fit\_transform(dataset[['YearlyIncome','Age']])

df\_normalized = pd.DataFrame(dataset)

'''Giving Commute distance ranks based on the distance they travel

0-1 Miles as 1

1-2 Miles as 2

2-5 Miles as 3

5-10 Miles as 4

10+ Miles as 5

1,2.. 5 are in ascending order '''

dataset.CommuteDistance.loc[dataset.CommuteDistance == '0-1 Miles'] = 1

dataset.CommuteDistance.loc[dataset.CommuteDistance == '1-2 Miles'] = 2

dataset.CommuteDistance.loc[dataset.CommuteDistance == '2-5 Miles'] = 3

dataset.CommuteDistance.loc[dataset.CommuteDistance == '5-10 Miles'] = 4

dataset.CommuteDistance.loc[dataset.CommuteDistance == '10+ Miles'] = 5

''' One hot encoding of region data '''

dataset = pd.concat([dataset,pd.get\_dummies(dataset['Region'], prefix = 'country')],axis =1)

dataset.drop(['Region'],axis = 1, inplace = True)

'''Transfoming TotalChildren ,NumberCarsOwned, CommuteDistance'''

dataset[['TotalChildren','NumberChildrenAtHome','NumberCarsOwned','CommuteDistance']]=scaler.fit\_transform(dataset[['TotalChildren','NumberChildrenAtHome','NumberCarsOwned','CommuteDistance']])

export\_csv = dataset.to\_csv(r'C:\Users\BHUVAN\Desktop\study zone\DataMinig\Lab 1\export\_dataframe.csv',index = None ,header = True)

customerkey1=input('enter first customerkey to find out Simple Matching , Extended Jackard Similarity and Cosine Similarity')

customerkey2 = input('enter second customerkey to find out Simple Matching , Extended Jackard Similarity and Cosine Similarity')

''' Result for Sample matching'''

sm1 = dataset.loc[int(customerkey1)]

sm2 = dataset.loc[int(customerkey2)]

res= sm1 == sm2

count = 0

i =0

print(res)

for i in range(0,8):

if res[i] == True:

count =count+1

if (res[9] == True | res[10] == True | res[11] == True):

count = count + 1

print('Simple Matching Coefficient',count/10)

'''Result for Jaccard Similarity'''

intersection =0

union =0

for i in range(0,8):

if res[i] == True:

intersection = intersection + 1

union = union + 1

elif res[i]== False:

union = union +1

if (res[9] == True|res[10] == True |res[11] == True):

intersection = intersection +1

union = union +1

print('Jaccard Similarity',(intersection/union))

pd.set\_option('display.max\_columns',None)

print(dataset.corr())

file\_handler = open("corelation.csv","r")

dataset1 = pd.read\_csv(file\_handler,sep = ",")

file\_handler.close()

dataset\_bike1 = dataset1[dataset1['BikeBuyer'] ==1]

dataset\_bike0= dataset1[dataset1['BikeBuyer'] ==0]

dataset\_bike1.drop(['BikeBuyer'],axis = 1, inplace = True)

dataset\_bike0.drop(['BikeBuyer'],axis = 1, inplace = True)

print(dataset\_bike1.corr())

print(dataset\_bike0.corr())