CIS 660 DATA MINING

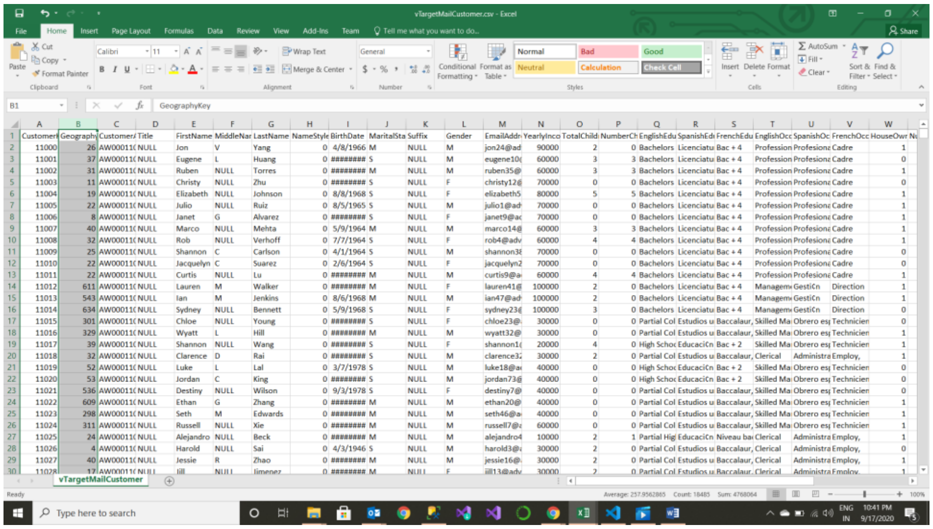
Lab 3: Designing and Building a Prediction Model with a Classifier and Feature Selection with PCA tools (Extra Credit)

Fathima Syeda

CSU ID-2790024

Part 1: Feature Selection, Cleaning, and Preprocessing to Construct an Input from Data Source:

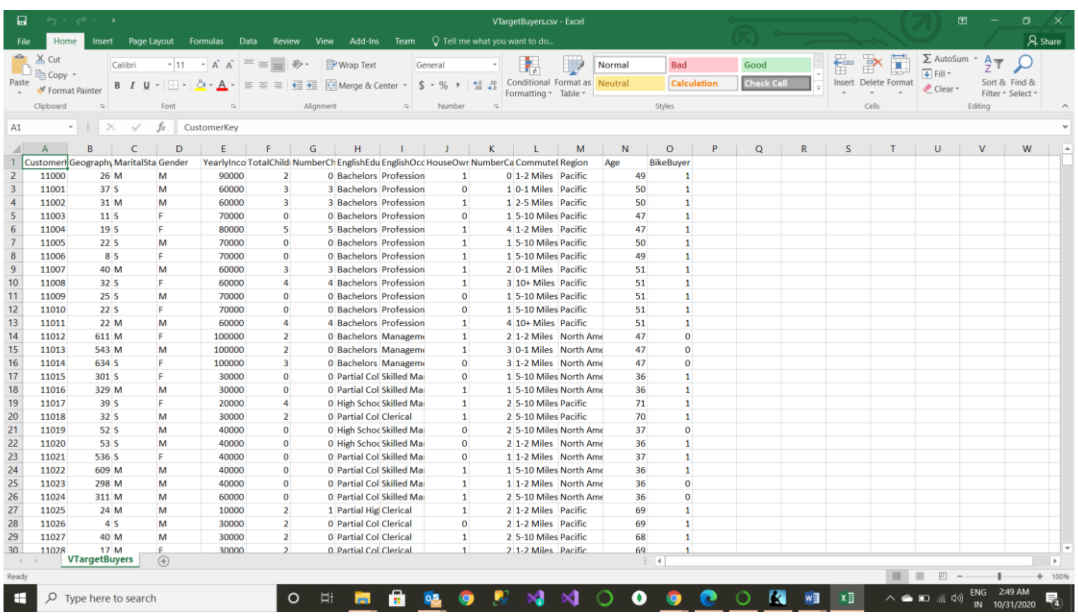
The given data VTargetCustomerMail data consists of data about customers who had already purchased bikes in the past. This data consists of 18000 objects and a total of 32 features.



We must get a list of potential buyers from the given data. Of the given features we choose those features that are most relevant in finding the target buyers list.

We choose 12 features that are most relevant in predicting future bike buyers.

The following shows the **VTargetBuyersList** file with the chosen features.



Of the selected features the **Yearly Income** is a continuous attributes and the rest of the attributes are discrete attributes.

Nominal: Marital Status, Gender, House Owner Flag, Region, Bike Buyer

Ordinal: Geography Key, English Education, Spanish Education, French Education, English Occupation, French Occupation, Spanish Occupation

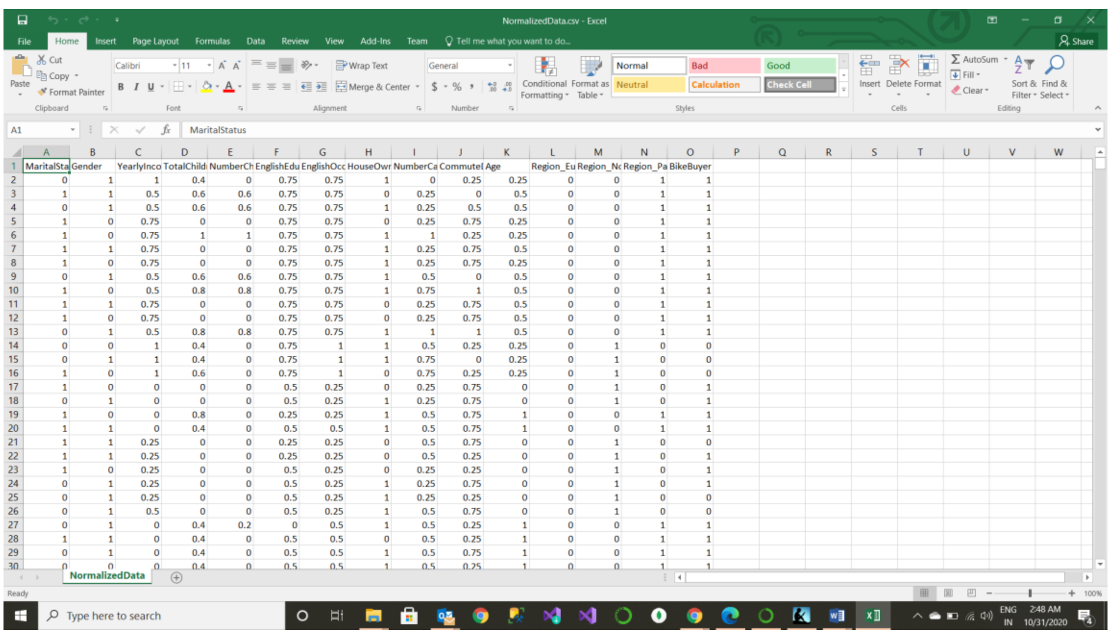
Interval: Yearly Income, Commute Distance

Ratio: Total Children, Number Cars Owned, Age

The chosen attributes are:, Marital Status, Gender, Yearly Income, Total Children, English Occupation, , House Owner Flag, Number Cars Owned, Commute Distance, Region, Age, Bike buyer.

- **Marital Status, Gender** –are Binary attributes and hence they are transformed using **<u>LabelEncoder</u>** form sklearn
- **YearlyIncome and Age** are continuous values and hence they are **Discretised** using Binning. Then these bins are again **Normalized using MinMax** scaling
- **English Occupation, Commute Distance**- Ordinal values hence we first rank them in order in numbers and then they are Normalized using **MinMax** scaling
- **Region** is a nominal data type so it is One Hot Encoded
- The rest of the attributes **Total Children House Owner Flag, Number Cars Owned** are numerical values and hence they are Normalized using **MinMax** scaling.

After all the transformations and pre-processing the transformed data is written into a csv file.



	MaritalSta	Gender	YearlyInco	TotalChild	NumberCh	EnglishEdu	EnglishOcc	HouseOwr	NumberCa	Commutel	Age	Region_Eu	Region_Nc	Region_Pa	BikeBuyer
2	0	1	1	0.4	0	0.75	0.75	1	0	0.25	0.25	0	0	1	1
3	1	1	0.5	0.6	0.6	0.75	0.75	0	0.25	0	0.5	0	0	1	1
4	0	1	0.5	0.6	0.6	0.75	0.75	1	0.25	0.5	0.5	0	0	1	1
5	1	0	0.75	0	0	0.75	0.75	0	0.25	0.75	0.25	0	0	1	1
6	1	0	0.75	1	1	0.75	0.75	1	1	0.25	0.25	0	0	1	1
7	1	1	0.75	0	0	0.75	0.75	1	0.25	0.75	0.5	0	0	1	1
8	1	0	0.75	0	0	0.75	0.75	1	0.25	0.75	0.25	0	0	1	1
9	0	1	0.5	0.6	0.6	0.75	0.75	1	0.5	0	0.5	0	0	1	1
10	1	0	0.5	0.8	0.8	0.75	0.75	1	0.75	1	0.5	0	0	1	1
11	1	1	0.75	0	0	0.75	0.75	0	0.25	0.75	0.5	0	0	1	1
12	1	0	0.75	0	0	0.75	0.75	0	0.25	0.75	0.5	0	0	1	1
13	0	1	0.5	0.8	0.8	0.75	0.75	1	1	1	0.5	0	0	1	1
14	0	0	1	0.4	0	0.75	1	1	0.5	0.25	0.25	0	1	0	0
15	0	1	1	0.4	0	0.75	1	1	0.75	0	0.25	0	1	0	0
16	1	0	1	0.6	0	0.75	1	0	0.75	0.25	0.25	0	1	0	0
17	1	0	0	0	0	0.5	0.25	0	0.25	0.75	0	0	1	0	1
18	0	1	0	0	0	0.5	0.25	1	0.25	0.75	0	0	1	0	1
19	1	0	0	0.8	0	0.25	0.25	1	0.5	0.75	1	0	0	1	1
20	1	1	0	0.4	0	0.5	0.5	1	0.5	0.75	1	0	0	1	1
21	1	1	0.25	0	0	0.25	0.25	0	0.5	0.75	0	0	1	0	0
22	1	1	0.25	0	0	0.25	0.25	0	0.5	0.25	0	0	1	0	1
23	1	0	0.25	0	0	0.5	0.25	0	0.25	0.25	0	0	1	0	1
24	0	1	0.25	0	0	0.5	0.25	1	0.25	0.75	0	0	1	0	1
25	0	1	0.25	0	0	0.5	0.25	1	0.25	0.25	0	0	1	0	0
26	0	1	0.5	0	0	0.5	0.25	1	0.5	0.75	0	0	1	0	0
27	0	1	0	0.4	0.2	0	0.5	1	0.5	0.25	1	0	0	1	1
28	1	1	0	0.4	0	0.5	0.5	0	0.5	0.25	1	0	0	1	1
29	0	1	0	0.4	0	0.5	0.5	1	0.5	0.75	1	0	0	1	1
30	0	0	0	0.4	0	0.5	0.5	1	0.5	0.25	1	0	0	1	1

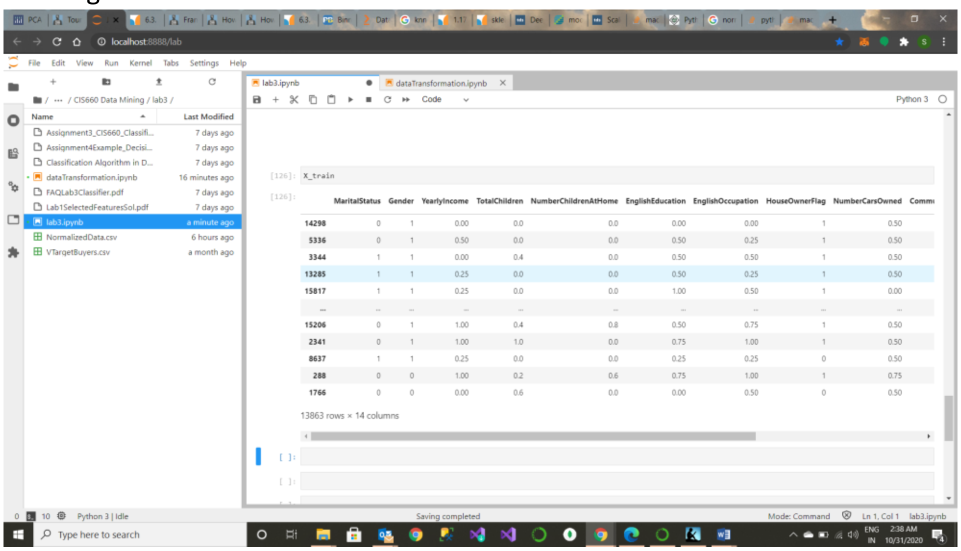
<u>**PART-2_1 Data Analytic Experiment with Two different Classifiers of Your Choice.**</u>

In this section we perform classification of the pre-processed data using two classifiers SVM and Neural Networks using various settings.

Before classifying the Transformed(pre-processed) data we remove the **BikeBuyer** column from the dataset as Labels.

The dataset is also <u>split into train-test dataset</u> using **sklearn.model_selection import train_test_split,** where the <u>test set size is 25 %</u> of the entire dataset and the rest is train set .

Training data consists of 13863 rows

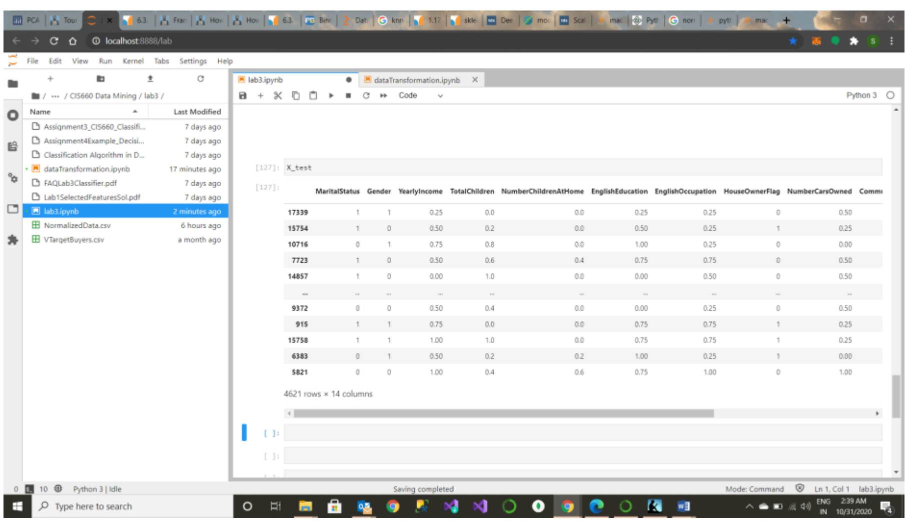


```
[126]: X_train
```

	MaritalStatus	Gender	YearlyIncome	TotalChildren	NumberChildrenAtHome	EnglishEducation	EnglishOccupation	HouseOwnerFlag	NumberCarsOwned
14298	0	1	0.00	0.0	0.0	0.00	0.00	1	0.50
5336	0	1	0.50	0.0	0.0	0.50	0.25	1	0.50
3344	1	1	0.00	0.4	0.0	0.50	0.50	1	0.50
13285	1	1	0.25	0.0	0.0	0.50	0.25	1	0.50
15817	1	1	0.25	0.0	0.0	1.00	0.50	1	0.00
...	...	...	...	...	...	...	...	...	...
15206	0	1	1.00	0.4	0.8	0.50	0.75	1	0.50
2341	0	1	1.00	1.0	0.0	0.75	1.00	1	0.50
8637	1	1	0.25	0.0	0.0	0.25	0.25	0	0.50
288	0	0	1.00	0.2	0.6	0.75	1.00	1	0.75
1766	0	0	0.00	0.6	0.0	0.00	0.50	0	0.50

13863 rows × 14 columns

Test data consists of 4621 rows



```
[127]: X_test
```

	MaritalStatus	Gender	YearlyIncome	TotalChildren	NumberChildrenAtHome	EnglishEducation	EnglishOccupation	HouseOwnerFlag	NumberCarsOwned
17339	1	1	0.25	0.0	0.0	0.25	0.25	0	0.50
15754	1	0	0.50	0.2	0.0	0.50	0.25	1	0.25
10716	0	1	0.75	0.8	0.0	1.00	0.25	0	0.00
7723	1	0	0.50	0.6	0.4	0.75	0.75	0	0.50
14857	1	0	0.00	1.0	0.0	0.00	0.50	0	0.50
...	...	...	...	...	...	...	...	...	...
9372	0	0	0.50	0.4	0.0	0.00	0.25	0	0.50
915	1	1	0.75	0.0	0.0	0.75	0.75	1	0.25
15758	1	1	1.00	1.0	0.0	0.75	0.75	1	0.25
6383	0	1	0.50	0.2	0.2	1.00	0.25	1	0.00
5821	0	0	1.00	0.4	0.6	0.75	1.00	0	1.00

4621 rows × 14 columns

SVM classifier :

The pre-processed data is fed into the SVM classifier on three times with various kernel settings. The kernel values used in the experiment are – linear, rbf and polynomial. The gamma value for SVM is set to 'scale' all throughout the classifications.

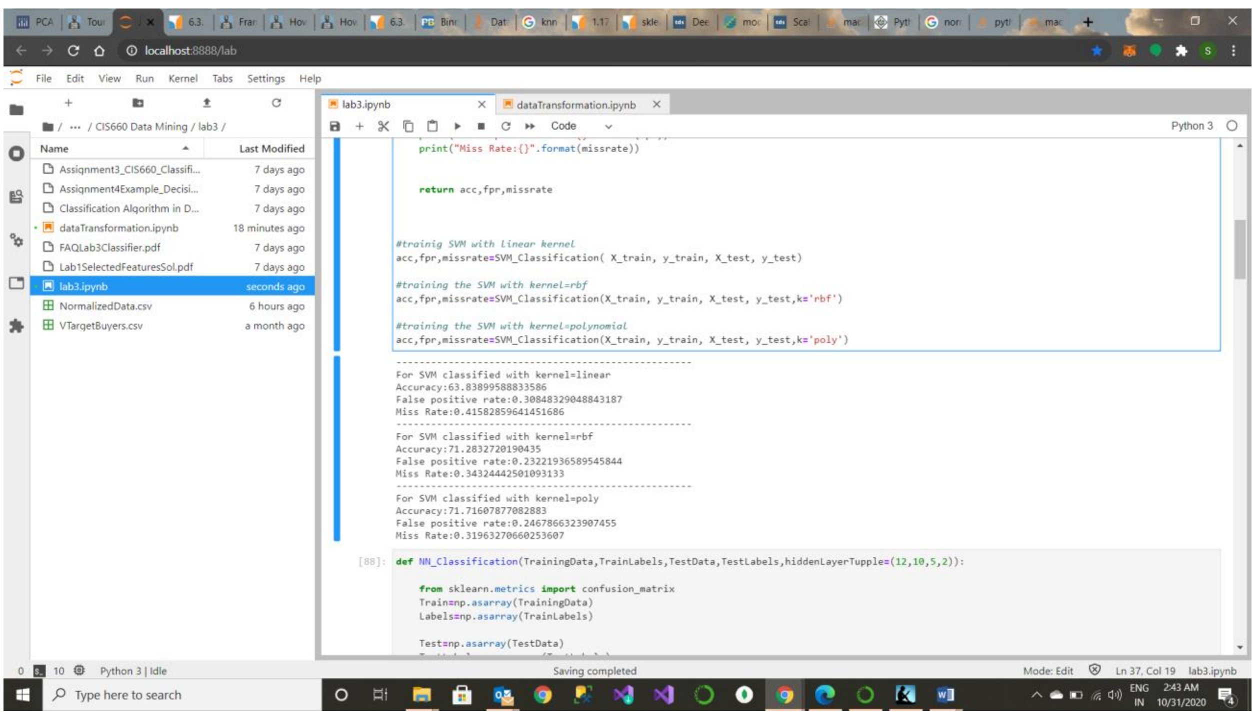
SVM classifier works by vectorization of each record. The following are the Accuracy and False positive and Miss Rate values when the data is classified using various Kernel settings of the SVM.

For SVM classified with kernel=linear
Accuracy:63.83899588833586
False positive rate:0.30848329048843187
Miss Rate:0.41582859641451686

For SVM classified with kernel=rbf
Accuracy:71.2832720190435
False positive rate:0.23221936589545844
Miss Rate:0.34324442501093133

For SVM classified with kernel=poly
Accuracy:71.71607877082883
False positive rate:0.2467866323907455
Miss Rate:0.31963270660253607

The highest Accuracy is with SVM kernels= rbf and polynomial. This is because the



Neural Network classifier :

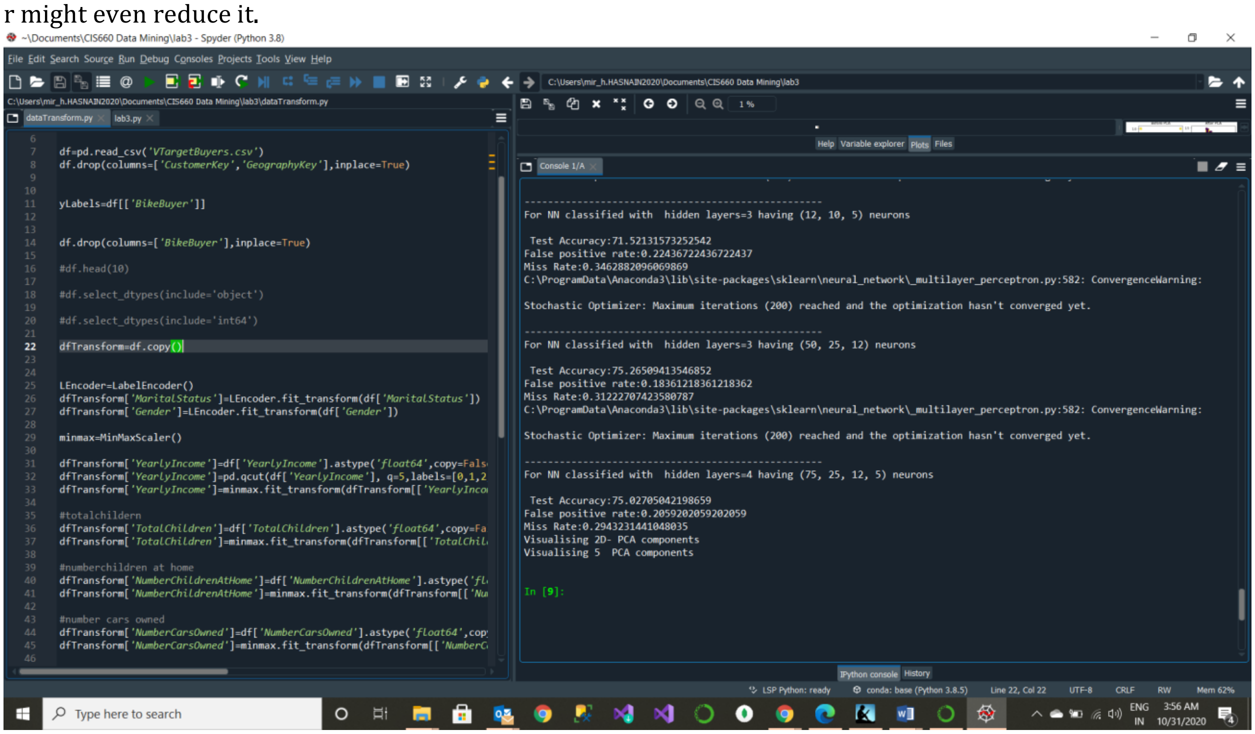
Three neural network topologies are designed with different hidden layers and different units in each layer. The neural network topologies are:

- NN with hidden layers=3 having (12, 10, 5) neurons
- NN classified with hidden layers=3 having (50, 25, 12) neurons
- NN classified with hidden layers=4 having (75, 25, 12, 5) neurons

\---
For NN classified with hidden layers=3 having (12, 10, 5) neurons
Accuracy:71.9974031594893
False positive rate:0.23005487547488393
Miss Rate:0.33259325044404975
\---
For NN classified with hidden layers=3 having (50, 25, 12) neurons
Accuracy:75.89266392555723
False positive rate:0.16040523427606584
Miss Rate:0.3259325044404973
\---
For NN classified with hidden layers=4 having (75, 25, 12, 5) neurons
Accuracy:75.80610257520017
False positive rate:0.25707049387927394
Miss Rate:0.22602131438721138

The highest accuracy is with the Neural network topology having hidden layers=3 having (50, 25, 12) units, f ollowed by a very close margin by the neural network having hidden layers=4 having (75, 25, 12, 5) units.

Thus we see that after a point the increase in number of layers doesn't impact the accuracy much and counter intuitively adding too many layers and units could increase the execution time but not increase the accuracy o r might even reduce it.



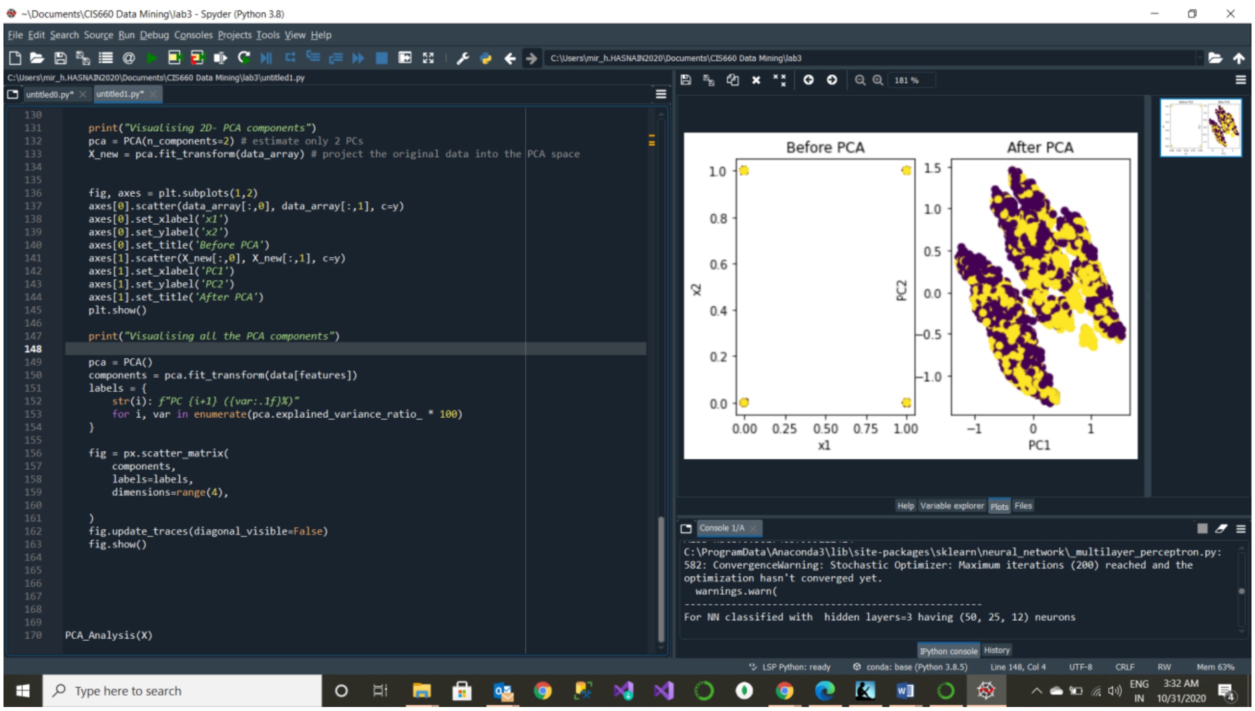
PART-2_3 Feature significance analysis with PCA tools (Extra Credit):

PCA technique is particularly useful in processing data where multi-colinearity exists between the features/variables.

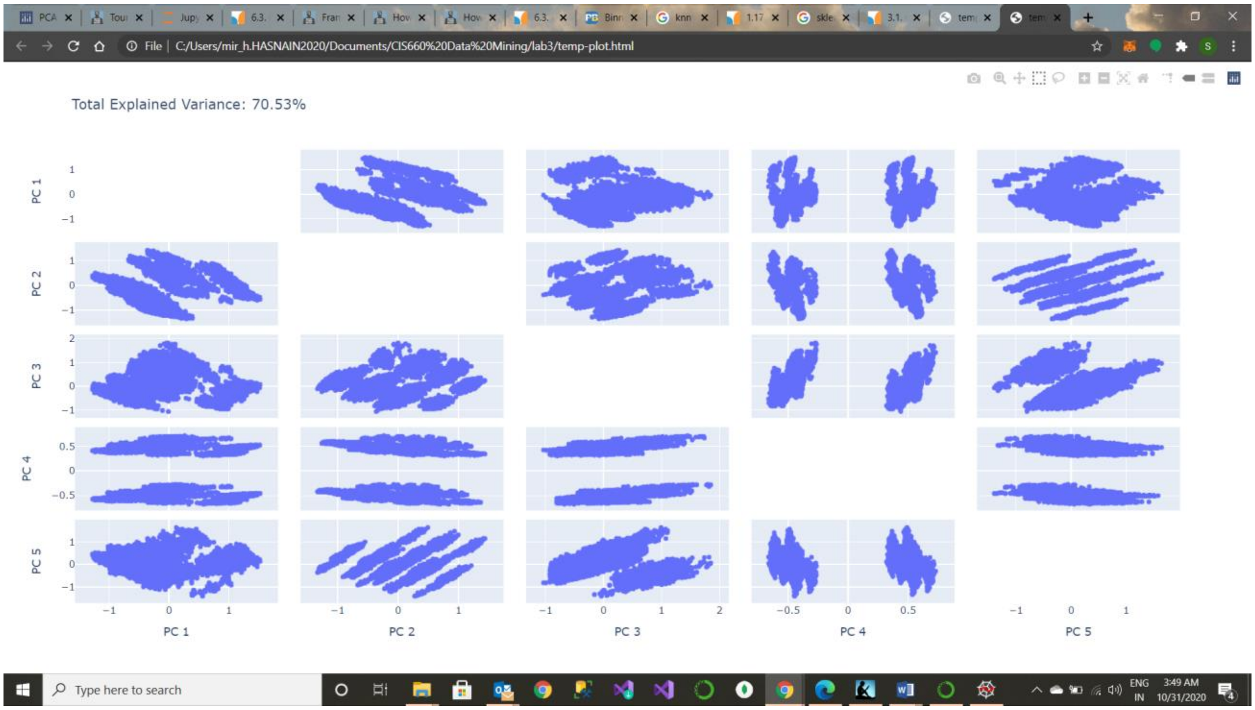
PCA analysis is done on 12 features of our pre-processed BikeBuyer dataset. The features selected are :

['MaritalStatus', 'Gender', 'YearlyIncome', 'TotalChildren',
'NumberChildrenAtHome', 'EnglishEducation', 'EnglishOccupation',
'HouseOwnerFlag', 'NumberCarsOwned', 'CommuteDistance', 'Age',
'Region_Europe', 'Region_North America', 'Region_Pacific']

In the following we plot a 2D plot by doing a PCA analysis for feature selection on our features using number of PCA components **nPCA=2**



We also show the PCA analysis with **n PCA=4** for four features of all the features in the dataset.



PART-3 Accuracy of models for each classifier with different Parameter settings or different transformation method.

1. SVM

The highest Accuracy for the SVM classifier is with **kernels= rbf and polynomial.** This is because rbf is better suited to handle data with outliers.

1. For SVM classified with **kernel=linear**

Accuracy:63.83899588833586
False positive rate:0.30848329048843187
Miss Rate:0.41582859641451686

2. For SVM classified with **kernel=rbf**

Accuracy:71.2832720190435
False positive rate:0.23221936589545844
Miss Rate:0.34324442501093133

3. For SVM classified with **kernel=poly**

Accuracy:71.71607877082883
False positive rate:0.2467866323907455
Miss Rate:0.31963270660253607

2. Neural Network classifier :

The highest accuracy is with the Neural network topology having hidden layers=3 having (50, 25, 12) units, fo llowed by a very close margin by the neural network having hidden layers=4 having (75, 25, 12, 5) units.

Thus we see that after a point the increase in number of layers doesn't impact the accuracy much and counter intuitively adding too many layers and units could increase the execution time but not increase the accuracy o r might even reduce it.

For NN classified with

1. hidden layers=3 having (12, 10, 5) neurons

Accuracy:71.9974031594893
False positive rate:0.23005487547488393
Miss Rate:0.33259325044404975

2. hidden layers=3 having (50, 25, 12) neurons

Accuracy:75.89266392555723
False positive rate:0.16040523427606584
Miss Rate:0.3259325044404973

3. hidden layers=4 having (75, 25, 12, 5) neurons

Accuracy:75.80610257520017
False positive rate:0.25707049387927394
Miss Rate:0.22602131438721138

PART -4

- NN have better accuracy than SVM on the same data because SVM uses only a subset of a dataset as training data thus the number of observations required to train an SVM isn't high whereas a NN uses the entire dataset.
- SVM works best when the preprocessed input data is vectorized whereas Neural networks work best with normalized data usually between [0,1]
- The highest accuracy of 75.89 % on the same dataset was yielded by the Neural network classifier with hidden layers=3 having (50, 25, 12) units in the layers respectively.
- The lowest accuracy 63.8% is found in the SVM classifier trained with kernel=linear, however the accuracy with rbf is much better, this is because rbf is better suited to handle data with outliers.
- The highest accuracy is with the Neural network topology having hidden layers=3 having (50, 25, 12) units, followed by a very close margin by the neural network having hidden layers=4 having (75, 25, 12, 5) units.
- after a point the increase in number of layers in the Neural Network didn't increase the accuracy of the BikeBuyer dataset much
- NN yielded better results when the optimizer used was Adam or lbfgs instead of sgd on the bike buyer dataset.

CODE:

The code is in 2 files :

1. **Datatransformation.py**

```
import pandas as pd

import numpy as np

from sklearn import preprocessing

from sklearn.preprocessing import OneHotEncoder, LabelEncoder, MinMaxScaler



df=pd.read_csv('VTargetBuyers.csv')

df.drop(columns=['CustomerKey','GeographyKey'],inplace=True)

yLabels=df[['BikeBuyer']]

df.drop(columns=['BikeBuyer'],inplace=True)

dfTransform=df.copy()


LEncoder=LabelEncoder()

dfTransform['MaritalStatus']=LEncoder.fit_transform(df['MaritalStatus'])

dfTransform['Gender']=LEncoder.fit_transform(df['Gender'])


minmax=MinMaxScaler()


dfTransform['YearlyIncome']=df['YearlyIncome'].astype('float64',copy=False)

dfTransform['YearlyIncome']=pd.qcut(df['YearlyIncome'], q=5,labels=[0,1,2,3,4])

dfTransform['YearlyIncome']=minmax.fit_transform(dfTransform[['YearlyIncome']])


#totalchildern

dfTransform['TotalChildren']=df['TotalChildren'].astype('float64',copy=False)
```

```
dfTransform['TotalChildren']=minmax.fit_transform(dfTransform[['TotalChildren']])

#numberchildren at home
dfTransform['NumberChildrenAtHome']=df['NumberChildrenAtHome'].astype('float64',copy=False)
dfTransform['NumberChildrenAtHome']=minmax.fit_transform(dfTransform[['NumberChildrenAtHome']
])

#number cars owned
dfTransform['NumberCarsOwned']=df['NumberCarsOwned'].astype('float64',copy=False)
dfTransform['NumberCarsOwned']=minmax.fit_transform(dfTransform[['NumberCarsOwned']])

dfTransform['CommuteDistance']=df['CommuteDistance'].replace(['0-1 Miles','1-2 Miles','2-5 Miles', '5-
10 Miles', '10+ Miles'],[0,1,2,3,4])
dfTransform['CommuteDistance']=minmax.fit_transform(dfTransform[['CommuteDistance']])

#English education
dfTransform['EnglishEducation']=df['EnglishEducation'].replace(['Partial High School','High
School','Partial College', 'Bachelors', 'Graduate Degree'],[0,1,2,3,4])
dfTransform['EnglishEducation']=minmax.fit_transform(dfTransform[['EnglishEducation']])

#English occupation
dfTransform['EnglishOccupation']=df['EnglishOccupation'].replace(['Manual','Skilled
Manual','Clerical','Professional', 'Management'],[0,1,2,3,4])
dfTransform['EnglishOccupation']=minmax.fit_transform(dfTransform[['EnglishOccupation']])



OHE=pd.get_dummies(df['Region'],prefix='Region')
dfTransform=dfTransform.drop('Region',axis=1)
dfTransform=dfTransform.join(OHE)
```

```
dfTransform['Age']=pd.qcut(df['Age'], q=5,labels=[0,1,2,3,4])
dfTransform['Age']=minmax.fit_transform(dfTransform[['Age']])

#dfTransform.head(12)

#joining class labels to the transfomed df
dfTransform=dfTransform.join(yLabels)

dfTransform.to_csv('NormalizedData.csv',index=False)
```

2. **lab3.py**

```
import pandas as pd
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from numpy import absolute,mean,std

df=pd.read_csv('NormalizedData.csv')

y=df['BikeBuyer'].ravel()
X=df.drop(columns=['BikeBuyer'])
```

```
X_train, X_test, y_train, y_test = train_test_split( X, y)



def SVM_Classification(TrainingData,TrainLabels,TestData,TestLabels,k='linear'):

    from sklearn.metrics import confusion_matrix
    Train=np.asarray(TrainingData)
    Labels=np.asarray(TrainLabels)

    Test=np.asarray(TestData)
    TestLabels=np.asarray(TestLabels)

    clf=SVC(kernel=k,gamma='scale')
    clf.fit(Train,Labels)

    prediction=clf.predict(Test)

    #true positve, true negative, false positive, false negative
    tn, fp, fn, tp = confusion_matrix(TestLabels, prediction).ravel()
    #accuracy
    acc=(tp+tn)/(tp+tn+fp+fn)
    acc=acc*100
    #false positive rate & miss rate
    fpr=fp/(fp+tn)
    missrate=fn/(tp+fn)
```

```
    print("-----------------------------------------------------")
    print("For SVM classified with kernel={}".format(k))
    #print("Cross Validation of Train Dataset ")
    #scores = cross_val_score(clf, X, y, cv=5)
    # summarize the model performance
    #print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

    print("\nTest Data")
    print("\n Test Accuracy:{}".format(acc))
    print("False positive rate:{}".format(fpr))
    print("Miss Rate:{}".format(missrate))




    return acc,fpr,missrate



#trainig SVM with linear kernel
acc,fpr,missrate=SVM_Classification( X_train, y_train, X_test, y_test)

#training the SVM with kernel=rbf
acc,fpr,missrate=SVM_Classification(X_train, y_train, X_test, y_test,k='rbf')

#training the SVM with kernel=polynomial
acc,fpr,missrate=SVM_Classification(X_train, y_train, X_test, y_test,k='poly')
```

```
def NN_Classification(TrainingData,TrainLabels,TestData,TestLabels,hiddenLayerTupple=(12,10,5,2)):

    from sklearn.metrics import confusion_matrix
    Train=np.asarray(TrainingData)
    Labels=np.asarray(TrainLabels)

    Test=np.asarray(TestData)
    TestLabels=np.asarray(TestLabels)

    clf=MLPClassifier(solver='adam', alpha=1e-5, hidden_layer_sizes=hiddenLayerTupple,
random_state=1)
    clf.fit(Train,Labels)

    prediction=clf.predict(Test)

    #true positve, true negative, false positive, false negative
    tn, fp, fn, tp = confusion_matrix(TestLabels, prediction).ravel()
    #accuracy
    acc=(tp+tn)/(tp+tn+fp+fn)
    acc=acc*100
    #false positive rate & miss rate
    fpr=fp/(fp+tn)
    missrate=fn/(tp+fn)

    print("--------------------------------------------------")
    print("For NN classified with  hidden layers={} having {}
neurons".format(len(hiddenLayerTupple),hiddenLayerTupple))
```

```
    #print("Cross Validation of Train Dataset ")
    #scores = cross_val_score(clf, X, y, cv=5)
    # summarize the model performance
    #print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
    print("\n Test Accuracy:{}".format(acc))
    print("False positive rate:{}".format(fpr))
    print("Miss Rate:{}".format(missrate))


    return acc,fpr,missrate




#trainig NN with 4 layers
acc,fpr,missrate=NN_Classification( X_train, y_train, X_test, y_test,hiddenLayerTupple=(12,10,5))

#training the NN with 4 layers
acc,fpr,missrate=NN_Classification(X_train, y_train, X_test, y_test,hiddenLayerTupple=(50,25,12))

#training the NN with 4 layers
acc,fpr,missrate=NN_Classification(X_train, y_train, X_test, y_test,hiddenLayerTupple=(75,25,12,5))

def PCA_Analysis(data):
    import matplotlib.pyplot as plt
    from sklearn.decomposition import PCA
    import plotly.express as px
    from plotly.offline import plot
```

```
features=list(data.columns)
data_array=np.asarray(data)

print("Visualising 2D- PCA components")
pca = PCA(n_components=2) # estimate only 2 PCs
X_new = pca.fit_transform(data_array) # project the original data into the PCA space



fig, axes = plt.subplots(1,2)
axes[0].scatter(data_array[:,0], data_array[:,1], c=y)
axes[0].set_xlabel('x1')
axes[0].set_ylabel('x2')
axes[0].set_title('Before PCA')
axes[1].scatter(X_new[:,0], X_new[:,1], c=y)
axes[1].set_xlabel('PC1')
axes[1].set_ylabel('PC2')
axes[1].set_title('After PCA')
plt.show()

print("Visualising 5  PCA components")

pca = PCA(n_components=5)
components = pca.fit_transform(data[features])
total_var = pca.explained_variance_ratio_.sum() * 100
labels = {str(i): f"PC {i+1}" for i in range(5)}

fig=px.scatter_matrix(
   components,
   labels=labels,
```

```
        dimensions=range(5),
        title=f'Total Explained Variance: {total_var:.2f}%',
    )
    fig.update_traces(diagonal_visible=False)
    fig.show()
    plot(fig)
PCA_Analysis(X)
```