**ASSIGNMENT–4**

**]**

|  |  |
| --- | --- |
| **AssignmentDate** | 27October2022 |
| **Student Name** | Pathuman Sethupathi R |
| **Team ID** | PNT2022TMID15092 |
| **MaximumMarks** | 2Marks |

ProblemStatement:CustomerSegmentationAnalysis

**ProblemStatement:**

Chronic Kidney Disease prediction is one of the most importants inhealthcare analytics. The most interesting and challenging tasks in day-to-daylife is prediction in the medical field. 10% of the world is affected by chronickidneydisease(CHD),andmillionsdieeachyearbecausetheydonothave

accesstoaffordabletreatment.ChronicKidneyDiscanbecured,iftreatedintheearly stages. The main aim of this project is to predict whether the patients havechronic kidney disease or not, in a more accurate and faster way based oncertaindiagnosticmeasurementslikeBloodPressure(Bp),Albumin(Al).

**Clusteringthedataand performingclassification**

**Algorithms**

# Downloadthedataset:Dataset

importpandasaspdimportnumpyasnp

importmatplotlib.pyplotaspltimportseabornassns

fromgoogle.colabimportdrive

drive.mount('/content/drive')

# Loadthedatasetintothetool.

data.head()

CustomerIDGenderAgeAnnualIncome(k$)SpendingScore(1-100)01Male19153912Male21158123Female2016634Female231677

45Female311740data.info()

<class'pandas.core.frame.DataFrame'>

RangeIndex:200entries,0to199Datacolumns(total5columns):

#ColumnNon-NullCountDtype

* 1. CustomerID200non-nullint64
  2. Gender200non-nullobject
  3. Age200non-nullint64
  4. AnnualIncome(k$)200non-nullint64
  5. SpendingScore(1-100)200non-nullint64dtypes:int64(4),object(1)

memoryusage:7.9+KBdata.describe()

CustomerIDAgeAnnualIncome(k$)SpendingScore(1-100)count200.000000200.000000200.000000200.000000mean100.50000038.850000

60.56000050.200000std57.87918513.96900726.26472125.823522min

1.00000018.00000015.0000001.00000025%50.75000028.75000041.500000

34.75000050%100.50000036.00000061.50000050.00000075%150.250000

49.00000078.00000073.000000max200.00000070.000000137.000000

99.000000

# PerformBelowVisualizations.

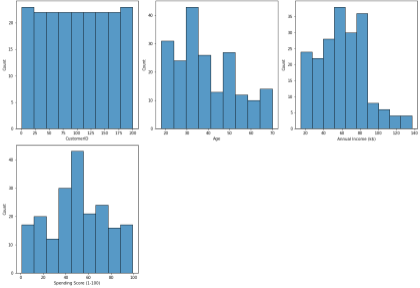
**univariate analysis***#univariateanalysis*cols=3

rows=3

num\_cols=data.select\_dtypes(exclude='object').columnsfig=plt.figure(figsize=(cols\*5,rows\*5))

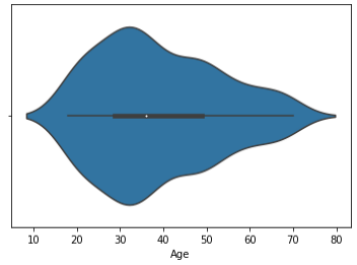
**for** i, col **in** enumerate(num\_cols):ax=fig.add\_subplot(rows,cols,i+1)

sns.histplot(x=data[col],ax=ax)

fig.tight\_layout()plt.show()

sns.violinplot(x=data["Age"])

<matplotlib.axes.\_subplots.AxesSubplotat0x7fa3726b9490>

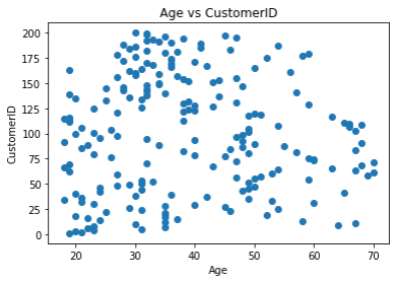


# Bivariateanalysis

importmatplotlib.pyplotasplt

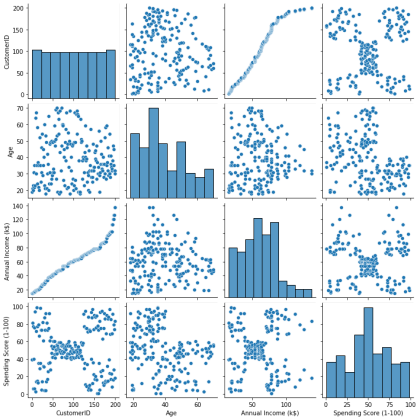
*#create scatterplot of hours vs. score*plt.scatter(data.Age,data.CustomerID)plt.title('Age vs CustomerID')plt.xlabel('Age')plt.ylabel('CustomerID')

Text(0,0.5,'CustomerID')



# Multivariateanalysis

sns.pairplot(data);



# Performdescriptivestatisticsonthedataset.

data.mean()

CustomerID100.50

Age38.85

Annual Income (k$) 60.56SpendingScore(1-100)50.20dtype:float64data.median()

CustomerID100.5

Age36.0

Annual Income (k$) 61.5SpendingScore(1-100)50.0dtype:float64

norm\_data=pd.DataFrame(np.random.normal(size=100000))

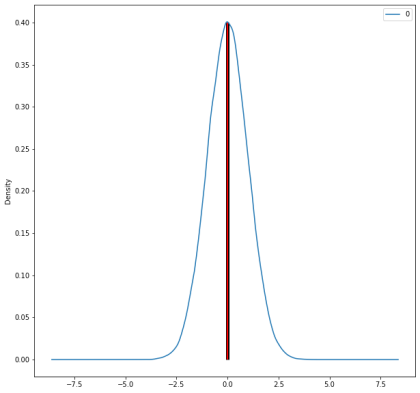
norm\_data.plot(kind="density",

figsize=(10,10));

plt.vlines(norm\_data.mean(),*#Plotblacklineatmean*

ymin=0,ymax=0.4,linewidth=5.0);

plt.vlines(norm\_data.median(),*#Plotredlineatmedian*

ymin=0,ymax=0.4,linewidth=2.0,color="red");

# CheckforMissingvaluesanddealwiththem.Identifyingthe missing value

df=pd.DataFrame(data)df.isnull()

CustomerID Gender Age Annual Income (k$) Spending Score (1-100) 0FalseFalseFalseFalseFalse1FalseFalseFalseFalseFalse2FalseFalseFalseFalseFalse3FalseFalseFalseFalseFalse4FalseFalseFalseFalseFalse.................195FalseFalseFalseFalseFalse

196FalseFalseFalseFalseFalse197FalseFalseFalseFalseFalse

198FalseFalseFalseFalseFalse199FalseFalseFalseFalseFalse[200rowsx5columns]

# Fillingthemissingvaluewithpreviousvalue

df.fillna(method='pad')

CustomerIDGenderAgeAnnualIncome(k$)SpendingScore(1-100)01

|  |  |  |
| --- | --- | --- |
| Male19153912Male21158123Female2016634Female | 2316774 | |
| 5Female311740.................195196Female35 | 12079 | 196 |
| 197Female4512628197198Male3212674198199Male32  200Male3013783 | 13718 | 199 |
| [200rowsx5columns] |  |  |
| **Fillingnullvaluesinmissingvalues** |  |  |
| data[0:] |  |  |
| CustomerIDGenderAgeAnnualIncome(k$)SpendingScore | (1-100) | 01 |
| Male19153912Male21158123Female2016634Female | 2316774 | |
| 5Female311740.................195196Female35 | 12079196 | |
| 197Female4512628197198Male3212674198199Male32  200Male3013783 | 13718199 | |

[200rowsx5columns]

# Findtheoutliersandreplace themoutliers

**Identifying the outliers**

print(df['AnnualIncome(k$)'].skew())df['AnnualIncome(k$)'].describe()

0.3218425498619055

count200.000000

mean60.560000

std26.264721

min15.000000

|  |  |
| --- | --- |
| 25% | 41.500000 |
| 50% | 61.500000 |
| 75% | 78.000000 |
| max | 137.000000 |

Name:AnnualIncome(k$),dtype:float64

# Replacingtheoutliers

print(df['AnnualIncome(k$)'].quantile(0.50))print(df['AnnualIncome(k$)'].quantile(0.95))

df['AnnualIncome(k$)']=np.where(df['AnnualIncome(k$)']>325,140,df['AnnualIncome(k$)'])

df.describe()

61.5

103.0

CustomerIDAgeAnnualIncome(k$)SpendingScore(1-100)count200.000000200.000000200.000000200.000000mean100.50000038.850000

60.56000050.200000std57.87918513.96900726.26472125.823522min

1.00000018.00000015.0000001.00000025%50.75000028.75000041.500000

34.75000050%100.50000036.00000061.50000050.00000075%150.250000

49.00000078.00000073.000000max200.00000070.000000137.000000

99.000000

# CheckforCategorical columnsandperform encoding.

**Performencoding**

fromsklearn.composeimportmake\_column\_selectorasselectorcategorical\_columns\_selector =selector(dtype\_include=object)

categorical\_columns=categorical\_columns\_selector(data)

categorical\_columns['Gender']

data\_categorical=data[categorical\_columns]

data\_categorical.head()

Gender

1. Male
2. Male
3. Female
4. Female
5. Female

fromsklearnimportpreprocessing

*#label\_encoderobjectknowshowtounderstandwordlabels.*

label\_encoder=preprocessing.LabelEncoder()

*#Encodelabelsincolumn'species'.*

df['Gender']= label\_encoder.fit\_transform(df['Gender'])df['Gender'].unique()

array([1,0])

*# import packages*import numpy as npimportpandasaspd

fromsklearn.model\_selectionimporttrain\_test\_split

*#importingdata*

print(df.shape)

*#headofthedata*

print('Headofthedataframe:')print(df.head())

print(df.columns)

X= df['Annual Income (k$)']y=df['SpendingScore(1-100)']

*#usingthetraintestsplitfunction*

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,random\_state=104,test\_size=0.25,shuffle=True)

*#printingouttrainandtestsets*

print('X\_train : ')print(X\_train.head())print(X\_train.shape)

print('')

print('X\_test : ')print(X\_test.head())print(X\_test.shape)print('')

print('y\_train : ')print(y\_train.head())print(y\_train.shape)

print('')

print('y\_test : ')print(y\_test.head())print(y\_test.shape)

(200,5)

Headofthedataframe:

CustomerIDGenderAgeAnnualIncome(k$)SpendingScore(1-100)01119153912121158123020166340231677450311740

Index(['CustomerID','Gender','Age','AnnualIncome(k$)','SpendingScore(1-100)'],

dtype='object')X\_train:

7350

3030

2325

15578

15778

Name:AnnualIncome(k$),dtype:int64(150,)

X\_test:

10462

12871

4940

3433

6448

Name:AnnualIncome(k$),dtype:int64(50,)

y\_train:

7356

304

2373

15589

15778

Name:SpendingScore(1-100),dtype:int64(150,)

y\_test :10456

12811

4942

3414

6451

Name:SpendingScore(1-100),dtype:int64(50,)

# Scaling the dataScaling

df\_scaled=df.copy()

col\_names=['AnnualIncome(k$)','SpendingScore(1-100)']features=df\_scaled[col\_names]

fromsklearn.preprocessingimportMinMaxScalerscaler=MinMaxScaler()

df\_scaled[col\_names]=scaler.fit\_transform(features.values)fromsklearn.preprocessingimportMinMaxScaler

scaler=MinMaxScaler(feature\_range=(5,10))

df\_scaled[col\_names]=scaler.fit\_transform(features.values)df\_scaled

CustomerIDGenderAgeAnnualIncome(k$)SpendingScore(1-100)011195.0000006.938776121215.0000009.081633230205.040984

5.255102340235.0409848.877551450315.0819676.989796.....

|  |  |  |  |
| --- | --- | --- | --- |
| ......... | ...1951960359.3032798.979592196 | 197045 | 9.549180 |
| 6.377551197 | 1981329.5491808.724490198199132 | 10.000000 | 5.867347 |
| 199200130 | 10.0000009.183673 |  |  |

[200rowsx5columns]

# Performanyoftheclusteringalgorithms

*# k-means clustering*fromnumpyimportuniquefromnumpyimportwhere

fromsklearn.datasetsimportmake\_classificationfromsklearn.clusterimportKMeans

frommatplotlibimportpyplot

# Addtheclusterdatawiththeprimarydataset

*#definedataset*

X,\_=make\_classification(n\_samples=1000,n\_features=2,n\_informative=2,n\_redundant=0,n\_clusters\_per\_class=1,random\_state=4)

# Splitthedataintodependentandindependentvariables.

*#definethemodel*

model=KMeans(n\_clusters=2)

*#fitthemodel*

model.fit(X)

*#assignaclustertoeachexample*

yhat=model.predict(X)

*#retrieveuniqueclusters*

clusters=unique(yhat)

*#createscatterplotforsamplesfromeachcluster*

**for**cluster**in**clusters:

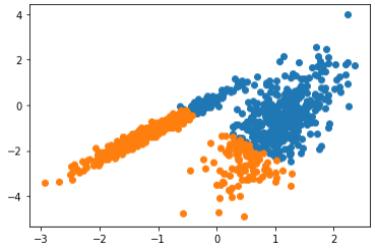
*#getrowindexesforsampleswiththiscluster*

row\_ix=where(yhat==cluster)

*#createscatterofthesesamples*

pyplot.scatter(X[row\_ix,0],X[row\_ix,1])

*#showtheplot*

pyplot.show()

# Splitthedataintotrainingandtesting

*#testingandtraining*

X=df.iloc[:,:-1]

y=df.iloc[:,-1]

*#splitthedataset*

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.05,random\_state=0)

print(X\_train,X\_test,y\_train,y\_test)

CustomerIDGenderAgeAnnualIncome(k$)616211946

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 125 | 126 | 0 | 31 | 70 |
| 180 | 181 | 0 | 37 | 97 |
| 154 | 155 | 0 | 47 | 78 |

808115754

..............

|  |  |
| --- | --- |
| 67680 | 6848 |
| 192193 | 133113 |
| 117118 | 04965 |
| 47480 | 2740 |

17217313687

[190rowsx4columns]CustomerIDGenderAgeAnnualIncome(k$)1819

15223

|  |  |  |  |
| --- | --- | --- | --- |
| 170171 | | 14087 | |
| 107108 | | 15463 | |
| 98991 | | 4861 | |
| 177 | 178 | 127 | 88 |
| 182 | 183 | 146 | 98 |
| 56 | 02217 | |  |
| 146 | 147148 | | 77 |
| 121305820 | | | |
| 152 | 153 | 04478615512577 | |
| 180 | 32 |  | |
| 154 | 16 |  | |

8051

..

|  |  |  |
| --- | --- | --- |
| 6748 |  | |
| 1928 |
| 11759 |
| 4747 |
| 17210 |
| Name:SpendingScore(1-100), | Length:190,dtype:int6418 | 29170 |
| 13 |  | |
| 10746 |
| 9842 |
| 17769 |
| 18215 |
| 576 |
| 14636 |
| 1215 |
| 15220  **13. Build the Model** |

Name:SpendingScore(1-100),dtype:int64

fromsklearn.linear\_modelimportLogisticRegressionlogreg= LogisticRegression()logreg.fit(X\_train,y\_train)y\_pred=logreg.predict(X\_test)

print(X\_test)*#testdataset*

print(y\_pred)*#predictedvalues*

CustomerIDGenderAgeAnnualIncome(k$)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 18191 | | 5223 | | | |
| 170171 | | 14087 | | | |
| 107108 | | 15463 | | | |
| 98991 | | 4861 | | | |
| 177 | 178 | 127 | 88 | | |
| 182 | 183 | 146 | 98 | | |
| 56 | 02217 | |  | | |
| 146 | 147148 | | 77 | | |
| 12130 | | 5820 | |  |  |
| 152153 | | 04478[4148558675 | | 8154 | 5] |

# TraintheModel

X\_train

CustomerIDGenderAgeAnnualIncome(k$)6162119461251260317018018103797154

|  |  |
| --- | --- |
| 15504778808115754.............. | 67 |
| 680684819219313311311711804965 | 47 |
| 480274017217313687 |  |
| [190rowsx4columns] |  |
| X\_test |  |
| CustomerIDGenderAgeAnnualIncome(k$) | 18 |
| 1915223170171140871071081546398 | 99 |
| 148611771781278818218314698560 | 22 |
| 1714614714877121305820152153044  y\_train | 78 |
| 6155 |  |
| 12577 |  |

18032

15416

8051

..

|  |  |  |
| --- | --- | --- |
| 6748 |  | |
| 1928 |
| 11759 |
| 4747 |
| 17210 |
| Name:Spending | Score(1-100),Length:190,dtype: | int64 |
| y\_test |  |  |
| 1829 |  |  |
| 17013 |  |  |
| 10746 |  |  |
| 9842 |  |  |
| 17769 |  |  |
| 18215 |  |  |
| 576 |  |  |
| 14636 |  |  |
| 1215 |  |  |
| 15220  Name:Spending | Score(1-100),dtype:int64 |  |

*#Selectalgorithm*

fromsklearn.treeimportDecisionTreeClassifierfrom sklearn.metrics import accuracy\_scoremodel=DecisionTreeClassifier()

*#Fitmodeltothedata*

model.fit(X\_train,y\_train)

# TesttheModel

*# Check model performance on training data*predictions = model.predict(X\_train)print(accuracy\_score(y\_train,predictions))

1.0

1. **Measure the performance using EvaluationMetrics.***#Evaluatethemodelonthetestdata*

predictions=model.predict(X\_test)predictions

array([14,95,56,49,75,28,77,16,14,28])

print(accuracy\_score(y\_test,predictions))0.0df=X\_test.copy()

df['Actual']=y\_test

df['Prediction']=predictionsdf

CustomerIDGenderAgeAnnualIncome(k$)ActualPrediction18

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 19 | 152232914170171140871395107 | 10815463465698991 | | |
| 48 | 614249177178127886975182183 | 146981528560 | 22 | 17 |
| 76 | 77146147148773616121305820 | 1514152153044 | 78 | 20 |
| 28 |  |  |  |  |

fromsklearn.metricsimportconfusion\_matrixfromsklearn.metricsimportaccuracy\_score

fromsklearn.metricsimportclassification\_reportfromsklearn.metricsimportroc\_auc\_score

from sklearn.metrics import log\_lossX\_actual=[0,1,0,1,0,0,1,0,0,0]

Y\_predic=[1,0,1,1,1,0,1,1,0,0]

results=confusion\_matrix(X\_actual,Y\_predic)print('ConfusionMatrix:')

print(results)

print('AccuracyScoreis',accuracy\_score(X\_actual,Y\_predic))print('ClassificationReport:')

print (classification\_report(X\_actual, Y\_predic))print('AUC-ROC:',roc\_auc\_score(X\_actual, Y\_predic))print('LOGLOSSValueis',log\_loss(X\_actual,Y\_predic))

ConfusionMatrix:

[[34]

[12]]

Accuracy Score is 0.5ClassificationReport:

precisionrecallf1-scoresupport

00.750.430.557

10.330.670.443

accuracy0.5010

macroavg0.540.550.4910

weightedavg0.620.500.5210