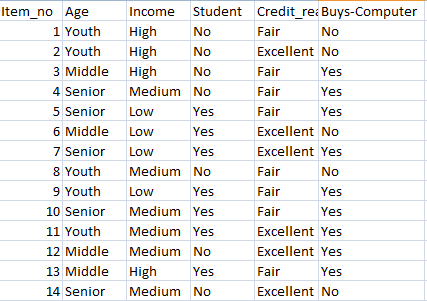
**Appendix-A**

**Sample Practical List**

1. **Write a python code to implement decision tree for below given dataset. Identify the root node and all subpart or children of node and draw the tree**.

**Dataset**

****

**Code**

import pandas as pd

from sklearn import tree

from sklearn.tree import DecisionTreeClassifier

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

import matplotlib.image as pltimg

df=pd.read\_csv("prac1.csv")

print("Data before mapping")

age = {'Youth': 0, 'Middle': 1, 'Senior': 2}

income = {'High': 3, 'Medium': 2, 'Low': 1}

student = {'Yes': 1, 'No': 0}

credit\_rating={'Fair':1,'Excellent':2}

buys\_computer={'Yes': 1, 'No': 0}

df['Credit Rating'] = df['Credit Rating'].map(credit\_rating)

df['Buys Computer'] = df['Buys Computer'].map(buys\_computer)

df['Age'] = df['Age'].map(age)

df['Income'] = df['Income'].map(income)

df['Student'] = df['Student'].map(student)

features = ['Age', 'Income', 'Student', 'Credit Rating','Buys Computer']

x=df[features]

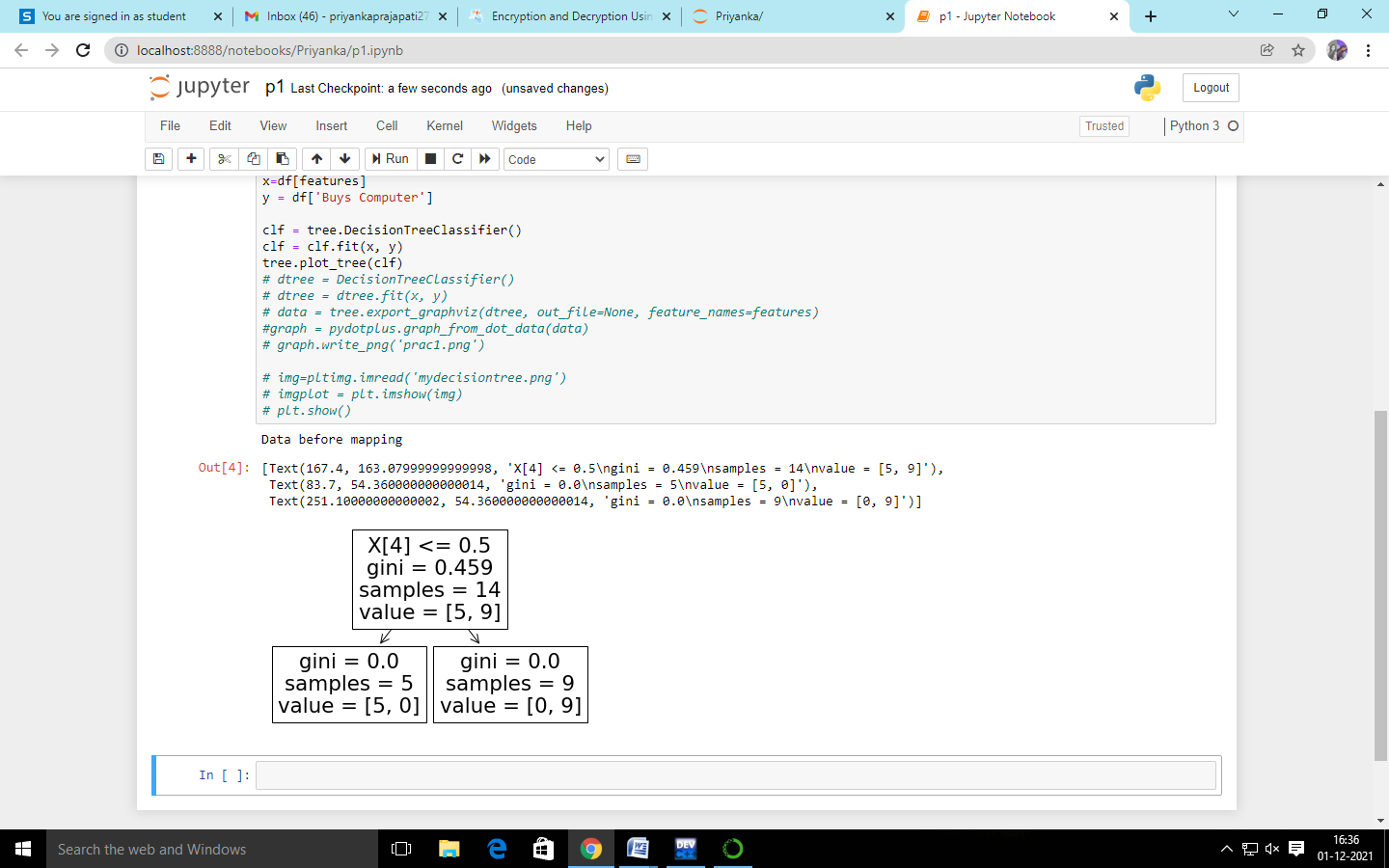
y = df['Buys Computer']

clf = tree.DecisionTreeClassifier()

clf = clf.fit(x, y)

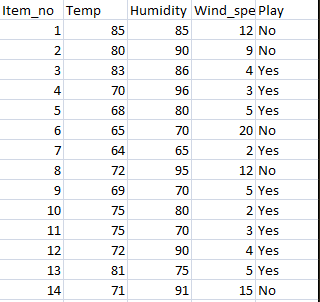
tree.plot\_tree(clf)

**Output:-**



**2.Write a python code to implement K-nearest neighbourhood program for the given dataset.**

**Data set**



**Code**

import pandas as pd

import numpy as np

import math

import csv

from csv import writer

from csv import reader

data = pd.read\_csv("knn\_dataset.csv")

#print(data)

k = 3

distance = []

minimum\_distance = []

x1 = 91

y1 =15

for x2 in data.Humidity:

global x

x = pow((x1-x2),2)

for y2 in data.Wind\_speed:

global y

y = pow((y1-y2),2)

xy = x + y

d = np.sqrt(xy)

distance.append(d)

print(distance)

print("\n")

print("\n")

data.insert(5,column = "Distance",value = distance)

print(data)

min\_dist = sorted(distance,key = float)

minimum\_distance.append(min\_dist[1])

print("\n")

print("k minimum distance ",minimum\_distance)

r = len(distance)

print("\n")

#print(data.Distance[0])

#print(minimum\_distance[0])

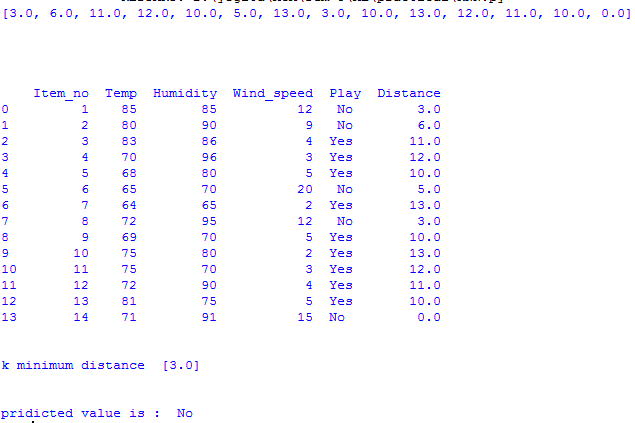
for i in range(r):

if data.Distance[i] == minimum\_distance[0]:

print("pridicted value is : ",data.Play[i])

break

**Output:-**



**3.Write a python code to implement Apriori algorithm, apply join and prune method and find frequent item set**

**Dataset**

|  |  |
| --- | --- |
| TID | Item |
| 100 | 1,3,4 |
| 200 | 2,3,5 |
| 300 | 1,2,3,5 |
| 400 | 2,5 |

**Code**

import pandas as pd

df = pd.read\_csv ('apriori\_algo.csv')

item\_set=df['Item']

len=df['Item']

frequecy=[];

min\_support=50;

confidance=70;

temp=0;

ans={'1':0,'2':0,'3':0,'4':0,'5':0};

support = 0

final\_list=[];

for value in df['Item']:

array\_value=value.split (",")

for item\_value in array\_value:

final\_list.append(item\_value)

for i in ans:

temp=0

temp=final\_list.count(i)

ans[str(i)]=ans[str(i)]+temp

df.insert(2,column="frequncy",value = ans)

for j in ans:

support = ans[j]/4 \* 100

#print(support)

print(j,ans[j],support)

**output:-**

1 2 [50.0]

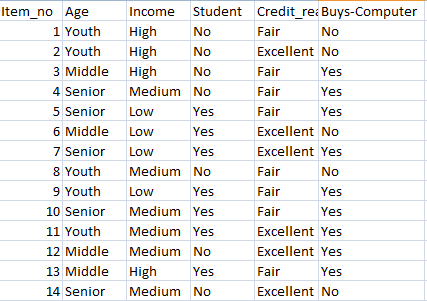
2 3 [50.0, 75.0]

3 3 [50.0, 75.0, 75.0]

4 1 [50.0, 75.0, 75.0, 25.0]

5 3 [50.0, 75.0, 75.0, 25.0, 75.0]

**4. Write a python code to apply Naive Bayesian and Logistic Regression algorithm to classify that whether a person can buy computer or not based on given test data:**

****

**Code**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

#import dataset

dataset = pd.read\_csv('student\_details.csv')

#separate two column

X = dataset.iloc[:, 1:5].values

Y = dataset.iloc[:, 5].values

#make data frame

X = pd.DataFrame(X)

y = pd.DataFrame(Y)

#import lable encoder library

from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()

X.values[:, 0] = labelencoder.fit\_transform(X.values[:, 0])

X.values[:, 1] = labelencoder.fit\_transform(X.values[:, 1])

X.values[:, 2] = labelencoder.fit\_transform(X.values[:, 2])

X.values[:, 3] = labelencoder.fit\_transform(X.values[:, 3])

y.values[:, 0] = labelencoder.fit\_transform(y.values[:, 0])

#import model split libaray

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 125)

X\_train=X\_train.astype(int)

y\_train=y\_train.astype(int)

X\_test=X\_test.astype(int)

y\_test=y\_test.astype(int)

#import decisionTree Alogorith Libaray

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier()

classifier = classifier.fit(X\_train,y\_train)

y\_pred = classifier.predict(X\_test)

#import library for sklern

from sklearn.metrics import confusion\_matrix

from sklearn import metrics

cm = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix : \n {}".format(cm))

#print accuracy

print("Accuracy Score: {} %".format(metrics.accuracy\_score(y\_test, y\_pred)\*100))

#import Graph Library

from sklearn.tree import export\_graphviz

from sklearn.externals.six import StringIO

from IPython.display import Image

import pydotplus

dot\_data = StringIO()

export\_graphviz(classifier, out\_file=dot\_data,

filled=True, rounded=True,

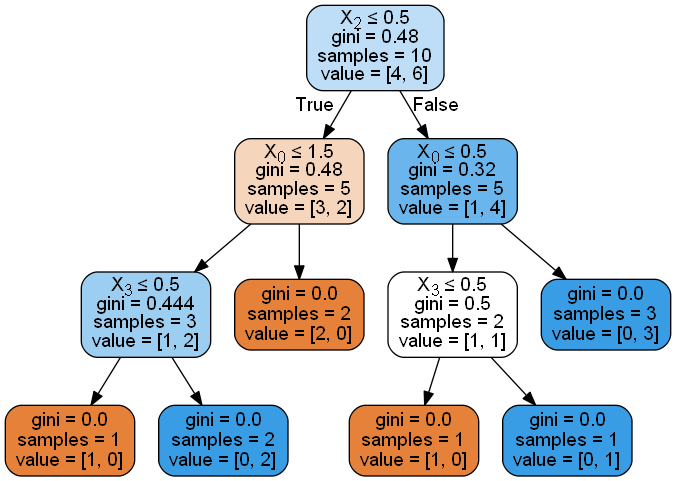
special\_characters=True,)

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

# graph.write\_png('computer.png')

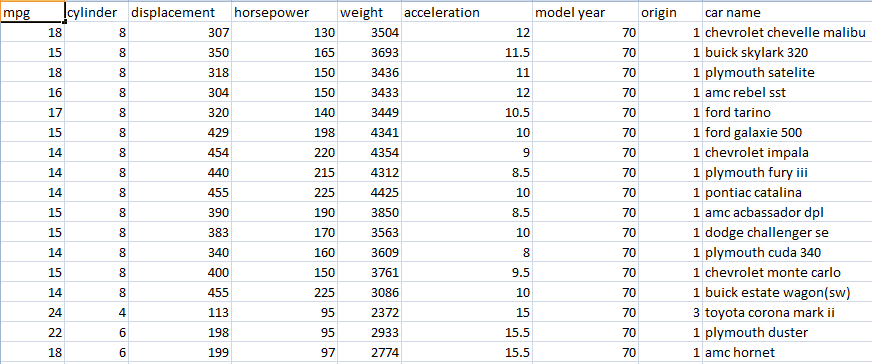
Image(graph.create\_png())

**Output:-**



**5 .Python code for Preparing to Model and Feature Engineering.**

**auto-mpg.csv**

****

**basic statestical function**

import pandas as pd

import numpy as np

data = pd.read\_csv("auto-mpg.csv")

data.info()

print("\n\n")

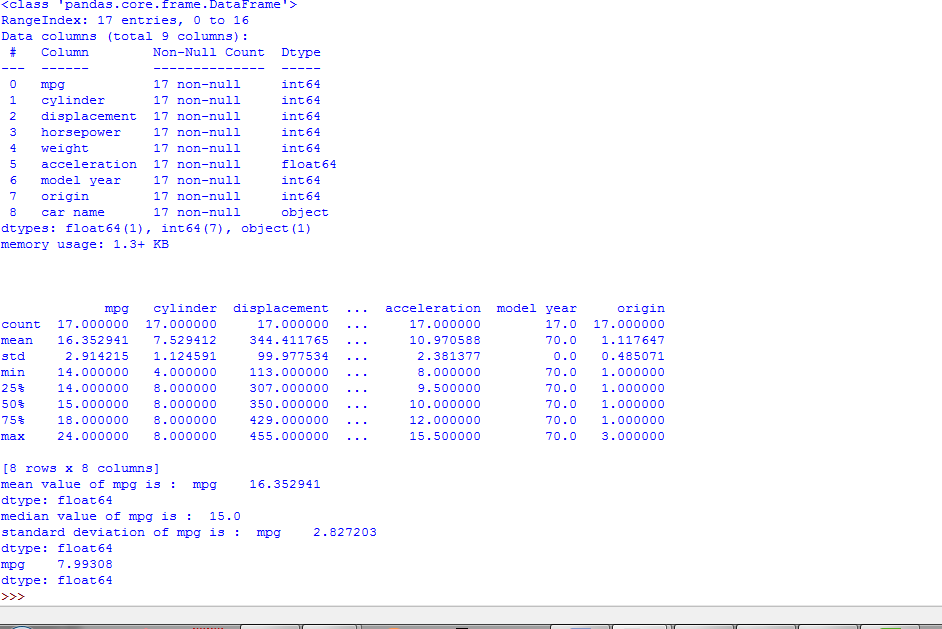
print(data.describe())#to get quick analysis of dataset

print("mean value of mpg is : ",np.mean(data[["mpg"]]))

print("median value of mpg is : ",np.median(data[["mpg"]]))

print("standard deviation of mpg is : ",np.std(data[["mpg"]]))

print(np.var(data[["mpg"]]))

**Output:-** 

**Box-plot**

from sklearn import datasets

iris = datasets.load\_iris()

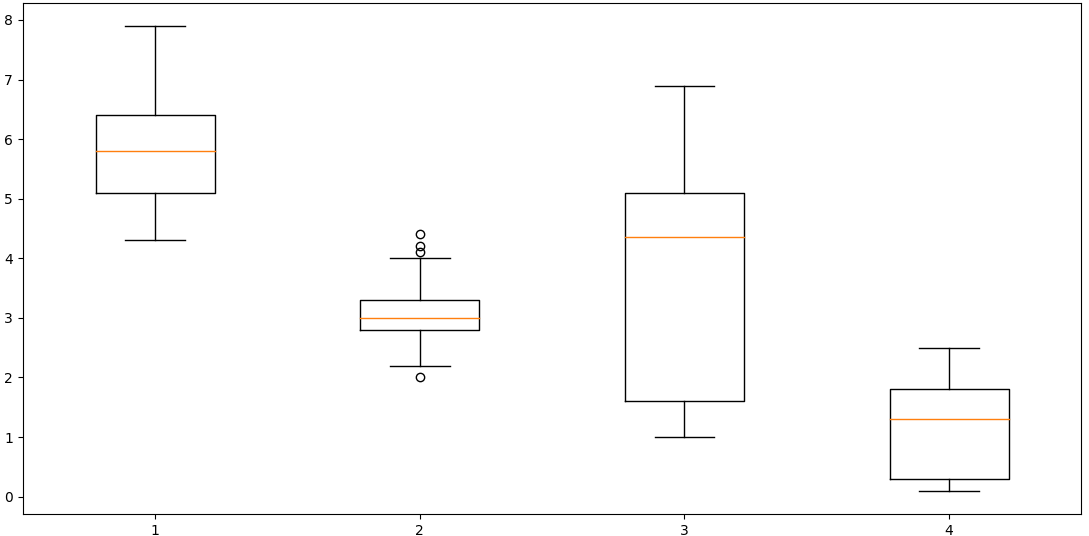
import matplotlib.pyplot as plt

x = iris.data[:,:4]

plt.boxplot(x)

plt.show()

**Output:-**



**Histogram**

import matplotlib.pyplot as plt

from sklearn import datasets

iris = datasets.load\_iris()

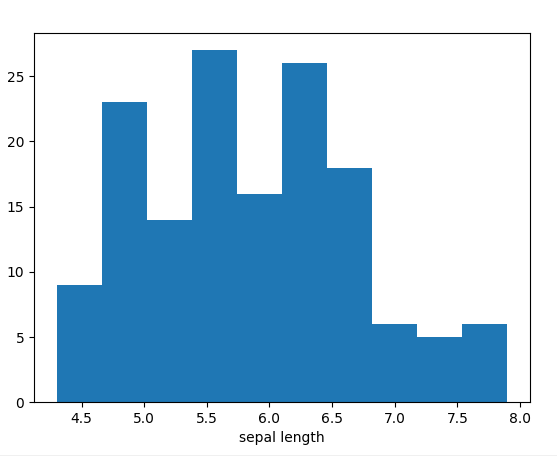
x = iris.data[:,:1]

plt.hist(x)

plt.xlabel('sepal length')

plt.show()

OutPut:-



**Scatterplot**

import matplotlib.pyplot as plt

from sklearn import datasets

iris = datasets.load\_iris()

x = iris.data[:,:4]

y = iris.target

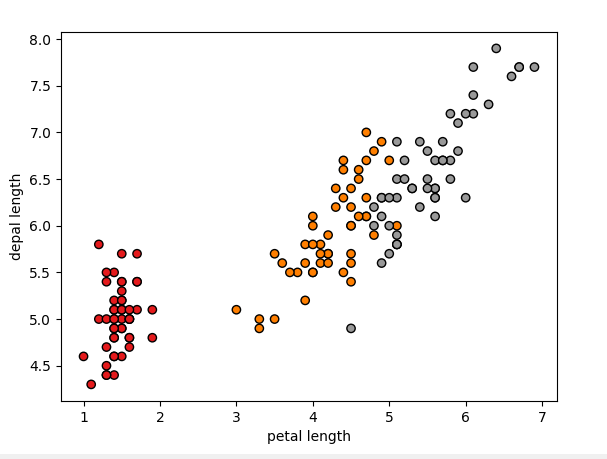
plt.scatter(x[:,2],x[:,0],c = y,cmap=plt.cm.Set1,edgecolor='k')

plt.xlabel('petal length')

plt.ylabel('depal length')

plt.show()

Output:-



**Removing outliers/missing values**

import matplotlib.pyplot as plt

import pandas as pd

data = pd.read\_csv("auto-mpg.csv")

def find\_outlier(ds,col):

quart1 = ds[col].quantile(0.25)

quart3 = ds[col].quantile(0.75)

IQR = quart1-quart3 #inter quartile rang

low\_val = quart1-1.5\*IQR

high\_val = quart3-1.5\*IQR

ds = ds.loc[(ds[col]<low\_val) | (ds[col] > high\_val)]

return ds

outliers = find\_outlier(data,"mpg")

def remove\_outlier(ds,col):

quart1 = ds[col].quantile(0.25)

quart3 = ds[col].quantile(0.75)

IQR = quart1-quart3 #inter quartile rang

low\_val = quart1-1.5\*IQR

high\_val = quart3-1.5\*IQR

ds\_out = ds.loc[(ds[col]>low\_val) & (ds[col] < high\_val)]

return ds\_out

outliers = remove\_outlier(data,"mpg")

print(outliers)

Output:-

mpg cylinder displacement ... model year origin car name

15 22 6 198 ... 70 1 plymouth duster

[1 rows x 9 columns]

**Feature construction**

import pandas as pd

room\_length = [18,20,10,12,18,11]

room\_breadth = [20,20,10,11,19,10]

room\_type = ['big','big','normal','normal','big','normal']

data = pd.DataFrame({'Length':room\_length,'Breath':room\_breadth,'Type':roo m\_type})

print(data)

data['Area'] = data['Length']\*data['Breath']

print(data)

**Output:-**

Length Breath Type

0 18 20 big

1 20 20 big

2 10 10 normal

3 12 11 normal

4 18 19 big

5 11 10 normal

Length Breath Type Area

0 18 20 big 360

1 20 20 big 400

2 10 10 normal 100

3 12 11 normal 132

4 18 19 big 342

5 11 10 normal 110

**Encoding categorical variables**

import pandas as pd

from sklearn import preprocessing

marks\_science = [78,56,87,91,45,62]

marks\_maths = [75,62,90,95,42,57]

grade = ['B','C','A','A','D','B']

data = pd.DataFrame({'science marks' : marks\_science,'maths marks':marks\_maths,'total grad':grade})

print(data)

#optin - 1

le = preprocessing.LabelEncoder()

le.fit(data['total grad'])

data['total grad'] = le.transform(data['total grad'])

print(data)

#option - 2

target = data['total grad'].replace(['A','B','C','D'],[0,1,2,3])

**Output:-**

science marks maths marks total grad

0 78 75 B

1 56 62 C

2 87 90 A

3 91 95 A

4 45 42 D

5 62 57 B

science marks maths marks total grad

0 78 75 1

1 56 62 2

2 87 90 0

3 91 95 0

4 45 42 3

5 62 57 1

**Transforming numeric features to categorical features**

import pandas as pd

import numpy as np

apartment\_area = [4720,2430,4368,3969,6142,7912]

apartment\_price = [2360000,1215000,2184000,1984500,3071000,3956000]

data = pd.DataFrame({'Area':apartment\_area,'price':apartment\_price})

print(data)

data['price'] = np.where(data['price'] > 3000000, 'High',np.where(data['price']<2000000, 'low','medium'))

print(data)

**Output:-**

Area price

0 4720 2360000

1 2430 1215000

2 4368 2184000

3 3969 1984500

4 6142 3071000

5 7912 3956000

Area price

0 4720 medium

1 2430 low

2 4368 medium

3 3969 low

4 6142 High

5 7912 High

**Principal component Analysis**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn import datasets

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

iris = datasets.load\_iris()

predictors = iris.data[:,0:4]

target = iris.target

predictors = StandardScaler().fit\_transform(predictors)

#STANDERDIZING THE FEATURES

pca = PCA(n\_components=2)

princomp = pca.fit\_transform(predictors)

princomp\_ds = pd.DataFrame(data = princomp,columns = ['PC 1','PC 2'])

print(princomp\_ds)

target\_ds = pd.DataFrame(data = target,columns = ['class'])

data\_mod = pd.concat([princomp\_ds.reset\_index(drop=True),target\_ds],axis = 1)

fig = plt.figure(figsize = (8,8))

pca\_plot = fig.add\_subplot(1,1,1)

pca\_plot.set\_xlabel('PC1')

pca\_plot.set\_ylabel('PC2')

pca\_plot.set\_title('PCA(2 component)',fontsize = 20)

classes = [0,1,2]

colors = ['y','r','b']

for target,color in zip(classes,colors):

indices = data\_mod['class'] == target

pca\_plot.scatter(data\_mod.loc[indices,'PC 1'],data\_mod.loc[indices,'PC 2'],c = color,s = 50)

legends = ['setosa','versicolor','virginica']

pca\_plot.legend(legends)

**Feature subset selection**

import pandas as pd

import numpy as np

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

data = pd.read\_csv('auto-mpg.csv')

predictor = data.iloc[:,0:7] #seggretating the predictor variable

target = data.iloc[:,7] #seggretsting the target/class variable

test = SelectKBest(chi2,k=2)

fit = test.fit(predictor,target)

features = fit.transform(predictor)

print(predictor.columns)

np.set\_printoptions(precision=3)

print(fit.scores\_)

**Output:-**

Index(['mpg', 'cylinder', 'displacement', 'horsepower', 'weight',

'acceleration', 'model year'],

dtype='object')

[ 3.799 1.758 165.204 30.306 434.319 1.572 0. ]