**CYCLE-4**

**1. Using the iris data set implement the KNN algorithm. Take different values for Test and training data set .Also use different values for k. Also find the accuracy level.**

**Program**

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

irist = sns.load\_dataset('iris')

dataset = pd.DataFrame(irist)

dataset.head()

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

y = dataset.iloc[:, 4].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors=5)

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

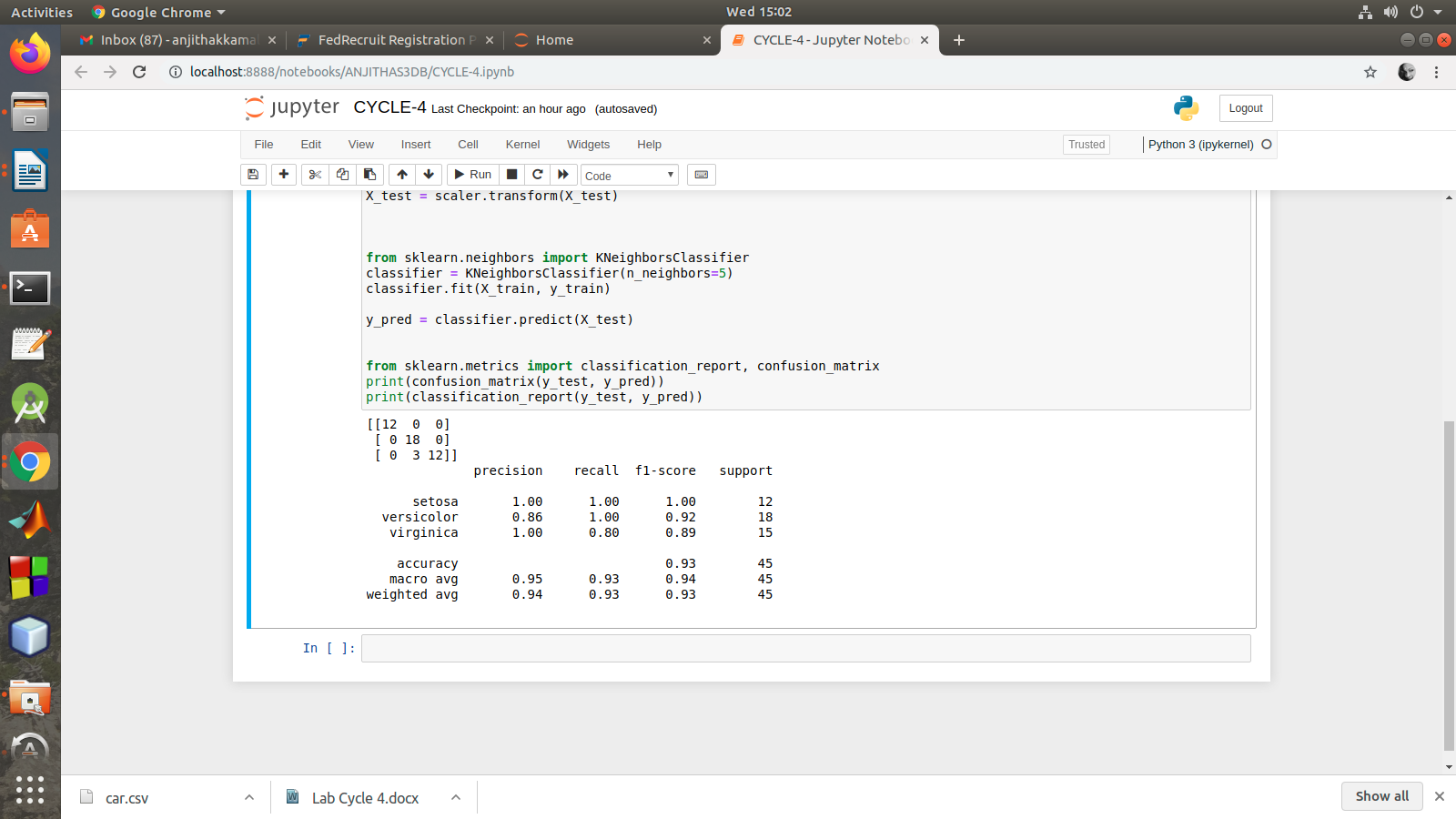
y\_pred = classifier.predict(X\_test)

from sklearn.metrics import classification\_report, confusion\_matrix

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

**output**

****

1. **Download another data set suitable for the KNN and implement the KNN algorithm. Take different values for Test and training data set. Also use different values for k.**

**CODE**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv("cancer.csv")

dataset.head()

dataset.info()

X = dataset.iloc[:, 2:35].values

print(X)

y = dataset.iloc[:, 1].values

print(y)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors=5)

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

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

from sklearn.metrics import classification\_report, confusion\_matrix

print(classification\_report(y\_test, y\_pred))

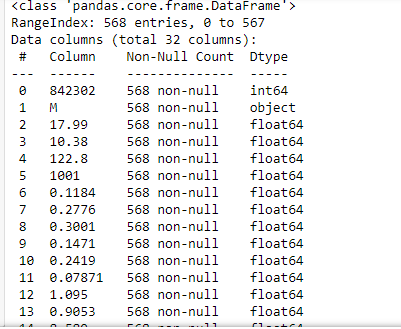
from sklearn.metrics import accuracy\_score

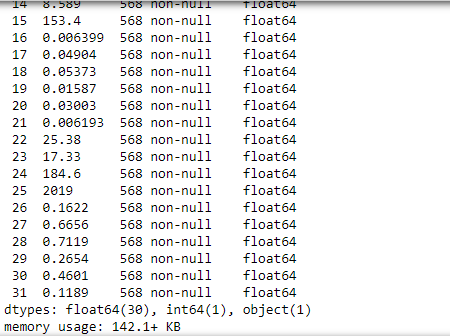
print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))

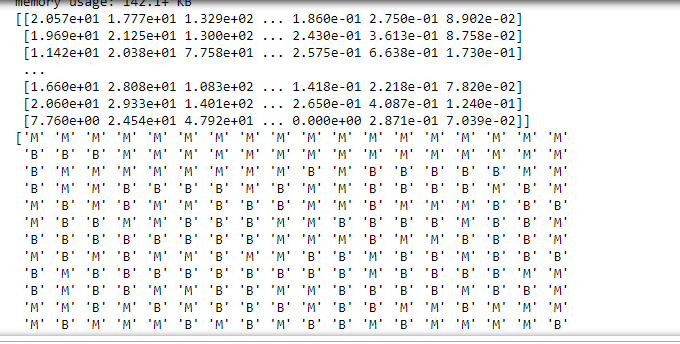
df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

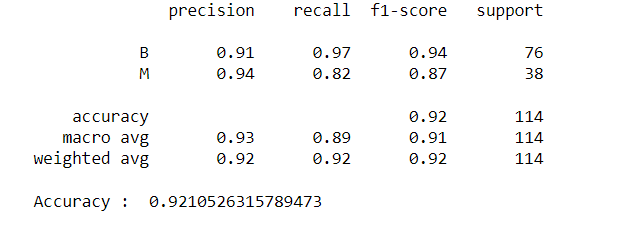
df

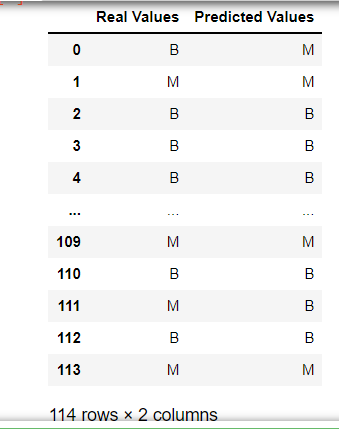
**OUTPUT**











1. **Using iris data set, implement naive bayes classification for different naive Bayes classification algorithms. ((i) gaussian (ii) bernoulli etc)**

* **Find out the accuracy level w.r.t to each algorithm**
* **Display the no:of mislabeled classification from test data set**
* **List out the class labels of the mismatching records**

**i)CODE**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('iris.csv')

X = dataset.iloc[:,:4].values

y = dataset['variety'].values

dataset.head(5)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2)

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

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

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

y\_pred

from sklearn.metrics import confusion\_matrix

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

from sklearn.metrics import accuracy\_score

print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))

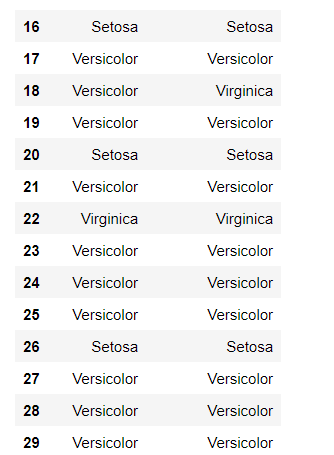
cm

df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

df

**OUTPUT**





**CODE**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('iris.csv')

X = dataset.iloc[:,:4].values

y = dataset['variety'].values

dataset.head(5)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2)

from sklearn.naive\_bayes import BernoulliNB

classifier = BernoulliNB()

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

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

y\_pred

from sklearn.metrics import confusion\_matrix

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

from sklearn.metrics import accuracy\_score

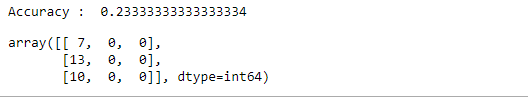
print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))

cm

# df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

# df

**OUTPUT**



**ii)CODE**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('iris.csv')

X = dataset.iloc[:,:4].values

y = dataset['variety'].values

dataset.head(5)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2)

from sklearn.naive\_bayes import BernoulliNB

classifier = BernoulliNB()

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

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

y\_pred

#from sklearn.metrics import confusion\_matrix

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

from sklearn.metrics import accuracy\_score

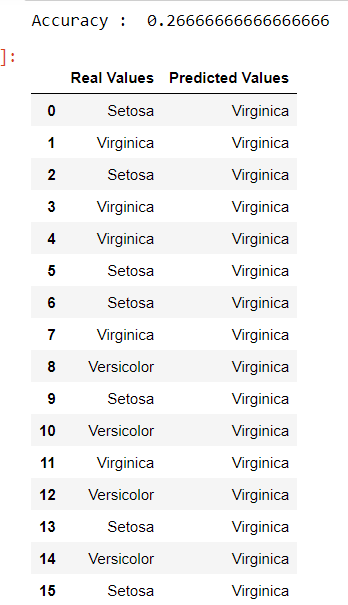
print ("Accuracy : ", accuracy\_score(y\_test, y\_pred))

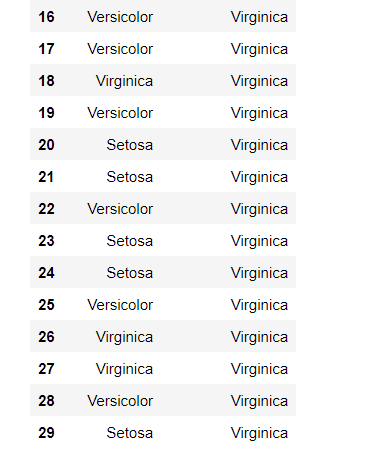
#cm

df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

df

**OUTPUT**





References:

<https://towardsdatascience.com/machine-learning-basics-naive-bayes-classification-964af6f2a965>

<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive_bayes>

1. **Use car details CSV file and implement decision tree algorithm**

* **Find out the accuracy level.**
* **Display the no: of mislabelled classification from test data set**
* **List out the class labels of the mismatching records**

References:

<https://www.24tutorials.com/machine-learning/case-study-decision-tree-model-for-car-quality/>

<https://notebook.community/bMzi/ML_in_Finance/0210_DecisionTrees>

<https://stackabuse.com/decision-trees-in-python-with-scikit-learn/>

For Data Sets Refer:

<https://www.kaggle.com> ( for data set)

<http://archive.ics.uci.edu/ml/datasets.php>

**REGRESSION**

Implement Simple and multiple linear regression for the data sets ‘student\_score.csv’ and ‘company\_data .csv’ respectively

Ref:  <https://stackabuse.com/linear-regression-in-python-with-scikit-learn/>

**Simple Linear Regression**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

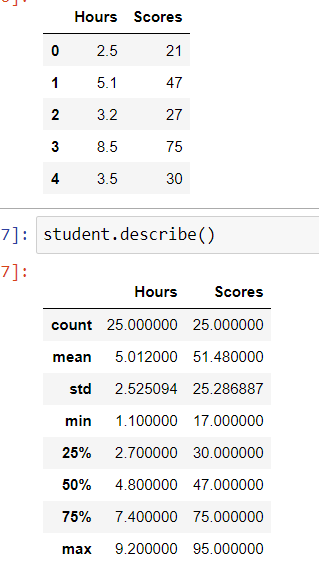
#data set contains details of no.of hours spend by students for studt and their marks

student = pd.read\_csv('student\_scores.csv')

student.head()

student.describe()

**OUTPUT:**



student.info()

**OUTPUT:**

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

RangeIndex: 25 entries, 0 to 24

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Hours 25 non-null float64

1 Scores 25 non-null int64

dtypes: float64(1), int64(1)

memory usage: 528.0 bytes

import matplotlib.pyplot as plt

Xax=student.iloc[:,0]

Yax=student.iloc[:,1]

plt.scatter(Xax,Yax)

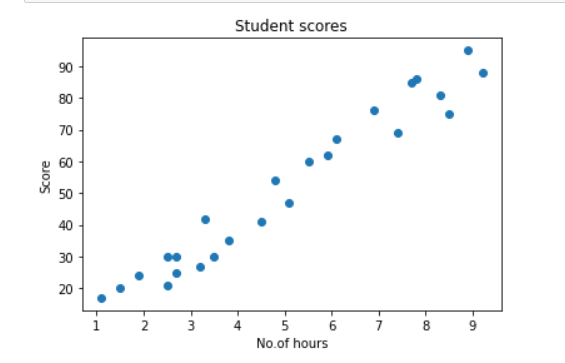
plt.xlabel("No.of hours")

plt.ylabel("Score")

plt.title("Student scores")

plt.show()

**OUTPUT:**



#Perform the simple linear regression model

#Equation: Y=w0+w1.x

#Here Y(marks)=w0+w1.x

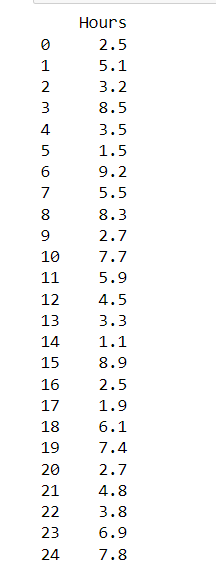
#Create x as hours and Y as marks

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

y = student.iloc[:, 1]

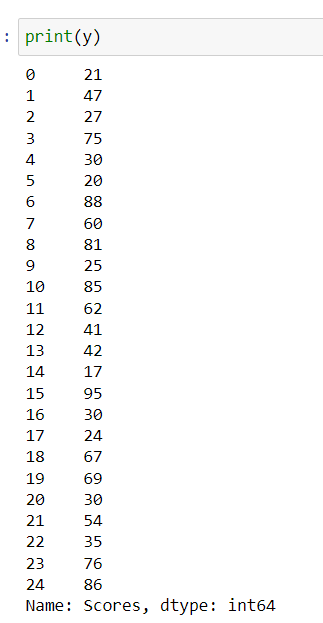
print(X)

**OUTPUT:**



print(y)

**OUTPUT:**

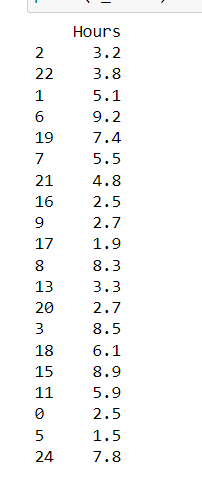


from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

print(X\_train)

**OUTPUT:**



from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

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

**OUTPUT:**

LinearRegression()

print(regressor.intercept\_)

**OUTPUT:**

3.4157789720979608

print(regressor.coef\_)

**OUTPUT:**

[9.544138]

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

for(i,j) in zip(y\_test,y\_pred):

if i!=j:

print("Actual value :",i,"Predicted value :",j)

print("Number of mislabeled points from test data set :", (y\_test != y\_pred).sum())

**OUTPUT:**

Actual value : 76 Predicted value : 69.2703311556657

Actual value : 41 Predicted value : 46.36439996138126

Actual value : 85 Predicted value : 76.90564155376049

Actual value : 17 Predicted value : 13.914330769478324

Actual value : 30 Predicted value : 36.82026196376275

Number of mislabeled points from test data set : 5

from sklearn import metrics

print("Mean Absolute error :", metrics.mean\_absolute\_error(y\_test,y\_pred))

print("Mean Squared error :", metrics.mean\_squared\_error(y\_test,y\_pred))

print("Root Mean Squared error :", np.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))

**OUTPUT:**

Mean Absolute error : 6.0188716892478995

Mean Squared error : 39.124239242163426

Root Mean Squared error : 6.254937189306014

import matplotlib.pyplot as plt

c=X\_test['Hours'].count()

xax=np.arange(c)

print(xax)

X\_axis = np.arange(len(xax))

plt.bar(X\_axis-0.2, y\_test, 0.6, label='Actual')

plt.bar(X\_axis+0.2, y\_pred, 0.6, label='Predicted')

plt.xlabel("Test Records")

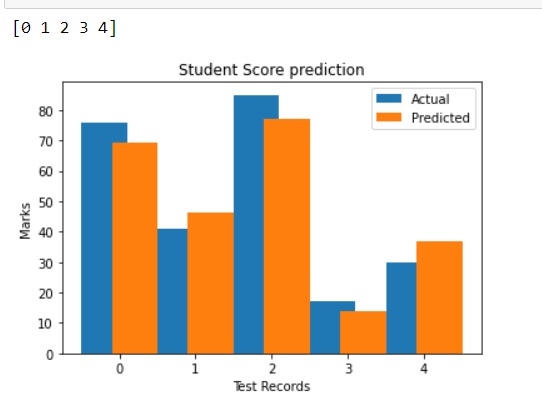
plt.ylabel("Marks")

plt.title("Student Score prediction")

plt.legend()

plt.show()

**OUTPUT:**



**Multiple Linear Regression**