SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMKUR-572103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MACHINE LEARNING TECHNIQUES LABORATORY (7RCSL02)

Student Name: YASHASWINI G R	**USN: 1SI19CS143**	**Batch No: B4**	**Date: 23-11-2022**

Evaluation:

Clarity in concepts (10 Marks)	**Execution and Results (10 Marks)**	**Maintain of observation book & Records (05 Marks)**	**Viva (10 Marks)**	**Total (35 Marks)**

Sl.No	**Name of the Faculty In-Charge**	**Signature**
1.	Dr. K Bhargavi	
2.	Dr. M B Nirmala	

Question No: 1

Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

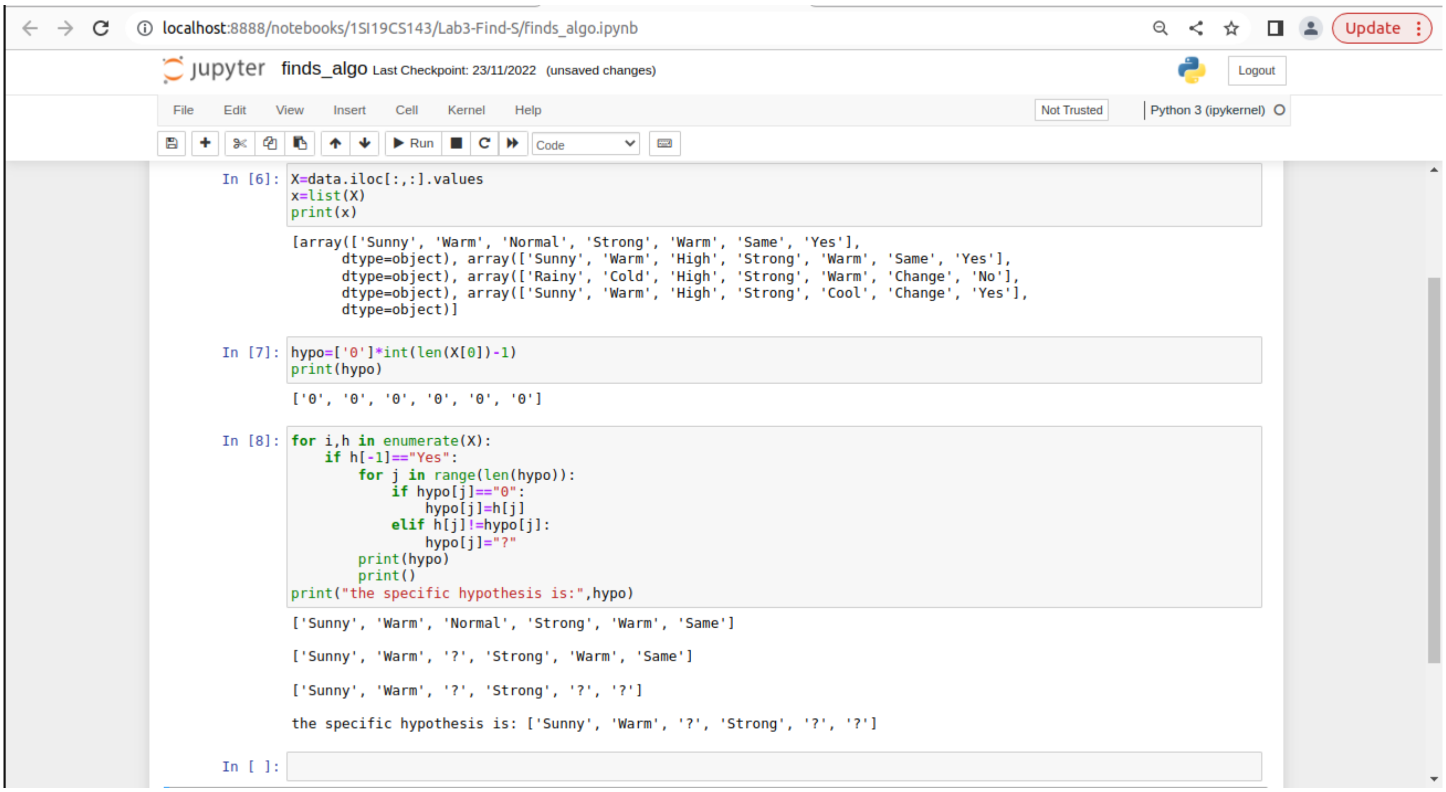
Algorithm:

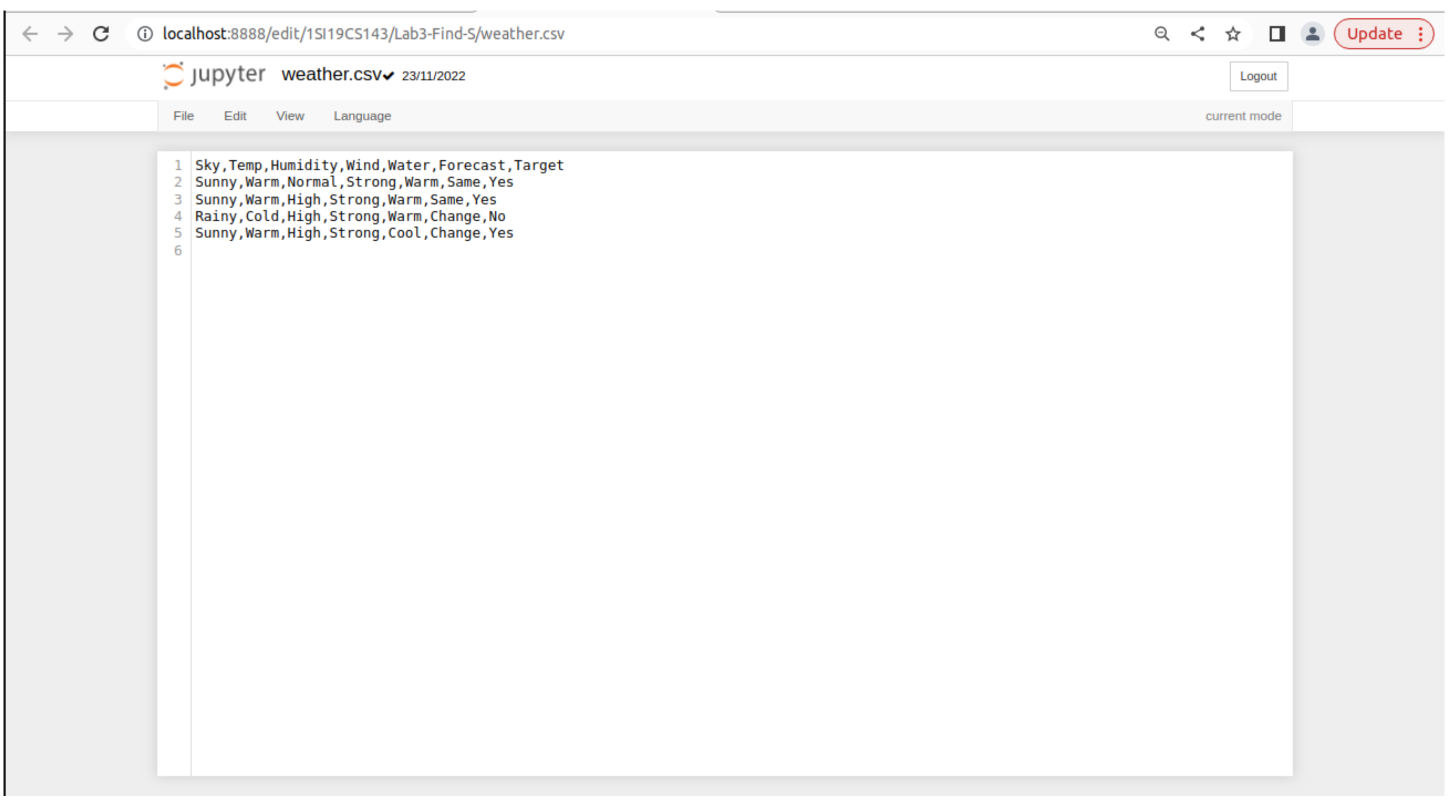
- Start with the most specific hypothesis. h = {ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}
- Take the next example and if it is negative, then no changes occur to the hypothesis.
- If the example is positive and we find that our initial hypothesis is too specific then we update our current hypothesis to a general condition, i.e each attribute in the example is checked equal to the hypothesis value
    1. If the value matches, then no changes are made
    2. If the value does not match, the value is changed to '?'
- Keep repeating the above steps till all the training examples are complete.
- After all the training examples are completed, the final hypothesis is produced which can be used to classify the new examples.

CODE:

```
import pandas as pd
file = pd.read_csv('weather.csv')


print(file)


target=file['Target'].values
attributes=file.drop("Target",axis=1).columns
num_of_attributes = len(attributes)
print(target)
hypothesis = ['0']*num_of_attributes
for i in range(len(target)):
    if target[i] == 'Yes':
        for x in range(num_of_attributes):
            if(hypothesis[x] == '0'):
                hypothesis[x] = file.iloc[i,x]
            if(hypothesis[x]!=file.iloc[i,x]):
                hypothesis[x] = '?'
    print(i+1, "hypothesis", hypothesis)
print("final hypothesis", hypothesis)
```

Output Screenshots:





Student Name: YASHASWINI G R	USN: 1SI19CS143	Batch No: B4	Date: 30-11-2022

Evaluation:

Clarity in concepts (10 Marks)	Execution and Results (10 Marks)	Maintain of observation book & Records (05 Marks)	Viva (10 Marks)	Total (35 Marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	Dr. K Bhargavi	
2.	Dr. M B Nirmala	

Question No: 2

For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

Algorithm:

Initialize G to the set of maximally general hypotheses in H
Initialize S to the set of maximally specific hypotheses in H
For each training example d, do

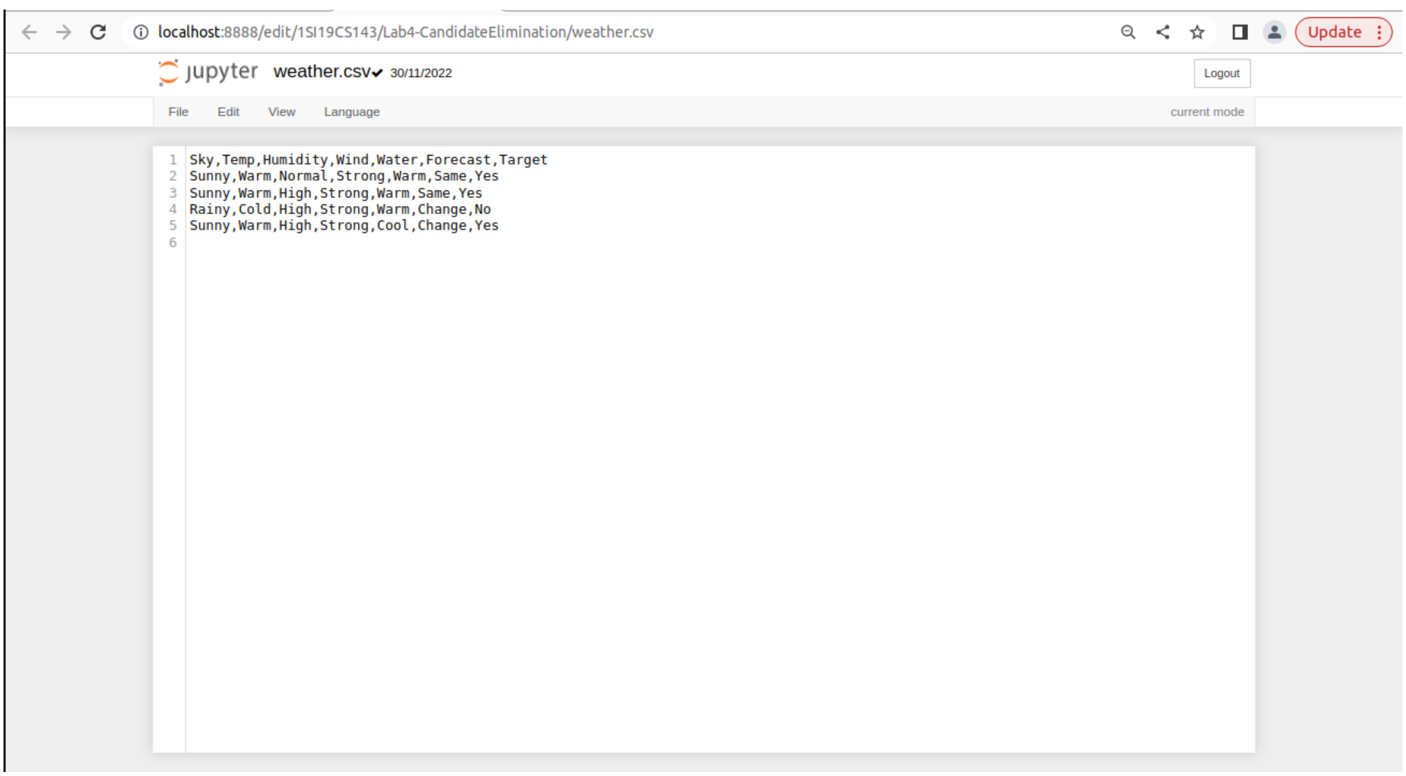
- If d is a positive example
  - Remove from G any hypothesis inconsistent with d
  - For each hypothesis s in S that is not consistent with d
    - Remove s from S
    - Add to S all minimal generalizations h of s such that
      - h is consistent with d, and some member of G is more general than h
    - Remove from S any hypothesis that is more general than another hypothesis in S
- If d is a negative example
  - Remove from S any hypothesis inconsistent with d
  - For each hypothesis g in G that is not consistent with d
    - Remove g from G
    - Add to G all minimal specializations h of g such that
      - h is consistent with d, and some member of S is more specific than h
    - Remove from G any hypothesis that is less general than another hypothesis in G

CODE:

```
import pandas as pd

import numpy as np

file = pd.read_csv('weather.csv')

concept = np.array(file.iloc[:,0:-1])

print(concept)

target = np.array(file.iloc[:,-1])

print(target)


def learn(concept, target):

    print("specific and general h")

    specific = concept[0].copy()

    print(specific)

    general = [['?' for i in range(len(specific))] for i in range(len(specific))]

    print(specific)

    for i,h in enumerate(concept):

        if target[i] == 'Yes':

            for x in range(len(specific)):

                if h[x]!= specific[x]:

                    specific[x] = '?'

                    general[x][x] = '?'

        if target[i] == 'No':

            for x in range(len(specific)):

                if h[x]!=specific[x]:

                    general[x][x] = specific[x]

                else:
```

```
            general[x][x] = '?'
        print("hypothesis,", i+1)
        print(specific)
        print(general)

    indices = [i for i,val in enumerate(general) if val == ['?','?','?','?','?','?']]

    for i in indices:
        general.remove(['?','?','?','?','?','?'])

    return (general,specific)

f_g, f_s = learn(concept,target)
print("\n")
print("Final specific :", f_s)
print("Final general : ", f_g)
```

Output Screenshots:



Student Name: YASHASWINI G R	USN: 1SI19CS143	Batch No: B4	Date: 04-01-2023

Evaluation:

Clarity in concepts (10 Marks)	Execution and Results (10 Marks)	Maintain of observation book & Records (05 Marks)	Viva (10 Marks)	Total (35 Marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	Dr. K Bhargavi	
2.	Dr. M B Nirmala	

Question No: 3

Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets.

Algorithm:

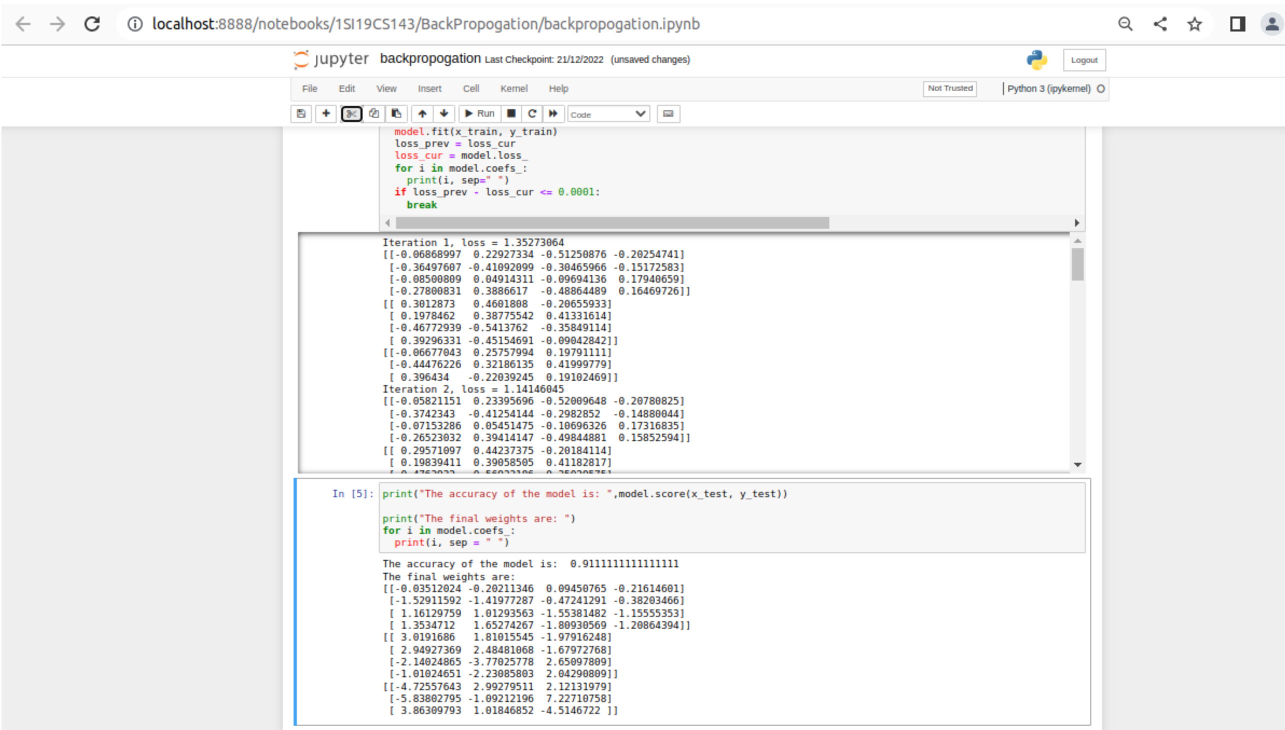
- Inputs X, arrive through the preconnected path.
- The input is modeled using true weights W. Weights are usually chosen randomly.
- Calculate the output of each neuron from the input layer to the hidden layer to the output layer.
- Calculate the error in the outputs

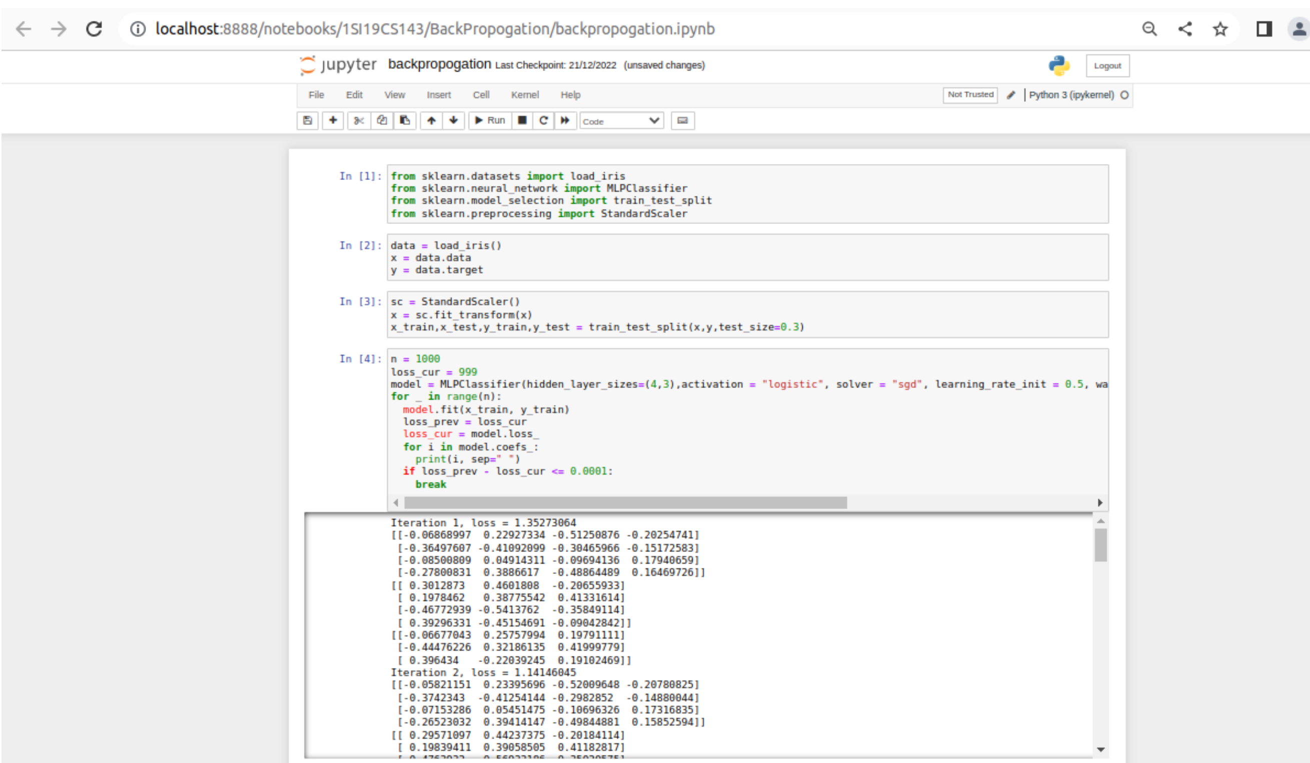
  Backpropagation Error= Actual Output – Desired Output
- From the output layer, go back to the hidden layer to adjust the weights to reduce the error.
- Step 6: Repeat the process until the desired output is achieved.

CODE:

```
from sklearn.datasets import load_iris

from sklearn.neural_network import MLPClassifier

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

data = load_iris()

x =  data.data

y = data.target

sc = StandardScaler()

x = sc.fit_transform(x)

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)

n = 1000

loss_cur = 999

model = MLPClassifier(hidden_layer_sizes=(4,3),activation = "logistic", solver =
"sgd",learning_rate_init = 0.5, warm_start = True, max_iter = 1, verbose = True, random_state = 1)

for _ in range(n):

    model.fit(x_train, y_train)

    loss_prev = loss_cur

    loss_cur  =  model.loss_

    for i in model.coefs_:

        print(i, sep=" ")

    if loss_prev - loss_cur <= 0.0001:

        break

print("The accuracy of the model is: ",model.score(x_test, y_test))

print("The final weights are: ")

for i in model.coefs_:

    print(i, sep = " ")
```

Output Screenshots:





Student Name: YASHASWINI G R	USN: 1SI19CS143	Batch No: B4	Date: 21-12-2022

Evaluation:

Clarity in concepts (10 Marks)	Execution and Results (10 Marks)	Maintain of observation book & Records (05 Marks)	Viva (10 Marks)	Total (35 Marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	Dr. K Bhargavi	
2.	Dr. M B Nirmala	

Question No: 4

Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

Algorithm:

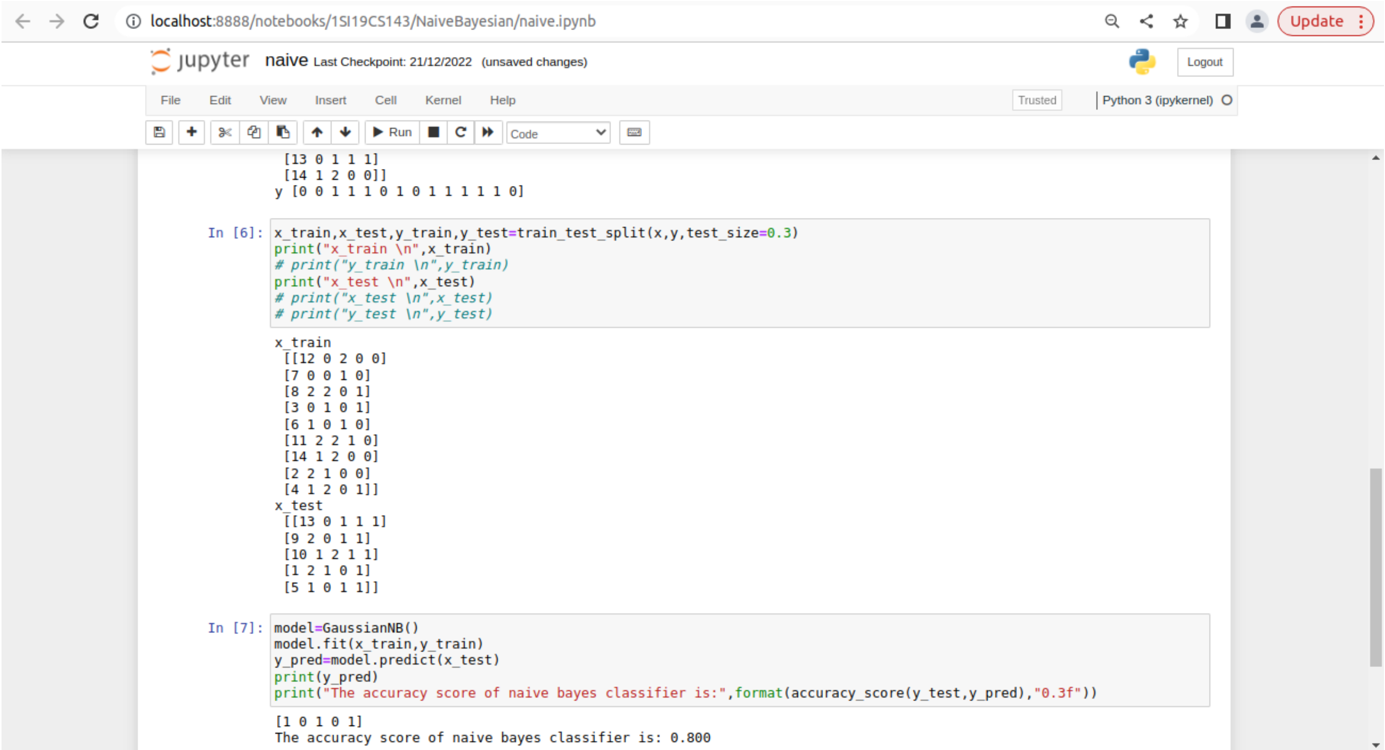
- Convert the given dataset into frequency tables.
- Generate Likelihood table by finding the probabilities of given features.
- Now, use Bayes theorem to calculate the posterior probability.

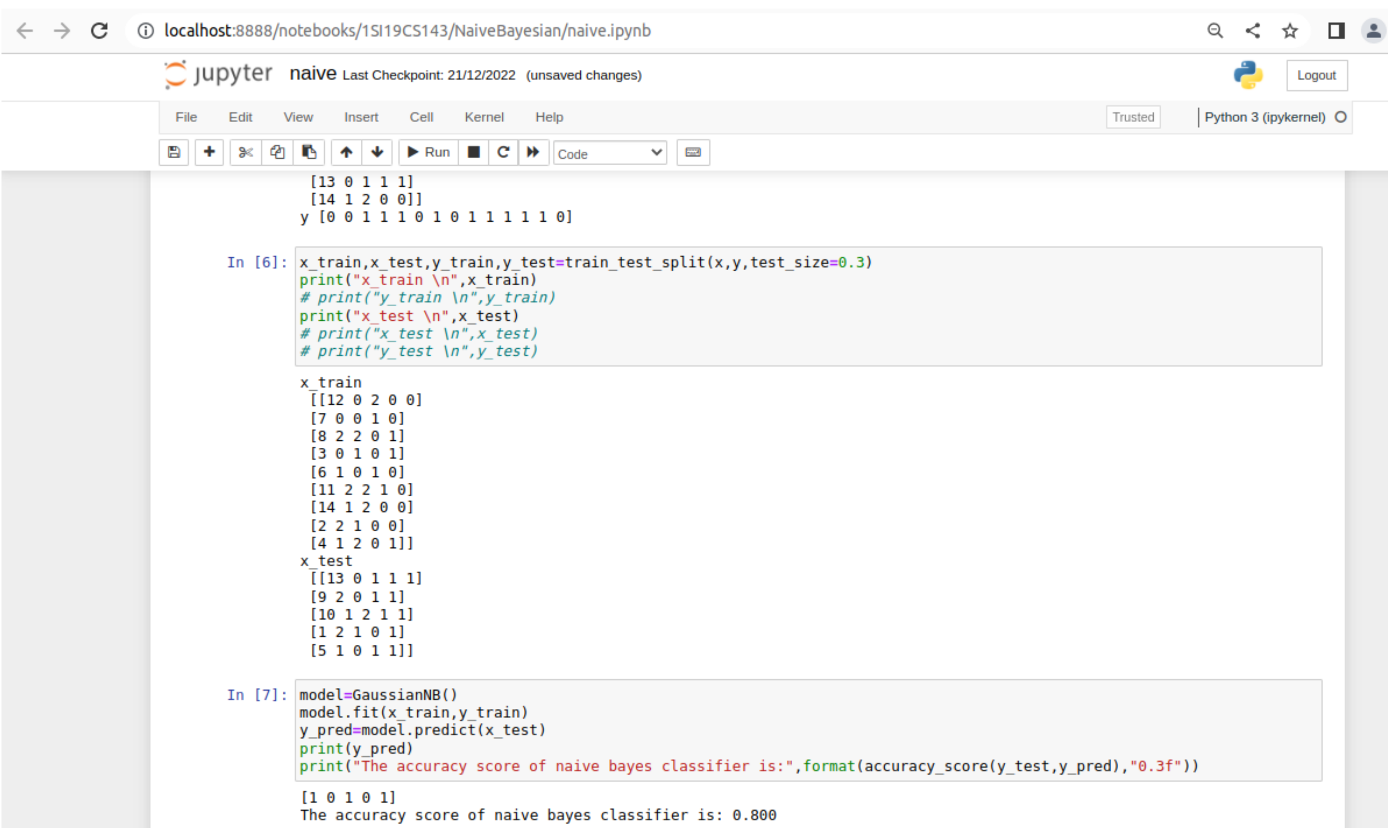
$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

CODE:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn import preprocessing

from sklearn.naive_bayes import GaussianNB

from sklearn import metrics


dataset = pd.read_csv("naive.csv")

dataset_df = pd.DataFrame(dataset)

en = preprocessing.LabelEncoder()

dataset_df_encoded = dataset_df.apply(en.fit_transform)


data = dataset_df_encoded.drop(['play'], axis=1)

target = dataset_df_encoded['play']


print(data)

print("target:-")

print(target)

X_train,X_test,Y_train,Y_test = train_test_split(data,target,test_size = 0.25)

model = GaussianNB()

learntModel = model.fit(X_train,Y_train)

prediction = learntModel.predict(X_test)

print(list(prediction))

print(list(Y_test))

print("Accuracy  :  ",  metrics.accuracy_score(prediction,Y_test))
```

Output Screenshots:





Student Name: YASHASWINI G R	USN: 1SI19CS143	Batch No: B4	Date: 18-01-2023

Evaluation:

Clarity in Concepts (10 Marks)	Execution and Results (10 Marks)	Maintain of observation book & Records (05 Marks)	Viva (10 Marks)	Total (35 Marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	Dr. K Bhargavi	
2.	Dr. M B Nirmala	

Question No: 5

Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.

Algorithm:

- Define the structure of the Bayesian Belief Network (BBN) in terms of a directed acyclic graph (DAG) with nodes representing variables and edges representing dependencies between variables.
- Define the conditional probability tables (CPTs) for each variable, which give the probability of the variable taking on a certain value given the values of its parent variables in the graph.
- Initialize the network by setting the initial belief state of each variable to a prior probability distribution.
- Perform probabilistic inference using the network by computing the marginal probability distributions of the variables of interest given the evidence.

  a. Use the CPTs to compute the likelihood of the evidence given the current belief state of the network.

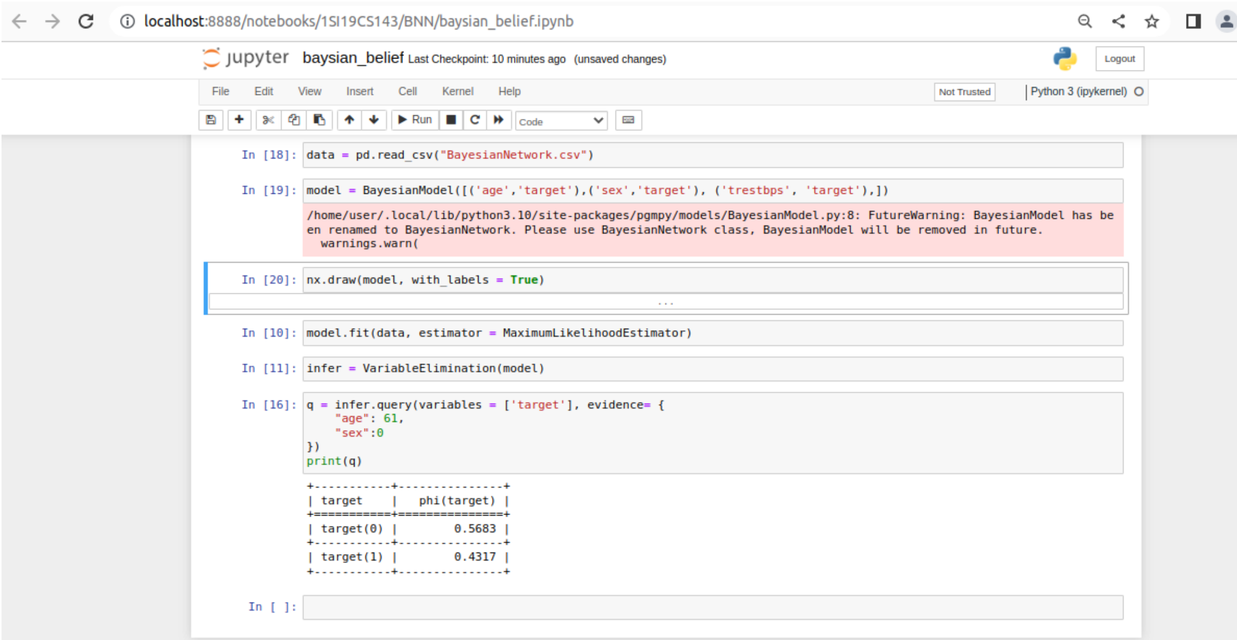
  b. Use Bayes' theorem to update the belief state of each variable based on the new evidence.

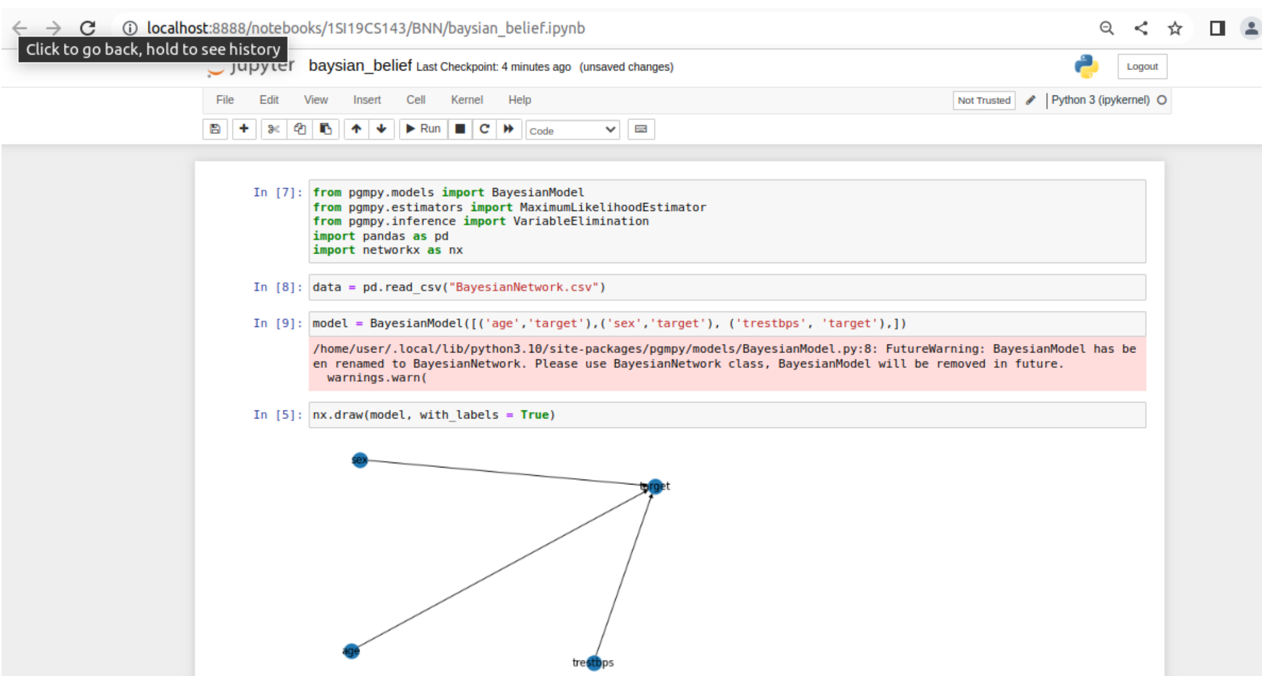
  c. Repeat steps a and b until the belief state of the network converges or a stopping criterion is met.
- Use the final belief state of the network to make decisions or predictions.

CODE:

```
pip install pgmpy

from pgmpy.models import BayesianNetwork

from pgmpy.estimators import MaximumLikelihoodEstimator

from pgmpy.inference import VariableElimination

import pandas as pd

import networkx as nx


data = pd.read_csv("BayesianNetwork.csv")

model = BayesianNetwork([('age','target'),('sex','target'), ('trestbps', 'target'),])


nx.draw(model, with_labels = True)

model.fit(data, estimator = MaximumLikelihoodEstimator)

infer = VariableElimination(model

q = infer.query(variables = ['target'], evidence= {

   "age": 29,

   "sex":0

})

print(q)
```

Output Screenshots:





Student Name: YASHASWINI G R	**USN: 1SI19CS143**	**Batch No: B4**	**Date: 18-01-2023**

Evaluation:

Clarity in concepts (10 Marks)	Execution and Results (10 Marks)	Maintain of observation book & Records (05 Marks)	Viva (10 Marks)	Total (35 Marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	Dr. K Bhargavi	
2.	Dr. M B Nirmala	

Question No: 6

Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

Algorithm:

The algorithm for Gaussian Mixture Model (GMM) involves the following steps:

1. Initialize the parameters of the Gaussian distributions (mean, covariance, and mixing coefficients)
2. Expectation step: Compute the probability of each data point belonging to each of the Gaussian distri-butions
3. Maximization step: Re-estimate the parameters of the Gaussian distributions using the probabilitiescomputed in step 2
4. Repeat steps 2 and 3 until the parameters converge to a stable solution
5. Final step: Assign each data point to the Gaussian distribution with the highest probability of member-ship

The algorithm for K-Means involves the following steps:

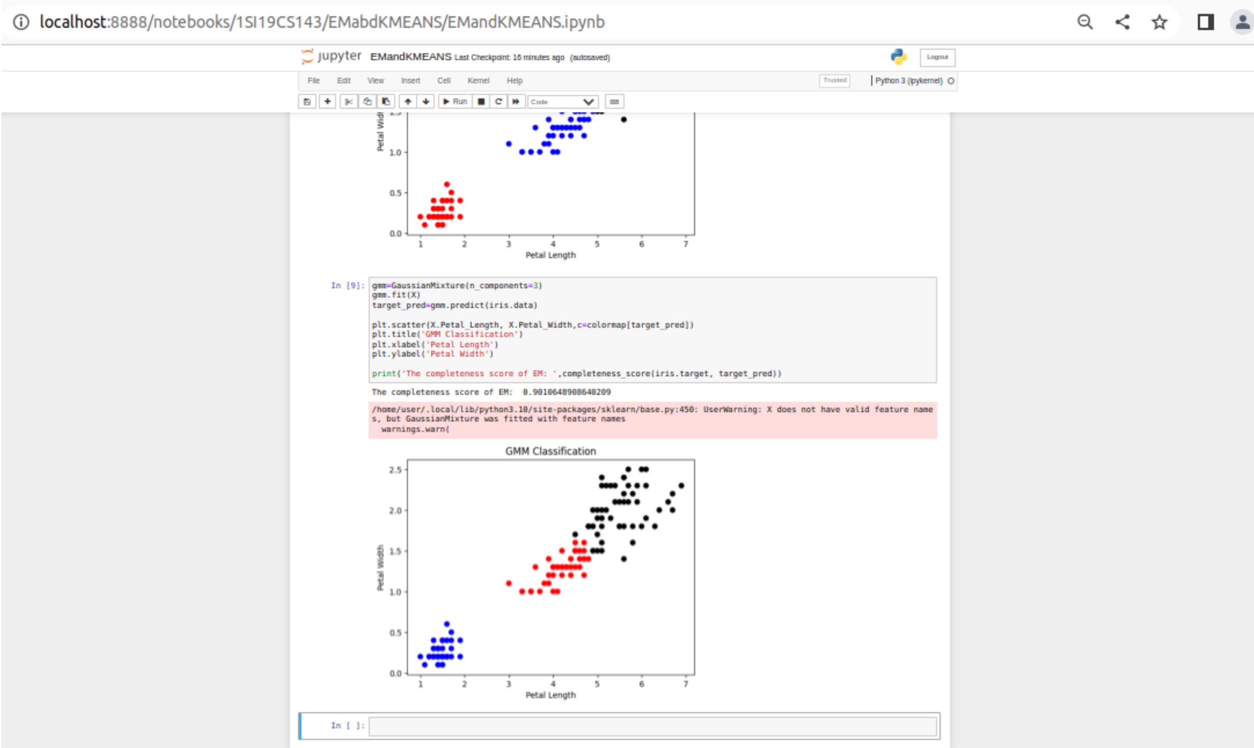
1. Select the number K to decide the number of clusters.
2. Select random K points or centroids. (It can be other from the input dataset).
3. Assign each data point to their closest centroid, which will form the predefined K clusters.
4. Calculate the variance and place a new centroid of each cluster.
5. Repeat the third steps, which means reassign each data-point to the new closest centroid of each cluster.
6. If any reassignment occurs, then go to step-4 else go to FINISH

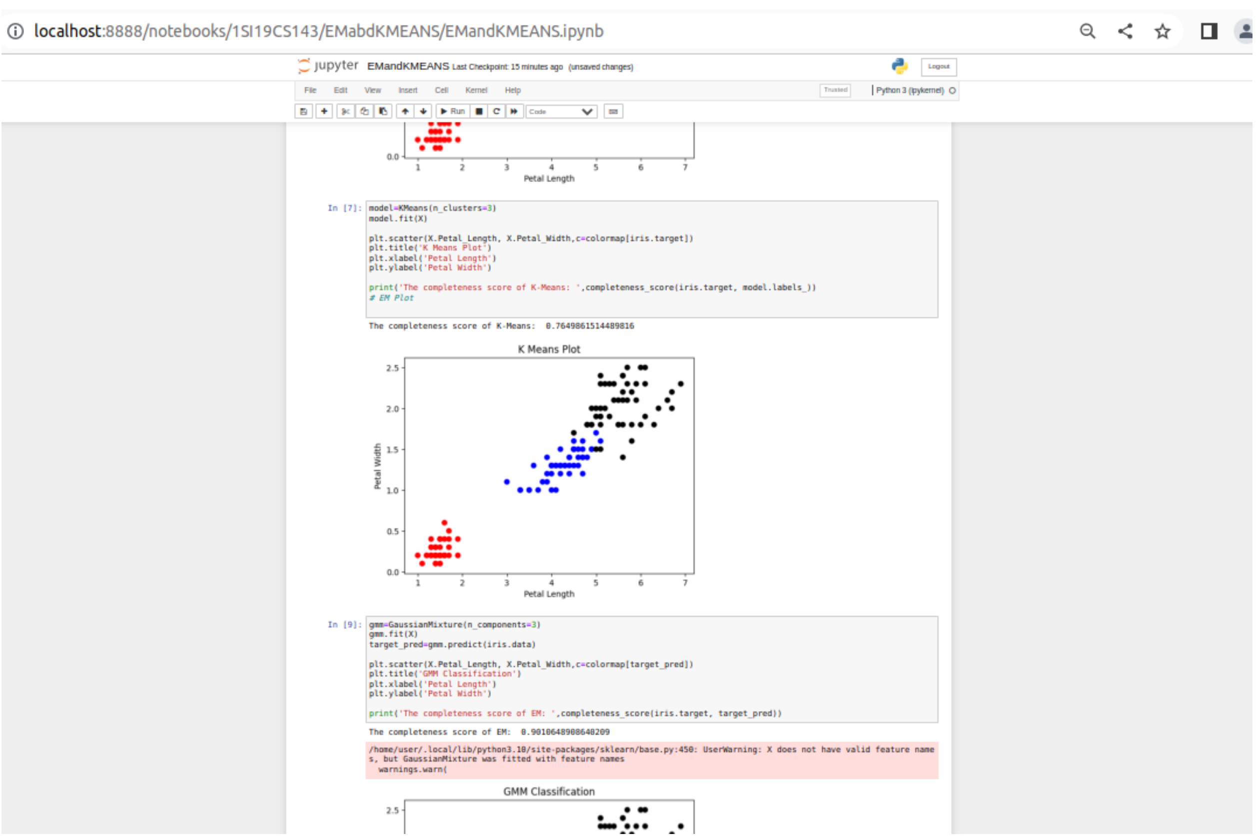
CODE:

```
from sklearn.datasets import load_iris

from sklearn.cluster import KMeans

from sklearn.mixture import GaussianMixture

from sklearn.metrics import completeness_score

import matplotlib.pyplot as plt

import numpy as np


data = load_iris()

x = data.data

y = data.target


wcss = []

for i in range(2,11):

  model = KMeans(n_clusters = i)

  model.fit(x)

  wcss.append(model.inertia_)


plt.figure()

plt.plot(range(2,11), wcss)


model = KMeans(n_clusters = 3)

model.fit(x)

print("The completeness score of KMeans is: ",completeness_score(y,model.labels_))


gmm = GaussianMixture(n_components = 3, random_state = 1)

gmm.fit(x)
```

```
y_pred = gmm.predict(x)
print("The completeness score of Gaussian Mixture is: ",completeness_score(y,y_pred))


plt.figure(figsize=(21,7))
colorMap = np.array(["lime","red","black"])


plt.subplot(1,3,1)
plt.scatter(x[:,2],x[:,3],c = colorMap[y])
plt.title("Real Classification")
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")


plt.subplot(1,3,2)
plt.scatter(x[:,2],x[:,3],c = colorMap[model.labels_])
plt.title("KMeans Classification")
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")


plt.subplot(1,3,3)
plt.scatter(x[:,2],x[:,3],c = colorMap[gmm.predict(x)],s=40)
plt.title("GaussianMixture Classification")
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
```

Output Screenshots:





localhost:8888/notebooks/045/EMANDKMEANS.ipynb

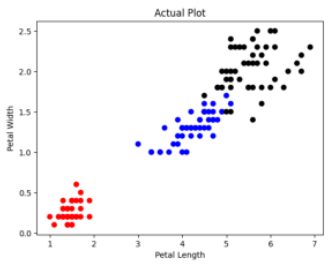
jupyter EMANDKMEANS Last Checkpoint: 19 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Help | Not Trusted | Python 3 (ipykernel)

In [6]:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.metrics import completeness_score
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

iris = load_iris()

X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']

y = pd.DataFrame(iris.target)
y.columns = ['Targets']
colormap=np.array(['red','blue','black'])
# Actual Plot
plt.scatter(X.Petal_Length, X.Petal_Width,c=colormap[iris.target])
plt.title('Actual Plot')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
# K-Means Plot
```

Out[6]: Text(0, 0.5, 'Petal Width')



In [3]:

```
model=KMeans(n_clusters=3)
model.fit(X)

plt.scatter(X.Petal_Length, X.Petal_Width,c=colormap[iris.target])
plt.title('K Means Plot')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')

print('The completeness score of K-Means: ',completeness_score(iris.target, model.labels_))
# EM Plot
```

The completeness score of K-Means: 0.7649861514489816

Student Name: YASHASWINI G R	USN: 1SI19CS143	Batch No: B4	Date: 16-11-2022	
Evaluation:				
Clarity in concepts (10 Marks)	**Execution and Results (10 Marks)**	**Maintain of observation book & Records (05 Marks)**	**Viva (10 Marks)**	**Total (35 Marks)**

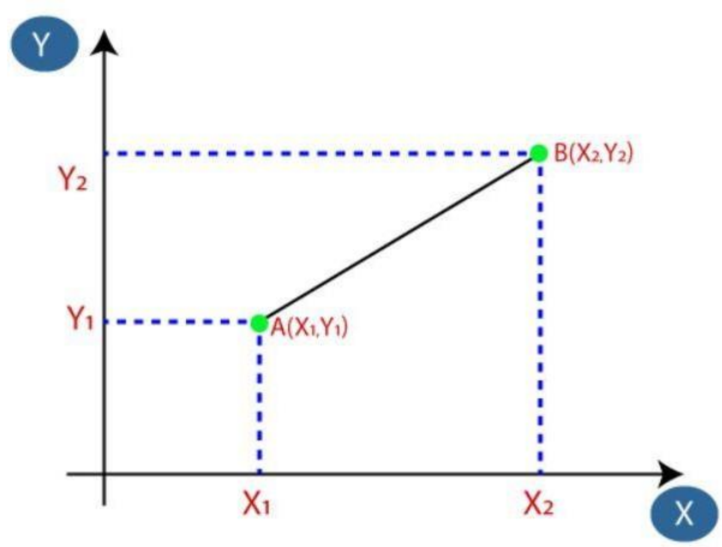
Sl.No	Name of the Faculty In-Charge	Signature
1.	Dr. K Bhargavi	
2.	Dr. M B Nirmala	

Question No: 7

Write a program to implement k-Nearest Neighbor algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

Algorithm:

- Select the number K of the neighbors
- Calculate the Euclidean distance of K number of neighbors



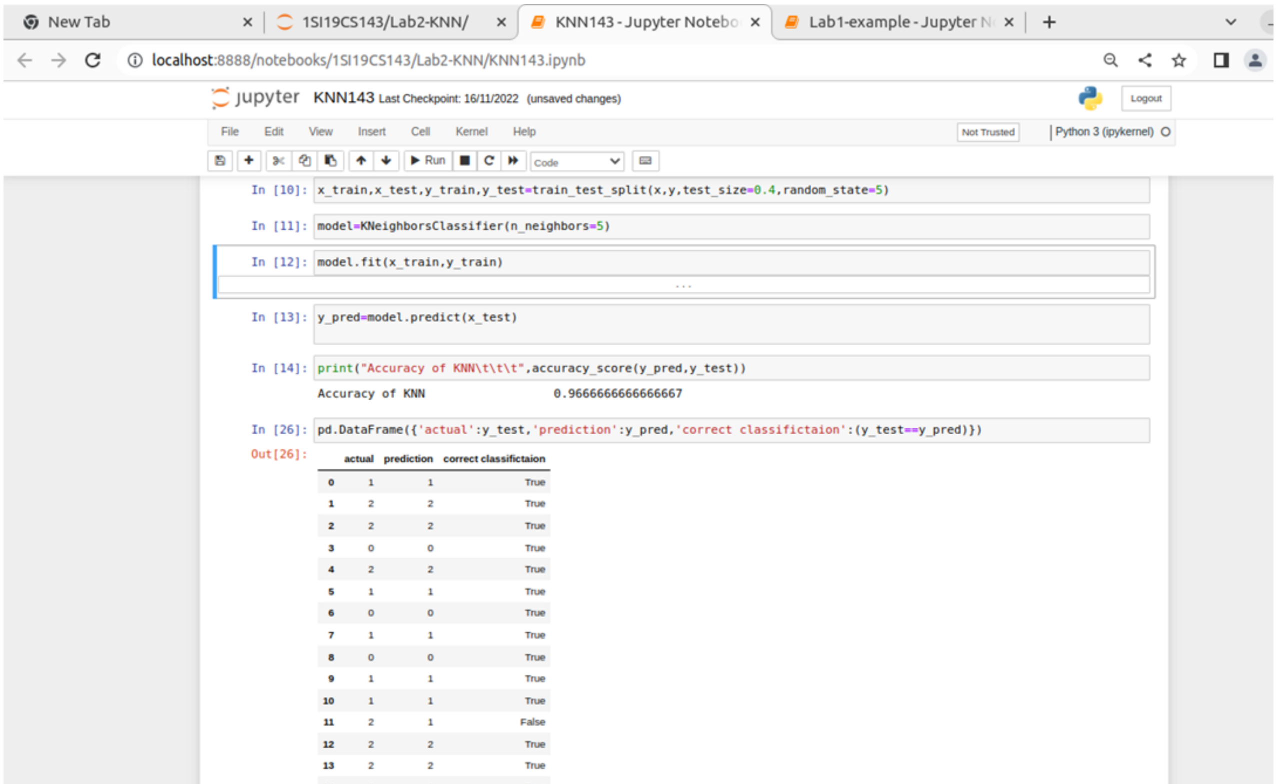
Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

- Take the K nearest neighbors as per the calculated Euclidean distance.
- Among these k neighbors, count the number of the data points in each category.
- Assign the new data points to that category for which the number of the neighbor is maximum

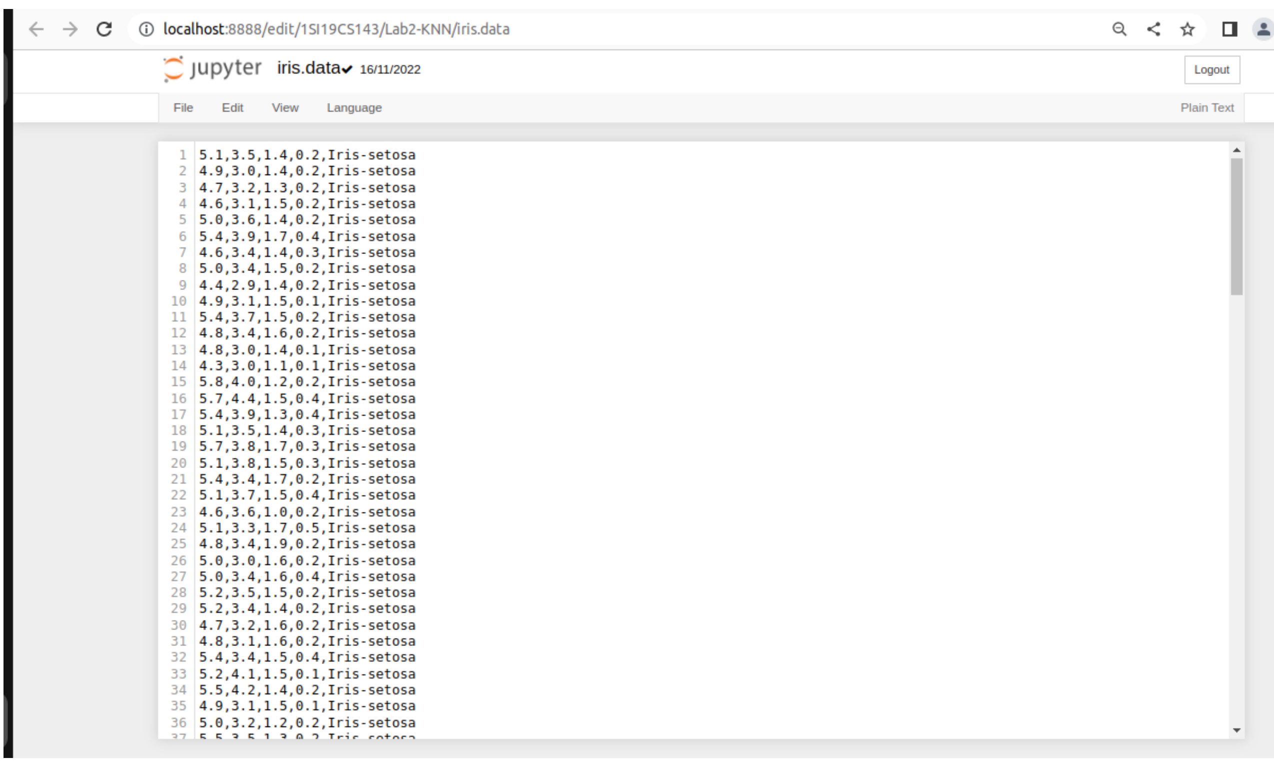
CODE:

```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

data =load_iris()

x_train,x_test,y_train,y_test=train_test_split(data.data,data.target,test_size=0.3,random_state=5)
model=KNeighborsClassifier(n_neighbors=5)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
mismatch = y_pred - y_test
print("Total missclassified, : " , sum(abs(mismatch)))
print("Acuraccy of KNN\t\t\t\t",accuracy_score(y_pred,y_test))
pd.DataFrame({'actual':y_test,'predction':y_pred,'correct    classification':(y_test==y_pred)})
```

Output Screenshots:



	actual	prediction	correct classifictaion
0	1	1	True
1	2	2	True
2	2	2	True
3	0	0	True
4	2	2	True
5	1	1	True
6	0	0	True
7	1	1	True
8	0	0	True
9	1	1	True
10	1	1	True
11	2	1	False
12	2	2	True
13	2	2	True
14	0	0	True




```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
5.0,3.4,1.6,0.4,Iris-setosa
5.2,3.5,1.5,0.2,Iris-setosa
5.2,3.4,1.4,0.2,Iris-setosa
4.7,3.2,1.6,0.2,Iris-setosa
4.8,3.1,1.6,0.2,Iris-setosa
5.4,3.4,1.5,0.4,Iris-setosa
5.2,4.1,1.5,0.1,Iris-setosa
5.5,4.2,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.0,3.2,1.2,0.2,Iris-setosa
```

Student Name: YASHASWINI G R	USN: 1SI19CS143	Batch No: B4	Date: 28-12-2022

Evaluation:

Clarity in concepts (10 Marks)	Execution and Results (10 Marks)	Maintain of observation book & Records (05 Marks)	Viva (10 Marks)	Total (35 Marks)

Sl.No	Name of the Faculty In-Charge	Signature
1.	Dr. K Bhargavi	
2.	Dr. M B Nirmala	

Question No: 8

Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

Algorithm:

- Read the given data sample to X and add the curve to Y.
- Set the value for smoothening parameter or free parameter.
- Set the bias/point of interest X_0 which is a subset of X.
- Determine the weight matrix using

$$w_i = e^{-\left(\frac{(x_i - x)^2}{2\tau^2}\right)}$$

- Determine the value of model term parameter Θ using

$$\theta = (X^T W X)^{-1} X^T W Y$$

- Prediction: $X_0 * \Theta$

CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("tips.csv")
bill = data.total_bill
tip = data.tip
mBill = np.mat(bill)
mTip = np.mat(tip)
m = mBill.shape[1]
one = np.mat(np.ones(m))
X = np.hstack((one.T, mBill.T))

def kernel(point,xmat,k):
    m,n = xmat.shape
    weights = np.mat(np.eye(m))
    for j in range(m):
        diff = point - xmat[j]
        weights[j,j] = np.exp(diff*diff.T/(-2*k**2))
    return weights

def  Beta(x_value,x,y,k):
    weight = kernel(x_value,x,k)
    W = (X.T * (weight * X)).I*(X.T *(weight * y.T))
    return W

def localWeightRegression(x,y,k):
    m,n = x.shape
    ypred = np.zeros(m)
    for i in range(m):
```

```
        ypred[i] = x[i] * Beta(x[i],x,y,k)
    return ypred

ypred = localWeightRegression(X,mTip,2)
SortIndex = X[:,1].argsort(0)
xsort = X[SortIndex][:,0]

fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(bill,tip,  color='blue')
ax.plot(xsort[:,1],ypred[SortIndex], color = 'red', linewidth=1)
plt.xlabel('Total bill')
plt.ylabel('Tip')
```

Output Screenshots: