MACHINE LEARNING LABORATORY
(Effective from the academic year 2018 -2019)
SEMESTER – VI

Subject Code	18AIL66	**CIE Marks**	40
Number of Contact Hours/Week	0:2:2	**SEE Marks**	60
Total Number of Lab Contact Hours		**Exam Hours**	3 Hrs

Credits – 2

Course Learning Objectives: This course will enable students to:

- Implement and evaluate ML algorithms in Python/Java programming language.

Descriptions (if any):
1. The programs can be implemented in either JAVA or Python.
2. Data sets can be taken from standard repository such as UCI

Installation procedure of the required software must be demonstrated, carried out in groups and documented in the journal.

Programs List:

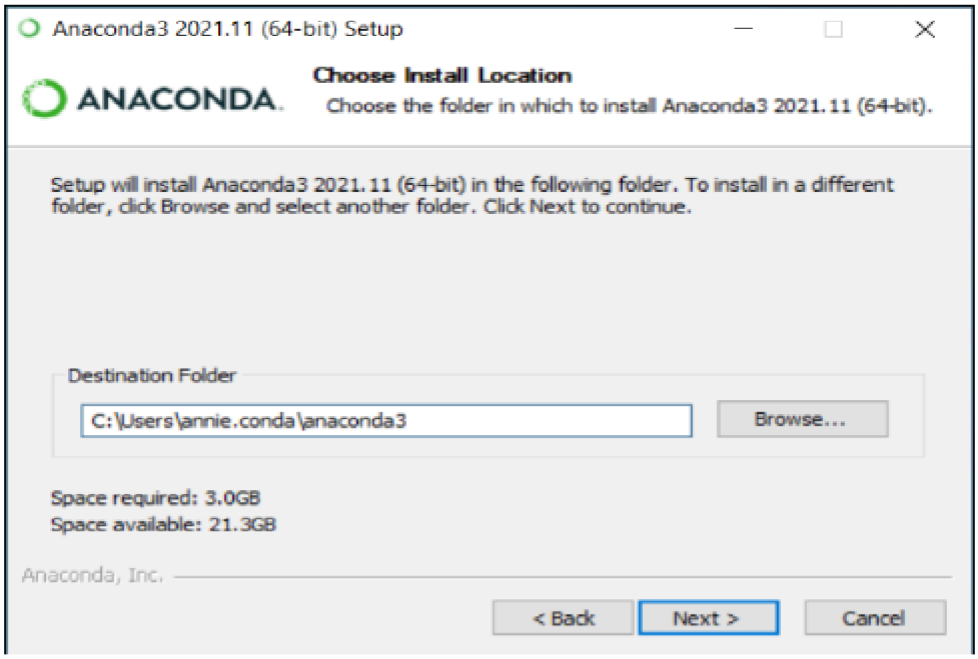
1.	Implement and demonstratethe**FIND-Salgorithm** for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file and show the output for test cases. Develop an interactive program by Compareing the result by implementing **LIST THEN ELIMINATE** algorithm.
2	For a given set of training data examples stored in a .CSV file, implement and demonstrate the **Candidate-Elimination**algorithm. Output a description of the set of all hypotheses consistent with the training examples.
3	Demonstrate Pre processing (Data Cleaning, Integration and Transformation) activity on suitable data: For example: Identify and Delete **Rows that Contain Duplicate Data** by considering an appropriate dataset. Identify and Delete **Columns That Contain a Single Value** by considering an appropriate dataset.
4	Demonstrate the working of the decision tree based **ID3 algorithm**. Use an appropriate data set for building the decision tree and apply this knowledge toclassify a new sample.
5	Demonstrate the working of the Random forest **algorithm**. Use an appropriate data set for building and apply this knowledge toclassify a new sample.
6	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.
7	Assuming a set of documents that need to be classified, use the **naive Bayesian Classifier** model to perform this task. Calculate the accuracy, precision, and recall for your data set.
8	Construct a**Bayesian network** considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.
9	Demonstrate the working of EM algorithm to cluster a set of data stored in a .CSV file.
10	Demonstrate the working of SVM classifier for a suitable data set

Installation

1. Download the Anaconda installer.
2. Go to your Downloads folder and double-click the installer to launch. To prevent permission errors, do not launch the installer from the Favorites folder.

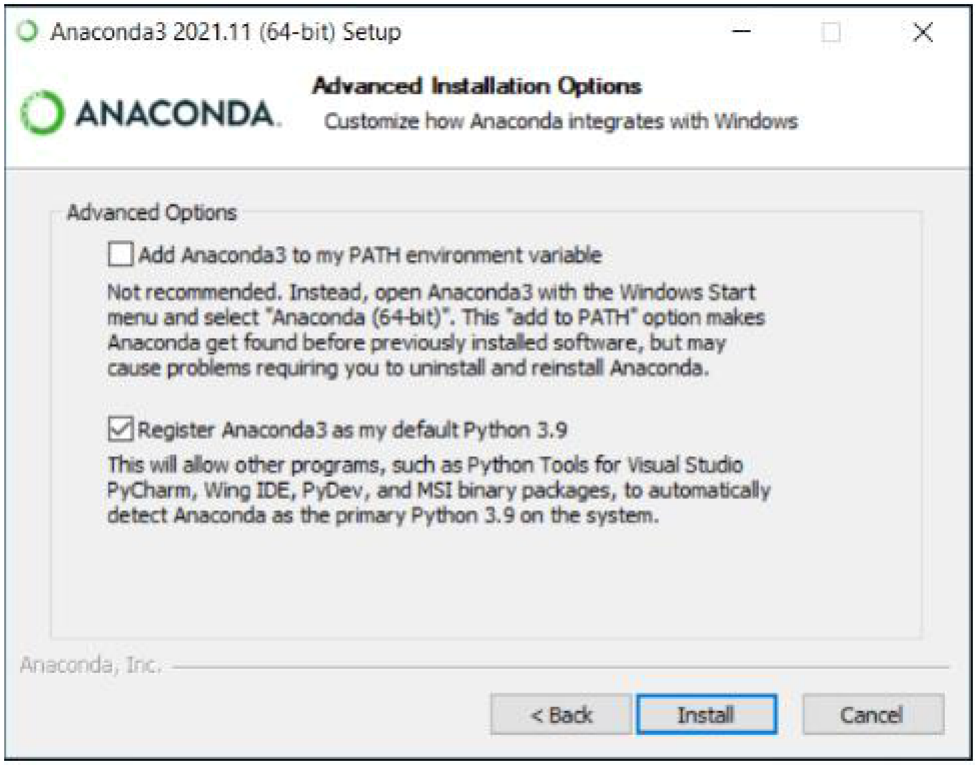
> **Note**
>
> If you encounter issues during installation, temporarily disable your anti-virus software during install, then re-enable it after the installation concludes. If you installed for all users, uninstall Anaconda and re-install it for your user only.

3. Click **Next**.
4. Read the licensing terms and click **I Agree**.
5. It is recommended that you install for **Just Me**, which will install Anaconda Distribution to just the current user account. Only select an install for **All Users** if you need to install for all users' accounts on the computer (which requires Windows Administrator privileges).
6. Click **Next**.
7. Select a destination folder to install Anaconda and click **Next**. Install Anaconda to a directory path that does not contain spaces or unicode characters. For more information on destination folders, see the FAQ.



8. Choose whether to add Anaconda to your PATH environment variable or register Anaconda as your default Python. We **don't recommend** adding Anaconda to your PATH environment variable, since this can interfere with other software. Unless you plan on installing and running multiple versions of Anaconda or multiple versions of Python, accept the default and leave this box checked. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.

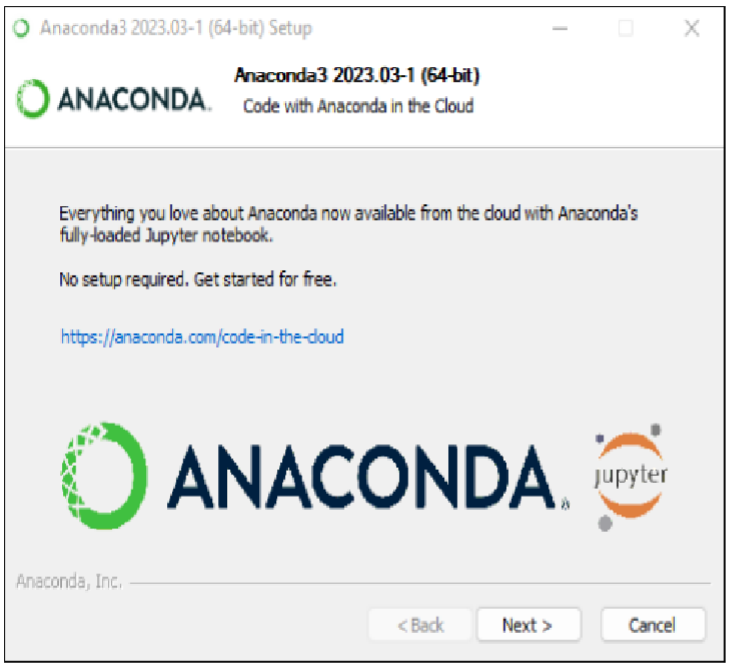
> **Note**
>
> As of `Anaconda Distribution 2022.05`, the option to add Anaconda to the PATH environment variable during an **All Users** installation has been disabled. This was done to address a security exploit. You can still add Anaconda to the PATH environment variable during a **Just Me** installation.



9. Click **Install**. If you want to watch the packages Anaconda is installing, click Show Details.

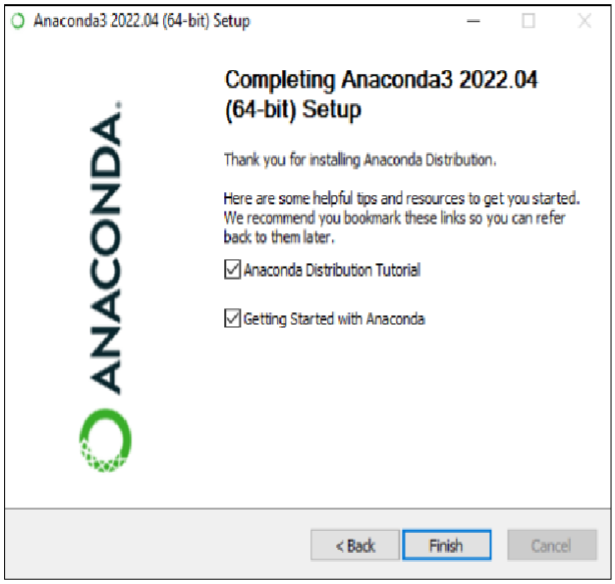
10. Click **Next**.

11. Optional: To learn more about Anaconda's cloud notebook service, go to https://www.anaconda.com/code-in-the-cloud.



Or click **Continue** to proceed.

12. After a successful installation you will see the "Thanks for installing Anaconda" dialog box:



13. If you wish to read more about Anaconda.org and how to get started with Anaconda, check the boxes "Anaconda Distribution Tutorial" and "Learn more about Anaconda". Click the **Finish** button.

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 and show the output for test cases. Develop an interactive program by Comparing the result by implementing LIST THEN ELIMINATE algorithm.

FIND-S algorithm

```
import csv
a = []
with open('enjoysport.csv', 'r') as csvfile:
    for row in csv.reader(csvfile):
        a.append(row)
    print(a)
print("\n The total number of training instances are : ",len(a))
num_attribute = len(a[0])-1
print("\n The initial hypothesis is : ")
hypothesis = ['0']*num_attribute
print(hypothesis)
for i in range(0, len(a)):
    if a[i][num_attribute] == 'yes':
        for j in range(0, num_attribute):
            if hypothesis[j] == '0' or hypothesis[j] == a[i][j]:
                hypothesis[j] = a[i][j]
            else:
                hypothesis[j] = '?'
    print("\n The hypothesis for the training instance {} is : \n" .format(i+1),hypothesis)
print("\n The Maximally specific hypothesis for the training instance is ")
print(hypothesis)
```

Datasets:

sky	airtemp	humidity	wind	water	forcast	enjoysport
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	change	yes

Output :

```
 The total number of training instances are :  5

 The initial hypothesis is :
['0', '0', '0', '0', '0', '0']

 The hypothesis for the training instance 1 is :
 ['0', '0', '0', '0', '0', '0']

 The hypothesis for the training instance 2 is :
 ['sunny', 'warm', 'normal', 'strong', 'warm', 'same']

 The hypothesis for the training instance 3 is :
 ['sunny', 'warm', '?', 'strong', 'warm', 'same']

 The hypothesis for the training instance 4 is :
 ['sunny', 'warm', '?', 'strong', 'warm', 'same']

 The hypothesis for the training instance 5 is :
 ['sunny', 'warm', '?', 'strong', '?', '?']

 The Maximally specific hypothesis for the training instance is
['sunny', 'warm', '?', 'strong', '?', '?']
```

LIST-THEN-ELIMINATE algorithm

```
def list_then_eliminate(elements):
    # Step 1: Display the original list
    print("Original list:", elements)

    # Step 2: Compare and eliminate
    elements result = []
    for i in range(len(elements)):
        # Compare the current element with the rest of the
        list is_unique = True
        for j in range(len(elements)):
            if i != j and elements[i] == elements[j]:
                is_unique = False
                break

        # Add the element to the result if it is unique
        if is_unique:
            result.append(elements[i])

    # Step 3: Display the final
    result print("Result:", result)

# Interactive part
input_list = input("Enter a list of elements (space-separated): ").split()
list_then_eliminate(input_list)
```

Output :

```
Enter a list of elements (space-separated): 1 2 3 3 4 5 4 6
Original list: ['1', '2', '3', '3', '4', '5', '4', '6']
Result: ['1', '2', '3', '4', '5', '6']
```

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

```
import numpy as np
import pandas as pd
data = pd.DataFrame(data=pd.read_csv('enjoysport.csv'))
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
target = np.array(data.iloc[:,-1])
print(target)
def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("initialization of specific_h and general_h")
    print(specific_h)
    general_h = [["?" for i in range(len(specific_h))]
    for i in range(len(specific_h))]
    print(general_h)
    for i, h in enumerate(concepts):
        if target[i] == "yes":
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'
                print(specific_h)
        print(specific_h)
        if target[i] == "no":
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'
        print(" steps of Candidate Elimination Algorithm",i+1)
        print(specific_h)
        print(general_h)
        indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?', '?', '?']]
        for i in indices:
            general_h.remove(['?', '?', '?', '?', '?', '?'])
        return specific_h, general_h
s_final, g_final = learn(concepts, target)
print("Final Specific_h:", s_final, sep="\n")
print("Final General_h:", g_final, sep="\n")
```

Datasets :

sky	airtemp	humidity	wind	water	forcast	enjoysport
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	change	yes

Output:

```
[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
 ['sunny' 'warm' 'high' 'strong' 'warm' 'same']
 ['rainy' 'cold' 'high' 'strong' 'warm' 'change']
 ['sunny' 'warm' 'high' 'strong' 'cool' 'change']]
['yes' 'yes' 'no' 'yes']
initialization of specific_h and general_h
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?']]
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
 steps of Candidate Elimination Algorithm 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?']]
Final Specific_h:
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
Final General_h:
[]
```

3.Demonstrate Pre-processing (Data Cleaning, Integration and Transformation) activity on suitable data: For example: Identify and Delete Rows that Contain Duplicate Data by considering an appropriate dataset. Identify and Delete Columns That Contain a Single Value by considering an appropriate dataset.

```
import pandas as pd

# Load the dataset
df = pd.read_csv('customer_data.csv')

# Display the original dataset
print("Original dataset:")
print(df)

# Identify and delete rows that contain duplicate data
for column in df.columns:
        df[columns]= df[columns].drop_duplicates()

# Identify and delete columns that contain a single
value df = df.loc[:, df.nunique() > 1]

# Display the pre-processed dataset
print("\nPre-processed dataset:")
print(df)
```

Datasets :

usn	name	age	gender	email-id	mobile	address
66	anusha	20	female	anusha123@gmail.com	9854367524	Davanagere
67	akbar	21	male	akbar@gmail.com	9658765356	Chitradurga
68	anusha	20	female	anusha123@gmail.com	9854367524	Davanagere
69	anusha	20	female	anusha123@gmail.com	9854367524	Davanagere
70	kiran	21	male	kiran@yahoo.com	9342567845	Bangalore
71	sinchana	20	female	sinchana@gmail.com	9657453219	Davanagere
72	sourav	21	male	sourav@gmail.com	6743265788	Davanagere
73	surabi	20	female	surabi@gmail.com	9567438265	Chitradurga

Output :

```
Original dataset:
   usn      name  age  gender             email-id      mobile      address
0   66    anusha   20  female  anusha123@gmail.com  9854367524   Davanagere
1   67     akbar   21    male      akbar@gmail.com  9658765356  Chitradurga
2   68    anusha   20  female  anusha123@gmail.com  9854367524   Davanagere
3   69    anusha   20  female  anusha123@gmail.com  9854367524   Davanagere
4   70     kiran   21    male      kiran@yahoo.com  9342567845    Bangalore
5   71  sinchana   20  female   sinchana@gmail.com  9657453219   Davanagere
6   72    sourav   21    male     sourav@gmail.com  6743265788   Davanagere
7   73    surabi   20  female     surabi@gmail.com  9567438265  Chitradurga

Pre-processed dataset:
   usn      name   age  gender             email-id        mobile      address
0   66    anusha  20.0  female  anusha123@gmail.com  9.854368e+09   Davanagere
1   67     akbar  21.0    male      akbar@gmail.com  9.658765e+09  Chitradurga
2   68       NaN   NaN     NaN                  NaN           NaN          NaN
3   69       NaN   NaN     NaN                  NaN           NaN          NaN
4   70     kiran   NaN     NaN      kiran@yahoo.com  9.342568e+09    Bangalore
5   71  sinchana   NaN     NaN   sinchana@gmail.com  9.657453e+09          NaN
6   72    sourav   NaN     NaN     sourav@gmail.com  6.743266e+09          NaN
7   73    surabi   NaN     NaN     surabi@gmail.com  9.567438e+09          NaN
```

4. Demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

```
import math
import pandas as pd
from operator import itemgetter

class DecisionTree:
        def __init__(self, df, target, positive, parent_val, parent):
                self.data = df
                self.target = target
                self.positive = positive
                self.parent_val = parent_val
                self.parent = parent
                self.childs = []
                self.decision = '';

def _get_entropy(self, data):
        p = sum(data[self.target] == self.positive)
        n = data.shape[0] - p
        p_ratio = p / (p + n)
        n_ratio = 1 - p_ratio
        entropy_p = -p_ratio * math.log2(p_ratio) if p_ratio != 0 else
        0 entropy_n = -n_ratio * math.log2(n_ratio) if n_ratio != 0
        else 0 return entropy_p + entropy_n

def _get_gain(self, feat):
        avg_info = 0
        for val in self.data[feat].unique():
                subset = self.data[self.data[feat] == val]
                avg_info += self._get_entropy(subset) * len(subset) /
        self.data.shape[0] return self._get_entropy(self.data) - avg_info

def _get_splitter(self):
self.splitter = max(self.gains, key=itemgetter(1))[0]

def update_nodes(self):
        self.features = [col for col in self.data.columns if col !=
        self.target] self.entropy = self._get_entropy(self.data) if
        self.entropy != 0:
                self.gains = [(feat, self._get_gain(feat)) for feat in
                self.features] self._get_splitter()
                residual_columns = [k for k in self.data.columns if k != self.splitter]
```

```
                for val in self.data[self.splitter].unique():
                        df_tmp = self.data[self.data[self.splitter] == val][residual_columns]
                        tmp_node = DecisionTree(df_tmp, self.target, self.positive, val,
self.splitter)
        self.childs.append(tmp_node)

def print_tree(n):
        print(n.parent)
        print(n.parent_val, '\n';)
        for child in n.childs:
                if child:
                        print_tree(child)

df = pd.read_csv('C:/ML/Prg-4') # Corrected the file path
dt = DecisionTree(df,'play','yes')
dt.update_nodes()
print_tree(dt)
```

Datasets :

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Output :

```
outlook
sunny

outlook
overcast

outlook
rainy
```

5. Demonstrate the working of the Random forest algorithm. Use an appropriate data set for building and apply this knowledge to classify a new sample.

```
import pandas as pd
import numpy as np
dataset=pd.read_csv('Boston1.csv')
dataset.head()

X=pd.DataFrame(dataset.iloc[:,:-1])
X
y=pd.DataFrame(dataset.iloc[:,-1])
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.ensemble import RandomForestRegressor
regressor= RandomForestRegressor(n_estimators=20,random_state=0)
regressor.fit(X_train,y_train)
y_pred=regressor.predict(X_test)

from sklearn import metrics
print('mean absolut 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)))
```

Dataset: Boston House Prices Dataset

Let us have a quick look at the dataset:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
2	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
3	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
4	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
5	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
6	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
7	0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
8	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
9	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
10	0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5
11	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
12	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15
13	0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9
14	0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7
15	0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26	20.4
16	0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26	18.2
17	0.62739	0	8.14	0	0.538	5.834	56.5	4.4986	4	307	21	395.62	8.47	19.9
18	1.05393	0	8.14	0	0.538	5.935	29.3	4.4986	4	307	21	386.85	6.58	23.1

Output:

Accuracy score: 0.8268156424581006

6. 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.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB from
sklearn import metrics

df = pd.read_csv("pima_indian.csv")
feature_col_names = ['num_preg', 'glucose_conc', 'diastolic_bp', 'thickness', 'insulin', 'bmi',
'diab_pred', 'age']
predicted_class_names = ['diabetes']

X = df[feature_col_names].values # these are factors for the prediction
y = df[predicted_class_names].values # this is what we want to predict

#splitting the dataset into train and test data

xtrain,xtest,ytrain,ytest=train_test_split(X,y,test_size=0.33)

print ('\n the total number of Training Data :',ytrain.shape)
print ('\n the total number of Test Data :',ytest.shape)

# Training Naive Bayes (NB) classifier on training data.

clf = GaussianNB().fit(xtrain,ytrain.ravel())
predicted = clf.predict(xtest)
predictTestData= clf.predict([[6,148,72,35,0,33.6,0.627,50]])

#printing Confusion matrix, accuracy, Precision and Recall

print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('\n Accuracy of the classifier is',metrics.accuracy_score(ytest,predicted))
print('\n The value of Precision', metrics.precision_score(ytest,predicted))
print('\n The value of Recall', metrics.recall_score(ytest,predicted))
print("Predicted Value for individual Test Data:", predictTestData)
```

Datasets :

num_preg	glucose_c	diastolic_bp	thickness	insulin	bmi	diab_pred	age	diabetes
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1

Output:

Split 768 rows into train=514 and test=254 rows

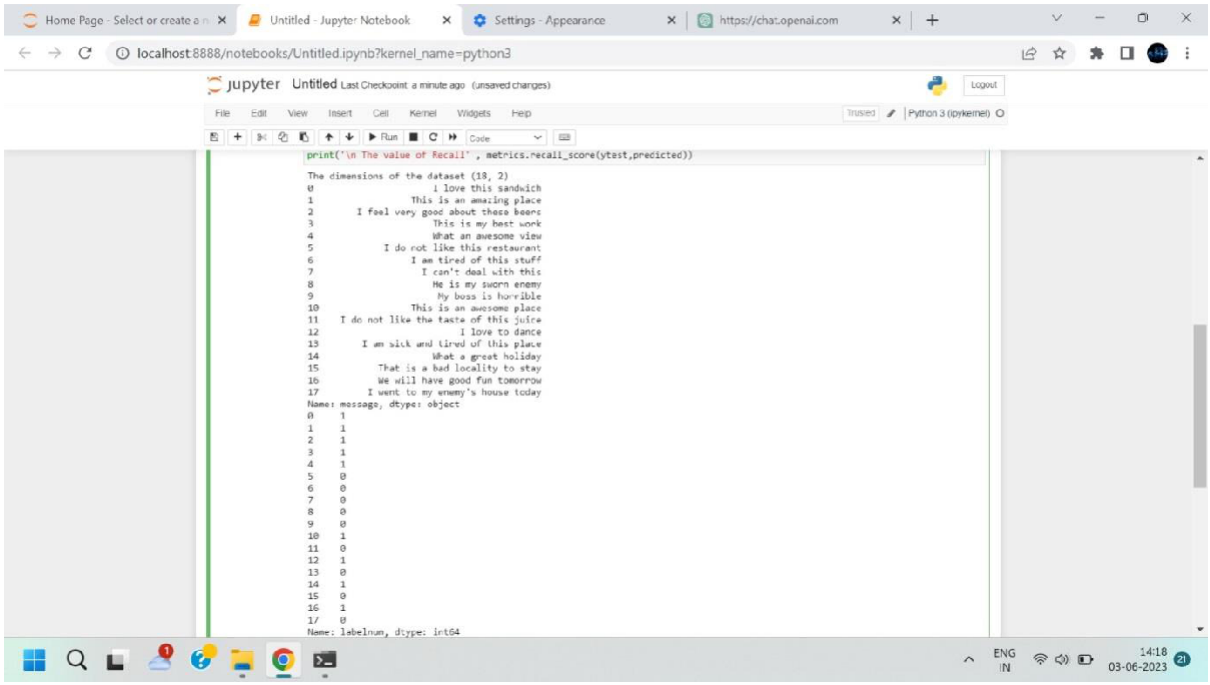
Accuracy of the classifier is : 71.65354330708661%

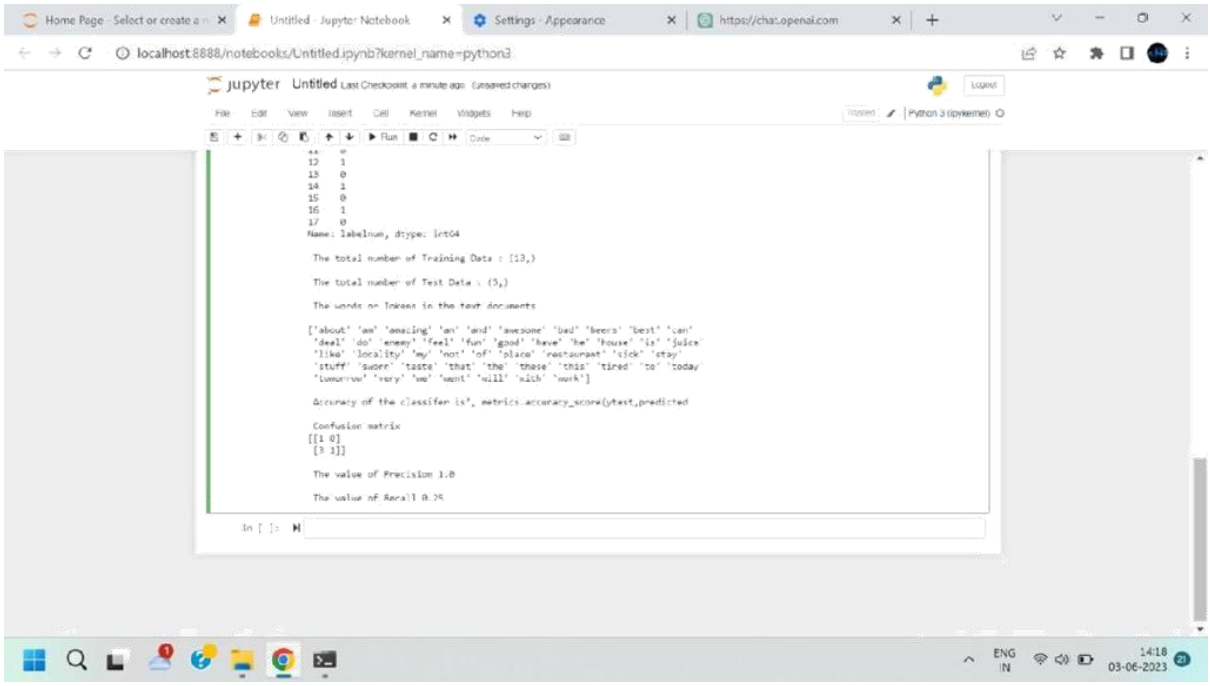
7. Assuming a set of documents that need to be classified, use the naive Bayesian Classifier model to perform this task. Calculate the accuracy, precision, and recall for your data set.

```
import pandas as pd
msg=pd.read_csv('naivetext.csv',names=['message','label'])
print('The dimensions of the dataset',msg.shape)
msg['labelnum']=msg.label.map({'pos':1,'neg':0})
X=msg.message
y=msg.labelnum
print(X)
print(y)
#splitting the dataset into train and test data
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(X,y)
print ('\n The total number of Training Data :',ytrain.shape)
print ('\n The total number of Test Data :',ytest.shape)

#output of count vectoriser is a sparse matrix
from sklearn.feature_extraction.text import
CountVectorizer count_vect = CountVectorizer()
xtrain_dtm = count_vect.fit_transform(xtrain)
xtest_dtm=count_vect.transform(xtest)
print('\n The words or Tokens in the text documents \n')
print(count_vect.get_feature_names())
df=pd.DataFrame(xtrain_dtm.toarray(),columns=count_vect.get_feature_names())

# Training Naive Bayes (NB) classifier on training
data. from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)

#printing accuracy, Confusion matrix, Precision and
Recall from sklearn import metrics
print('\n Accuracy of the classifer is', metrics.accuracy_score(ytest,predicted')
print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))
print('\n The value of Precision' , metrics.precision_score(ytest,predicted))
print('\n The value of Recall' , metrics.recall_score(ytest,predicted))
```

Datasets :

I love this sandwich	pos
This is an amazing place	pos
I feel very good about these beers	pos
This is my best work	pos
What an awesome view	pos
I do not like this restaurant	neg
I am tired of this stuff	neg
I can't deal with this	neg
He is my sworn enemy	neg
My boss is horrible	neg
This is an awesome place	pos
I do not like the taste of this juice	neg
I love to dance	pos
I am sick and tired of this place	neg
What a great holiday	pos
That is a bad locality to stay	neg
We will have good fun tomorrow	pos
I went to my enemy's house today	neg




Home Page - Select or create a n
Untitled - Jupyter Notebook
Settings - Appearance
https://chat.openai.com
localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3
jupyter Untitled Last Checkpoint: a minute ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
Logout
Python 3 (ipykernel)
12 1
13 0
14 1
15 0
16 1
17 0
Name: labelnum, dtype: int64
The total number of Training Data : (13,)
The total number of Test Data : (5,)
The words or Tokens in the text documents
['about' 'am' 'amazing' 'an' 'and' 'awesome' 'bad' 'beers' 'best' 'can'
'deal' 'do' 'enemy' 'feel' 'fun' 'good' 'have' 'he' 'house' 'is' 'juice'
'like' 'locality' 'my' 'not' 'of' 'place' 'restaurant' 'sick' 'stay'
'stuff' 'sucks' 'taste' 'that' 'the' 'these' 'this' 'tired' 'to' 'today'
'tomorrow' 'very' 'we' 'went' 'will' 'with' 'work']
Accuracy of the classifier is', metrics.accuracy_score(ytest,predicted
Confusion matrix
[[1 0]
[3 1]]
The value of Precision 1.0
The value of Recall 0.25
In [ ]:
ENG
IN
14:18
03-06-2023

8. Construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.

```
import numpy as np
import pandas as pd
import csv
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination

#read Cleveland Heart Disease data
heartDisease = pd.read_csv('heart.csv')
heartDisease = heartDisease.replace('?',np.nan)

#display the data
print('Sample instances from the dataset are given below')
print(heartDisease.head())

#display the Attributes names and datatyes
print('\n Attributes and datatypes')
print(heartDisease.dtypes)

#Creat Model- Bayesian Network
model=BayesianModel([('age','heartdisease'),('sex','heartdisease'),(
'exang','heartdisease'),('cp','heartdisease'),('heartdisease', 'restecg'),('heartdisease','chol')])

#Learning CPDs using Maximum Likelihood Estimators print('\n
Learning CPD using Maximum likelihood estimators')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

# Inferencing with Bayesian Network print('\n
Inferencing with Bayesian Network:')
HeartDiseasetest_infer = VariableElimination(model)

#computing the Probability of HeartDisease given restecg
print('\n 1.Probability of HeartDisease given evidence= restecg :1')
q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restecg':1})
print(q1)

#computing the Probability of HeartDisease given cp
print('\n 2.Probability of HeartDisease given evidence= cp:2 ')
q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2})
print(q2)
```

Datasets :

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	heartdisease
63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
56	1	3	130	256	1	2	142	1	0.6	2	1	6	2
44	1	2	120	263	0	0	173	0	0	1	0	7	0
52	1	3	172	199	1	0	162	0	0.5	1	0	7	0
57	1	3	150	168	0	0	174	0	1.6	1	0	3	0
48	1	2	110	229	0	0	168	0	1	3	0	7	1
54	1	4	140	239	0	0	160	0	1.2	1	0	3	0
48	0	3	130	275	0	0	139	0	0.2	1	0	3	0
49	1	2	130	266	0	0	171	0	0.6	1	0	3	0
64	1	1	110	211	0	2	144	1	1.8	2	0	3	0
58	0	1	150	283	1	2	162	0	1	1	0	3	0
58	1	2	120	284	0	2	160	0	1.8	2	0	3	1
58	1	3	132	224	0	2	173	0	3.2	1	2	7	3

Output :

```
Sample instances from the dataset are given below
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   63    1   1       145   233    1        2      150      0      2.3      3
1   67    1   4       160   286    0        2      108      1      1.5      2
2   67    1   4       120   229    0        2      129      1      2.6      2
3   37    1   3       130   250    0        0      187      0      3.5      3
4   41    0   2       130   204    0        2      172      0      1.4      1

  ca thal  heartdisease
0  0    6             0
1  3    3             2
2  2    7             1
3  0    3             0
4  0    3             0

 Attributes and datatypes
age               int64
sex               int64
cp                int64
trestbps          int64
chol              int64
```

```
 Attributes and datatypes
age               int64
sex               int64
cp                int64
trestbps          int64
chol              int64
fbs               int64
restecg           int64
thalach           int64
exang             int64
oldpeak         float64
slope             int64
ca               object
thal             object
heartdisease      int64
dtype: object
```

```
 Inferencing with Bayesian Network:

 1.Probability of HeartDisease given evidence= restecg :1
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.1012 |
+-----------------+---------------------+
| heartdisease(1) |              0.0000 |
+-----------------+---------------------+
| heartdisease(2) |              0.2392 |
+-----------------+---------------------+
| heartdisease(3) |              0.2015 |
+-----------------+---------------------+
| heartdisease(4) |              0.4581 |
+-----------------+---------------------+

 2.Probability of HeartDisease given evidence= cp:2
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.3610 |
+-----------------+---------------------+
| heartdisease(1) |              0.2159 |
+-----------------+---------------------+
| heartdisease(2) |              0.1373 |
+-----------------+---------------------+
| heartdisease(3) |              0.1537 |
+-----------------+---------------------+
| heartdisease(4) |              0.1321 |
+-----------------+---------------------+
```

9. Demonstrate the working of EM algorithm to cluster a set of data stored in a .CSV file.

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import pandas as pd
import numpy as np

iris = datasets.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']

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


plt.figure(figsize=(14,7))

colormap = np.array(['red', 'lime', 'black'])

# Plot the Original Classifications
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets],
s=40) plt.title('Real Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')


# Plot the Models Classifications
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K Mean Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
print('The accuracy score of K-Mean: ',sm.accuracy_score(y, model.labels_))
print('The Confusion matrixof K-Mean: ',sm.confusion_matrix(y, model.labels_))
```

```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
xs = pd.DataFrame(xsa, columns = X.columns)
#xs.sample(5)

from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(xs)

y_gmm = gmm.predict(xs)
#y_cluster_gmm

plt.subplot(2, 2, 3)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_gmm], s=40)
plt.title('GMM Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')

print('The accuracy score of EM: ',sm.accuracy_score(y, y_gmm))
print('The Confusion matrix of EM: ',sm.confusion_matrix(y, y_gmm))
```

Datasets :

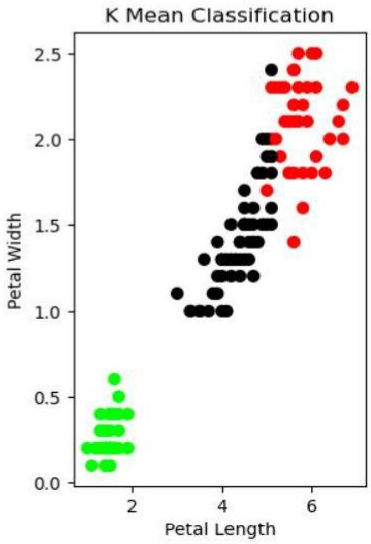
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.8	1.4	0.1	Iris-setosa
5.8	3.5	1.2	0.3	Iris-setosa
5.6	3.6	1.1	0.2	Iris-setosa
5.1	3.5	1.4	0.2	Iris-setosa
5.7	3.2	1.6	0.7	Iris-setosa
5.8	3.9	1.7	0.5	Iris-setosa
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.8	1.4	0.1	Iris-setosa
5.8	3.5	1.2	0.3	Iris-setosa
5.6	3.6	1.1	0.2	Iris-setosa
5.1	3.5	1.4	0.2	Iris-setosa
5.7	3.2	1.6	0.7	Iris-setosa
5.8	3.9	1.7	0.5	Iris-setosa
5.1	3.5	1.4	0.2	Iris-setosa
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.8	1.4	0.1	Iris-setosa
5.8	3.5	1.2	0.3	Iris-setosa
5.6	3.6	1.1	0.2	Iris-setosa
5.1	3.5	1.4	0.2	Iris-setosa
5.7	3.2	1.6	0.7	Iris-setosa
5.8	3.9	1.7	0.5	Iris-setosa
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.8	1.4	0.1	Iris-setosa
5.8	3.5	1.2	0.3	Iris-setosa

Output:

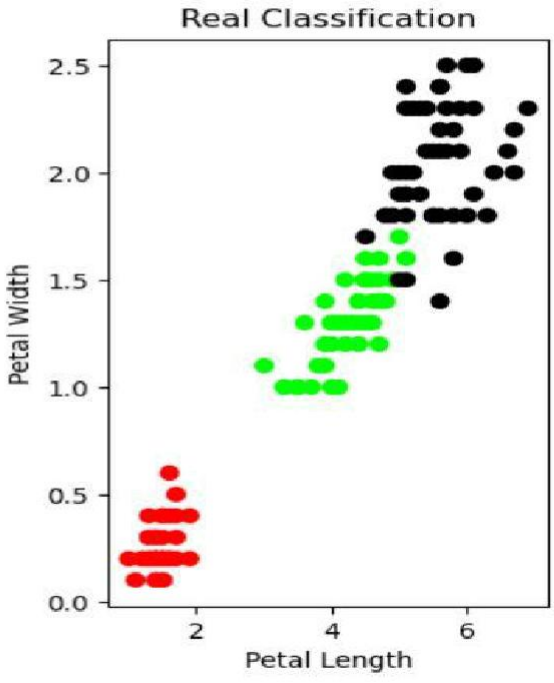
click to scroll output; double click to hide

```
In [20]: print('The accuracy score of EM: ',sm.accuracy_score(y, y_gmm))
         print('The Confusion matrix of EM: ',sm.confusion_matrix(y, y_gmm))

         The accuracy score of EM:  0.0
         The Confusion matrix of EM:  [[ 0 50  0]
          [ 5  0 45]
          [50  0  0]]

In [ ]:
```

```
The accuracy score of K-Mean:  0.09333333333333334
The Confusion matrixof K-Mean:  [[ 0 50  0]
 [ 2  0 48]
 [36  0 14]]
```




```
Out[14]: Text(0, 0.5, 'Petal Width')
```



```
P_NUM_THREADS=1.
  warnings.warn(
```

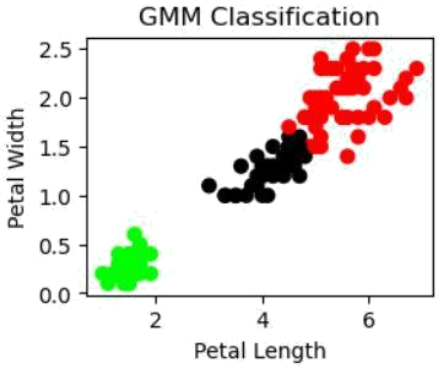
Out[17]:

```
GaussianMixture(n_components=3)
```

In [18]:
```
y_gmm = gmm.predict(xs)
#y_cluster_gmm
```

In [19]:
```
plt.subplot(2, 2, 3)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_gmm], s=40)
plt.title('GMM Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```

Out[19]:
```
Text(0, 0.5, 'Petal Width')
```



10. Demonstrate the working of SVM classifier for a suitable data set

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Step 1: Dataset Preparation
data = {
   'Weight': [150, 200, 250, 180, 300, 220],
   'Color': ['Red', 'Red', 'Orange', 'Orange', 'Red', 'Orange'],
   'Label': ['Apple', 'Apple', 'Orange', 'Orange', 'Apple', 'Orange']
}

df = pd.DataFrame(data)

# Step 2: Feature Extraction
df['Color'] = df['Color'].map({'Red': 0, 'Orange': 1})

# Step 3: Data Split
X = df[['Weight', 'Color']]
y = df['Label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Step 4: Model Training
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)

 # Step 5: Model Evaluation
y_pred = svm.predict(X_test)
accuracy = accuracy_score(y_test,
y_pred) print("Accuracy:", accuracy)

# Step 6: Prediction
new_data = {
   'Weight': [190],
   'Color': [0]
}

new_df = pd.DataFrame(new_data)
new_prediction = svm.predict(new_df)
print("New prediction:", new_prediction)
```

Output :

```
# Step 4: Model Training
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)


# Step 5: Model Evaluation
y_pred = svm.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)


# Step 6: Prediction
new_data = {
    'Weight': [190],
    'Color': [0]
}


new_df = pd.DataFrame(new_data)
new_prediction = svm.predict(new_df)
print("New prediction:", new_prediction)
```

```
Accuracy: 0.0
New prediction: ['Orange']
```