7)Demonstrate the text classifier using Naive Bayes classifier algorithm.

```python
#NAIVE BAYES CLASSIFIER
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_20newsgroups_vectorized
from sklearn.metrics import accuracy_score,classification_report

data=fetch_20newsgroups_vectorized()

x=data.data
y=data.target

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
ate=42)

clf=MultinomialNB()
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)


print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
0.7445868316394167
              precision    recall  f1-score   support

           0       0.88      0.38      0.53        93
           1       0.82      0.62      0.71       118
           2       0.89      0.66      0.76       128
           3       0.62      0.77      0.69       120
           4       0.72      0.82      0.77       102
           5       0.88      0.73      0.80       124
           6       0.88      0.66      0.76       112
           7       0.65      0.95      0.77       112
           8       0.91      0.88      0.90       118
           9       0.97      0.93      0.95       125
          10       0.95      0.94      0.94       117
          11       0.52      0.97      0.68       120
          12       0.92      0.50      0.65       138
          13       0.87      0.90      0.88       118
          14       0.90      0.87      0.88       122
          15       0.38      0.98      0.55       120
          16       0.81      0.84      0.83       105
          17       0.95      0.85      0.90       115
          18       1.00      0.16      0.27        90
          19       1.00      0.02      0.03        66

    accuracy                           0.74      2263
   macro avg       0.83      0.72      0.71      2263
weighted avg       0.82      0.74      0.73      2263
```

6)Implement Random Forest classifier using python programming.

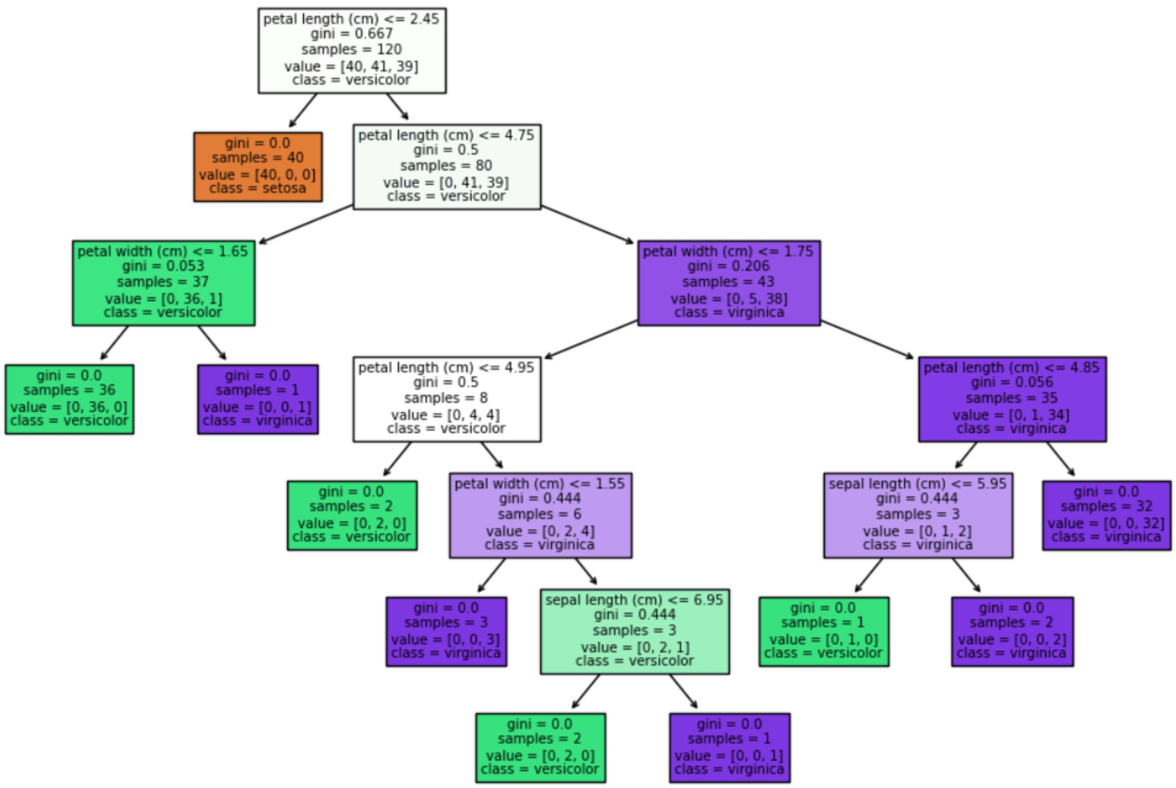
```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report


data=load_breast_cancer()

x=data.data
y=data.target

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
ate=42)


clf=RandomForestClassifier(n_estimators=500,max_leaf_nodes=16,n_jobs=-1)

clf.fit(x_train,y_train)

y_pred=clf.predict(x_test)


print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
0.9649122807017544
              precision    recall  f1-score   support

           0       0.98      0.93      0.95        43
           1       0.96      0.99      0.97        71

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

5. Implement and demonstrate the working of the Decision Tree algorithm

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report
import matplotlib.pyplot as plt
from sklearn import tree

data=load_iris()

x=data.data
y=data.target


x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
ate=42)

clf=DecisionTreeClassifier()

clf.fit(x_train,y_train)

y_pred=clf.predict(x_test)

print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))

plt.figure(figsize=(12,8))
tree.plot_tree(clf,feature_names=data.feature_names,class_names=data.targe
t_names,filled=True)
plt.show()
```

```
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```



4. Demonstrate the working of SVM classifier for a suitable dataset

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report

data=load_breast_cancer()
x=data.data
y=data.target

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
ate=42)

svm=SVC(kernel='linear')
svm.fit(x_train,y_train)

y_pred=svm.predict(x_test)

print("PREDICTED:", y_pred)
print("CONFUSION MATRIX: \n",confusion_matrix(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
PREDICTED: [1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
 1 0 0]
CONFUSION MATRIX:
 [[39  4]
 [ 1 70]]
ACCURACY: 0.956140350877193
              precision    recall  f1-score   support

           0       0.97      0.91      0.94        43
           1       0.95      0.99      0.97        71

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

3. Demonstrate data Preprocessing (Data Cleaning, Integration and Transformation) operations on a suitable data.

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import LabelEncoder

data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

print("Dataset:")
print(df.head())

X = df.drop(columns=['target'])
y = df['target']

label_encoder = LabelEncoder()
y_label_encoded = label_encoder.fit_transform(y)
print("Target\n", y_label_encoded)
```

```
Dataset:
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840   
1        20.57         17.77          132.90     1326.0          0.08474   
2        19.69         21.25          130.00     1203.0          0.10960   
3        11.42         20.38           77.58      386.1          0.14250   
4        20.29         14.34          135.10     1297.0          0.10030   

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419   
1           0.07864          0.0869              0.07017         0.1812   
2           0.15990          0.1974              0.12790         0.2069   
3           0.28390          0.2414              0.10520         0.2597   
4           0.13280          0.1980              0.10430         0.1809   

   mean fractal dimension  ...  worst texture  worst perimeter  worst area  \
0                 0.07871  ...          17.33           184.60      2019.0   
1                 0.05667  ...          23.41           158.80      1956.0   
2                 0.05999  ...          25.53           152.50      1709.0   
3                 0.09744  ...          26.50            98.87       567.7   
4                 0.05883  ...          16.67           152.20      1575.0   

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654   
1            0.1238             0.1866           0.2416                0.1860   
2            0.1444             0.4245           0.4504                0.2430   
3            0.2098             0.8663           0.6869                0.2575   
4            0.1374             0.2050           0.4000                0.1625   

   worst symmetry  worst fractal dimension  target  
0          0.4601                  0.11890       0  
1          0.2750                  0.08902       0  
2          0.3613                  0.08758       0  
3          0.6638                  0.17300       0  
4          0.2364                  0.07678       0  
```


```
   worst symmetry  worst fractal dimension  target  
0          0.4601                  0.11890       0  
1          0.2750                  0.08902       0  
2          0.3613                  0.08758       0  
3          0.6638                  0.17300       0  
4          0.2364                  0.07678       0  
```


```
[5 rows x 31 columns]
Target
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

8.Implement the Naive Bayesian classifier for a sample training data set stored as a .CSV file.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
# Load the Dataset
url = "iris.csv"
column_names = ['sepal_length', 'sepal_width', 'petal_length',
'petal_width',
'class']
df = pd.read_csv(url, header=None, names=column_names)
#print(df.head())
# Split the Data
X = df.drop('class', axis=1)
y = df['class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Train the Model
model = GaussianNB()
model.fit(X_train, y_train)
# Evaluate the Model
y_pred = model.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))


Output:
Accuracy: 1.0
Classification Report:
                 precision    recall  f1-score   support
    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11
```

```
     accuracy                           1.00        30
    macro avg       1.00      1.00      1.00        30
 weighted avg       1.00      1.00      1.00        30
```

10..Implement KNN classification algorithm with an appropriate dataset and analyze the results.

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

data = load_breast_cancer()
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

Output:
Accuracy: 0.9590643274853801
```

```
Confusion Matrix:
 [[ 57   6]
 [  1 107]]
Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.90      0.94        63
           1       0.95      0.99      0.97       108

    accuracy                           0.96       171
   macro avg       0.96      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```

1.Implement and demonstrate the Find-S algorithm for finding the most specific hypothesis.

```
import csv
with open('enjoysport.csv', 'r') as csvfile:
    data = list(csv.reader(csvfile))
print(data)
print("\nThe total number of training instances are:", len(data))
num_attributes = len(data[0]) - 1
hypothesis = ['0'] * num_attributes
print("\nThe initial hypothesis is:", hypothesis)
for i, instance in enumerate(data):
    if instance[num_attributes] == 'yes':
        for j in range(num_attributes):
            if hypothesis[j] == '0' or hypothesis[j] == instance[j]:
                hypothesis[j] = instance[j]
            else:
                hypothesis[j] = '?'
    print(f"\nThe hypothesis after training instance {i + 1} is:",
hypothesis)
print("\n\nThe Maximally specific hypothesis is:", hypothesis)
```

```
Output:
[['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']]
The total number of training instances are: 5
The initial hypothesis is: ['0', '0', '0', '0', '0', '0']
The hypothesis after training instance 1 is: ['0', '0', '0', '0', '0',
'0']
The hypothesis after training instance 2 is: ['sunny', 'warm', 'normal',
'strong', 'warm', 'same']
The hypothesis after training instance 3 is: ['sunny', 'warm', '?',
'strong', 'warm', 'same']
The hypothesis after training instance 4 is: ['sunny', 'warm', '?',
'strong', 'warm', 'same']
The hypothesis after training instance 5 is: ['sunny', 'warm', '?',
'strong', '?', '?']
The Maximally specific hypothesis is: ['sunny', 'warm', '?', 'strong',
'?', '?']
```

2.Implement and demonstrate the Candidate Elimination algorithm using a data set stored as a .CSV file.

```
import csv
with open("finds.csv") as f:
    csv_file = csv.reader(f)
    data = list(csv_file)
specific = data[1][:-1]
general = [['?' for _ in range(len(specific))] for _ in
range(len(specific))]
for i in data[1:]:
    if i[-1] == "Yes":
        for j in range(len(specific)):
            if i[j] != specific[j]:
                specific[j] = '?'
                general[j][j] = '?'
    elif i[-1] == "No":
```

```
        for j in range(len(specific)):
            if i[j] != specific[j]:
                general[j][j] = '?'
            else:
                general[j][j] = specific[j]
gh = [g for g in general if g != ['?' for _ in range(len(specific))]]
print("\nFinal Specific Hypothesis:\n", specific)
print("\nFinal General Hypothesis:\n", gh)

Output:
Final Specific Hypothesis:
 ['Sunny', 'Warm', 'High', 'Strong', '?', '?']

Final General Hypothesis:
 [['?', '?', 'High', '?', '?', '?'], ['?', '?', '?', 'Strong', '?', '?']]
```

9.Construct a Bayesian network to analyze the diagnosis of heart patients using heart diseases dataset.

```
import pandas as pd
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination

data = pd.read_csv('heart.csv')[['age', 'sex', 'cp', 'thalach', 'exang',
'oldpeak', 'target']]
print(data.head())

model = BayesianNetwork([('age', 'target'), ('sex', 'target'), ('cp',
'target'),('thalach', 'target'), ('exang', 'target'), ('oldpeak',
'target')])
model.fit(data, estimator=MaximumLikelihoodEstimator)

inference = VariableElimination(model)
evidence = {'age': 63, 'sex': 1, 'cp': 1, 'thalach': 150, 'exang': 0,
'oldpeak': 2.3}
result = inference.query(variables=['target'], evidence=evidence)
print(result)

Output:
```

```
Found Intel OpenMP ('libiomp') and LLVM OpenMP ('libomp') loaded at
the same time. Both libraries are known to be incompatible and this
can cause random crashes or deadlocks on Linux when loaded in the
same Python program.
Using threadpoolctl may cause crashes or deadlocks. For more
information and possible workarounds, please see
    https://github.com/joblib/threadpoolctl/blob/master/multiple_openmp.md

  warnings.warn(msg, RuntimeWarning)
   age  sex  cp  thalach  exang  oldpeak  target
0   52    1   0      168      0      1.0       0
1   53    1   0      155      1      3.1       0
2   70    1   0      125      1      2.6       0
3   61    1   0      161      0      0.0       0
4   62    0   0      106      0      1.9       0
+-----------+---------------+
| target    |   phi(target) |
+===========+===============+
| target(0) |        0.5000 |
+-----------+---------------+
| target(1) |        0.5000 |
+-----------+---------------+
```