PROGRAM 1:

```
import csv
def loaddata(filename):
   with open(filename,'r')as f:
      reader=csv.reader(f)
      data=list(reader)
      header=data[0]
      instances=data[1:]
      return header,instances
def finds(data):
   for instances in data:
      if instances[-1].lower()=='yes':
         hypothesis=instances[:-1]
         break
   else:
      None
 for instance in data:
      if instance[-1].lower()=='yes':
         for i in range(len(hypothesis)):
            if hypothesis[i]!=instance[i]:
               hypothesis[i]='?'
   return hypothesis
filename='trainingdata.csv'
header,data=loaddata(filename)
print('attributes',header)
print('')
print('trainingdata')
for row in data:
     print(row)
hypothesis=finds(data)
if hypothesis:
```

```
    print('most specific hypothesis by FIND-S')
    print(hypothesis)
else:
    print ('no positive training examples in training data')
```

PROGRAM 2:

```
import pandas as pd
import numpy as np
def loaddata(filename):
   data=pd.read_csv(filename)
   print(data)
   concepts=data.iloc[:,:-1].values
   target=data.iloc[:,-1].values
   return concepts,target
def candidate(concepts,target):
   nf=len(concepts[0])
   s=concepts[0].copy()
   g=[["?" for _ in range(nf)] for _ in range(nf)]
   for i,example in enumerate(concepts):
      if target[i].lower()=='yes':
         for x in range(nf):
            if s[x]!=example[x]:
               s[x]="?"
               g[x][x]="?"
      else:
         for x in range(nf):
            if s[x]!=example[x]:
               g[x][x]=s[x]
            else:
               g[x][x]="?"
   g=[h for h in g if any (attr!="?" for attr in h)]
```

```
    return s,g
filename='traindata1.csv'
concepts,target=loaddata(filename)
s,g=candidate(concepts,target)
print('most specific hypothesis using candidate')
print(s)
print('most general hypothesis using candidate')
print(g)
```

PROGRAM 3:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
iris=load_iris()
x=iris.data
y=iris.target
print(iris)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
accuracy=accuracy_score(y_test,y_pred)
print(f"Accuracy:{accuracy:.2f}")
new_sample=[[5.1,2.5,4.6,1.5]]
predicted_class_index=clf.predict(new_sample)[0]
predicted_class_name=iris.target_names[predicted_class_index]
print(f"predicted class for the new sample{new_sample} is:{predicted_class_name}")
```

PROGRAM 5:

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score,precision_score,recall_score
data=pd.read_csv('data.csv')
print(data)
x=data.iloc[:,:-1]
y=data.iloc[:,-1]
x=x.copy()
le_outlook=LabelEncoder()
x.Outlook=le_outlook.fit_transform(x.Outlook)
le_Temperature=LabelEncoder()
x.Temperature=le_Temperature.fit_transform(x.Temperature)
print("\n now the train outputis\n",x)
le_playTennis=LabelEncoder()
y=le_playTennis.fit_transform(y)
print("\n now the  train output is\n",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)
classifier=GaussianNB()
classifier.fit(x_train,y_train)
print("Acuuracy is:",accuracy_score(classifier.predict(x_test),y_test))
print('recall:',recall_score(classifier.predict(x_test),y_test))
print('precision:',precision_score(classifier.predict(x_test),y_test))
```

PROGRAM 6:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
```

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,precision_score,recall_score
df=pd.read_csv('document.csv')
text = df['text'].values
labels=df['label'].values
vectorizer=CountVectorizer()
x=vectorizer.fit_transform(text)
x_train,x_text,y_train,y_test=train_test_split(x,labels,test_size=0.3,random_state=42)
model=MultinomialNB()
model.fit(x_train,y_train)
y_pred=model.predict(x_text)
accuracy=accuracy_score(y_test,y_pred)
precision=precision_score(y_test,y_pred,pos_label='positive')
recall=recall_score(y_test,y_pred,pos_label='positive')
print("test results:")
for text,true_label,pred_label in zip(vectorizer.inverse_transform(x_text),y_test,y_pred):
    print(f"Text:{' '.join(text)} | True:{true_label} | predicted:{pred_label}")
print("\n metrices:")
print(f"Accuracy:{accuracy:.2f}")
print(f"Precision:{precision:.2f}")
print(f"Recall:{recall:.2f}")
```

PROGRAM 9:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import   KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn import datasets
```

```
iris=datasets.load_iris()
print("Iris Data set loaded..")
x_train,x_test,y_train,y_test=train_test_split(iris.data,iris.target,random_state=0)
for i in range(len(iris.target_names)):
    print("Label",i,"-",str(iris.target_names[i]))
classifier=KNeighborsClassifier(n_neighbors=2)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
print("Results of classification using K-nn with k=2")
for r in range(0,len(x_test)):
    print("sample:",str(x_test[r])," Actual-label:",str(y_test[r])," predicted-label:",str(y_pred[r]))
print("\n classification accuracy:",classifier.score(x_test,y_test));
print("\n confusion matrix:\n", metrics.confusion_matrix(y_test,y_pred))
```