1.

```
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] == '1':
        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)
```

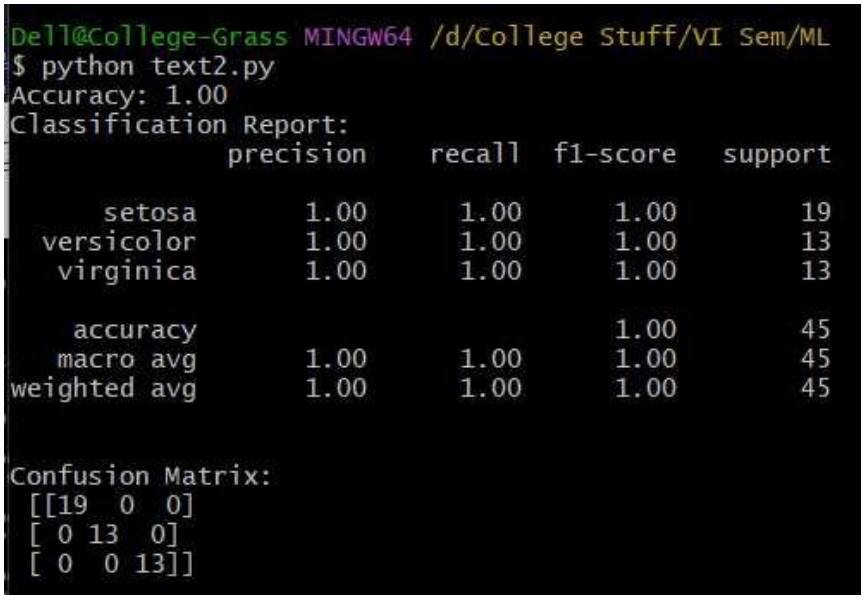
OUTPUT

```
Dell@College-Grass MINGW64 /d/College Stuff/VI Sem/ML
$ python text.py
[['sky', 'airtemp', 'humidity', 'wind', 'water', 'forcast', 'enjoysport'], ['sunny', 'warm', 'normal', 'strong',
 'high', 'strong', 'warm', 'change', '0'], ['sunny', 'warm', 'high', 'strong', 'cool', 'change', '1']]

 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', '?', '?']
```

4.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
# Train the SVM classifier
svm = SVC(kernel='linear', C=1.0, random_state=42)
svm.fit(X_train, y_train)
# Predict the test set results
y_pred =  svm.predict(X_test)
# Evaluate the classifier
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

OUTPUT



5.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report

# Load and split the Iris dataset
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.35, random_state=42)

# Train and evaluate the Random Forest classifier
rf_classifier = DecisionTreeClassifier(
random_state=42).fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)

print(f'Accuracy: {accuracy_score(y_test, y_pred):.2f}')
print('Classification Report:\n', classification_report(y_test,
y_pred))
```

OUTPUT

```
 Accuracy: 0.98
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.94      1.00      0.97        17
           2       1.00      0.94      0.97        17

    accuracy                           0.98        53
   macro avg       0.98      0.98      0.98        53
weighted avg       0.98      0.98      0.98        53
```

6.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
# Load and split the Iris dataset
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35,
random_state=42)
# Train and evaluate the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=42).fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
print(f'Accuracy: {accuracy_score(y_test, y_pred):.2f}')
print('Classification Report:\n', classification_report(y_test, y_pred))
```

OUTPUT

```
Dell@College-Grass MINGW64 /d/College Stuff/VI Sem/ML
$ python text.py
Accuracy: 0.98
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.94      1.00      0.97        17
           2       1.00      0.94      0.97        17

    accuracy                           0.98        53
   macro avg       0.98      0.98      0.98        53
weighted avg       0.98      0.98      0.98        53
```

7.

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
# Load the 20 newsgroups dataset
newsgroups = fetch_20newsgroups(subset='all')
# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(newsgroups.data,
newsgroups.target, test_size=0.2, random_state=42)
# Transform the text data to feature vectors using TF-IDF
vectorizer = TfidfVectorizer(stop_words='english')
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
# Train the Naive Bayes classifier
clf = MultinomialNB()
clf.fit(X_train_tfidf, y_train)
# Predict the test set results
y_pred = clf.predict(X_test_tfidf)
# Evaluate the classifier
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred,
target_names=newsgroups.target_names))
```

OUTPUT

```
Dell@College-Grass MINGW64 /d/College Stuff/VI Sem/ML
$ python text.py
Accuracy: 0.88
Classification Report:
                          precision    recall  f1-score   support

             alt.atheism       0.85      0.86      0.86       151
           comp.graphics       0.88      0.84      0.86       202
 comp.os.ms-windows.misc       0.86      0.85      0.85       195
comp.sys.ibm.pc.hardware       0.65      0.85      0.73       183
   comp.sys.mac.hardware       0.94      0.87      0.90       205
          comp.windows.x       0.95      0.85      0.90       215
            misc.forsale       0.93      0.72      0.81       193
               rec.autos       0.91      0.94      0.92       196
         rec.motorcycles       0.89      0.95      0.92       168
      rec.sport.baseball       0.95      0.95      0.95       211
        rec.sport.hockey       0.90      0.99      0.94       198
               sci.crypt       0.91      0.97      0.94       201
         sci.electronics       0.92      0.82      0.86       202
                 sci.med       0.97      0.94      0.96       194
               sci.space       0.88      0.99      0.93       189
  soc.religion.christian       0.72      0.99      0.83       202
      talk.politics.guns       0.81      0.97      0.88       188
   talk.politics.mideast       0.94      0.99      0.97       182
      talk.politics.misc       0.96      0.75      0.84       159
      talk.religion.misc       1.00      0.31      0.47       136

                accuracy                           0.88      3770
               macro avg       0.89      0.87      0.87      3770
            weighted avg       0.89      0.88      0.87      3770
```

8.

```
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)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f"Accuracy: {accuracy}\n")
print("Classification Report:")
print(report)
```

OUTPUT

```
Dell@College-Grass MINGW64 /d/College Stuff/VI Sem/ML
$ python text.py
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
```