Q1 Write a Python program using Scikit-learn to split the iris dataset into 80% train data and 20% test data. Out of total 150 records, the training set will contain 120 records and the test set contains 30 of those records. Train or fit the data into the model and calculate the accuracy of the model using the Decision Tree Algorithm

In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

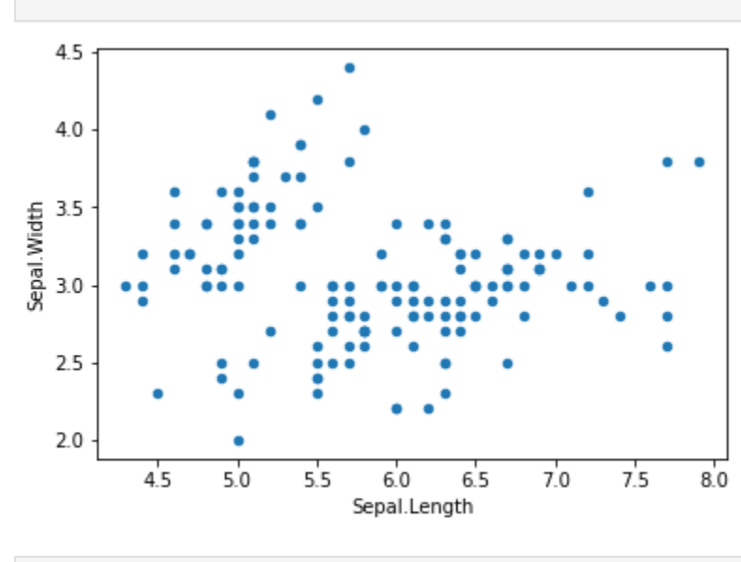
In [2]:
```python
iris =   pd.read_csv("E:\\Data Science\\Data Set\\Iris.csv") #Iris.csv is now a pandas dataframe
print(iris.head()) #prints first 5 values
```

```
   Sepal.Length  Sepal.Width  Petal.Length  Petal.Width Species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

In [3]:
```python
print(iris.describe())
```

```
       Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

In [6]:
```python
iris.plot(kind="scatter", x="Sepal.Length",   y="Sepal.Width")
plt.show()
```



In [9]:
```python
from sklearn.model_selection import train_test_split
```

In [10]:
```python
x = iris.iloc[:, :-1].values
```

In [11]:
```python
y = iris.iloc[:,   -1].values
```

In [12]:
```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

In [13]:
```python
from sklearn.tree import   DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

In [14]:
```python
classifier   = DecisionTreeClassifier()
```

In [15]:
```python
classifier.fit(x_train,   y_train)
```

Out[15]:
```
DecisionTreeClassifier()
```

In [16]:
```python
y_pred   = classifier.predict(x_test)
```

In [19]:
```python
from sklearn.metrics import classification_report
```

In [20]:
```python
print(classification_report(y_test,   y_pred))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00         6

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

In [21]:
```python
from sklearn.metrics import confusion_matrix
```

In [22]:
```python
print(confusion_matrix(y_test, y_pred))
```

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

In [23]:
```python
print('accuracy   is',accuracy_score(y_pred,y_test))
```

```
accuracy   is 1.0
```

Q2 For the dataset given below calculate Entropy an Information game, required to create a decision tree?

In [24]:
```python
# Load libraries
import numpy as np
import pandas as pd
from sklearn import metrics
```

In [25]:
```python
df=pd.read_csv("E:\\Data Science\\Data Set\\Play Tennis.csv")
value=['Outlook','Temprature','Humidity','Wind']
df
```

Out[25]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes
5	D6	Rain	Cool	Normal	Strong	No
6	D7	Overcast	Cool	Normal	Strong	Yes
7	D8	Sunny	Mild	High	Weak	No
8	D9	Sunny	Cool	Normal	Weak	Yes
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

In [26]:
```python
len(df)
```

Out[26]:
```
14
```

In [27]:
```python
df.shape
```

Out[27]:
```
(14, 6)
```

In [28]:
```python
df.head()
```

Out[28]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes

In [29]:
```python
df.describe()
```

Out[29]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
count	14	14	14	14	14	14
unique	14	3	3	2	2	2
top	D8	Rain	Mild	Normal	Weak	Yes
freq	1	5	6	7	8	9

In [30]:
```python
#machine learning algorithms can only learn from numbers (int, float, doubles .. )
#so let us encode it to int
from sklearn import preprocessing
string_to_int= preprocessing.LabelEncoder()                      #encode your data
df=df.apply(string_to_int.fit_transform) #fit and transform it
df
```

Out[30]:

	Day	Outlook	Temprature	Humidity	Wind	Play_Tennis
0	0	2	1	0	1	0
1	6	2	1	0	0	0
2	7	0	1	0	1	1
3	8	1	2	0	1	1
4	9	1	0	1	1	1
5	10	1	0	1	0	0
6	11	0	0	1	0	1
7	12	2	2	0	1	0
8	13	2	0	1	1	1
9	1	1	2	1	1	1
10	2	2	2	1	0	1
11	3	0	2	0	0	1
12	4	0	1	1	1	1
13	5	1	2	0	0	0

In [31]:
```python
#To divide our data into attribute set and Label:
feature_cols = ['Outlook','Temprature','Humidity','Wind']
X = df[feature_cols ]                              #contains the attribute
y = df.Play_Tennis
```

In [32]:
```python
#To divide our data into training and test sets:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

In [33]:
```python
# perform training
from sklearn.tree import DecisionTreeClassifier                              # import the classifier
classifier =DecisionTreeClassifier(criterion="entropy", random_state=100)     # create a classifier object
classifier.fit(X_train, y_train)
```

Out[33]:
```
DecisionTreeClassifier(criterion='entropy', random_state=100)
```

In [34]:
```python
#Predict the response for test dataset
y_pred= classifier.predict(X_test)
```

In [36]:
```python
# Model Accuracy, how often is the classifier correct?
from sklearn.metrics import accuracy_score
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.6
```

In [37]:
```python
data_p=pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
data_p
```

Out[37]:

	Actual	Predicted
1	0	0
11	1	0
10	1	1
5	0	1
4	1	1

In [39]:
```python
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1 1]
 [1 2]]
              precision    recall  f1-score   support

           0       0.50      0.50      0.50         2
           1       0.67      0.67      0.67         3

    accuracy                           0.60         5
   macro avg       0.58      0.58      0.58         5
weighted avg       0.60      0.60      0.60         5
```

Q9 Using disaster-tweet.csv, Write a code to classify tweets as Relevant or not Relevant using TF-IDF vectorization method

In [2]:
```python
# Import utility libraries
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sn

%matplotlib inline
# Import libraries for text manipulation
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
# Import modules for evaluation purposes
# Import libraries for predcton
from sklearn import metrics
from sklearn.metrics import confusion_matrix,accuracy_score,roc_auc_score,roc_curve,auc,f1_score

from wordcloud import WordCloud
```

In [35]:
```python
#Read the tweets of our train dataset
data = pd.read_csv("E:\Data Science\Data Set\\Tweets.csv")
data
```

Out[35]:

	id	keyword	location	text	target
0	0	ablaze	NaN	Communal violence in Bhainsa, Telangana. "Ston...	1.0
1	1	ablaze	NaN	Telangana: Section 144 has been imposed in Bha...	1.0
2	2	ablaze	New York City	Arsonist sets cars ablaze at dealership https:...	1.0
3	3	ablaze	Morgantown, WV	Arsonist sets cars ablaze at dealership https:...	1.0
4	4	ablaze	NaN	"Lord Jesus, your love brings freedom and pard...	0.0
...	...	...	...	...	...
11365	11365	wrecked	Blue State in a red sea	Media should have warned us well in advance. T...	0.0
11366	11366	wrecked	arohaonces	i feel directly attacked 💀 i consider moonbin ...	0.0
11367	11367	wrecked	🇵🇭	i feel directly attacked 💀 i consider moonbin ...	0.0
11368	11368	wrecked	auroraborealis	ok who remember "outcast" nd the "dora" au?? T...	0.0
11369	11369	wrecked	NaN	Jake Corway wreck	NaN

11370 rows × 5 columns

In [4]:
```python
#Relevant columns
TEXT_COLUMN = 'text'
TARGET_COLUMN = 'target'
```

In [ ]:

In [5]:
```python
#Extract only the text and target columns from our dataframe
data = data[[TEXT_COLUMN, TARGET_COLUMN]]
data.head()
```

Out[5]:

	text	target
0	Communal violence in Bhainsa, Telangana. "Ston...	1.0
1	Telangana: Section 144 has been imposed in Bha...	1.0
2	Arsonist sets cars ablaze at dealership https:...	1.0
3	Arsonist sets cars ablaze at dealership https:...	1.0
4	"Lord Jesus, your love brings freedom and pard...	0.0

In [31]:
```python
#check for NaN values
data.columns[data.isna().any()].tolist()
```

Out[31]:
```
['target']
```

In [32]:
```python
# Count total NaN at each column in a DataFrame
print(" \nCount total NaN at each column in a DataFrame : \n\n",
      data.isnull().sum())
```

```
Count total NaN at each column in a DataFrame :

 text      0
target    1
dtype: int64
```

In [33]:
```python
# Remove the NaN Values from the data set
data.dropna()
```

Out[33]:

	text	target
0	Communal violence in Bhainsa, Telangana. "Ston...	1.0
1	Telangana: Section 144 has been imposed in Bha...	1.0
2	Arsonist sets cars ablaze at dealership https:...	1.0
3	Arsonist sets cars ablaze at dealership https:...	1.0
4	"Lord Jesus, your love brings freedom and pard...	0.0
...	...	...
11364	Had these guys last game n fcked them. Talked ...	0.0
11365	Media should have warned us well in advance. T...	0.0
11366	i feel directly attacked 💀 i consider moonbin ...	0.0
11367	i feel directly attacked 💀 i consider moonbin ...	0.0
11368	ok who remember "outcast" nd the "dora" au?? T...	0.0

11369 rows × 2 columns

In [34]:
```python
# Count total NaN at each column in a DataFrame
print(" \nCount total NaN at each column in a DataFrame : \n\n",
      data.isnull().sum())
```

```
Count total NaN at each column in a DataFrame :

 text      0
target    1
dtype: int64
```

In [36]:
```python
np.isnan(data.any()) #and gets False
np.isfinite(data.all()) #and gets True
```

Out[36]:
```
id          True
keyword     True
location    True
text        True
target      True
dtype: bool
```

In [ ]:

In [64]:
```python
X_train, X_test, y_train, y_test = train_test_split(data[TEXT_COLUMN], data[TARGET_COLUMN].values , test_size=0.20, random_state=0)
# Show the size of our datasets
print('X Train Size:',X_train.shape)
print('X Test Size:',X_test.shape)
print('Y Train Size:',y_train.shape)
print('Y Test Size:',y_test.shape)
```

```
X Train Size: (6361,)
X Test Size: (1591,)
Y Train Size: (6361,)
Y Test Size: (1591,)
```

In [65]:
```python
# Create a Counter of tokens
count_vectorizer = CountVectorizer(decode_error='ignore', lowercase=True, min_df=2)
# Apply it on the train data to get the vocabulary and the mapping. This vocab and mapping is then applied to the test set.
# Before, we convert to Unicode to avoid issues with CountVectorizer
train = count_vectorizer.fit_transform(X_train.values.astype('U'))
test = count_vectorizer.transform(X_test.values.astype('U'))
```

In [66]:
```python
print('Train size: ',train.shape)
print('Test size: ',test.shape)
```

```
Train size:  (6361, 6697)
Test size:  (1591, 6697)
```

In [40]:
```python
vocab = list(count_vectorizer.vocabulary_.items())
print(vocab[:10])
```

```
[('hi', 3604), ('kevin', 4240), ('very', 8198), ('sorry', 7131), ('about', 260), ('this', 7727), ('freight', 3144), ('train', 7890), ('has', 3509), ('derailed', 2194)]
```

In [67]:
```python
# try multiple ways of calculating features
# Create the numericalizer TFIDF for lowercase
tfidf = TfidfVectorizer(decode_error='ignore', lowercase = True, min_df=2)
# Numericalize the train dataset
train = tfidf.fit_transform(X_train.values.astype('U'))
# Numericalize the test dataset
test = tfidf.transform(X_test.values.astype('U'))
```

In [68]:
```python
print('Train size: ',train.shape)
print('Test size: ',test.shape)
```

```
Train size:  (6361, 6697)
Test size:  (1591, 6697)
```

In [54]:
```python
print (data.dtypes)
```

```
id            int64
keyword      object
location     object
text         object
target      float64
dtype: object
```

In [57]:
```python
# Count total NaN at each column in a DataFrame
print(" \nCount total NaN at each column in a DataFrame : \n\n",
      data.isnull().sum())
```

```
Count total NaN at each column in a DataFrame :

 id             0
keyword        0
location    3418
text           0
target         1
dtype: int64
```

In [58]:
```python
# drop all rows with any NaN and NaT values
data = data.dropna()
print(data)
```

```
          id  keyword                 location  \
2          2   ablaze            New York City
3          3   ablaze           Morgantown, WV
5          5   ablaze                       OC
6          6   ablaze          London, England
7          7   ablaze                   Bharat
...      ...      ...                      ...
11362  11362  wrecked         feuille d'érable
11365  11365  wrecked  Blue State in a red sea
11366  11366  wrecked               arohaonces
11367  11367  wrecked                       🇵🇭
11368  11368  wrecked           auroraborealis

                                                    text  target
2      Arsonist sets cars ablaze at dealership https:...     1.0
3      Arsonist sets cars ablaze at dealership https:...     1.0
5      If this child was Chinese, this tweet would ha...     0.0
6      Several houses have been set ablaze in Ngemsib...     1.0
7      Asansol: A BJP office in Salanpur village was ...     1.0
...                                                  ...     ...
11362  Stell wrecked ako palagi sayo. Haha. #ALABTopS...     0.0
11365  Media should have warned us well in advance. T...     0.0
11366  i feel directly attacked 💀 i consider moonbin ...     0.0
11367  i feel directly attacked 💀 i consider moonbin ...     0.0
11368  ok who remember "outcast" nd the "dora" au?? T...     0.0

[7952 rows x 5 columns]
```

In [59]:
```python
data['target'] = data['target'].astype(int)
```

```
<ipython-input-59-b84843a1bd3a>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['target'] = data['target'].astype(int)
```

In [60]:
```python
# Extract the vocabulary as a list of (word, frequency)
vocab = list(count_vectorizer.vocabulary_.items())
print(vocab[:10])
```

```
[('hi', 3604), ('kevin', 4240), ('very', 8198), ('sorry', 7131), ('about', 260), ('this', 7727), ('freight', 3144), ('train', 7890), ('has', 3509), ('derailed', 2194)]
```

In [ ]:

In [69]:
```python
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

# Define the parameters to tune
parameters = {
    'C': [1.0, 10],
    'gamma': [1, 'auto', 'scale']
}
# Tune yyperparameters  using Grid Search and a SVM model
model = GridSearchCV(SVC(kernel='rbf'), parameters, cv=5, n_jobs=-1).fit(train, y_train)
```

In [70]:
```python
from sklearn.pipeline import Pipeline
from sklearn.svm import LinearSVC, SVC
text_clf = Pipeline([('tfidf', TfidfVectorizer()),
                     ('clf', LinearSVC()),
                    ])
text_clf.fit(X_train, y_train)
```

Out[70]:
```
Pipeline(steps=[('tfidf', TfidfVectorizer()), ('clf', LinearSVC())])
```

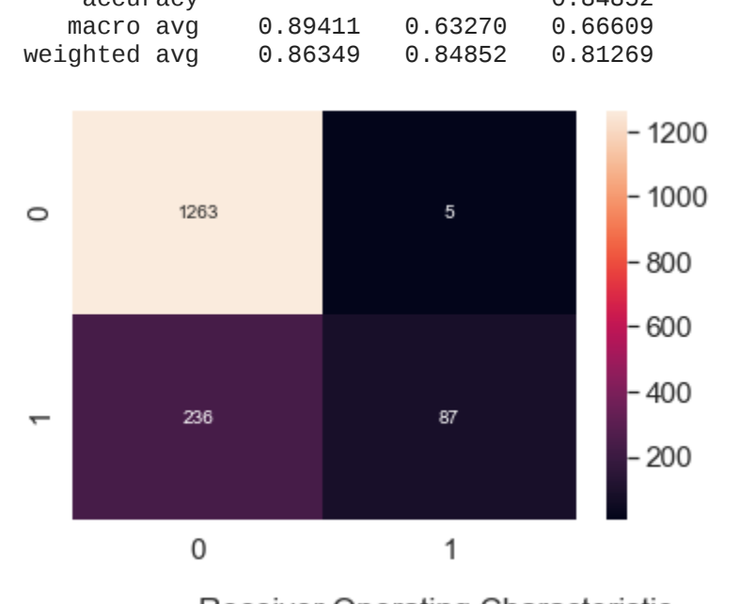
In [71]:
```python
X_train = X_train.replace((np.inf, -np.inf, np.nan), 0).reset_index(drop=True)
```

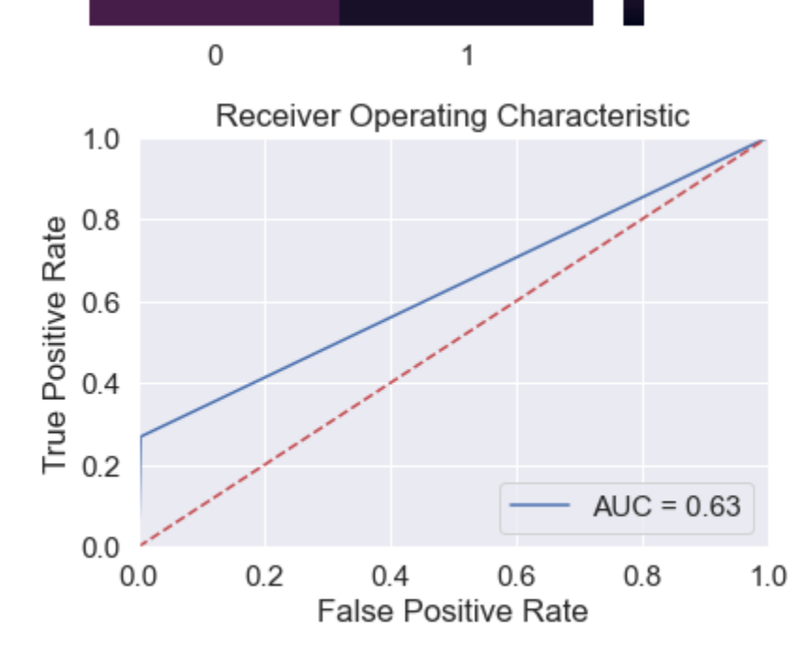
In [72]:
```python
model = MultinomialNB()
model.fit(train, y_train)
print("train score:", model.score(train, y_train))
print("test score:", model.score(test, y_test))
```

```
train score: 0.8880679138500236
test score: 0.8485229415461973
```

In [75]:
```python
# Predicting the Test set results
y_pred = model.predict(test)

print(metrics.classification_report(y_test, y_pred,  digits=5))
plot_confussion_matrix(y_test, y_pred)
plot_roc_curve(y_test, y_pred)
```

```
              precision    recall  f1-score   support

           0    0.84256   0.99606   0.91290      1268
           1    0.94565   0.26935   0.41928       323

    accuracy                        0.84852      1591
   macro avg    0.89411   0.63270   0.66609      1591
weighted avg    0.86349   0.84852   0.81269      1591
```





In [76]:
```python
def plot_confussion_matrix(y_test, y_pred):
    ''' Plot the confussion matrix for the target labels and predictions '''
    cm = confusion_matrix(y_test, y_pred)

    # Create a dataframe with the confussion matrix values
    df_cm = pd.DataFrame(cm, range(cm.shape[0]),
                  range(cm.shape[1]))
    #plt.figure(figsize = (10,7))
    # Plot the confussion matrix
    sn.set(font_scale=1.4) #for label size
    sn.heatmap(df_cm, annot=True,fmt='.0f',annot_kws={"size": 10})# font size
    plt.show()
```

In [77]:
```python
# ROC Curve
# plot no skill
# Calculate the points in the ROC curve
def plot_roc_curve(y_test, y_pred):
    ''' Plot the ROC curve for the target labels and predictions'''
    fpr, tpr, thresholds = roc_curve(y_test, y_pred, pos_label=1)
    roc_auc= auc(fpr,tpr)

    plt.title('Receiver Operating Characteristic')
    plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
    plt.legend(loc = 'lower right')
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0, 1])
    plt.ylim([0, 1])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```

In [ ]: