2. Introduce scikit-learn as a machine learning library.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = metrics.accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
```

Output :-

```
Accuracy: 0.9777777777777777
```

3. Install and set up scikit-learn and other necessary tools. Procedure.

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
iris = datasets.load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,
test_size=0.3)
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)
print(f"Accuracy: {accuracy}")
```

Output :-

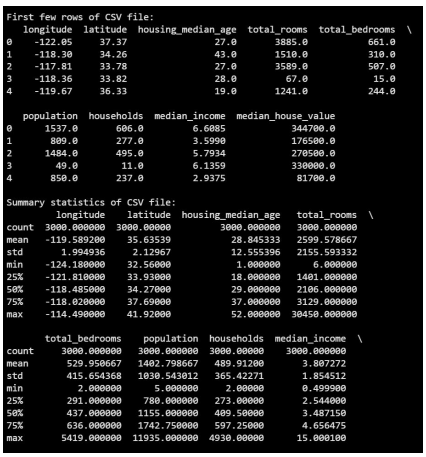
```
Accuracy: 0.9777777777777777
```

4. Write a program to Load and explore the dataset of .CVS and excel files using pandas.

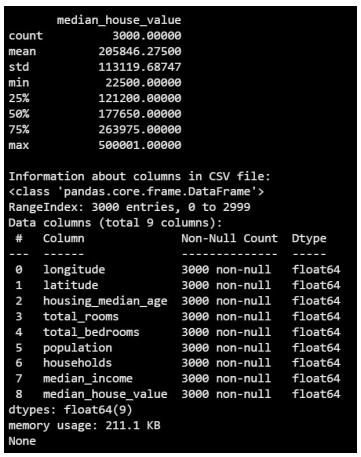
```
import pandas as pd
csv_data = pd.read_csv('california_housing_test.csv')
print("First few rows of CSV file:")
print(csv_data.head())
print("\nSummary statistics of CSV file:")
print(csv_data.describe())
print("\nInformation about columns in CSV file:")
print(csv_data.info())

excel_data = pd.read_excel('SampleData.xlsx')
```

```
print("First few rows of Excel file:")
print(excel_data.head())
print("\nSummary statistics of Excel file:")
print(excel_data.describe())
print("\nInformation about columns in Excel file:")
print(excel_data.info())
```

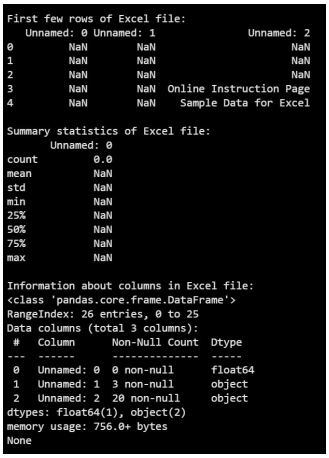
Output :-



```
First few rows of CSV file:
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.05     37.37                27.0       3885.0           661.0
1    -118.30     34.26                43.0       1510.0           310.0
2    -117.81     33.78                27.0       3589.0           507.0
3    -118.36     33.82                28.0         67.0            15.0
4    -119.67     36.33                19.0       1241.0           244.0

   population  households  median_income  median_house_value
0      1537.0       606.0         6.6085            344700.0
1       809.0       277.0         3.5990            176500.0
2      1484.0       495.0         5.7934            270500.0
3        49.0        11.0         6.1359            330000.0
4       850.0       237.0         2.9375             81700.0

Summary statistics of CSV file:
         longitude     latitude  housing_median_age   total_rooms  \
count  3000.000000  3000.000000         3000.000000   3000.000000
mean   -119.589200    35.63539           28.845333   2599.578667
std       1.994936     2.12967           12.555396   2155.593332
min    -124.180000    32.56000            1.000000      6.000000
25%    -121.810000    33.93000           18.000000   1401.000000
50%    -118.485000    34.27000           29.000000   2106.000000
75%    -118.020000    37.69000           37.000000   3129.000000
max    -114.490000    41.92000           52.000000  30450.000000

       total_bedrooms    population   households  median_income  \
count     3000.000000   3000.000000  3000.000000    3000.000000
mean       529.950667   1402.798667   489.91200       3.887272
std        415.654368   1030.543012   365.42271       1.854512
min          2.000000      5.000000     2.00000       0.499900
25%        291.000000    780.000000   273.00000       2.544000
50%        437.000000   1155.000000   409.50000       3.487150
75%        636.000000   1742.750000   597.25000       4.656475
max       5419.000000  11935.000000  4930.00000      15.000100
```



```
       median_house_value
count         3000.00000
mean        205846.27500
std         113119.68747
min          22500.00000
25%         121200.00000
50%         177650.00000
75%         263975.00000
max         500001.00000

Information about columns in CSV file:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           3000 non-null   float64
 1   latitude            3000 non-null   float64
 2   housing_median_age  3000 non-null   float64
 3   total_rooms         3000 non-null   float64
 4   total_bedrooms      3000 non-null   float64
 5   population          3000 non-null   float64
 6   households          3000 non-null   float64
 7   median_income       3000 non-null   float64
 8   median_house_value  3000 non-null   float64
dtypes: float64(9)
memory usage: 211.1 KB
None
```

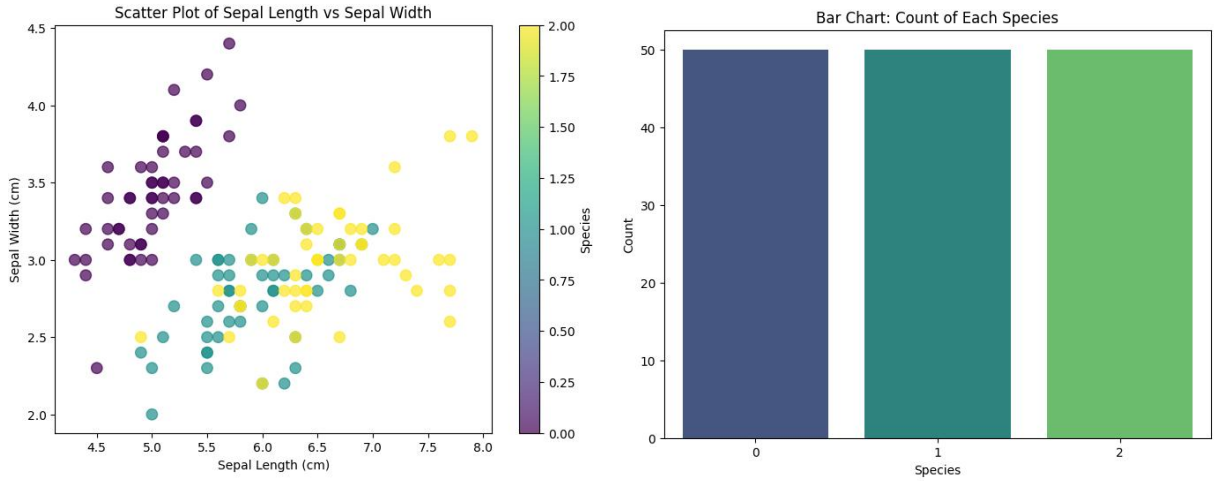


```
First few rows of Excel file:
   Unnamed: 0 Unnamed: 1                Unnamed: 2
0         NaN        NaN                       NaN
1         NaN        NaN                       NaN
2         NaN        NaN                       NaN
3         NaN        NaN  Online Instruction Page
4         NaN        NaN    Sample Data for Excel

Summary statistics of Excel file:
       Unnamed: 0
count         0.0
mean          NaN
std           NaN
min           NaN
25%           NaN
50%           NaN
75%           NaN
max           NaN

Information about columns in Excel file:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  0 non-null      float64
 1   Unnamed: 1  3 non-null      object
 2   Unnamed: 2  20 non-null     object
dtypes: float64(1), object(2)
memory usage: 756.0+ bytes
None
```

5. Write a program to Visualize the dataset to gain insights using Matplotlib or Seaborn by plotting scatter plots, bar charts.

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
import pandas as pd
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
plt.figure(figsize=(8, 6))
plt.scatter(iris_df['sepal length (cm)'], iris_df['sepal width (cm)'],
c=iris_df['target'], cmap='viridis', s=80, alpha=0.7)
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Scatter Plot of Sepal Length vs Sepal Width')
plt.colorbar(label='Species')
plt.show()
plt.figure(figsize=(8, 6))
sns.countplot(x='target', data=iris_df, palette='viridis')
plt.xlabel('Species')
plt.ylabel('Count')
plt.title('Bar Chart: Count of Each Species')
plt.show()
```

Output :-



6. Write a program to implement a k-Nearest Neighbours (k-NN) classifier using scikitlearn and Train the classifier on the dataset and evaluate its performance.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
iris = load_iris()
X = iris.data
y = iris.target
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
k = 3
knn_classifier = KNeighborsClassifier(n_neighbors=k)
knn_classifier.fit(X_train, y_train)
predictions = knn_classifier.predict(X_test)
accuracy = metrics.accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
print("Classification Report:")
print(metrics.classification_report(y_test, predictions))
print("Confusion Matrix:")
print(metrics.confusion_matrix(y_test, predictions))
```

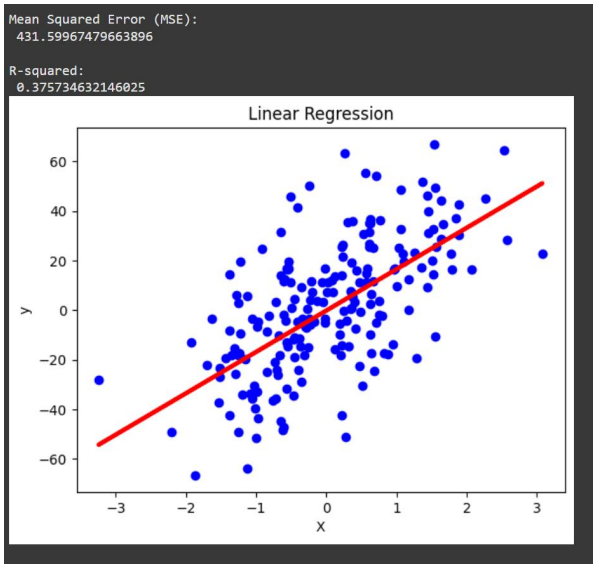
Output :-

```
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45


Confusion Matrix:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
```

7. Write a program to implement a linear regression model for regression tasks and Train the model on a dataset with continuous target variables.

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
X,y = make_regression(n_samples=1000, n_features=1, noise=20, random_state=42)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,
random_state=42)
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
predictions = linear_reg.predict(X_test)
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
print("Mean Squared Error (MSE):\n",mse)
print("R-squared:\n",r2)
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, predictions, color='red', linewidth=3)
plt.xlabel('X')
plt.ylabel('y')
plt.title('Linear Regression')
plt.show()
```

Output :-



8. Write a program to implement a decision tree classifier using scikitlearn and visualize the decision tree and understand its splits.

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
```

```
iris = load_iris()
X = iris.data
y = iris.target
class_names = [str(name) for name in iris.target_names]
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X, y)
plt.figure(figsize=(12, 8))
plot_tree(decision_tree, feature_names=iris.feature_names,
class_names=class_names, filled=True, rounded=True)
plt.title("Decision Tree Visualization")
plt.show()
```

Output :-