Assignment 1

- **Install and Import Pandas Library.**
- **Use Pandas for the following tasks:**
- **Create a Data Frame (choose data of your choice).**
- **Read and Write to CSV file.**
- **Read and write to Excel file.**
- **Rename columns in a Data Frame.**
- **Select a single column from the Data frame and print it.**

Solution:

```
import pandas as pd
# Create a DataFrame
data = {'Name': ['John', 'Alice', 'Bob', 'Emily'],
'Age': [25, 30, 35, 40],
'City': ['New York', 'Paris', 'London', 'Sydney']}
df = pd.DataFrame(data)
print(df)

# Write the DataFrame to a CSV file
df.to_csv('example.csv', index=False)
df_csv = pd.read_csv('example.csv')
df.to_excel('example.xlsx', index=False)
df_excel = pd.read_excel('example.xlsx')
df.rename(columns={'Name': 'First Name', 'City': 'Location'},
inplace=True)
print(df['Age'])
```

OUTPUT:

```
    Name  Age      City
0   John   25  New York
1  Alice   30     Paris
2    Bob   35    London
3  Emily   40    Sydney

0    25
1    30
2    35
3    40
Name: Age, dtype: int64
```

Assignment 2

- **Install and Import Numpy Library.**
- **Use Numpy for the following tasks:**
- **Create an empty and a full array of size 3x3.**
- **Generate an array of 25 random numbers sampled from a standard normal distribution.**
- **Find Dot product of two arrays.**
- **Sort the below array along the row, along the column and as a whole.**
- **[3, 7, 1] [10, 3, 2] [5, 6, 7]**
- **Make a list of 3 numpy arrays and find the mean of all the numpy arrays and output them as a list.**
- **Make a numpy array containing the string 'PHP C# Python C Java C++' as the only element, then split it on the basis of spaces.**

Solution:

```
import numpy as np
num = np.empty((3,3))
print(num)

num2 = np.zeros((3,3))
print(num2)

import numpy as np

random_array = np.random.normal(0.0, 1.0, 25)

print("1D Array with random values : \n", random_array)

import numpy as np
arr1 = np.matrix('[1, 2, 3; 4, 5, 6; 7, 8, 9]')
arr2 = np.matrix('[1, 2, 3; 4, 5, 6; 7, 8, 9]')
arr = arr1.dot(arr2)
print(arr)


a = np.array([[3, 7, 1], [10, 3,2],[5, 6, 7]])
b = np.sort(a, axis = 0)
b

b = np.sort(a, axis = -1)
b


b = np.sort(a, axis = None)
b
```

```
import numpy as np

Input = [np.array([1, 2, 3]),
         np.array([4, 5, 6]),
         np.array([7, 8, 9])]

b = []

for i in range(len(Input)):
   b.append(np.mean(Input[i]))


print(b)

print(np.char.split('PHP c# Python C++ Java HTML CSS Javascript Ruby
Chiken'))
```

OUTPUT:

```
[[0.00e+000 0.00e+000 0.00e+000]
 [0.00e+000 0.00e+000 4.33e-321]
 [0.00e+000 0.00e+000 0.00e+000]]

[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]

1D Array with random values :
 [-0.088618   -0.49471685  0.12885889 -1.19017752  1.17082229 -0.3868410
8
  1.25185868 -1.15853162 -1.42604111  1.27803133  0.55996665 -0.32010013
 -0.14863046  0.71897288  1.17108734  0.13672456  0.37114933  1.16026666
  0.17955642  0.07249019  0.47964567 -0.43793002  0.44137223 -0.11774771
  1.06238952]

[[ 30  36  42]
 [ 66  81  96]
 [102 126 150]]

[2.0, 5.0, 8.0]

['PHP', 'c#', 'Python', 'C++', 'Java', 'HTML', 'CSS', 'Javascript', 'Rub
y', 'Chiken']
```

Assignment 3

Install and Import Matplotlib Library.
Use Matplotlib library and the csv file provided to generate plots with the given instruction below:

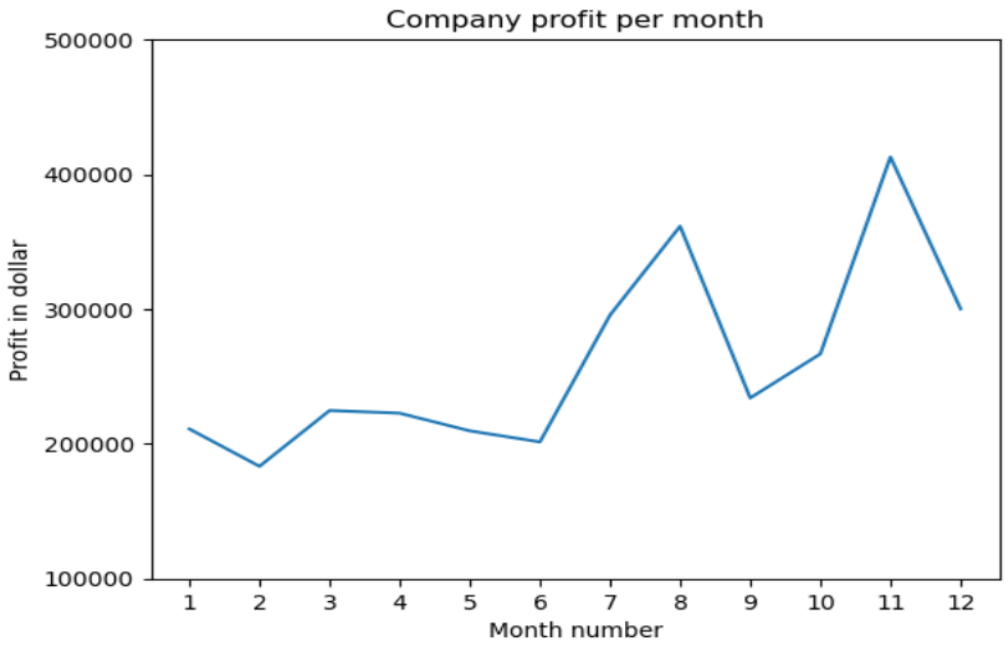
1. **Read Total profit of all months and show it using a line plot. Total profit data provided for each month. Generated line plot must include the following properties:**
    a. **X label name = Month Number**
    b. **Y label name = Total profit**
2. **Get total profit of all months and show line plot with the following Style properties. Total profit data provided for each month (same as 1st question). Generated line plot must include following Style properties:**
    a. **Line Style dotted and Line-color should be red**
    b. **Show legend at the lower right location.**
    c. **X label name = Month Number**
    d. **Y label name = Sold units number**
    e. **Add a circle marker.**
    f. **Line marker color as read**
    g. **Line width should be 3**
3. **Display the number of units sold per month for each product using multiline plots. (i.e., Separate Plotline for each product)**

4. **Read toothpaste sales data of each month and show it using a scatter plot. Also, add a grid in the plot, gridline style should be "–".**
5. **Read face cream and facewash product sales data and show it using the bar chart. The bar chart should display the number of units sold per month for each product. Add a separate bar for each product in the same chart.**

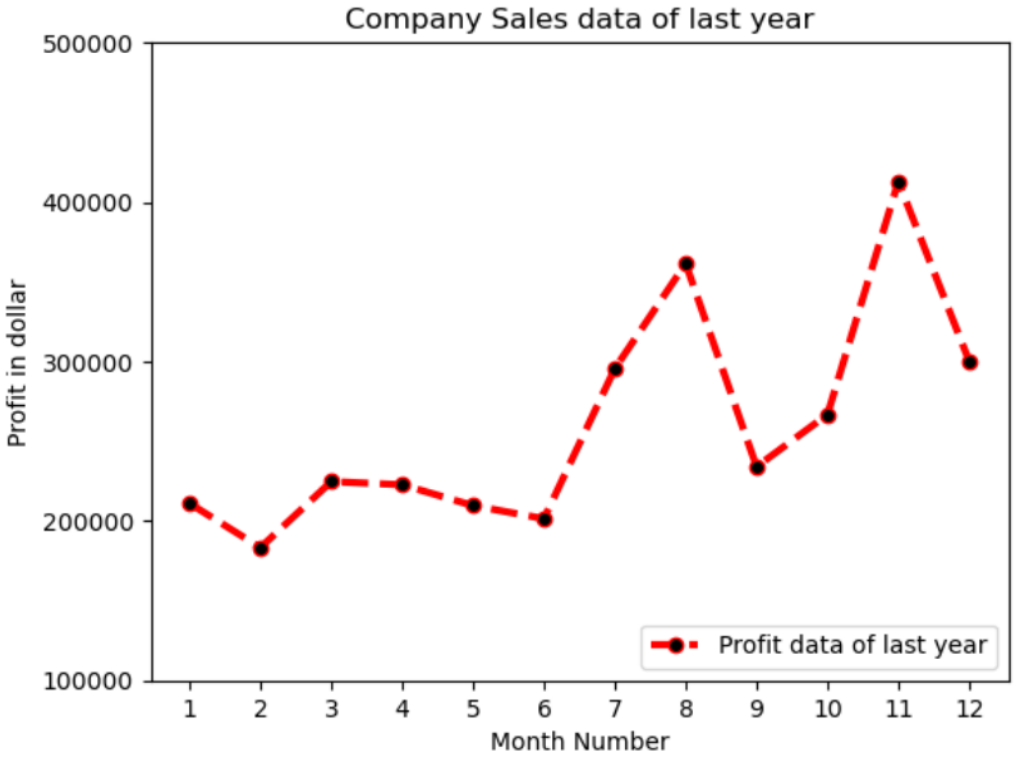
Solution:

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("Downloads\\cs313_assignmnet_3.csv")
profitList = df ['total_profit'].tolist()
monthList  = df ['month_number'].tolist()

plt.plot(monthList, profitList, label = 'Month-wise Profit data of last
year')
plt.xlabel('Month number')
plt.ylabel('Profit in dollar')
plt.xticks(monthList)
plt.title('Company profit per month')
plt.yticks([100000, 200000, 300000, 400000, 500000])
```

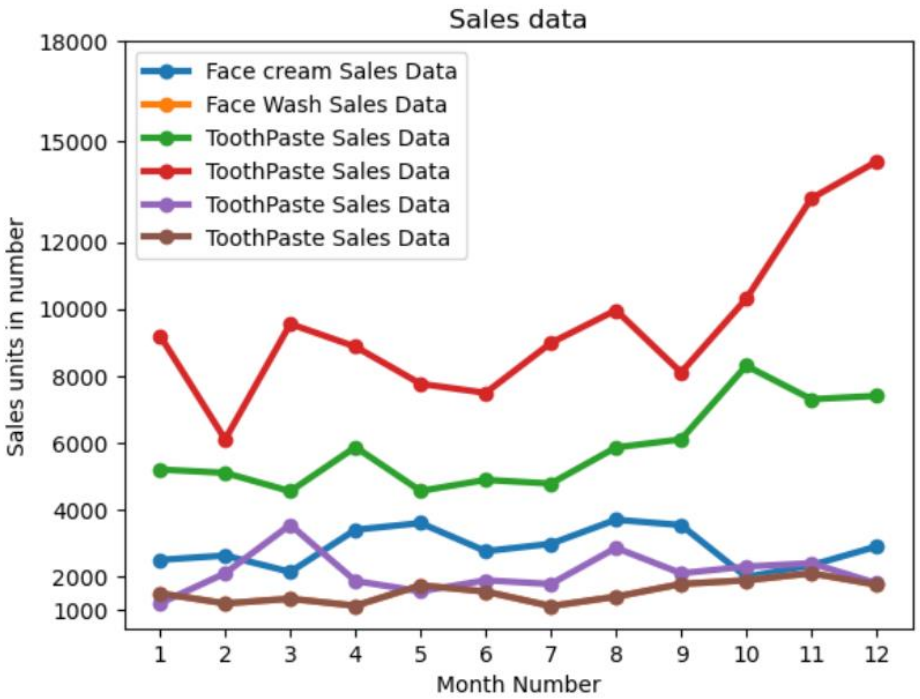
```
plt.show()
```




```
plt.plot(monthList, profitList, label = 'Profit data of last year',
      color='r', marker='o', markerfacecolor='k',
      linestyle='--', linewidth=3)

plt.xlabel('Month Number')
plt.ylabel('Profit in dollar')
plt.legend(loc='lower right')
plt.title('Company Sales data of last year')
plt.xticks(monthList)
plt.yticks([100000, 200000, 300000, 400000, 500000])
plt.show()
```

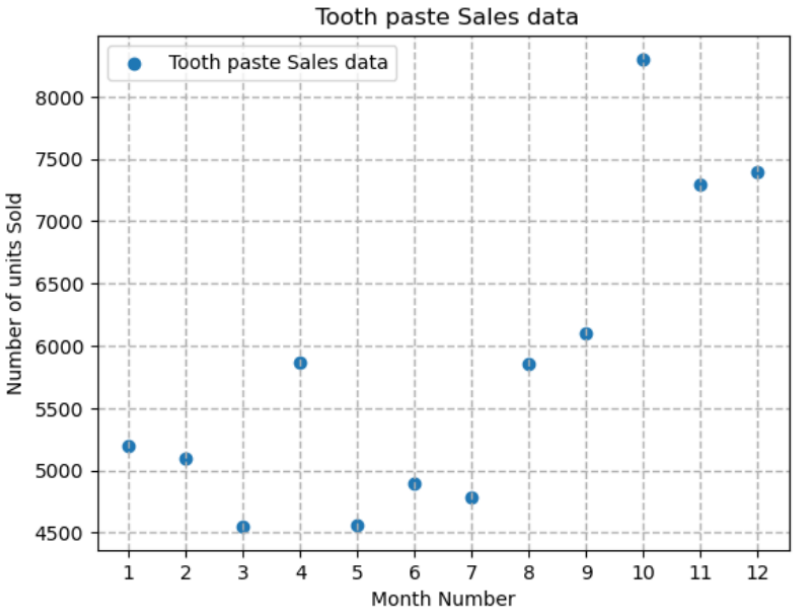


```
faceCremSalesData   = df ['facecream'].tolist()
faceWashSalesData   = df ['facewash'].tolist()
toothPasteSalesData = df ['toothpaste'].tolist()
bathingsoapSalesData   = df ['bathingsoap'].tolist()
shampooSalesData   = df ['shampoo'].tolist()
moisturizerSalesData = df ['moisturizer'].tolist()

plt.plot(monthList, faceCremSalesData,   label = 'Face cream Sales Data',
marker='o', linewidth=3)
plt.plot(monthList, faceWashSalesData,   label = 'Face Wash Sales
Data',  marker='o', linewidth=3)
plt.plot(monthList, toothPasteSalesData, label = 'ToothPaste Sales Data',
marker='o', linewidth=3)
plt.plot(monthList, bathingsoapSalesData, label = 'ToothPaste Sales
Data', marker='o', linewidth=3)
plt.plot(monthList, shampooSalesData, label = 'ToothPaste Sales Data',
marker='o', linewidth=3)
plt.plot(monthList, moisturizerSalesData, label = 'ToothPaste Sales
Data', marker='o', linewidth=3)

plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.legend(loc='upper left')
plt.xticks(monthList)
plt.yticks([1000, 2000, 4000, 6000, 8000, 10000, 12000, 15000, 18000])
plt.title('Sales data')
plt.show()
```

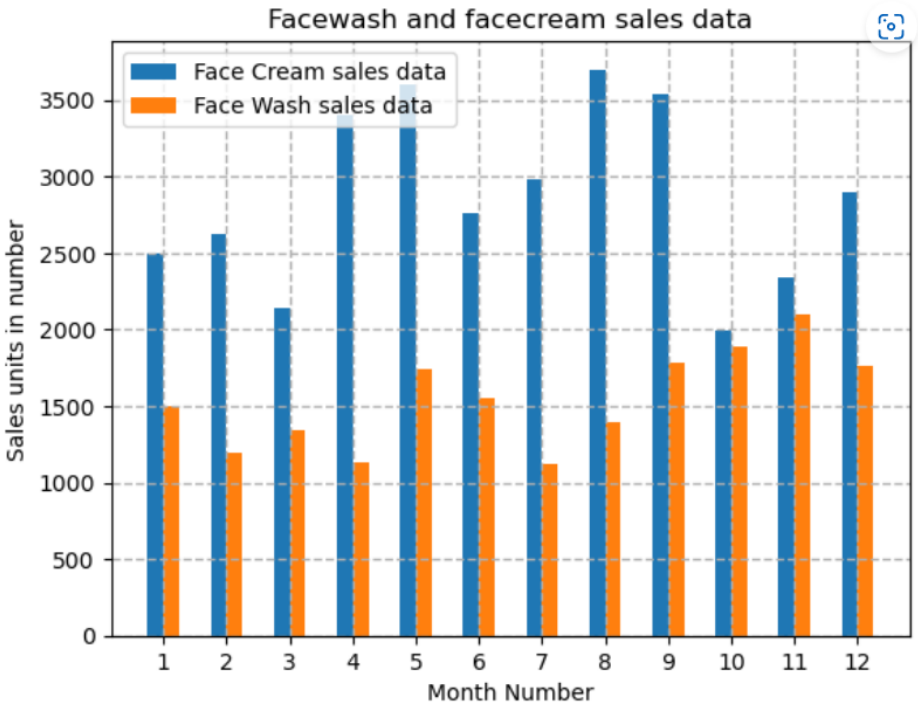



```
plt.scatter(monthList, toothPasteSalesData, label = 'Tooth paste Sales data')
plt.xlabel('Month Number')
plt.ylabel('Number of units Sold')
plt.legend(loc='upper left')
plt.title(' Tooth paste Sales data')
plt.xticks(monthList)
```

```
plt.grid(True, linewidth= 1, linestyle="--")
plt.show()
```




```
plt.bar([a-0.25 for a in monthList], faceCremSalesData, width= 0.25,
label = 'Face Cream sales data', align='edge')
plt.bar([a+0.25 for a in monthList], faceWashSalesData, width= -0.25,
label = 'Face Wash sales data', align='edge')
plt.xlabel('Month Number')
plt.ylabel('Sales units in number')
plt.legend(loc='upper left')
plt.title(' Sales data')

plt.xticks(monthList)
plt.grid(True, linewidth= 1, linestyle="--")
plt.title('Facewash and facecream sales data')
plt.show()
```



Assignment 4

- Perform Exploratory Data Analysis (EDA) for Iris Species Dataset provided as csv file (you can also download it from: https://www.kaggle.com/datasets/uciml/iris). Your EDA should include the following operations:
  1. Show size of the dataset.
  2. Show datatype for each column.
  3. Show distribution of data (use describe() function).
  4. Check if there are any null values.
  5. Check for duplicates.
  6. Check the number of instances for each species of flower.
  7. Compare sepal length and sepal width.
  8. Compare petal length and petal width.
  9. Use pairplot to show all comparisons.
  10. Use histograms to compare sepal length, sepal width, petal length and petal width across the species.
  11. Use boxplot to show distribution of data across the species.
  12. Use violinplot to show distribution of data across the species.

Solution:

```
# In[1]:
import pandas as pd
df = pd.read_csv("iris_csv.csv")
df.head()

# In[2]:
df.shape
```

```
Output: (150, 5)
```

```
# In[3]:
df.info()
```

```
Output: <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 150 entries, 0 to 149
        Data columns (total 5 columns):
         #   Column        Non-Null Count  Dtype
        ---  ------        --------------  -----
         0   sepal-length  150 non-null    float64
         1   sepal-width   150 non-null    float64
         2   petal-length  150 non-null    float64
         3   petal-width   150 non-null    float64
         4   Class         150 non-null    object
        dtypes: float64(4), object(1)
        memory usage: 6.0+ KB
```

```
# In[4]:
df.describe()
```

Output:

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
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[5]:
df.isnull().sum()
```

Output:
```
sepal-length    0
sepal-width     0
petal-length    0
petal-width     0
Class           0
dtype: int64
```

```
# In[6]:
data = df.drop_duplicates(subset ="Class",)
data
```

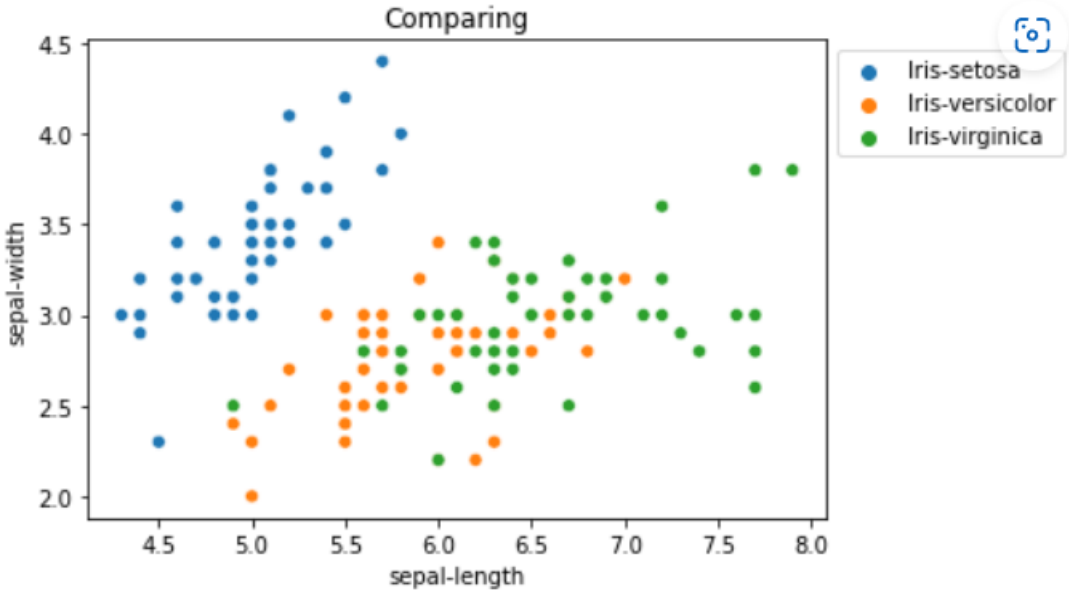
Output:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
50	7.0	3.2	4.7	1.4	Iris-versicolor
100	6.3	3.3	6.0	2.5	Iris-virginica

```
# In[7]:
df.value_counts("Class")

# In[8]:
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(x='sepal-length', y='sepal-width', hue='Class', data=df,
)
plt.legend(bbox_to_anchor=(1, 1), loc=2)
```

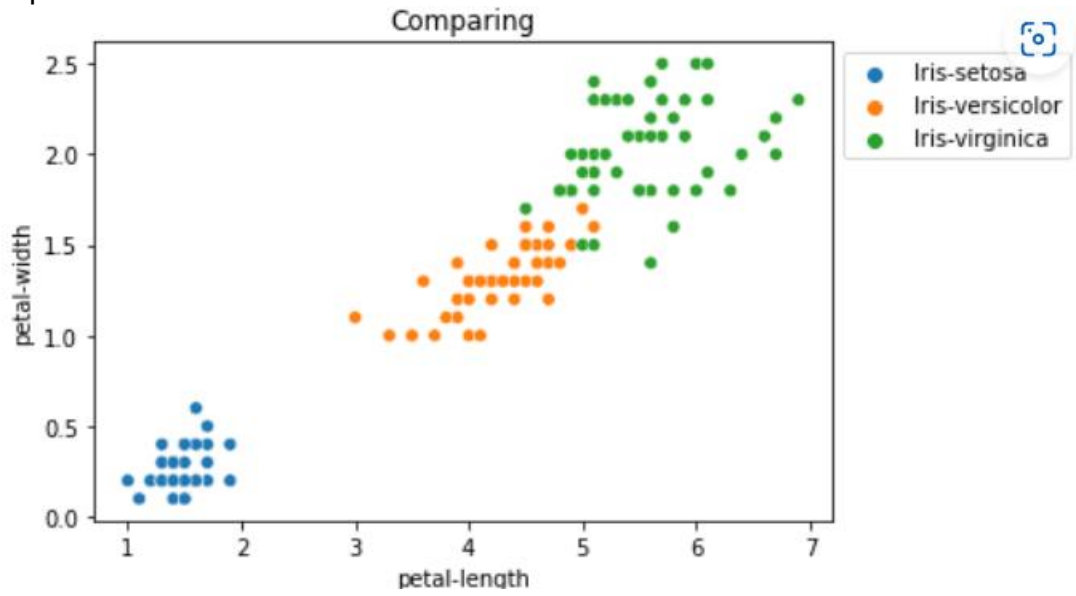
```
plt.title('Comparing')
plt.show()
```

Output:



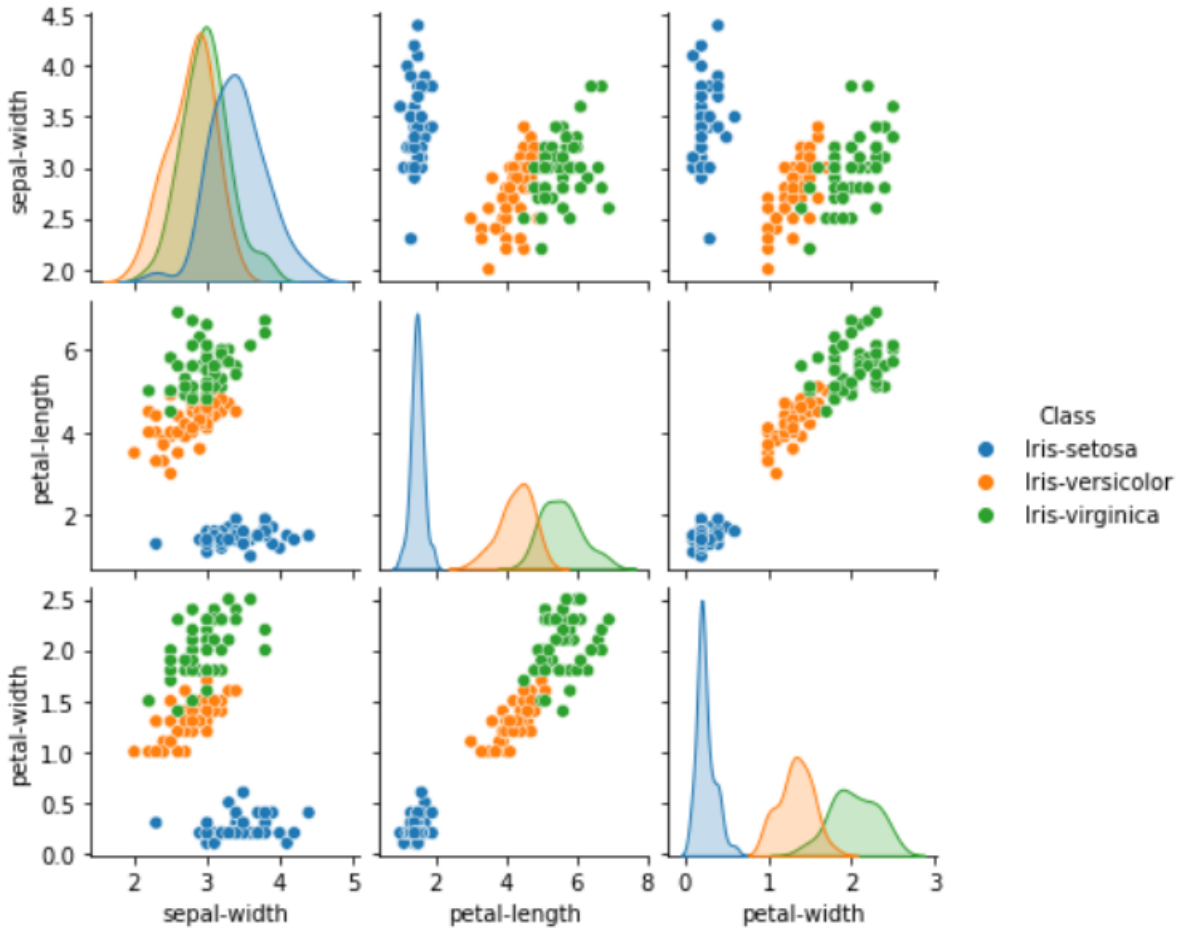

```
# In[9]:
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(x='petal-length', y='petal-width',
                hue='Class', data=df, )

plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.title('Comparing')
plt.show()
```

Output:



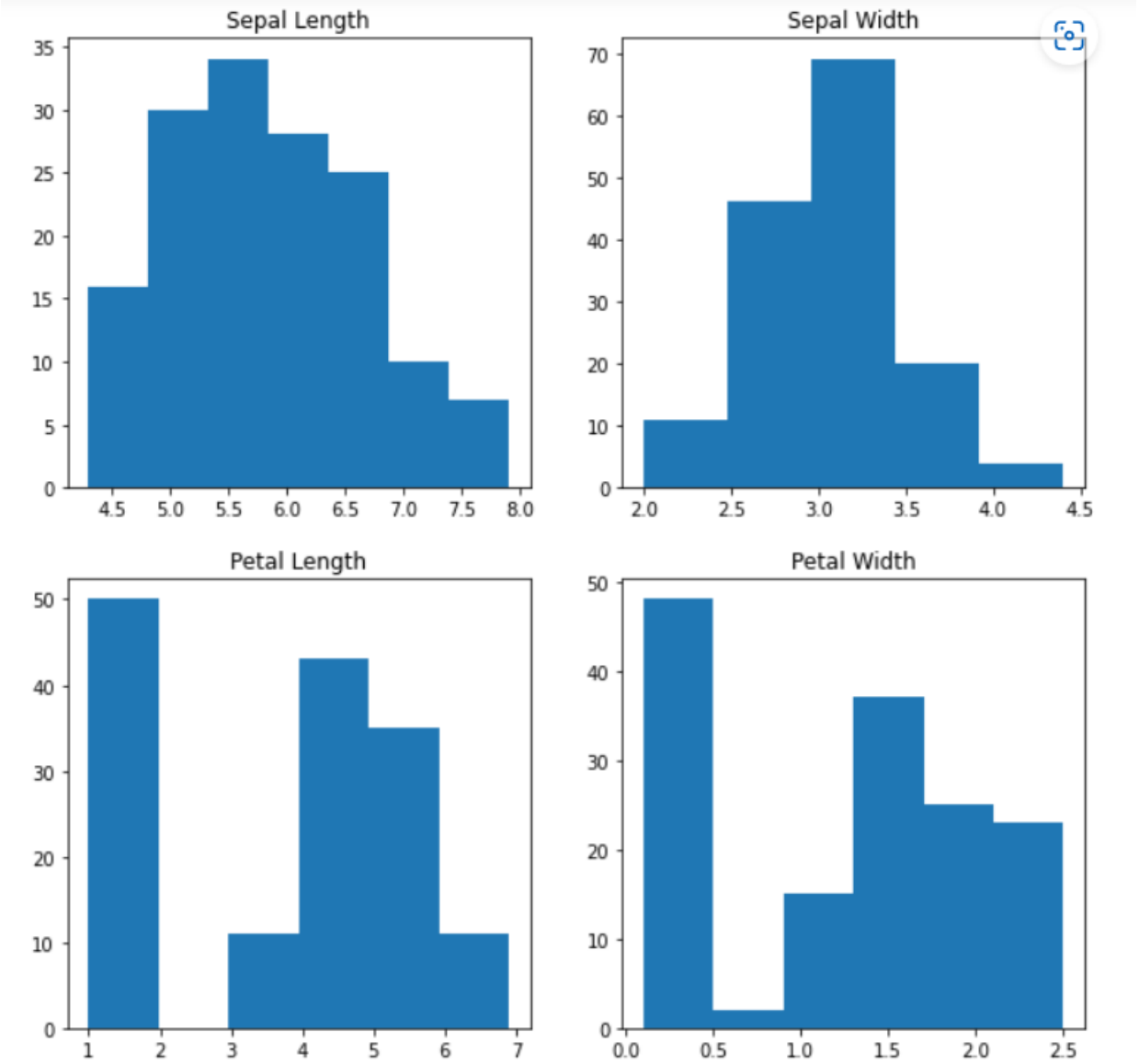

```
# In[10]:
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(df.drop(['sepal-length'], axis = 1),
             hue='Class', height=2)
```

Output:



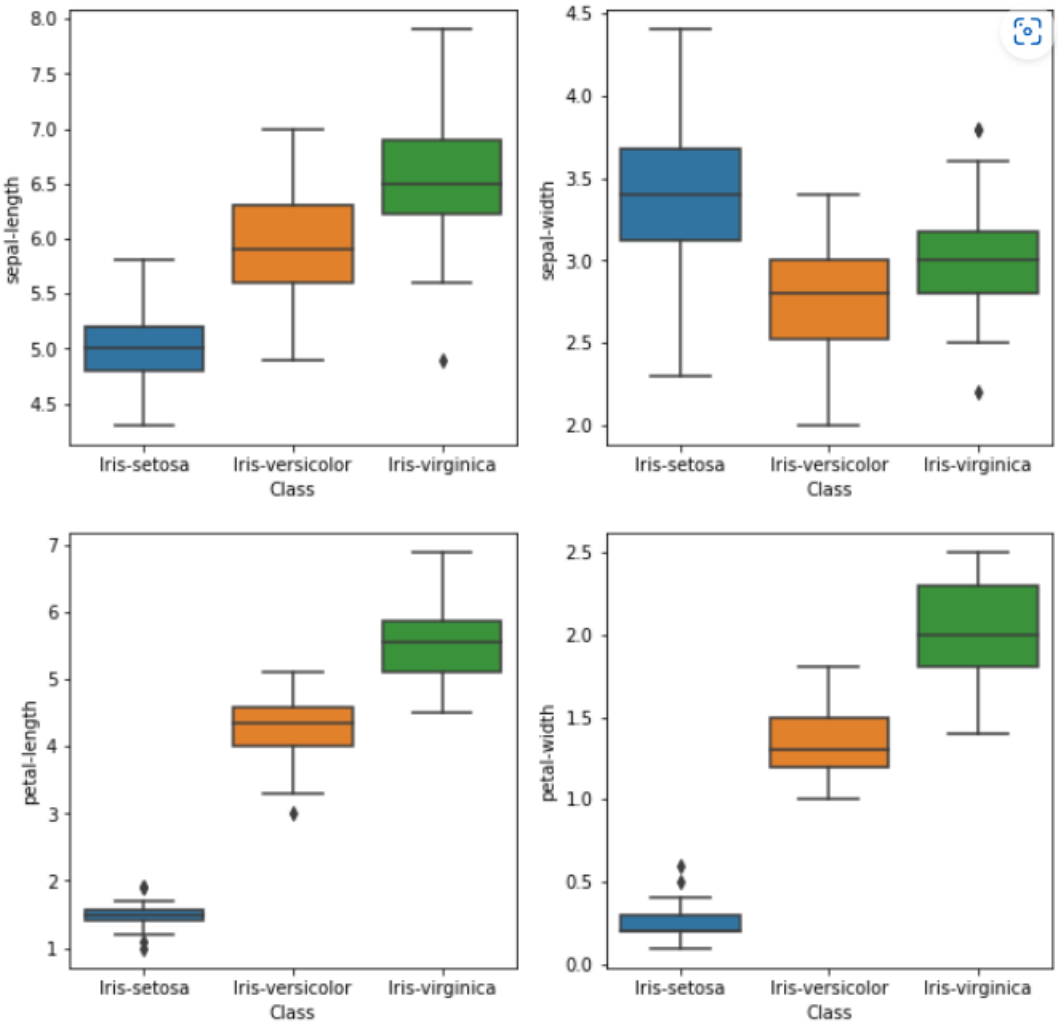

```
# In[11]:
import seaborn as sns
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2, 2, figsize=(10,10))
axes[0,0].set_title("Sepal Length")
axes[0,0].hist(df['sepal-length'], bins=7)
axes[0,1].set_title("Sepal Width")
axes[0,1].hist(df['sepal-width'], bins=5);
axes[1,0].set_title("Petal Length")
axes[1,0].hist(df['petal-length'], bins=6);
axes[1,1].set_title("Petal Width")
axes[1,1].hist(df['petal-width'], bins=6);
```

Output:



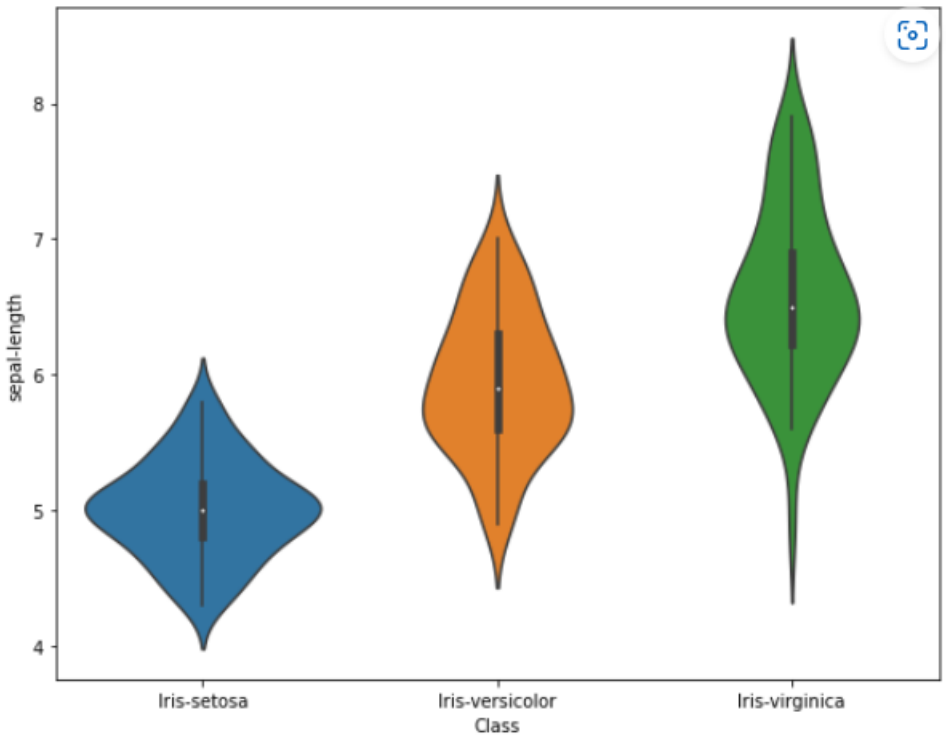

```
# In[12]:
import seaborn as sns
import matplotlib.pyplot as plt

def graph(y):
    sns.boxplot(x="Class", y=y, data=df)

plt.figure(figsize=(10,10))
plt.subplot(221)
graph('sepal-length')
plt.subplot(222)
graph('sepal-width')
plt.subplot(223)
graph('petal-length')
plt.subplot(224)
graph('petal-width')

plt.show()
```

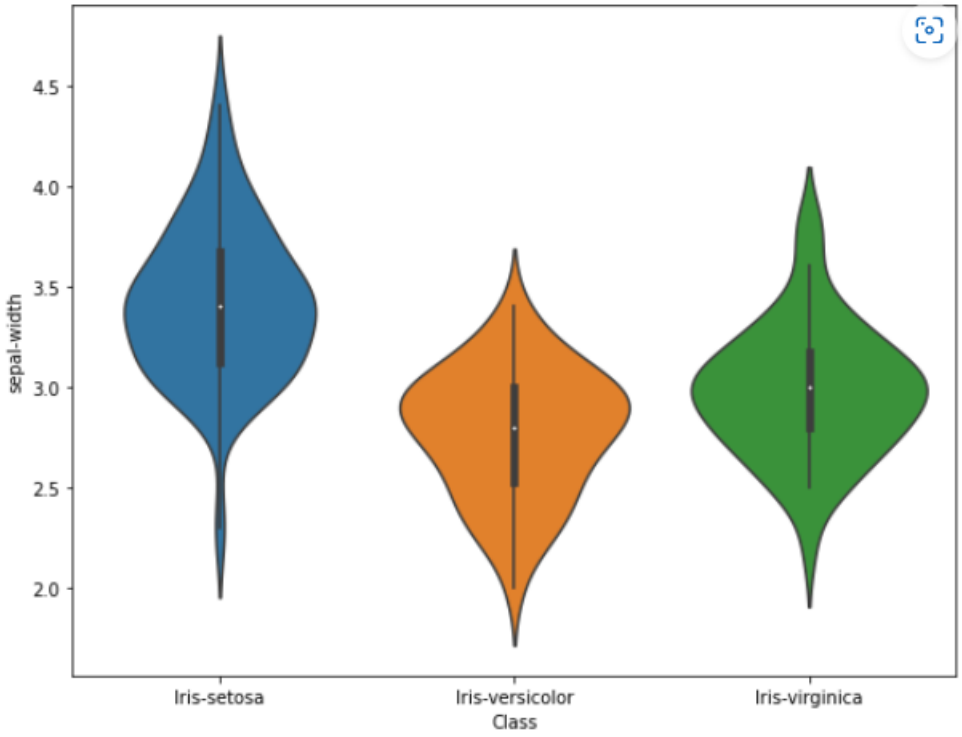
Output:




```
# In[13]:
import seaborn as sns
from matplotlib import pyplot
import seaborn
fig, ax = pyplot.subplots(figsize =(9, 7))
sns.violinplot( ax = ax, x = df["Class"], y = df["sepal-length"] )

# In[14]:
fig, ax = pyplot.subplots(figsize =(9, 7))
sns.violinplot( ax = ax, x = df["Class"], y = df["sepal-width"] )
```

Ouput:


sepal-length
8
7
6
5
4
Iris-setosa
Iris-versicolor
Iris-virginica
Class


sepal-width
4.5
4.0
3.5
3.0
2.5
2.0
Iris-setosa
Iris-versicolor
Iris-virginica
Class

Assignment 5

Write a python program to apply Decision Tree Classifier algorithm on the following datasets using Scikit-learn. (Find the datasets attached along with this assignment).

Solution :

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report


def importdata():
    balance_data = pd.read_csv(
        'https://archive.ics.uci.edu/ml/machine-learning-' +
        'databases/balance-scale/balance-scale.data',
        sep=',', header=None)

    print("Dataset Length: ", len(balance_data))
    print("Dataset Shape: ", balance_data.shape)

    print("Dataset: ", balance_data.head())
    return balance_data


def splitdataset(balance_data):

    X = balance_data.values[:, 1:5]
    Y = balance_data.values[:, 0]

    X_train, X_test, y_train, y_test = train_test_split(
        X, Y, test_size=0.3, random_state=100)

    return X, Y, X_train, X_test, y_train, y_test


def train_using_gini(X_train, X_test, y_train):

    clf_gini = DecisionTreeClassifier(criterion="gini",random_state=100,
max_depth=3, min_samples_leaf=5)

    clf_gini.fit(X_train, y_train)
    return clf_gini


def tarin_using_entropy(X_train, X_test, y_train):

    clf_entropy = DecisionTreeClassifier(
        criterion="entropy", random_state=100,
```

```
            max_depth=3, min_samples_leaf=5)

        clf_entropy.fit(X_train, y_train)
        return clf_entropy


    def prediction(X_test, clf_object):

        y_pred = clf_object.predict(X_test)
        print("Predicted values:")
        print(y_pred)
        return y_pred


    def cal_accuracy(y_test, y_pred):

        print("Confusion Matrix: ",
              confusion_matrix(y_test, y_pred))

        print("Accuracy : ",
              accuracy_score(y_test, y_pred)*100)

        print("Report : ",
              classification_report(y_test, y_pred))



    def main():


        data = importdata()
        X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
        clf_gini = train_using_gini(X_train, X_test, y_train)
        clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

        print("Results Using Gini Index:")

        y_pred_gini = prediction(X_test, clf_gini)
        cal_accuracy(y_test, y_pred_gini)

        print("Results Using Entropy:")
        y_pred_entropy = prediction(X_test, clf_entropy)
        cal_accuracy(y_test, y_pred_entropy)

    main()
```

output:

```
Dataset Length:  625
Dataset Shape:  (625, 5)
Dataset:      0  1  2  3  4
0  B  1  1  1  1
1  R  1  1  1  2
```

```
2  R  1  1  1  3
3  R  1  1  1  4
4  R  1  1  1  5
Results Using Gini Index:
Predicted values:
['R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'L'
 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'L'
 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'L' 'R'
 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L'
 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R'
 'L' 'R' 'R' 'L' 'L' 'R' 'R' 'R']
Confusion Matrix:  [[ 0  6  7]
 [ 0 67 18]
 [ 0 19 71]]
Accuracy :  73.40425531914893
Report :                precision    recall  f1-score   support

           B       0.00      0.00      0.00        13
           L       0.73      0.79      0.76        85
           R       0.74      0.79      0.76        90

    accuracy                           0.73       188
   macro avg       0.49      0.53      0.51       188
weighted avg       0.68      0.73      0.71       188

Results Using Entropy:
Predicted values:
['R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L'
 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L'
 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L'
 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L'
 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'L' 'R'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'R']
Confusion Matrix:  [[ 0  6  7]
 [ 0 63 22]
 [ 0 20 70]]
Accuracy :  70.74468085106383
Report :                precision    recall  f1-score   support

           B       0.00      0.00      0.00        13
           L       0.71      0.74      0.72        85
           R       0.71      0.78      0.74        90

    accuracy                           0.71       188
   macro avg       0.47      0.51      0.49       188
weighted avg       0.66      0.71      0.68       188
```

Assignment 6

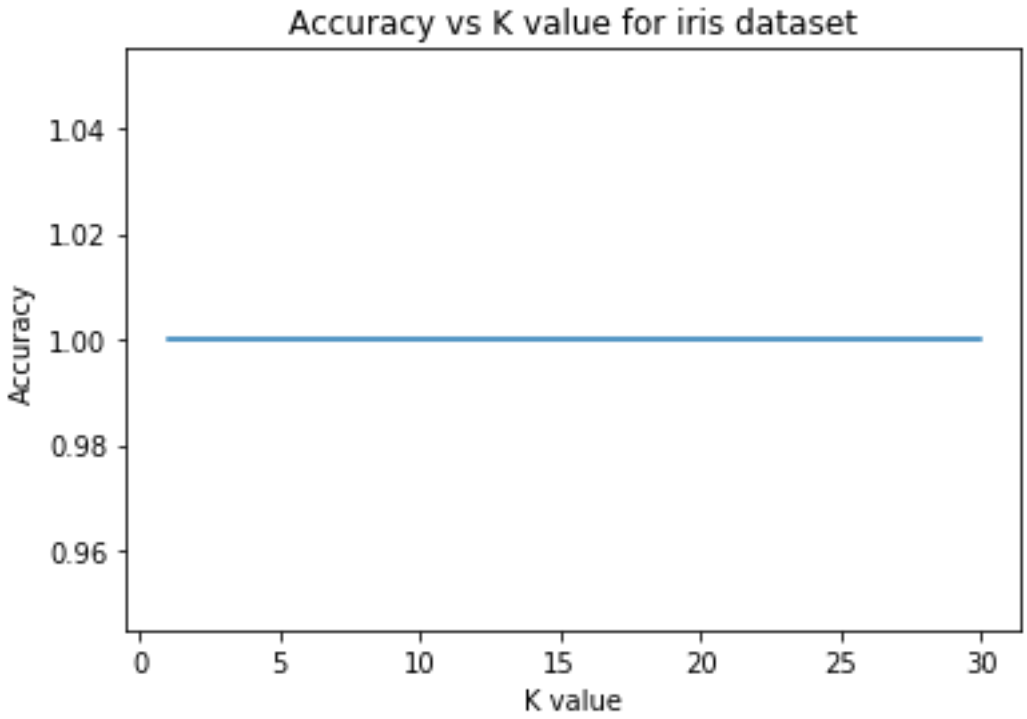
Write a python program to apply KNN (Classification + Regression) algorithm on the given datasets using Scikit-learn and also plot accuracy and mean error vs k value for both.(Find the datasets attached along with this assignment).

Solution:

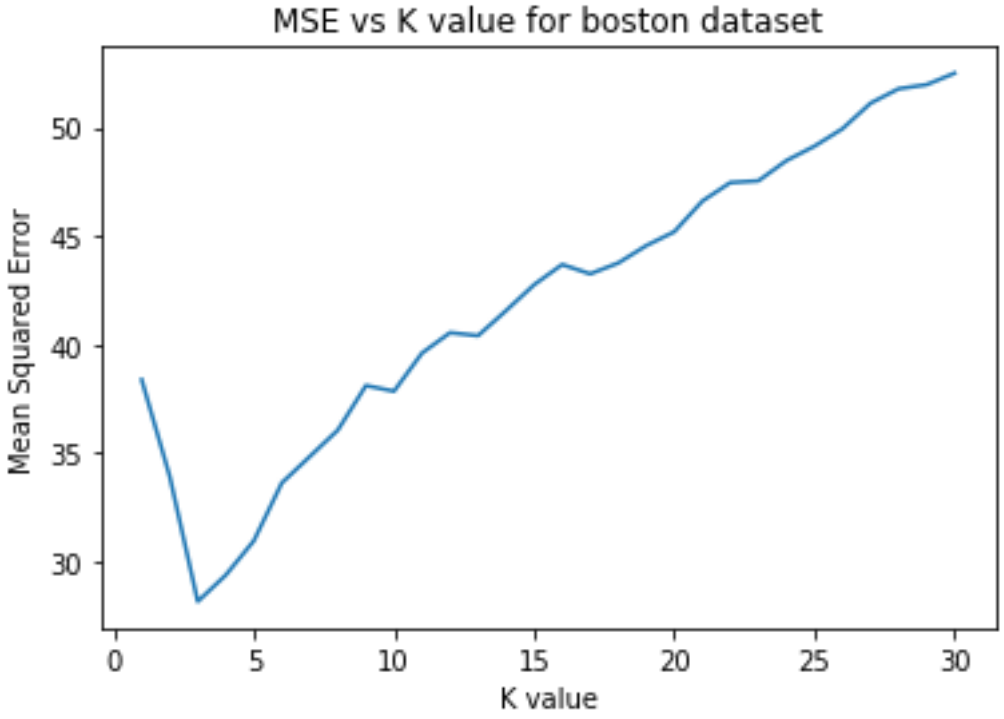
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris, load_boston
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.metrics import accuracy_score, mean_squared_error
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, test_size=0.3, random_state=42)
knn_clf = KNeighborsClassifier(n_neighbors=3)
knn_clf.fit(X_train, y_train)
y_pred = knn_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy for K=3 in iris dataset is: ", accuracy)
k_range = range(1, 31)
accuracy_scores = []
for k in k_range:
    knn_clf = KNeighborsClassifier(n_neighbors=k)
    knn_clf.fit(X_train, y_train)
    y_pred = knn_clf.predict(X_test)
    accuracy_scores.append(accuracy_score(y_test, y_pred))
plt.plot(k_range, accuracy_scores)
plt.xlabel('K value')
plt.ylabel('Accuracy')
plt.title('Accuracy vs K value for iris dataset')
plt.show()
boston = load_boston()
X_train, X_test, y_train, y_test = train_test_split(
    boston.data, boston.target, test_size=0.3, random_state=42)
knn_reg = KNeighborsRegressor(n_neighbors=3)
knn_reg.fit(X_train, y_train)
y_pred = knn_reg.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean squared error for K=3 in boston dataset is: ", mse)
mse_scores = []
for k in k_range:
    knn_reg = KNeighborsRegressor(n_neighbors=k)
    knn_reg.fit(X_train, y_train)
    y_pred = knn_reg.predict(X_test)
    mse_scores.append(mean_squared_error(y_test, y_pred))
plt.plot(k_range, mse_scores)
plt.xlabel('K value')
plt.ylabel('Mean Squared Error')
plt.title('MSE vs K value for boston dataset')
plt.show()
```

OUTPUT:-

```
Accuracy for K=3 in iris dataset is:  1.0
```




```
Mean squared error for K=3 in boston dataset is:  28.149334795321632
```



Assignment 7

Write a python program to apply Perceptron, SVM, Logistic Regression algorithm on the given datsets using Scikit-learn and also plot accuracy and show the precision, recall, f1-score, support table. (Find the datasets attached along with this assignment).

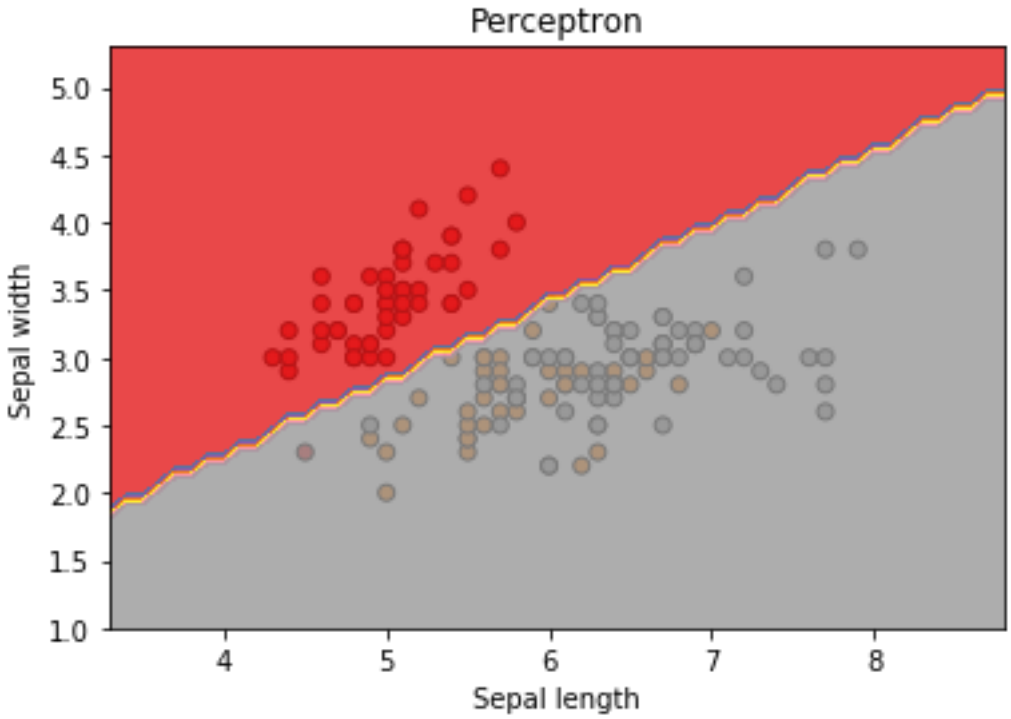
Solution:

```
from sklearn import datasets
from sklearn.linear_model import Perceptron, LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)
models = [
    ('Perceptron', Perceptron(random_state=42)),
    ('SVM', SVC(random_state=42)),
    ('Logistic Regression', LogisticRegression(random_state=42))
]
for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    print(f"{name} accuracy: {accuracy}")
    print(f"{name} classification report:\n{report}"
    plt.figure()
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor='k')
    plt.xlabel('Sepal length')
    plt.ylabel('Sepal width')
    plt.title(name)
    xx, yy=np.meshgrid(np.arange(X[:, 0].min() - 1, X[:, 0].max() + 1,
0.1),
                       np.arange(X[:, 1].min() - 1, X[:, 1].max() + 1,
0.1))
    Z=model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z=Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.Set1, alpha=0.8)
    plt.show()
```

OUTPUT:

```
Perceptron accuracy: 0.7111111111111111
Perceptron classification report:
              precision    recall  f1-score   support
```

0	1.00	1.00	1.00	19
1	0.50	1.00	0.67	13
2	0.00	0.00	0.00	13
micro avg	0.71	0.71	0.71	45
macro avg	0.50	0.67	0.56	45
weighted avg	0.57	0.71	0.61	45



SVM accuracy: 0.8
SVM classification report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	19
1	0.70	0.54	0.61	13
2	0.62	0.77	0.69	13
micro avg	0.80	0.80	0.80	45
macro avg	0.78	0.77	0.77	45
weighted avg	0.81	0.80	0.80	45