Importing libraries from sklearn

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

Loading iris Dataset from sklearn Library

In [4]:

```
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

Describing the Data from iris Dataset

In [5]:

```
iris_df = pd.DataFrame(data=X, columns=iris.feature_names)
iris_df['target'] = iris.target
iris_df['species'] = iris_df['target'].apply(lambda x: iris.target_names[x])
print(iris_df.describe())
```

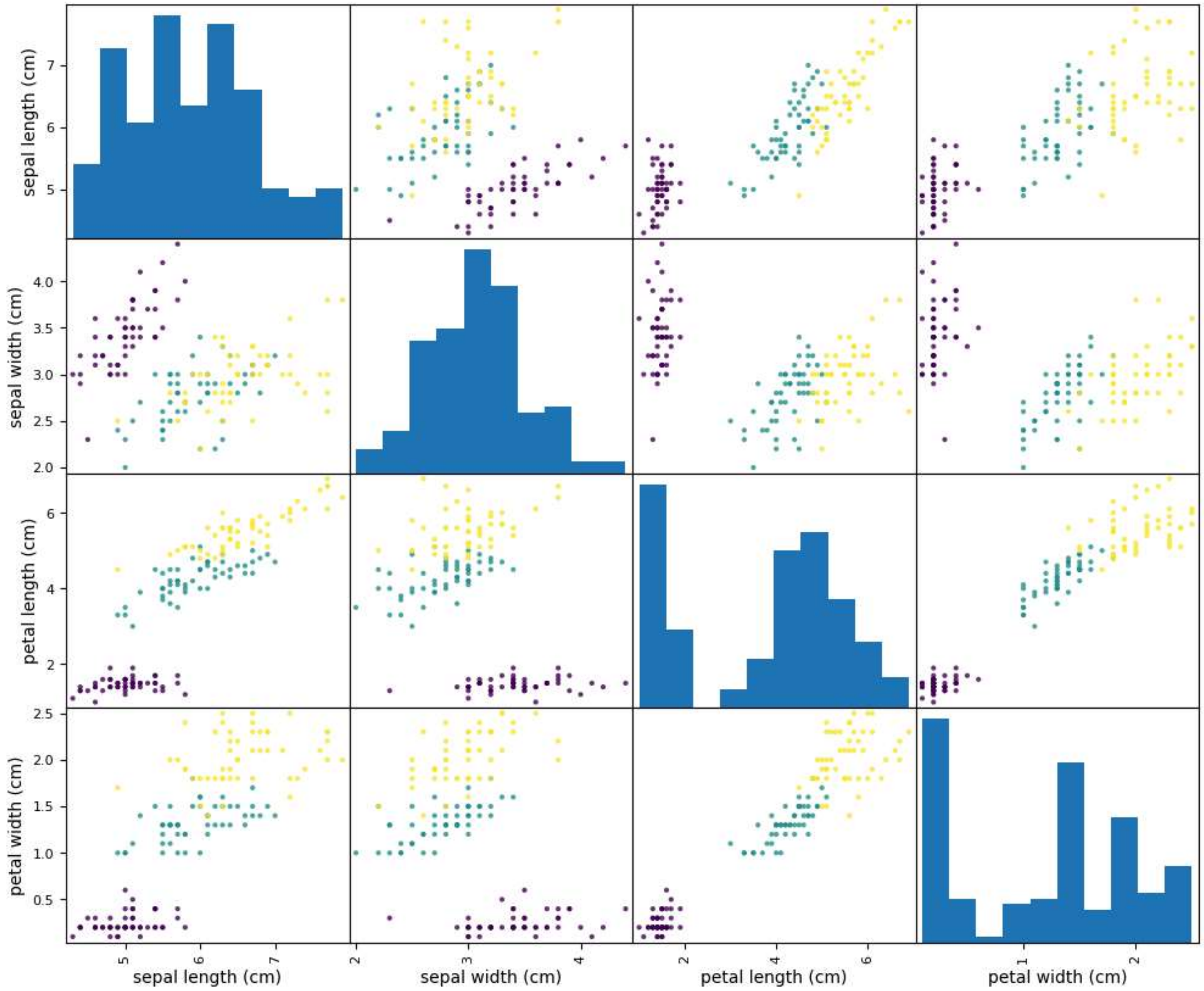
```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000   
mean            5.843333          3.057333           3.758000   
std             0.828066          0.435866           1.765298   
min             4.300000          2.000000           1.000000   
25%             5.100000          2.800000           1.600000   
50%             5.800000          3.000000           4.350000   
75%             6.400000          3.300000           5.100000   
max             7.900000          4.400000           6.900000   

       petal width (cm)      target  
count        150.000000  150.000000  
mean           1.199333    1.000000  
std            0.762238    0.819232  
min            0.100000    0.000000  
25%            0.300000    0.000000  
50%            1.300000    1.000000  
75%            1.800000    2.000000  
max            2.500000    2.000000  
```

Data Visualization

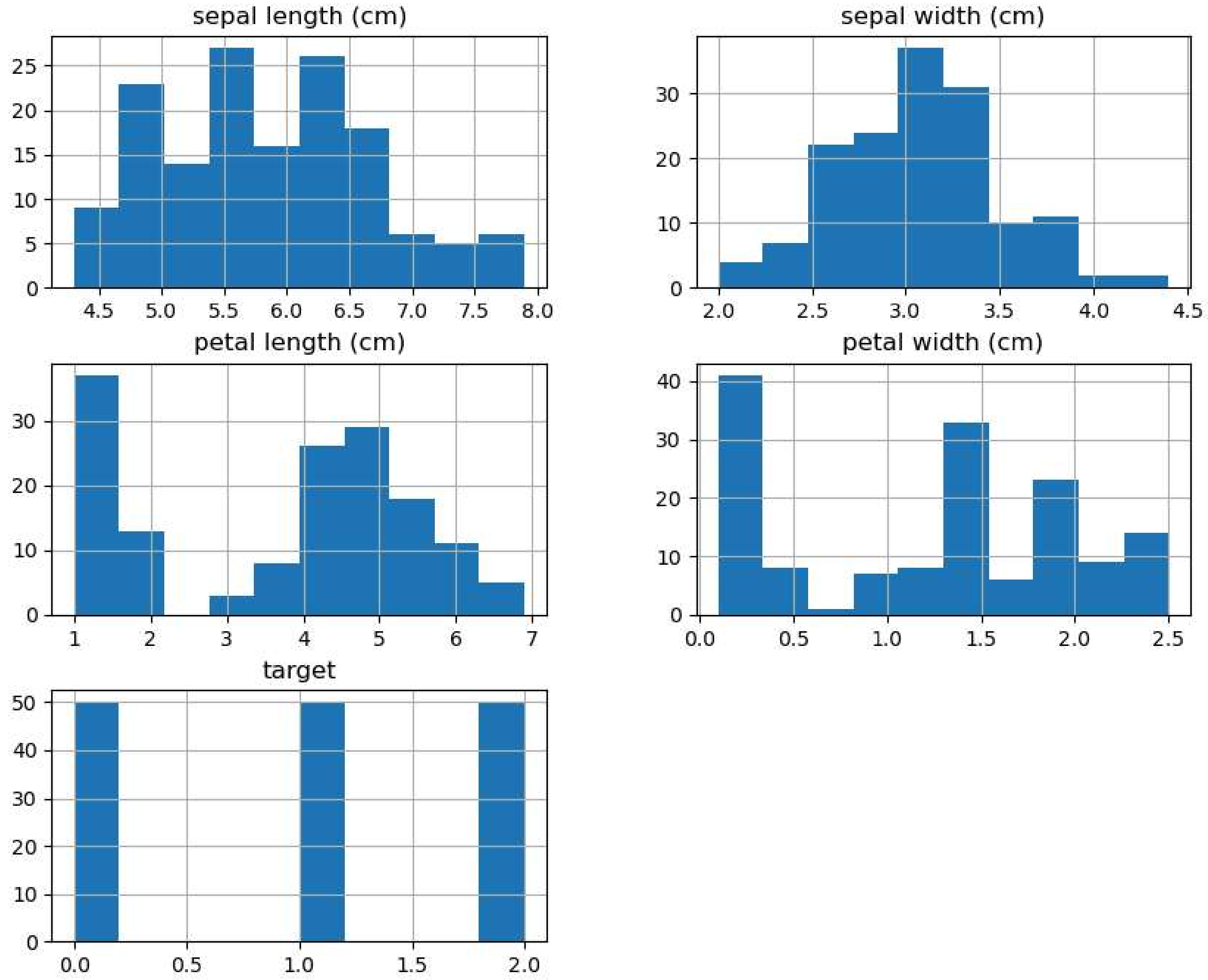
In [6]:

```
pd.plotting.scatter_matrix(iris_df[['sepal length (cm)', 'sepal width (cm)', 'petal le
plt.savefig("Graph1.png")
plt.show()
```



In [7]:

```
iris_df.hist(figsize=(10, 8))
plt.savefig("Graph2.png")
plt.show()
```



Data Preprocessing

In [8]:

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random
```

Data Modelling

In [9]:

```
# We'll use an SVM with a 'sigmoid' kernel for this example other kernels like 'poly',
model = SVC(kernel='sigmoid')
model.fit(X_train, y_train) # K-Fold Cross-Validation
kf = KFold(n_splits=10, shuffle=True, random_state=42)
cv_scores = []
```

Cross Validation

In [10]:

```
for train_index, test_index in kf.split(X_scaled):
    X_train_fold, X_test_fold = X_scaled[train_index], X_scaled[test_index]
    y_train_fold, y_test_fold = y[train_index], y[test_index]
    model.fit(X_train_fold, y_train_fold)
    y_pred = model.predict(X_test_fold)
    cv_scores.append(accuracy_score(y_test_fold, y_pred))
```

Model Evaluation

In [11]:

```
# Performance metrics
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
```

```
Accuracy: 0.9
Precision: 0.9013888888888889
Recall: 0.9
F1-score: 0.89923273657289
Cross-Validation Scores: [0.9333333333333333, 0.8666666666666667, 0.8, 0.866666666666
6667, 0.9333333333333333, 0.8666666666666667, 0.7333333333333333, 0.8666666666666667,
1.0, 0.8666666666666667]
```

Printing the results

In [12]:

```
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("Cross-Validation Scores:", cv_scores)
```

```
Accuracy: 0.9
Precision: 0.9013888888888889
Recall: 0.9
F1-score: 0.89923273657289
Cross-Validation Scores: [0.9333333333333333, 0.8666666666666667, 0.8, 0.866666666666
6667, 0.9333333333333333, 0.8666666666666667, 0.7333333333333333, 0.8666666666666667,
1.0, 0.8666666666666667]
```