SIA VASHIST

20190802107

In [58]:
```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
#from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

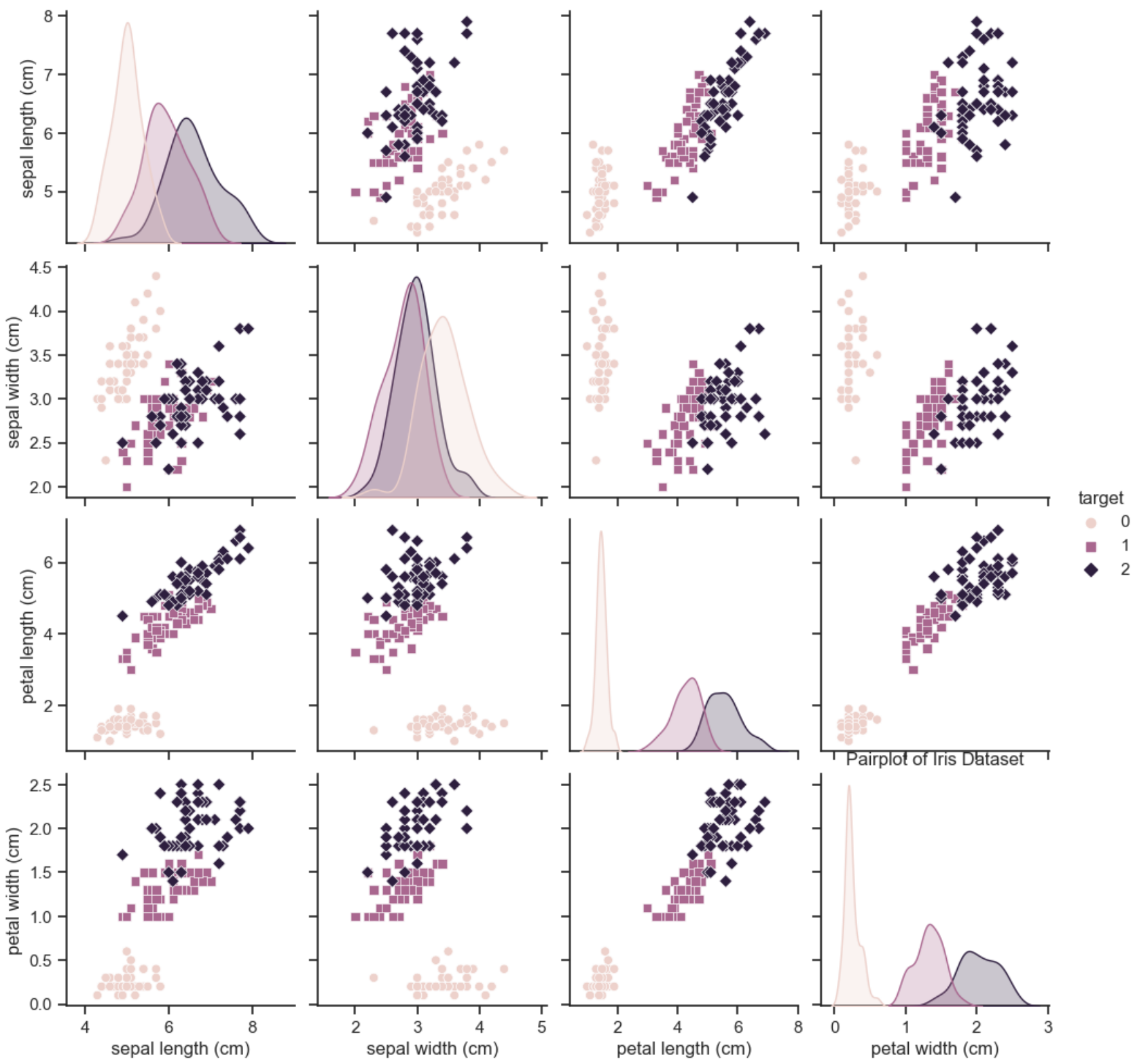
In [59]:
```
iris = load_iris()
data = iris.data
target = iris.target
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target

# Summary statistics
print("Summary Statistics:")
print(df.describe())
```

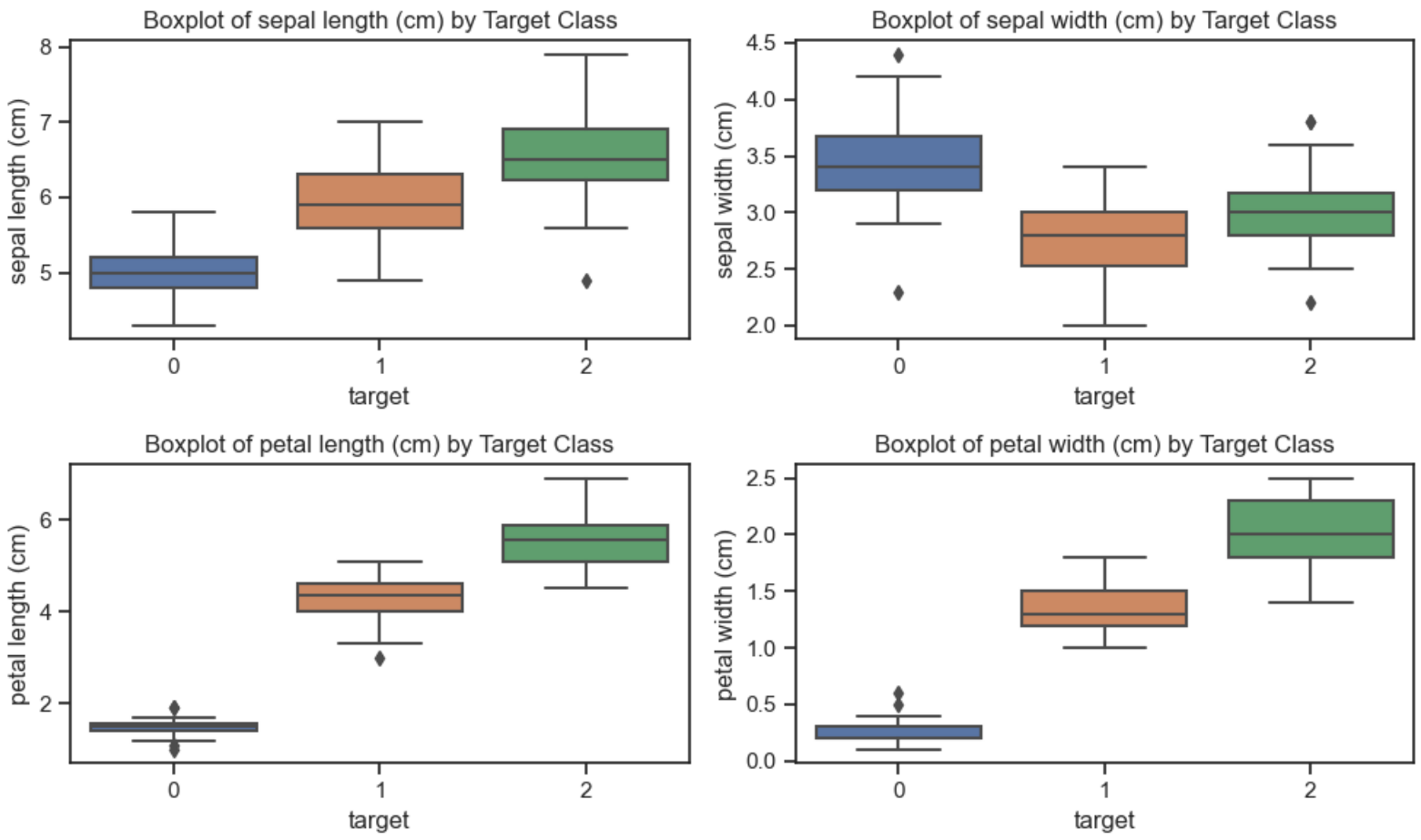
```
Summary Statistics:
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          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
```

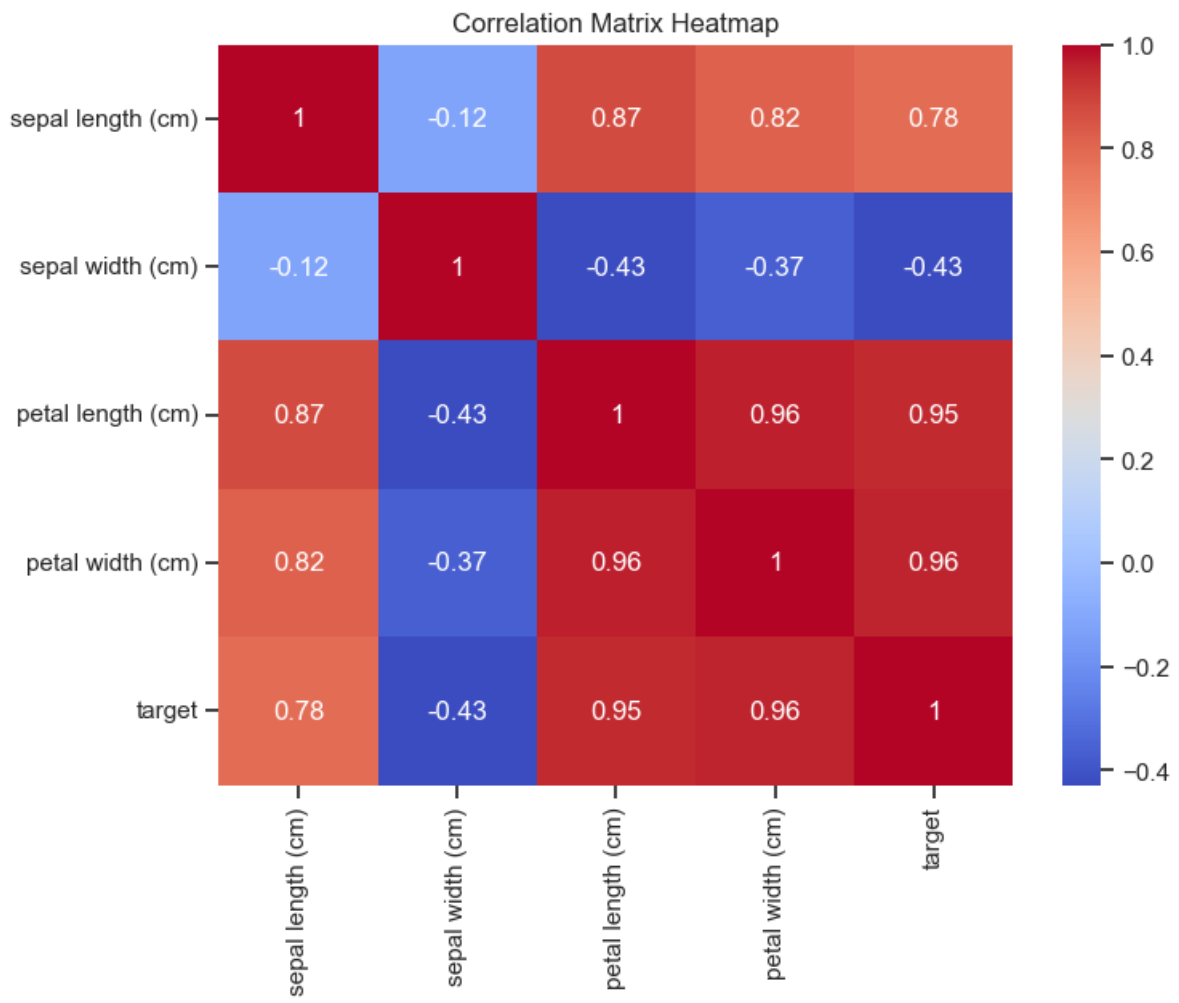
In [60]:
```
# Pairplot for visualizing relationships between features
sns.set(style="ticks")
sns.pairplot(df, hue="target", markers=["o", "s", "D"])
plt.title("Pairplot of Iris Dataset")
plt.show()
```



In [61]:
```
# Boxplot for each feature by target class
plt.figure(figsize=(10, 6))
for i, feature in enumerate(iris.feature_names):
    plt.subplot(2, 2, i + 1)
    sns.boxplot(x='target', y=feature, data=df)
    plt.title(f'Boxplot of {feature} by Target Class')
plt.tight_layout()
plt.show()
```



In [62]:
```
# Correlation matrix and heatmap
correlation_matrix = df.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix Heatmap")
plt.show()
```



Data Preprocessing:

- Feature Encoding: The Iris dataset is a standard dataset, and its features are already numeric, so there's no need for feature encoding.
- Feature Scaling: You can use StandardScaler from scikit-learn to scale your features.

Feature Encoding:

Task-3: Feature Scaling

Z-score Standardization:

In [63]:
```
from pandas.core.frame import DataFrame
scaler = StandardScaler()
data_scaled = scaler.fit_transform(df.iloc[: , :4])
z_score = pd.DataFrame(data_scaled, columns = iris.feature_names)
z_score
```

Out[63]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444
...	...	...	...	...
145	1.038005	-0.131979	0.819596	1.448832
146	0.553333	-1.282963	0.705921	0.922303
147	0.795669	-0.131979	0.819596	1.053935
148	0.432165	0.788808	0.933271	1.448832
149	0.068662	-0.131979	0.762758	0.790671

150 rows × 4 columns

Task-4: Feature Selection

In [64]:
```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

data = load_iris()
X = data.data
y = data.target

estimator = LogisticRegression(solver="liblinear")

num_features_to_select = 2
rfe = RFE(estimator, n_features_to_select=num_features_to_select)

rfe.fit(X, y)

selected_features = [data.feature_names[i] for i in range(len(rfe.support_)) if rfe.support_[i]]

print("Selected Features:")
for feature in selected_features:
    print("\t"+feature)
```

```
Selected Features:
        sepal width (cm)
        petal width (cm)
```

In [65]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier()
model.fit(X_train, y_train)

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')
```

In [67]:
```
print("Model Accuracy:", accuracy * 100)
print("Model Precision:", precision * 100)
print("Model Recall:", recall * 100)
print("Model F1 Score:", f1* 100)
```

```
Model Accuracy: 100.0
Model Precision: 100.0
Model Recall: 100.0
Model F1 Score: 100.0
```