10190900042 IRIS Dataset

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import random as rd
from sklearn import datasets
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

from warnings import filterwarnings
filterwarnings('ignore')


# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
target = iris.target


# Convert the NumPy array to a DataFrame
df = pd.DataFrame(X, columns=iris.feature_names)
df['target'] = target


print(df.head(5))
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
```

```
# Display general information about the dataset
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
```

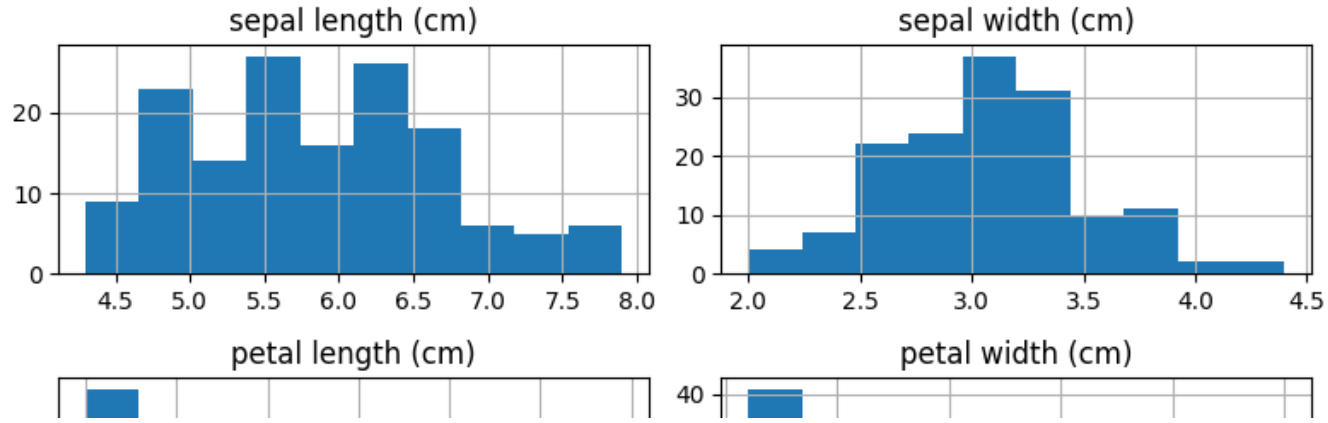
```
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   target             150 non-null    int64
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
None
```

```
# View summary statistics of the dataset
print(df.describe())
```

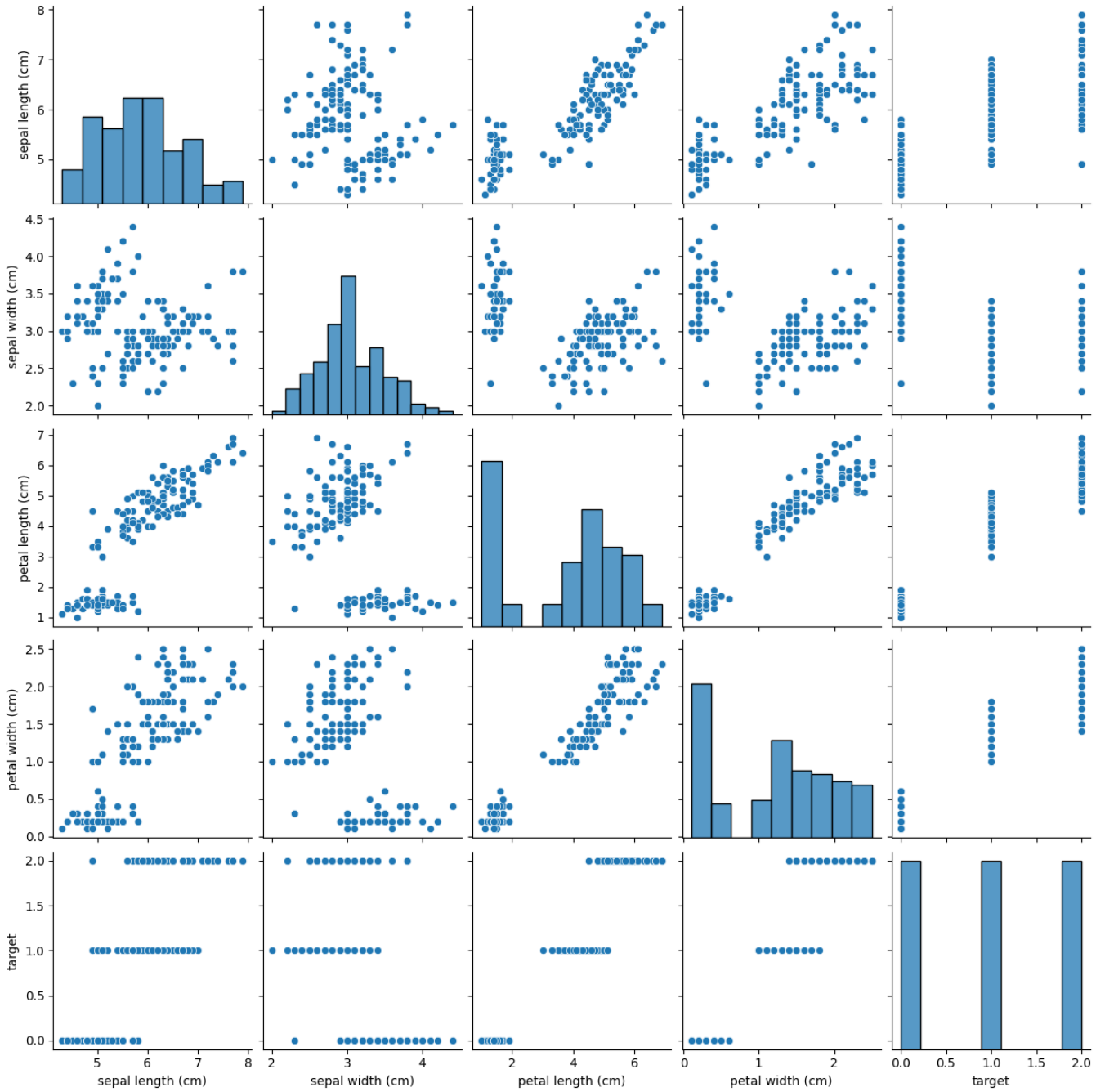
```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           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
```

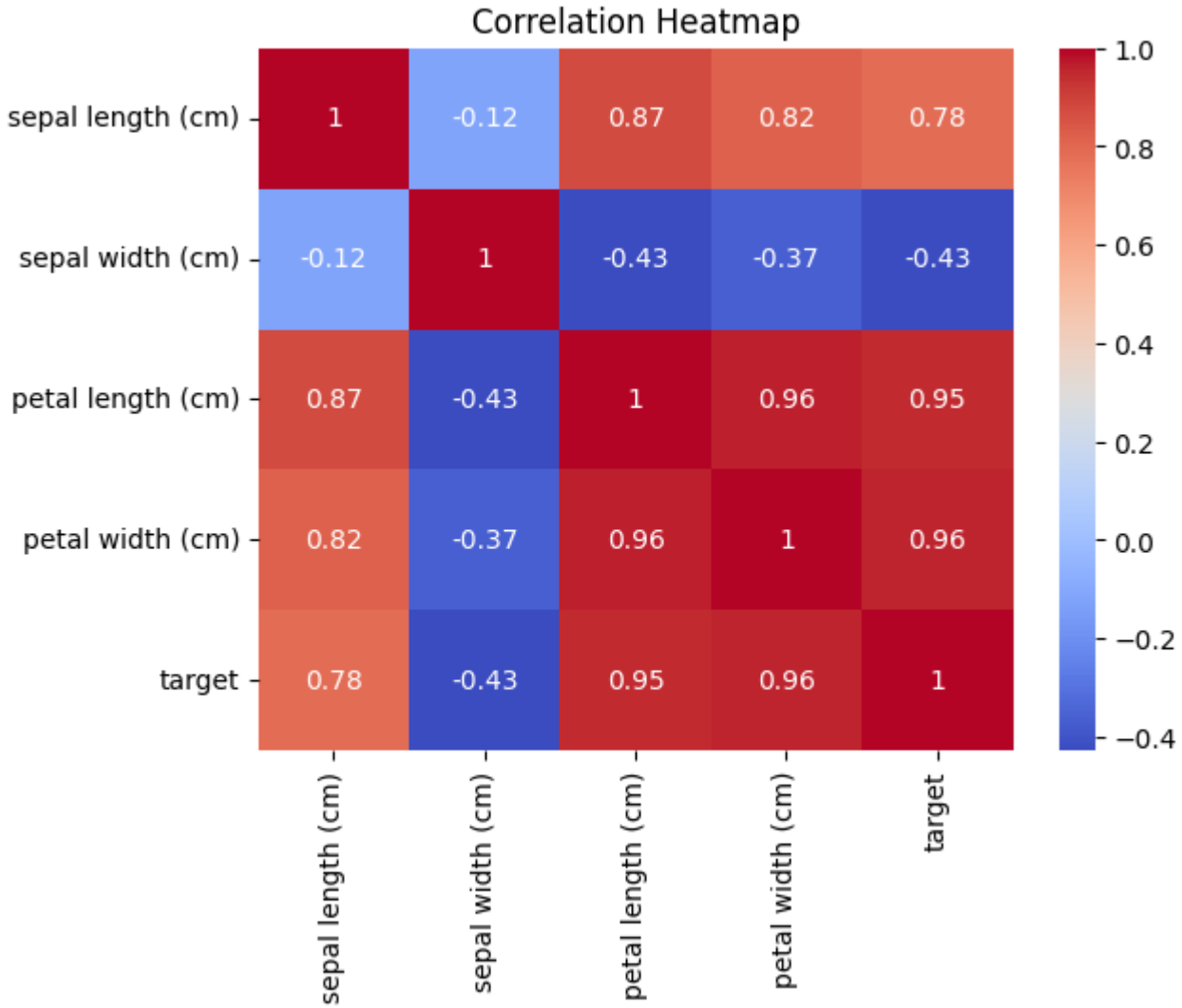
```
# Visualize the distribution of each feature using histograms
df.hist(figsize=(8, 6), bins=10)
plt.tight_layout()
plt.show()
```



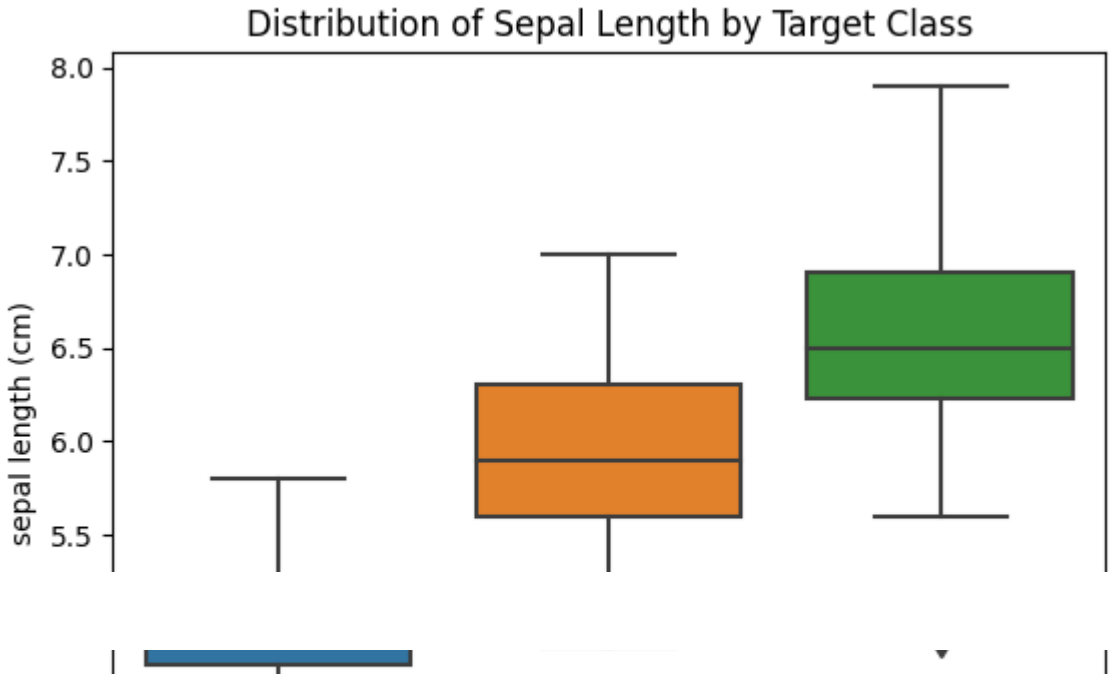

```
# Visualize the relationships between features using a pair plot
sns.pairplot(df)
plt.show()
```



```
# Calculate and visualize the correlation between features
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```




```
# Visualize the distribution of each feature by target class using box plots
sns.boxplot(data=df, x='target', y='sepal length (cm)')
plt.title('Distribution of Sepal Length by Target Class')
plt.show()
```



USING CLASSIFICATION ALGORITHMS FOR DATA MODELLING

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, silhc
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC


# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target


print(df.head())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
logreg_accuracy = accuracy_score(y_test, y_pred)
print("Logistic Regression Accuracy:", logreg_accuracy)
```

```
Logistic Regression Accuracy: 1.0
```

```
# Decision Tree Classifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
y_pred = dtree.predict(X_test)
dtree_accuracy = accuracy_score(y_test, y_pred)
print("Decision Tree Accuracy:", dtree_accuracy)
```

```
Decision Tree Accuracy: 1.0
```