load dataset

```
import pandas as pd

# Load the dataset
data = pd.read_csv('Iris.csv')
```

display first 5 rows

```
print("First few rows of the dataset:")
print(data.head())
```

```
First few rows of the dataset:
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

```
# Check the shape of the dataset
print("\nShape of the dataset:", data.shape)
```

```

Shape of the dataset: (150, 6)
```

```
# Get basic information about the dataset
print("\nDataset info:")
print(data.info())
```

```

Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

checking for missing values

```
# Check for missing values
print("\nMissing values in each column:")
print(data.isnull().sum())
```

```
Missing values in each column:
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

summary statistics

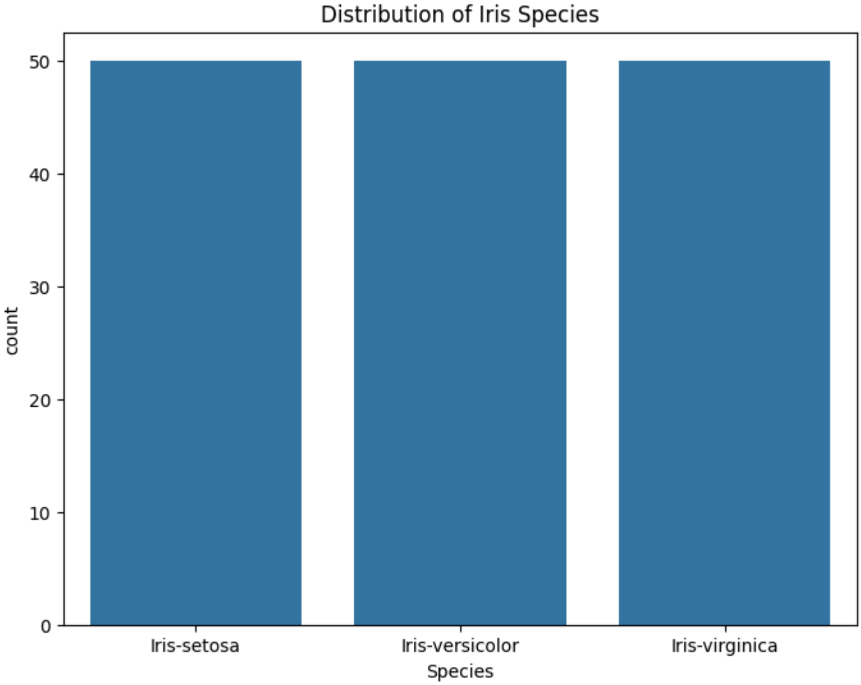
```
# Summary statistics
print("\nSummary statistics:")
print(data.describe())
```

```
Summary statistics:
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  150.000000     150.000000    150.000000     150.000000    150.000000
mean    75.500000       5.843333      3.054000       3.758667      1.198667
std     43.445368       0.828066      0.433594       1.764420      0.763161
min      1.000000       4.300000      2.000000       1.000000      0.100000
25%     38.250000       5.100000      2.800000       1.600000      0.300000
50%     75.500000       5.800000      3.000000       4.350000      1.300000
75%    112.750000       6.400000      3.300000       5.100000      1.800000
max    150.000000       7.900000      4.400000       6.900000      2.500000
```

Visualize the distribution of species

```
import seaborn as sns
import matplotlib.pyplot as plt  # Import matplotlib.pyplot

# Visualize the distribution of species
plt.figure(figsize=(8, 6))
sns.countplot(x='Species', data=data)
plt.title('Distribution of Iris Species')
plt.show()
```



```
# Drop the 'Id' column as it's not useful for classification
data = data.drop(columns=['Id'])
```

```
# Import the LabelEncoder class
from sklearn.preprocessing import LabelEncoder

# Encode the 'Species' column into numerical values
label_encoder = LabelEncoder()
data['Species'] = label_encoder.fit_transform(data['Species'])
```

```
# Split the dataset into features (X) and target (y)
X = data.drop(columns=['Species'])
y = data['Species']
```

```
# Import the necessary function
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

```
print("\nTraining set size:", X_train.shape)
```

```
Training set size: (120, 4)
```

```
print("Testing set size:", X_test.shape)
```

```
Testing set size: (30, 4)
```

train model

```
# Import the necessary class
from sklearn.ensemble import RandomForestClassifier

# Initialize the Random Forest Classifier
model = RandomForestClassifier(random_state=42)

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)
```

```
# Import the necessary class
from sklearn.ensemble import RandomForestClassifier
# Import accuracy_score
from sklearn.metrics import accuracy_score

# Initialize the Random Forest Classifier
model = RandomForestClassifier(random_state=42)

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy of the model:", round(accuracy * 100, 2), "%")
```

```

Accuracy of the model: 100.0 %
```

```
# Import the necessary class
from sklearn.ensemble import RandomForestClassifier
# Import accuracy_score
from sklearn.metrics import accuracy_score
# Import classification_report
from sklearn.metrics import classification_report # Importing the classification_repor

# Initialize the Random Forest Classifier
model = RandomForestClassifier(random_state=42)

# Train the model on the training data
```

```
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy of the model:", round(accuracy * 100, 2), "%")

# Print the classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))
```

```
Accuracy of the model: 100.0 %

Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

```
# Import the necessary class
from sklearn.ensemble import RandomForestClassifier
# Import accuracy_score
from sklearn.metrics import accuracy_score
# Import classification_report
from sklearn.metrics import classification_report # Importing the classification_repo
# Import confusion_matrix
from sklearn.metrics import confusion_matrix # Importing the confusion_matrix functio

# Initialize the Random Forest Classifier
model = RandomForestClassifier(random_state=42)

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy of the model:", round(accuracy * 100, 2), "%")

# Print the classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
cm = confusion_matrix(y_test, y_pred)
```

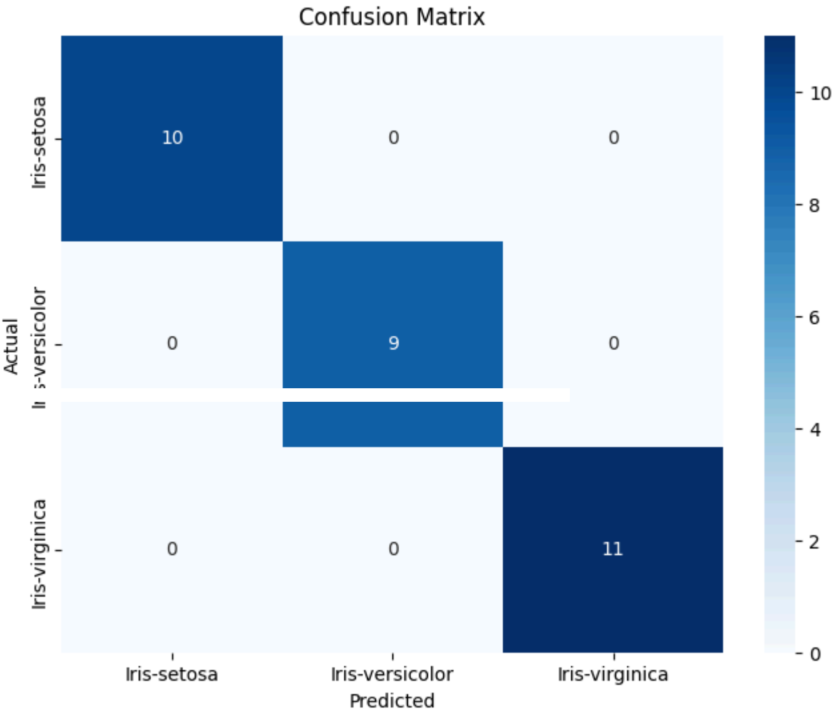
```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=label_encoder.classes_
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy of the model: 100.0 %

Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```


Confusion Matrix
Actual
Iris-setosa
Iris-versicolor
Iris-virginica
Predicted
Iris-setosa
Iris-versicolor
Iris-virginica