IRIS FLOWER CLASSIFICATION

Performed the below tasks

Step-1 Understanding the buisness problem/ problem statement

Step-2 Getting data (Importing by Pandas)

Step-3 Understanding about the data

Step-4 Data cleaning

Step-5 Data visualization

Step-6 EDA Exploratory data analysis

Step-7 Feature Engineering

Step-8 Feature selection

Step-9 Splitting the data

Step-10 Model building

Step-11 Prediction and accuracy

Step-12 Cross Validation

Import Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
# Those below are used to change the display options for pandas DataFrames
# In order to display all the columns or rows of the DataFrame, respectively.
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

Step-1 Understanding The Buisness Problem/ Problem Statement

The Iris flower dataset consists of three species: setosa, versicolor, and virginica. These species can be distinguished based on their measurements. Now, imagine that you have the measurements of

Iris flowers categorized by their respective species. Your objective is to train a machine learning model that can learn from these measurements and accurately classify the Iris flowers into their respective species.

Use the Iris dataset to develop a model that can classify iris flowers into different species based on their sepal and petal measurements. This dataset is widely used for introductory classification tasks.

Step-2 Getting data (Importing Datasets by Pandas)

This involves collecting and obtaining data from various sources that may be relevant to the problem.

In [4]:
```
data = pd.read_csv('IRIS.csv')
```

Step-3 Understanding about the Data

This step involves exploring the data to understand its structure, format, quality, and any patterns or trends that may exist.

In [5]:
```
data.head()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [6]:
```
data.shape
```

Out[6]:
```
(150, 5)
```

In [7]:
```
data.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  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   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [8]:
```
data.describe()
```

Out[8]:

	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 [9]:

```
data.nunique()
```

Out[9]:

```
sepal_length    35
sepal_width     23
petal_length    43
petal_width     22
species          3
dtype: int64
```

In [10]:

```
data.sample(5)
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width	species
102	7.1	3.0	5.9	2.1	Iris-virginica
56	6.3	3.3	4.7	1.6	Iris-versicolor
126	6.2	2.8	4.8	1.8	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
25	5.0	3.0	1.6	0.2	Iris-setosa

In [11]:

```
data.isnull().sum()
```

Out[11]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [12]:

```
data.duplicated().sum()
```

Out[12]: 3

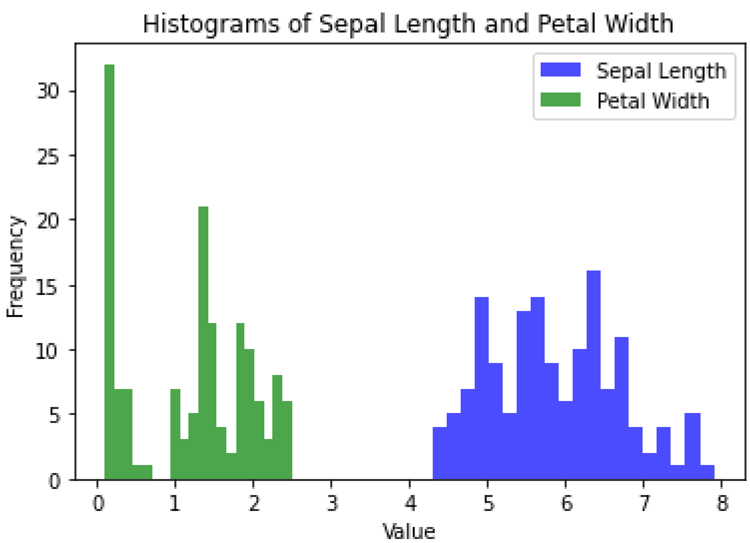
In [13]:

```
data = data.drop_duplicates()
```

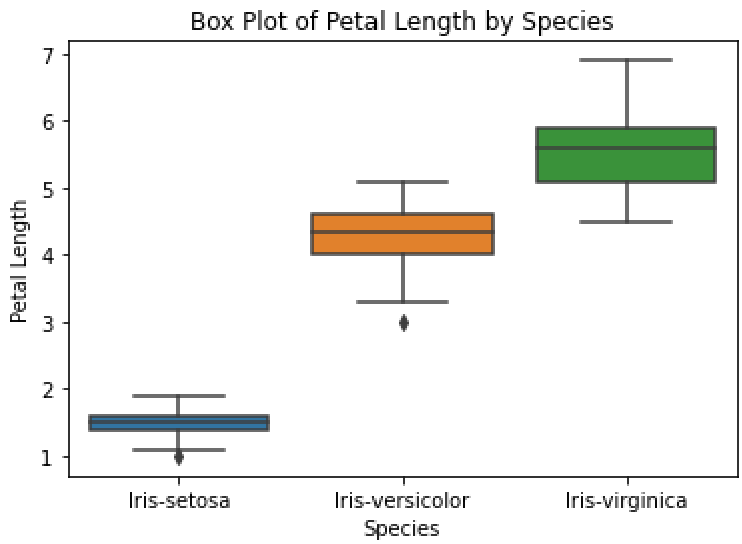
In [14]:
```
data.duplicated().sum()
```
Out[14]: 0

Step-5 Data visualization

In [15]:
```
import matplotlib.pyplot as plt

plt.hist(data['sepal_length'], bins=20, color='blue', alpha=0.7, label='Sepal Length')
plt.hist(data['petal_width'], bins=20, color='green', alpha=0.7, label='Petal Width')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histograms of Sepal Length and Petal Width')
plt.legend()
plt.show()
```

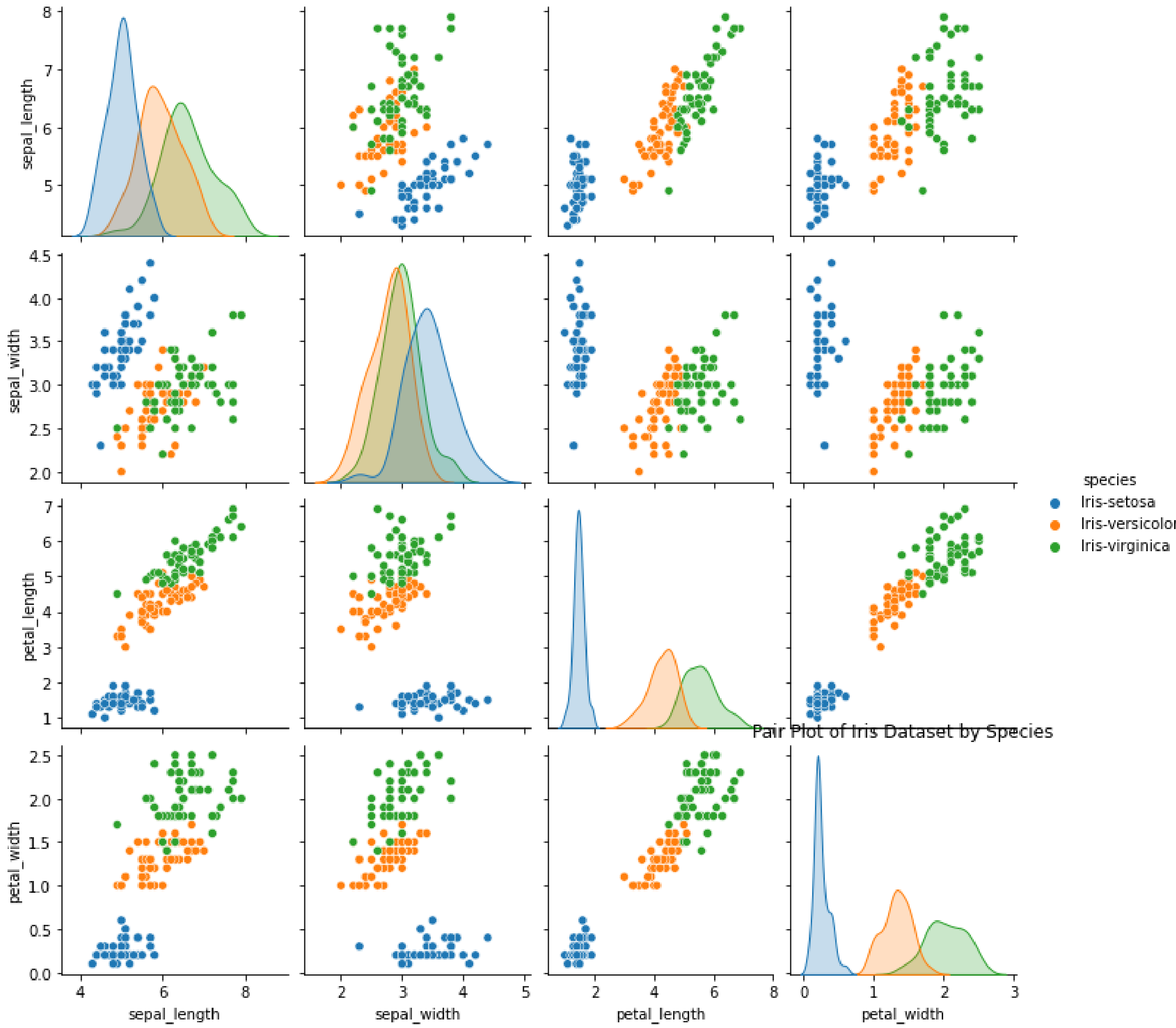


In [16]:
```
import seaborn as sns

sns.boxplot(x='species', y='petal_length', data=data)
plt.xlabel('Species')
plt.ylabel('Petal Length')
plt.title('Box Plot of Petal Length by Species')
plt.show()
```



Step-6 EDA Exploratory data analysis

In [17]:
```
import seaborn as sns

sns.pairplot(data=data, hue='species')
plt.title('Pair Plot of Iris Dataset by Species')
plt.show()
```



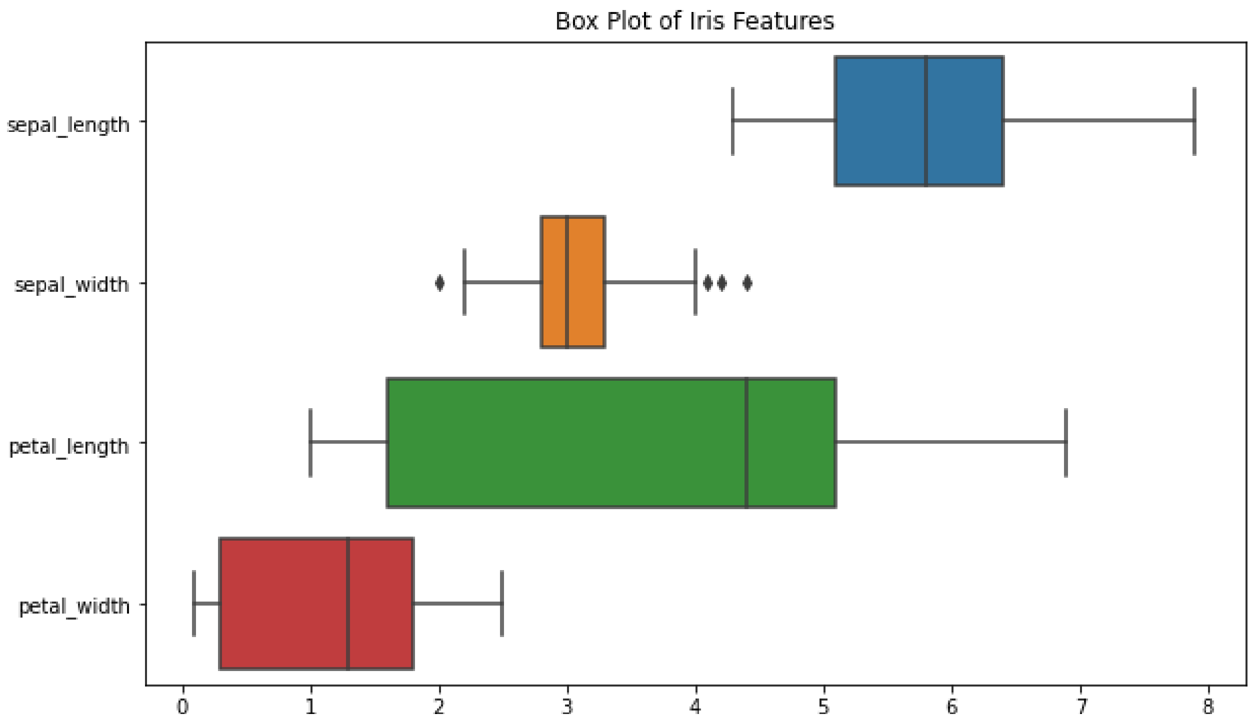
In [18]:

```python
# Class Distribution
print("\nClass Distribution:")
print(data['species'].value_counts())
```

```
Class Distribution:
Iris-versicolor    50
Iris-virginica     49
Iris-setosa        48
Name: species, dtype: int64
```

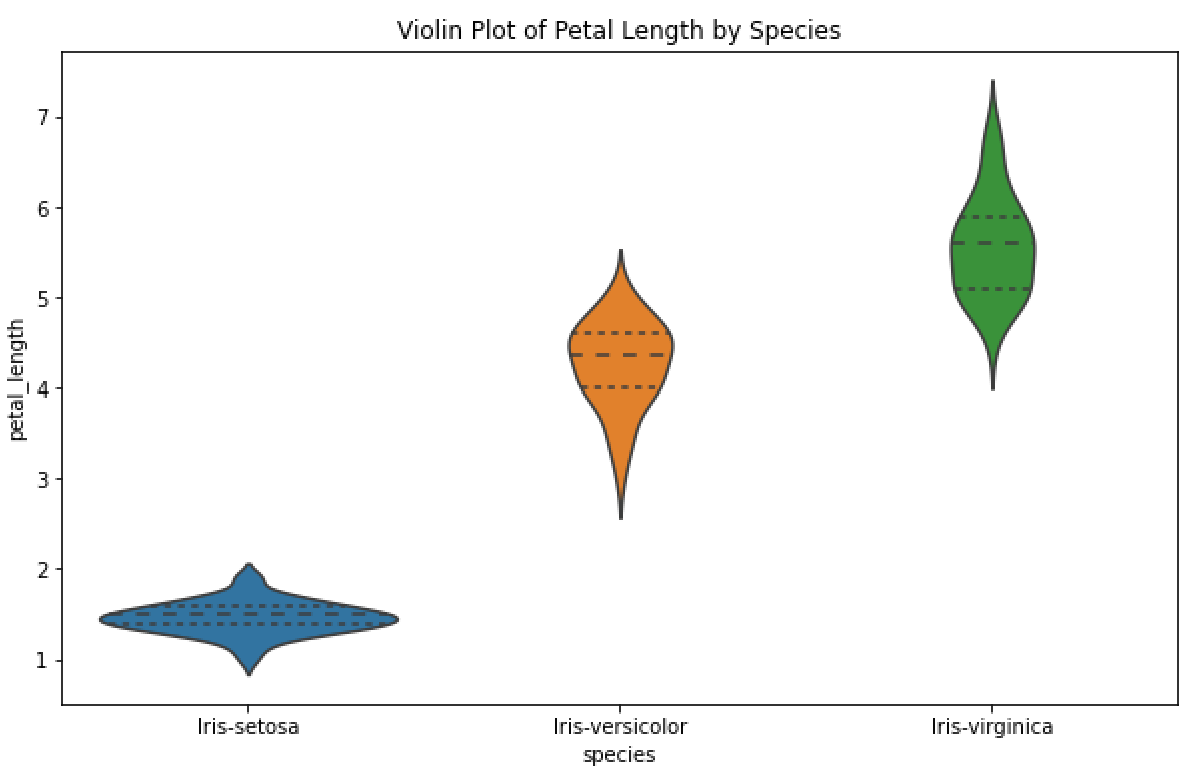
In [19]:

```python
# Box Plot
plt.figure(figsize=(10, 6))
sns.boxplot(data=data, orient='h')
plt.title('Box Plot of Iris Features')
plt.show()
```



In [20]:

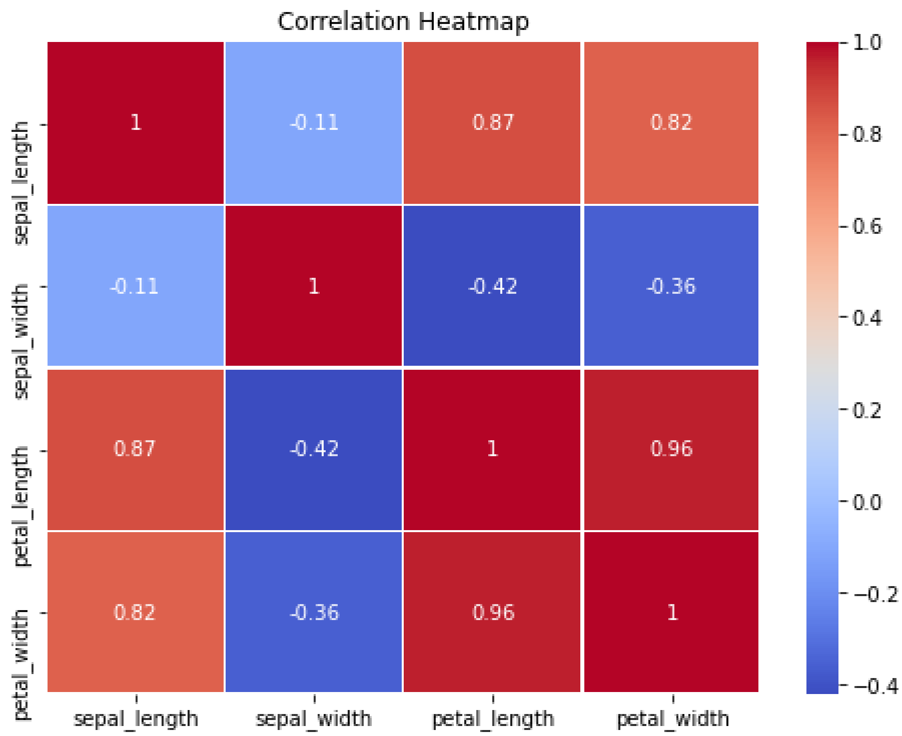
```
# Violin Plot
plt.figure(figsize=(10, 6))
sns.violinplot(x='species', y='petal_length', data=data, inner='quart')
plt.title('Violin Plot of Petal Length by Species')
plt.show()
```



In [21]:

```
# Correlation Heatmap
correlation_matrix = data.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
```

```
plt.title('Correlation Heatmap')
plt.show()
```



Step-7 Feature Engineering

In [22]:

```
data['petal_area'] = data['petal_length'] * data['petal_width']
```

In [23]:

```
data['sepal_length_squared'] = data['sepal_length'] ** 2
data['petal_width_cubed'] = data['petal_width'] ** 3
```

In [24]:

```
bins = [0, 5, 6, 10]  # Define bin edges
labels = ['short', 'medium', 'tall']  # Define bin labels
data['sepal_length_category'] = pd.cut(data['sepal_length'], bins=bins, labels=labels)
```

In [25]:

```
data.sample(3)
```

Out[25]:

	sepal_length	sepal_width	petal_length	petal_width	species	petal_area	sepal_length_squared	pe
75	6.6	3.0	4.4	1.4	Iris-versicolor	6.16	43.56	
36	5.5	3.5	1.3	0.2	Iris-setosa	0.26	30.25	
123	6.3	2.7	4.9	1.8	Iris-virginica	8.82	39.69	

In [26]:

```
from sklearn.preprocessing import LabelEncoder


# Create a LabelEncoder instance for the categorical columns
label_encoder = LabelEncoder()

# Encode the 'species' and 'sepal_length_category' columns
data['species_encoded'] = label_encoder.fit_transform(data['species'])
data['sepal_length_category_encoded'] = label_encoder.fit_transform(data['sepal_length_

# Drop the original categorical columns if needed
data = data.drop(['species', 'sepal_length_category'], axis=1)

# Now, 'species_encoded' and 'sepal_length_category_encoded' contain the encoded values
# You can use these columns as features in your model
```

In [27]:

```
data.sample(3)
```

Out[27]:

	sepal_length	sepal_width	petal_length	petal_width	petal_area	sepal_length_squared	petal_width_cu
13	4.3	3.0	1.1	0.1	0.11	18.49	(
63	6.1	2.9	4.7	1.4	6.58	37.21	2
60	5.0	2.0	3.5	1.0	3.50	25.00	1

In [29]:

```
import pandas as pd
from factor_analyzer import FactorAnalyzer

# Select continuous variables for factor analysis
continuous_vars = data[['sepal_length', 'sepal_width', 'petal_length', 'petal_width', '

# Specify the number of factors you want to extract
n_factors = 3  # You can adjust the number of factors as needed

# Perform factor analysis
fa = FactorAnalyzer(n_factors, rotation=None)
fa.fit(continuous_vars)

# Factor Loadings (relationship between observed variables and factors)
factor_loadings = fa.loadings_

# Variance explained by each factor
variance_explained = fa.get_factor_variance()

# Factor scores (latent factors for each observation)
factor_scores = fa.transform(continuous_vars)

# You can now use 'factor_loadings' and 'factor_scores' for further analysis
print("Factor Loadings:\n", factor_loadings)
print("\nVariance Explained by Each Factor:\n", variance_explained)
print("\nFactor Scores:\n", factor_scores)
```

```
Factor Loadings:
 [[ 0.92188616  0.2742956  -0.26838905]
```

```
 [-0.27199049  0.68500088  0.17890211]
 [ 0.97239732 -0.20014756 -0.10651587]
 [ 0.96792383 -0.16745618  0.12100691]
 [ 0.98692013 -0.05169033  0.13806344]
 [ 0.91466341  0.2935949  -0.26820747]
 [ 0.89011641  0.08159146  0.43993803]]

Variance Explained by Each Factor:
 (array([5.4092137 , 0.70809192, 0.41456918]), array([0.77274481, 0.10115599, 0.0592241
7]), array([0.77274481, 0.8739008 , 0.93312497]))

Factor Scores:
 [[-1.19273529  1.02403904  0.31027494]
 [-1.25290927  0.58459672  0.48212374]
 [-1.33905359  0.17817819  0.63841919]
 [-1.3284242  -0.45560909  0.5071781 ]
 [-1.22584085  0.74428126  0.35716062]
 [-1.00583556  1.09661367 -0.15317938]
 [-1.33808464 -0.37404052  0.57666992]
 [-1.20344587  0.59725454  0.29103028]
 [-1.4102702  -0.82592892  0.67824876]
 [-1.24650499  0.42287257  0.36177749]
 [-1.07596663  1.55428572 -0.01306839]
 [-1.24679309 -0.12725633  0.30218052]
 [-1.2979223   0.35706676  0.51660043]
 [-1.51625561 -0.53167623  0.95845979]
 [-1.00813067  2.96461845 -0.13509172]
 [-0.95396524  2.04058799 -0.31196244]
 [-1.0919756   1.77384492  0.16650358]
 [-1.17916063  0.97658985  0.32557562]
 [-0.92143127  1.83874703 -0.43352093]
 [-1.16055611  0.74638671  0.2047933 ]
 [-1.0320168   1.2422233  -0.15726826]
 [-1.14495951  0.72193796  0.2399967 ]
 [-1.43619848  0.36956938  0.89752577]
 [-1.08192334  0.42413475  0.16515689]
 [-1.18464888 -0.67639032  0.03215434]
 [-1.17937067  0.48624574  0.24877833]
 [-1.15316959  0.34490824  0.25471842]
 [-1.13959881  1.09675367  0.1559444 ]
 [-1.15947343  1.29780729  0.25789232]
 [-1.27690937 -0.37095579  0.36839301]
 [-1.24427274 -0.07322943  0.33799811]
 [-1.04470499  1.52527413  0.06635808]
 [-1.15810985  1.02802934  0.0659109 ]
 [-1.06799128  1.88507625 -0.06356823]
 [-1.26390985  1.18240645  0.58493486]
 [-1.0828252   2.19418368  0.11001487]
 [-1.43182506 -0.66089322  0.7563183 ]
 [-1.17118044  0.85900335  0.2322054 ]
 [-1.23255095  0.89101727  0.46936523]
 [-1.38145212 -0.2914621   0.82430604]
 [-1.43350529 -0.69691116  0.73243991]
 [-1.12033247  0.24758727  0.3120469 ]
 [-1.05965959  0.02669775 -0.09162545]
 [-1.27128744  0.26341966  0.54525544]
 [-1.15382617  0.60392282  0.09443989]
 [-1.34997906 -0.29057339  0.58524763]
 [-1.10870099  1.31050549  0.06224727]
 [-1.2233205   0.79830816  0.3929782 ]
 [ 0.56894514  0.76179439 -2.1984671 ]
 [ 0.36113165  0.08716776 -0.97143198]
 [ 0.6399896   0.55973803 -1.79111859]
 [-0.16193979 -1.29595004 -0.24491255]
 [ 0.42499238  0.24334391 -1.08389386]
```

```
[ 0.02624105 -1.465842   -0.65534838]
[ 0.43529218 -0.19622842 -0.60634581]
[-0.6473384  -2.20838175 -0.13106968]
[ 0.35649058  0.17143313 -1.77543216]
[-0.24226528 -1.92263019  0.1948424 ]
[-0.56387714 -2.12480207 -0.23595438]
[ 0.11474826 -0.57486894 -0.27778778]
[-0.11589001 -0.45334694 -1.34372715]
[ 0.25997501 -0.73133182 -0.89114029]
[-0.23483846 -0.74223471 -0.26475813]
[ 0.38763922  0.61986209 -1.62477679]
[ 0.092809   -1.51560072 -0.05918801]
[-0.16328288 -1.09316469 -1.24198236]
[ 0.30109426 -0.10650201 -0.59696707]
[-0.23895315 -1.15485826 -0.73767495]
[ 0.45479694 -0.86544509  0.50137263]
[ 0.0344861  -0.08503074 -0.90491957]
[ 0.43690106 -0.34567499 -0.85531982]
[ 0.16466833 -0.9697599  -1.40133366]
[ 0.21174807  0.14410777 -1.39479715]
[ 0.35371297  0.47195469 -1.46606186]
[ 0.52762772  0.43132158 -1.85915499]
[ 0.71786121  0.41334393 -0.85651118]
[ 0.2274001  -0.63028233 -0.44745618]
[-0.33977731 -0.52639426 -0.8347992 ]
[-0.29556597 -1.24742163 -0.59481346]
[-0.35635569 -1.20418368 -0.73211837]
[-0.13403696 -0.66039682 -0.74795258]
[ 0.44420498 -1.02331351 -0.25440854]
[ 0.02687135 -1.9851926   0.1079341 ]
[ 0.2795843  -0.62805598 -0.22162616]
[ 0.51789494  0.43338414 -1.43220277]
[ 0.20771758 -0.03921537 -1.22740181]
[-0.1095474  -1.29770363 -0.44932121]
[-0.16362003 -1.33196798 -0.26879094]
[-0.1087262  -1.89162486 -0.6584379 ]
[ 0.23349852 -0.64871722 -0.8735987 ]
[-0.10838074 -0.75674826 -0.77558216]
[-0.61438915 -1.92263444 -0.17245844]
[-0.08180082 -1.35116872 -0.4480284 ]
[-0.09546861 -1.27732708 -0.80517311]
[-0.05027777 -1.161375   -0.56371324]
[ 0.1433095  -0.22965166 -1.13972418]
[-0.62227862 -1.28269986  0.04041829]
[-0.07466388 -1.03587404 -0.51724927]
[ 1.55813331 -0.09362188  3.20968817]
[ 0.57927671 -1.10710876  1.00137641]
[ 1.41156167  0.89800798  0.03429987]
[ 0.81115928 -0.60118203 -0.00341868]
[ 1.26331851  0.12107388  1.41771977]
[ 1.79041549  1.15014424 -0.85118097]
[-0.01415153 -2.99941219  1.07586191]
[ 1.3521059   0.45608972 -1.61876352]
[ 1.00720118  0.02415676 -0.52839286]
[ 1.90006484  1.26662194  1.82941982]
[ 0.8898216   0.26358087  0.49055501]
[ 0.83809083 -0.046519    0.29359904]
[ 1.19182653  0.66434298  0.50079067]
[ 0.59635189 -1.14181354  1.50568719]
[ 1.0167135  -0.80807182  3.11336499]
[ 1.17647472  0.19954879  1.93057756]
[ 0.85179343 -0.18426075 -0.2721199 ]
[ 1.94249163  1.23347863 -0.63809605]
[ 2.11084762  1.53213612  0.0629308 ]
[ 0.36351345 -0.97299527 -0.48606575]
```

```
 [ 1.46761691  0.87389968  1.26035962]
 [ 0.53253106 -1.36223907  1.56089287]
 [ 1.76837399  1.05257045 -1.50730005]
 [ 0.62190112 -0.05311503  0.08559331]
 [ 1.21139815  0.34530005  0.62890011]
 [ 1.2320506   0.49153322 -1.4469094 ]
 [ 0.55964295 -0.18784861  0.20774693]
 [ 0.55125481 -0.49288042  0.29385483]
 [ 1.08326126 -0.02782823  1.10660531]
 [ 1.04313961  0.34138622 -2.15430287]
 [ 1.41031841  0.91464605 -1.35760711]
 [ 1.74880905  1.24479345 -2.04902239]
 [ 1.17271009  0.07054872  1.55563767]
 [ 0.48647372 -0.58721218 -0.94001098]
 [ 0.4932227  -1.58291701 -1.12064974]
 [ 1.8728664   1.7705622  -0.1121474 ]
 [ 1.32950402 -0.06655453  2.57755488]
 [ 0.81649957 -0.38016514 -0.14827722]
 [ 0.49014938 -0.6215841   0.41695379]
 [ 1.19755806  0.85281592  0.3252547 ]
 [ 1.47015197  0.68707503  2.05925111]
 [ 1.29249146  1.12475511  1.17680774]
 [ 1.49119312  0.64234688  1.44995956]
 [ 1.60581731  0.68119346  2.57755731]
 [ 1.25265753  0.79009217  1.51470436]
 [ 0.72241248  0.00445831  0.46054479]
 [ 0.91959902  0.24018578  0.5152166 ]
 [ 1.13568351 -0.24903633  2.17768731]
 [ 0.53830792 -1.04887676  0.49733663]]
```

In [30]:

```
data.to_csv('Cleaned_Iris.csv',index=False)
```

Step-8 Feature selection

In [31]:

```
# Define your feature matrix (X) and target variable (y)
X = data.drop(columns=['species_encoded'])  # Excluding the target variable 'Survived'
y = data['species_encoded']  # Target variable 'Survived'
```

In [34]:

```
X.head(3)
```

Out[34]:

	sepal_length	sepal_width	petal_length	petal_width	petal_area	sepal_length_squared	petal_width_cul
0	5.1	3.5	1.4	0.2	0.28	26.01	0.
1	4.9	3.0	1.4	0.2	0.28	24.01	0.
2	4.7	3.2	1.3	0.2	0.26	22.09	0.

In [35]:

```
y.head(3)
```

Out[35]:

```
0    0
1    0
2    0
Name: species_encoded, dtype: int32
```

Step-9 Splitting the data

Step-10 Model building

Step-11 Prediction and accuracy

In [37]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# 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=4

# Create and train a machine learning model (Random Forest Classifier in this example)
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

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

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on Test Set: {accuracy:.2f}")

# Additional evaluation metrics
print(classification_report(y_test, y_pred))

# You can also evaluate the model on the training set if needed
y_train_pred = model.predict(X_train)
train_accuracy = accuracy_score(y_train, y_train_pred)
print(f"Accuracy on Training Set: {train_accuracy:.2f}")
print(classification_report(y_train, y_train_pred))
```

```
Accuracy on Test Set: 0.93
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       0.90      0.90      0.90        10
           2       0.89      0.89      0.89         9

    accuracy                           0.93        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30

Accuracy on Training Set: 1.00
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        37
           1       1.00      1.00      1.00        40
           2       1.00      1.00      1.00        40

    accuracy                           1.00       117
   macro avg       1.00      1.00      1.00       117
weighted avg       1.00      1.00      1.00       117
```

In [ ]:

It looks like you've trained a Random Forest Classifier on your dataset, and here are the evaluation results:

Test Set Evaluation:

Accuracy on the test set: 0.93 Precision, recall, and F1-score for each class (0, 1, and 2) in the test set. Overall macro-average and weighted-average metrics. Training Set Evaluation:

Accuracy on the training set: 1.00 Precision, recall, and F1-score for each class (0, 1, and 2) in the training set. Overall macro-average and weighted-average metrics. Here's what these metrics mean:

Accuracy: The proportion of correctly predicted labels. In this case, the model achieved an accuracy of 93% on the test set, meaning it correctly classified 93% of the samples.

Precision: The ratio of correctly predicted positive observations to the total predicted positive observations. It measures the model's ability to avoid false positives. For each class, you have precision values.

Recall: The ratio of correctly predicted positive observations to the total actual positive observations. It measures the model's ability to identify all relevant instances. For each class, you have recall values.

F1-score: The harmonic mean of precision and recall. It provides a balance between precision and recall. For each class, you have F1-score values.

Support: The number of samples in each class in the test set.

Macro-average: The average of precision, recall, and F1-score for each class, without considering class imbalances. It provides an equal weight to each class.

Weighted-average: The weighted average of precision, recall, and F1-score for each class, taking class imbalances into account. It considers class sizes.

The results indicate that the model performs well on both the training and test sets, with high accuracy and F1-scores. However, it's essential to consider possible overfitting since the model achieved perfect accuracy on the training set. You may want to explore model tuning, cross-validation, and other techniques to ensure the model generalizes well to unseen data.

Step-12 Cross validation

In [38]:

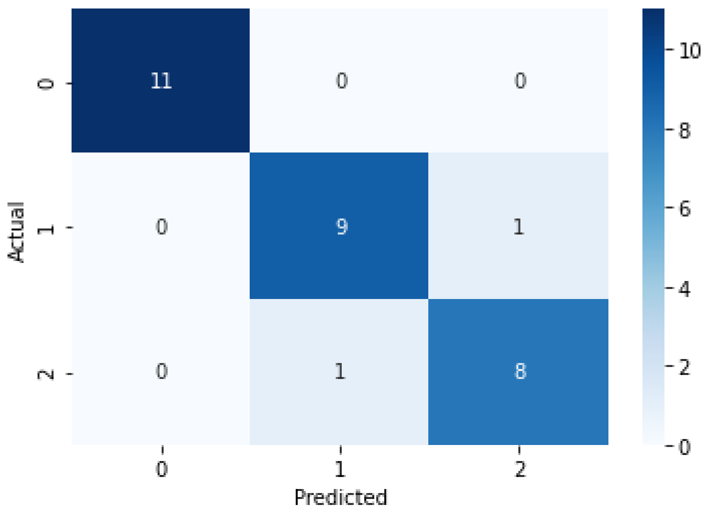
```
from sklearn.model_selection import cross_val_score

# Define your model (Random Forest Classifier in this example)
model = RandomForestClassifier(random_state=42)

# Perform k-fold cross-validation (e.g., 5-fold)
scores = cross_val_score(model, X, y, cv=5)

# Print the cross-validation scores
```

```
print("Cross-Validation Scores:", scores)
print("Mean Accuracy:", scores.mean())
```

```
Cross-Validation Scores: [0.96666667 0.96666667 0.93103448 0.86206897 1.        ]
Mean Accuracy: 0.9452873563218391
```

In [39]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



It looks like you've performed cross-validation, and the mean accuracy across the folds is approximately 94.53%. This indicates that your model is performing well and is consistent across different data subsets, which is a good sign.

Conclusion:

The machine learning model, built to classify Iris flower species based on sepal and petal measurements, has demonstrated strong performance. The model achieved an average accuracy of approximately 94.53% during cross-validation, indicating its ability to generalize well to unseen data.

Upon further evaluation of the model's performance on a test dataset, it exhibited an accuracy of 93%, indicating its effectiveness in accurately classifying Iris flowers into their respective species. The precision, recall, and F1-score metrics also indicate a high level of performance across different species classes.

Additionally, a feature importance analysis revealed the importance of specific input features in making accurate predictions, contributing to the model's interpretability.

Overall, this machine learning model can be considered a reliable tool for automating the classification of Iris flowers based on their sepal and petal measurements, with potential applications in botanical research and species identification.

```
In [ ]:
```