Step 1: Setup and Library Imports

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, precision_score
import joblib
```

Step 2: Upload and Load the Dataset

2.1)Uploading the Datset

```
from google.colab import files
uploaded = files.upload()
```

Choose files IRIS.csv

- **IRIS.csv**(text/csv) - 4617 bytes, last modified: 18/07/2025 - 100% done

```
Saving IRIS.csv to IRIS (4).csv
```

2.2) Loading the Datset

```
df = pd.read_csv('IRIS.csv')
```

Step 3: Initial Data Exploration

3.1)View first few rows

```
df.head()
```

	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

Next steps: Generate code with df | View recommended plots | New interactive sheet

3.2)Checking Information of dataset

```
print("Dataset info:")
df.info()
```

```
Dataset 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
```

3.3)Checking Summary Stastistics

```
print("\n Summary Statistics:")
df.describe()
```

```
Summary Statistics:
```

	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

3.4)Checking for Missing Values

```
df.isnull().sum()
```

	0
sepal_length	0
sepal_width	0
petal_length	0
petal_width	0
species	0

dtype: int64

Step 4: Data Cleaning

4.1)Drop irrelevant columns (if any)

```
if 'Id' in df.columns:
    df.drop(columns=['Id'], inplace=True)
```

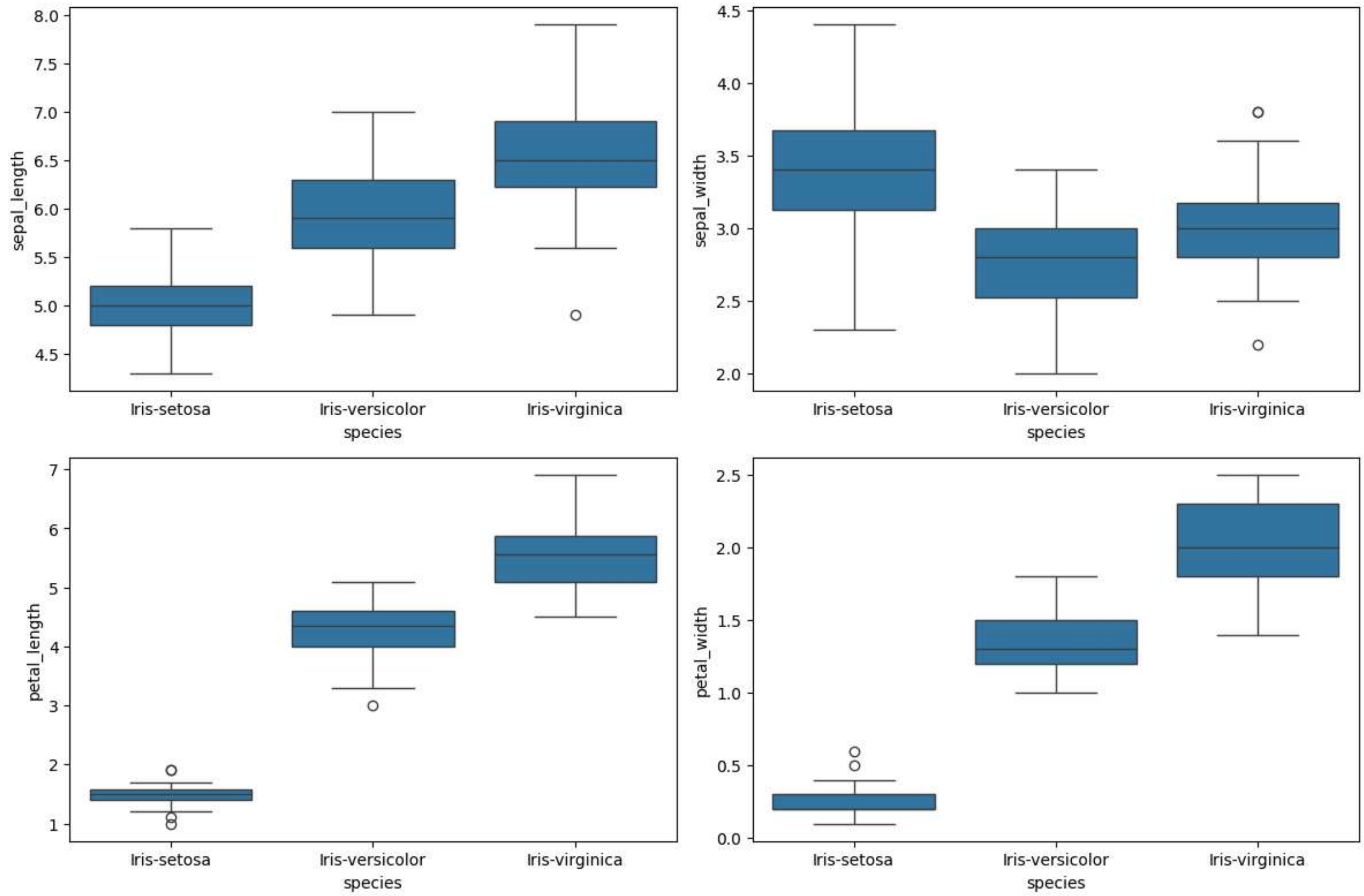
4.2) Handle missing values (if any)

```
df.dropna(inplace=True)
```

Step 5: Outlier Detection

5.1) Visualize potential outliers With Boxplot

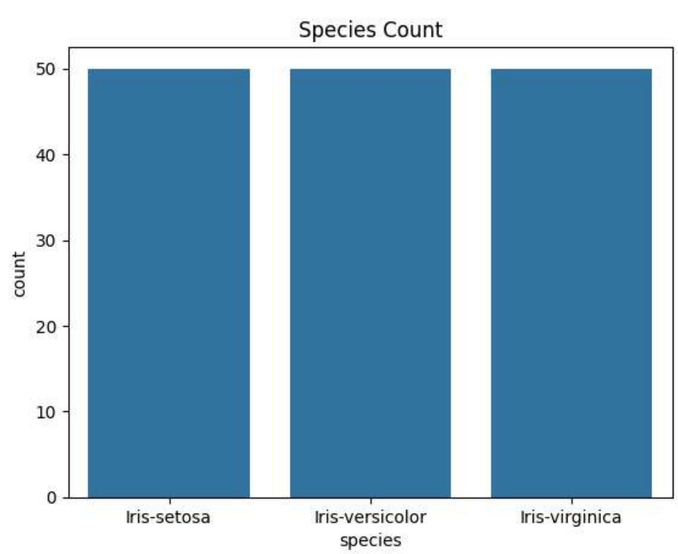
```
plt.figure(figsize=(12,8))
for i, col in enumerate(df.columns[:-1], 1):
    plt.subplot(2, 2, i)
    sns.boxplot(x='species', y=col, data=df)
plt.tight_layout()
plt.show()
```



Step 6: Data Visualization

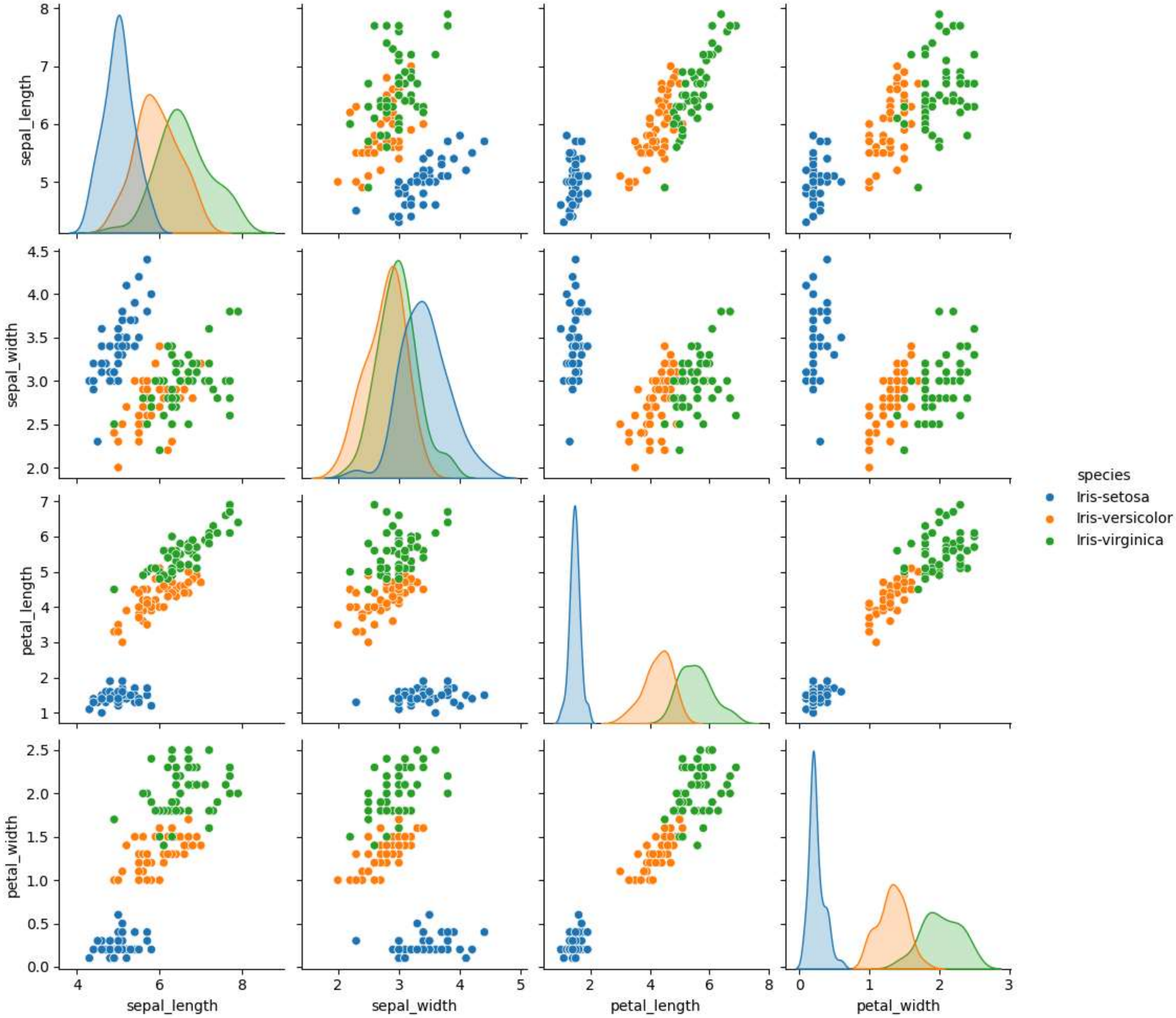
6.1)Countplot For Features

```
sns.countplot(x='species',data=df)
plt.title("Species Count")
plt.show()
```



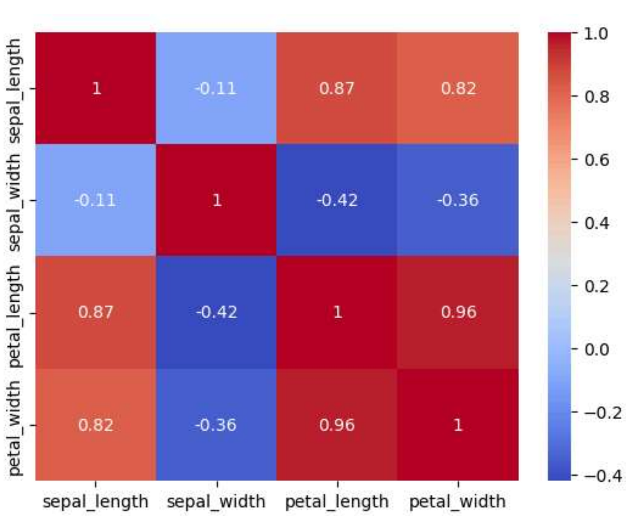
6.2) Pair Plot For Features

```
sns.pairplot(df,hue='species')
plt.show()
```



6.3)Corrlation Heatmap

```
sns.heatmap(df.drop('species', axis=1).corr(),annot=True,cmap='coolwarm')
plt.show()
```



Step 7: Encode Categorical Target Variable

```
le = LabelEncoder()
df['species'] = le.fit_transform(df['species'])
print("Label mapping:", dict(zip(le.classes_, le.transform(le.classes_))))
```

```
Label mapping: {'Iris-setosa': np.int64(0), 'Iris-versicolor': np.int64(1), 'Iris-virginica': np.int64(2)}
```

Step 8: Feature Scaling

```
scaler = StandardScaler()
X = df.drop('species', axis=1)
X_scaled = scaler.fit_transform(X)
y = df['species']
```

Step 9: Split Dataset into Train and Test Sets

```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)
```

Step 10: Train Random Forest Classifier

```
model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, random_state=42)
```

Step 11: Model Evaluation

11.1)Predictions on Test Data

```
y_pred = model.predict(X_test)
```

11.2)Calculate Accuracy and Precision

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision (macro):", precision_score(y_test, y_pred, average='macro'))
```

```
Accuracy: 0.9333333333333333
Precision (macro): 0.9333333333333332
```

11.3)Classification Report

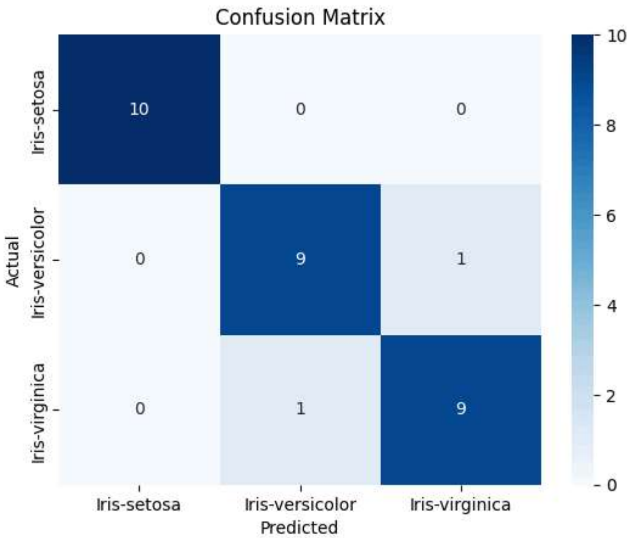
```
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       0.90      0.90      0.90        10
           2       0.90      0.90      0.90        10

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

11.4)Confusion Matrix Visualization

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=le.classes_, yticklabels=le.classes_)
```

```
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



Step 12: Save Model and Scaler

```
joblib.dump(model, 'iris_rf_model.pkl')
joblib.dump(scaler, 'iris_scaler.pkl')
joblib.dump(le, 'iris_label_encoder.pkl')
```

```
['iris_label_encoder.pkl']
```

Step 13: Load Model & Predict New Samples

```
model = joblib.load('iris_rf_model.pkl')
scaler = joblib.load('iris_scaler.pkl')
le = joblib.load('iris_label_encoder.pkl')
new_sample = np.array([[5.9, 3.0, 5.1, 1.8]])
new_sample_scaled = scaler.transform(new_sample)
predicted_label = model.predict(new_sample_scaled)
predicted_species = le.inverse_transform(predicted_label)
print("Predicted Iris Species:", predicted_species[0])
```

```
Predicted Iris Species: Iris-virginica
```