

# TASK 3

## IRIS FLOWER CLASSIFICATION

```
In [31]: # IRIS flower classification

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMa
from sklearn.ensemble import RandomForestClassifier
```

```
In [32]: df = pd.read_csv("IRIS.csv")
```

```
In [33]: df.head()
```

```
Out[33]:
```

	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 [34]: print("\nClass Distribution:")
print(df['species'].value_counts())
```

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

```
In [35]: 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    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 [36]: df.describe()
```

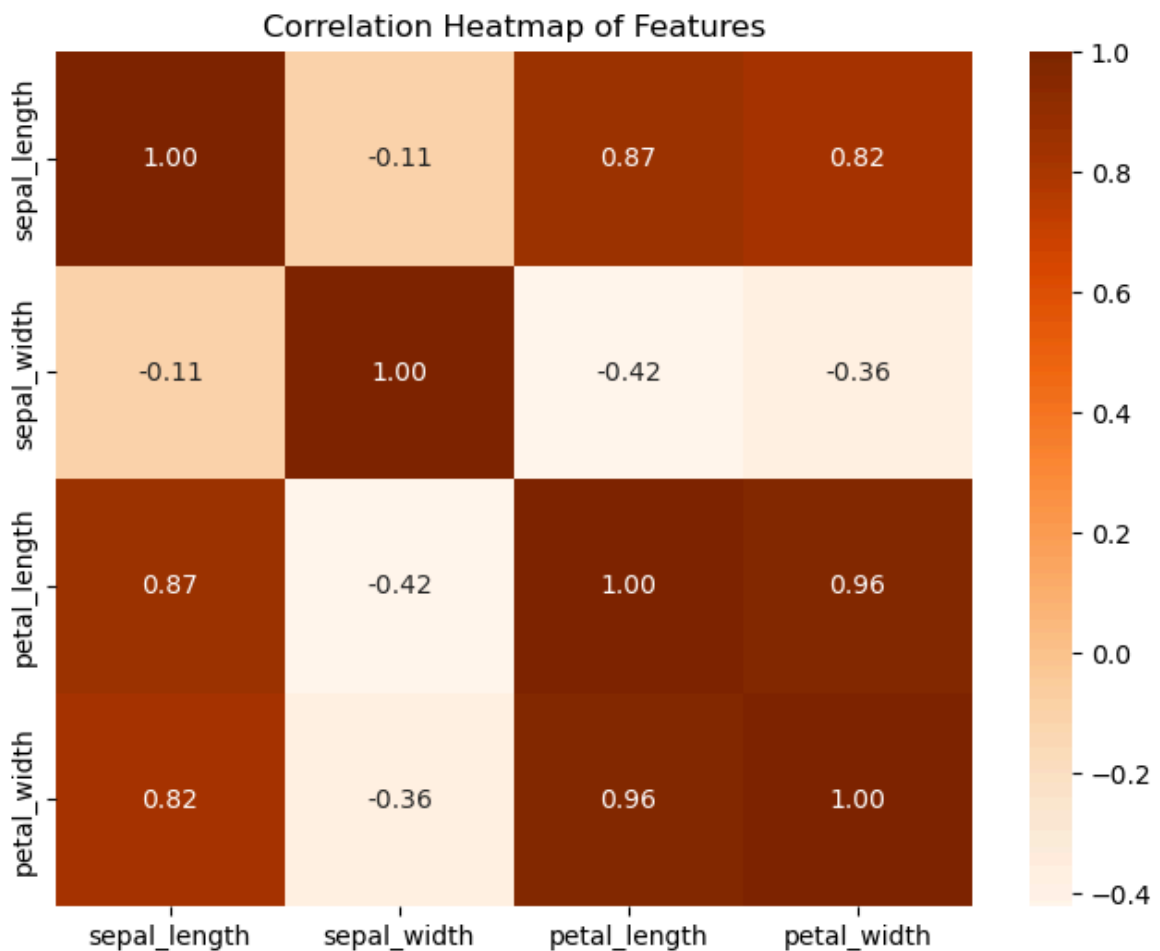
```
Out[36]:
```

	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

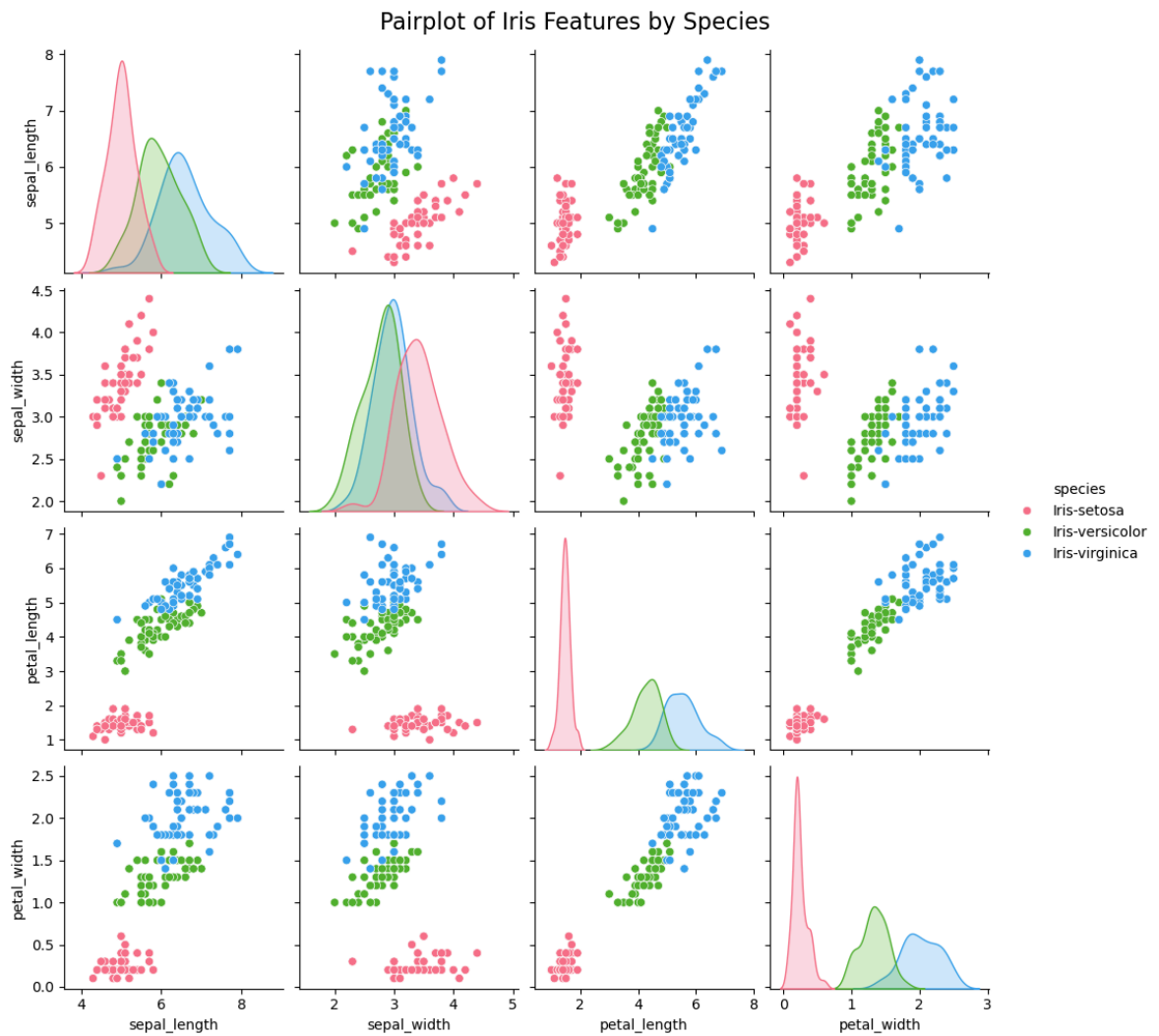
```
In [38]: df.columns
```

```
Out[38]: Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',  
               'species'],  
              dtype='object')
```

```
In [10]: plt.figure(figsize=(8, 6))  
sns.heatmap(df.drop('species', axis=1).corr(), annot=True, cmap='Oranges', fmt="")  
plt.title("Correlation Heatmap of Features")  
plt.show()
```

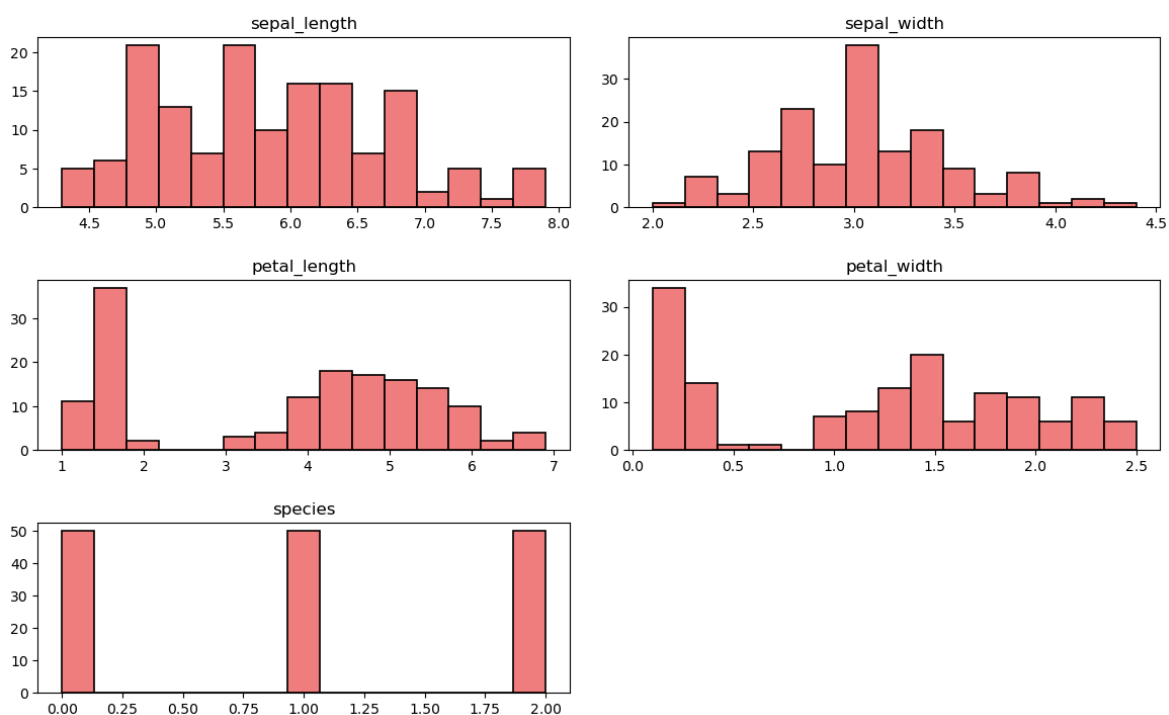


```
In [11]: sns.pairplot(df, hue='species', palette='husl')
plt.suptitle("Pairplot of Iris Features by Species", y=1.02, fontsize=16)
plt.show()
```



```
In [28]: df.hist(
    figsize=(12, 8),
    edgecolor='black',
    color='lightcoral',
    bins=15,
    linewidth=1.2,
    grid=False
)
plt.suptitle("Distribution of Iris Dataset Features", fontsize=18, fontweight='b')
plt.tight_layout(pad=2)
plt.show()
```

## Distribution of Iris Dataset Features

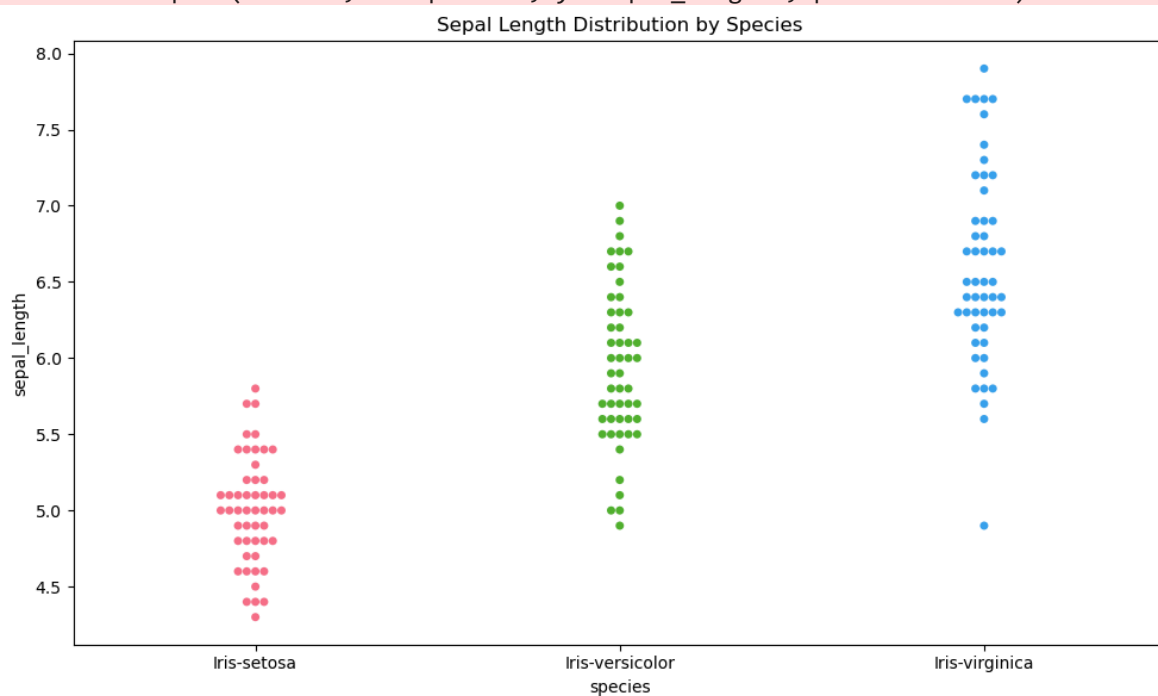


```
In [12]: plt.figure(figsize=(10, 6))
sns.swarmplot(data=df, x='species', y='sepal_length', palette='husl')
plt.title("Sepal Length Distribution by Species")
plt.tight_layout()
plt.show()
```

C:\Users\gonda\AppData\Local\Temp\ipykernel\_21648\3458291229.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v 0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.swarmplot(data=df, x='species', y='sepal_length', palette='husl')
```

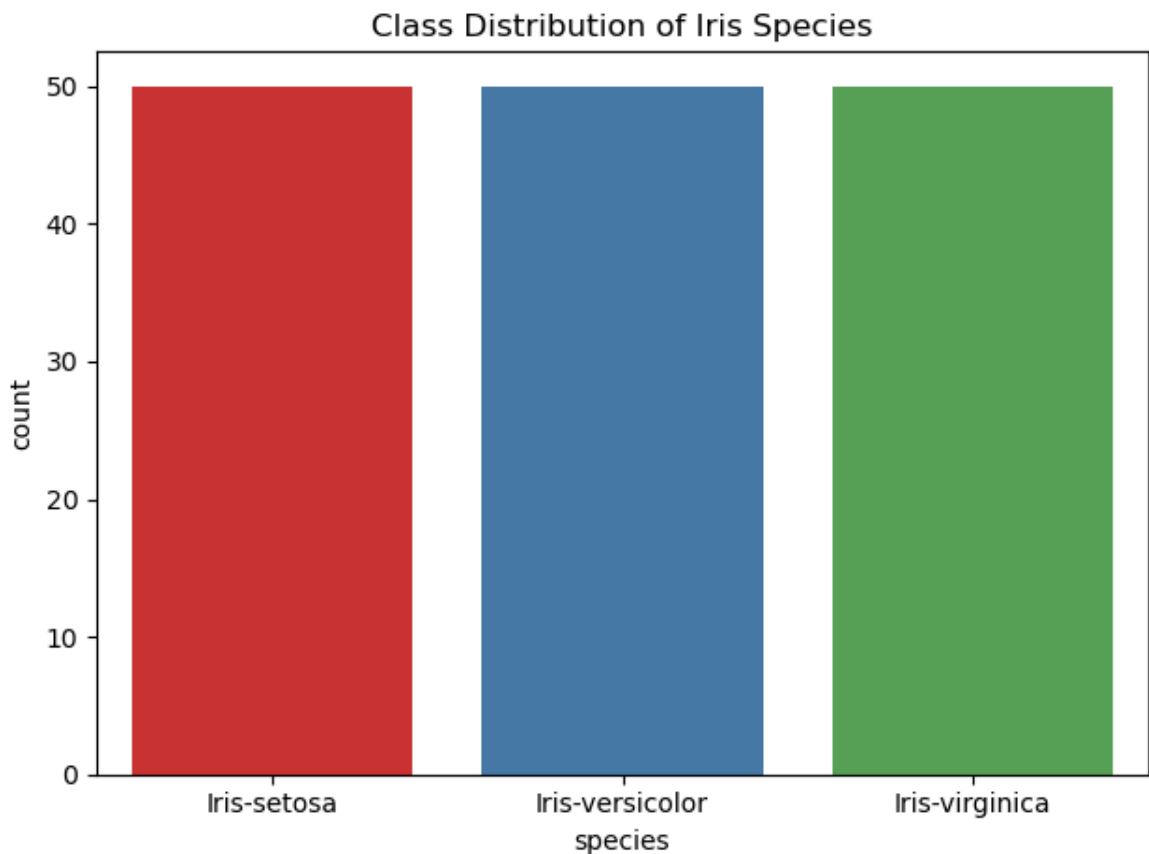


```
In [13]: sns.countplot(data=df, x='species', palette='Set1')
plt.title("Class Distribution of Iris Species")
plt.tight_layout()
plt.show()
```

C:\Users\gonda\AppData\Local\Temp\ipykernel\_21648\3854417594.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v 0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.countplot(data=df, x='species', palette='Set1')
```

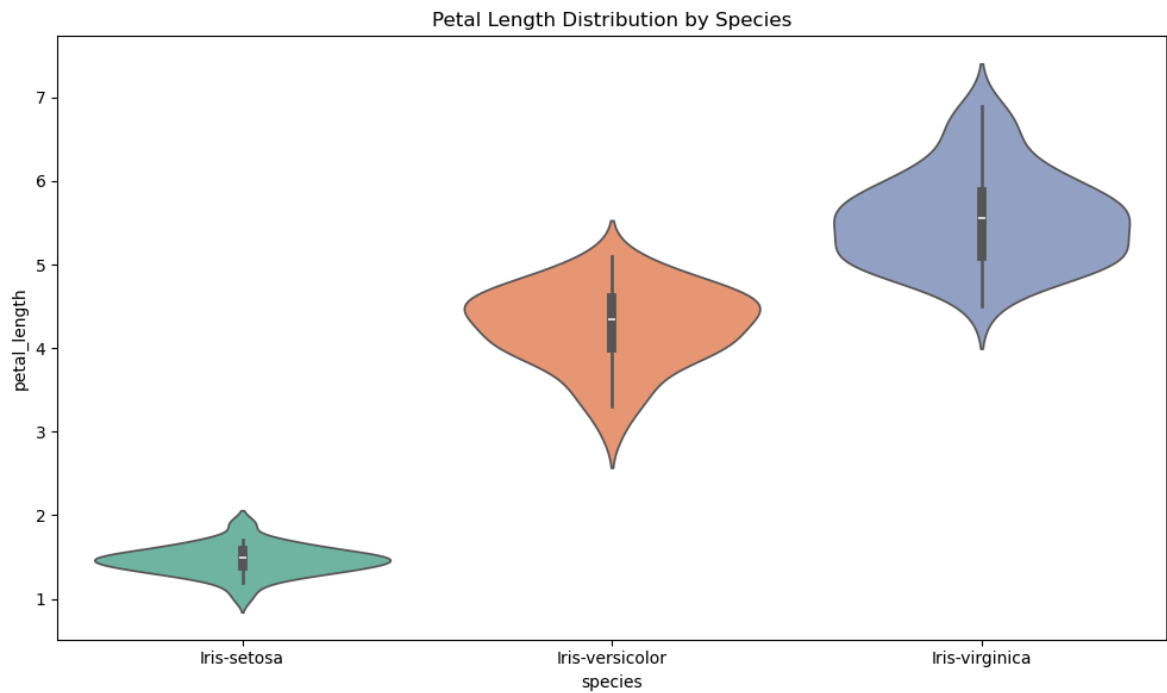


```
In [14]: plt.figure(figsize=(10, 6))
sns.violinplot(data=df, x='species', y='petal_length', palette='Set2')
plt.title("Petal Length Distribution by Species")
plt.tight_layout()
plt.show()
```

C:\Users\gonda\AppData\Local\Temp\ipykernel\_21648\2535030879.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v 0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.violinplot(data=df, x='species', y='petal_length', palette='Set2')
```



```
In [15]: le = LabelEncoder()
df['species'] = le.fit_transform(df['species'])
```

```
In [16]: X = df.drop('species', axis=1)
y = df['species']
```

```
In [17]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

```
In [18]: model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
Out[18]: Random Forest Classifier
RandomForestClassifier(random_state=42)
```

```
In [19]: y_pred = model.predict(X_test)
```

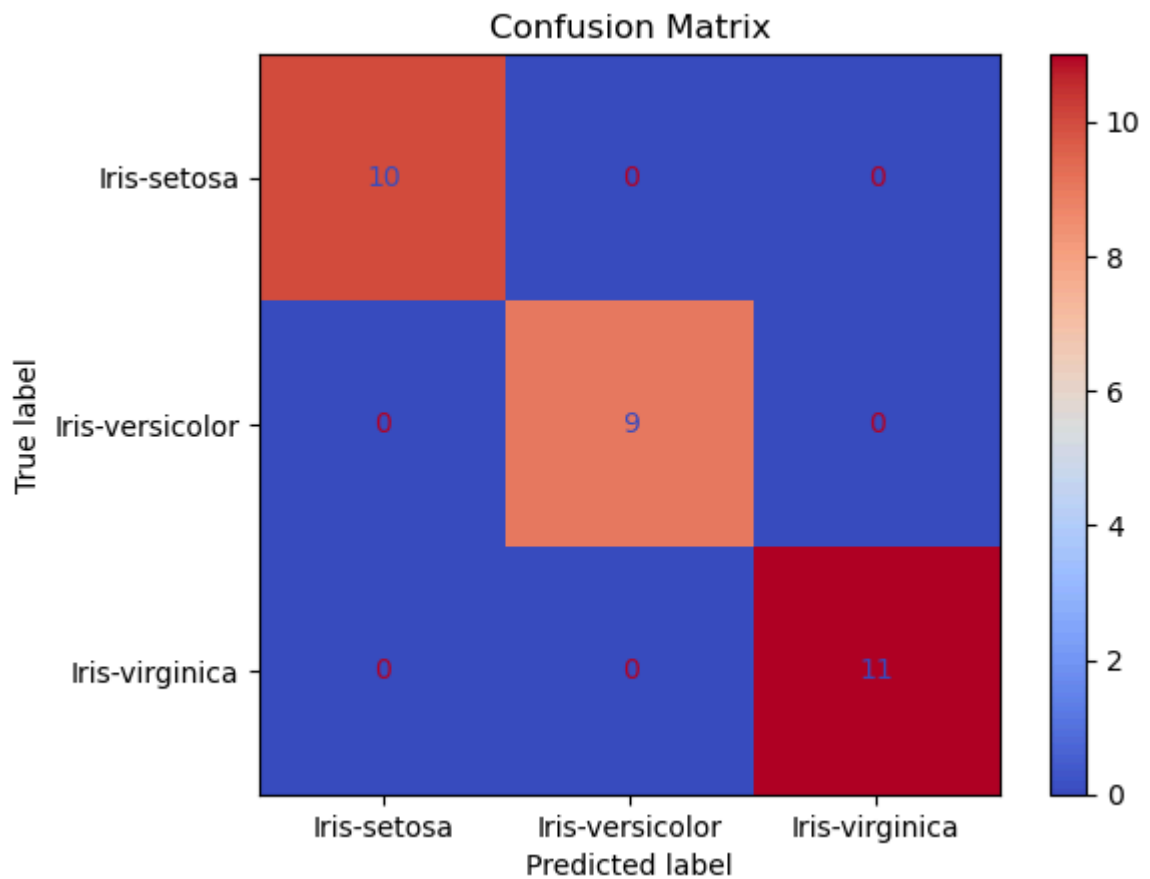
```
In [20]: print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

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

```
In [21]: conf_mat = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=conf_mat, display_labels=le.class
disp.plot(cmap='coolwarm')
```

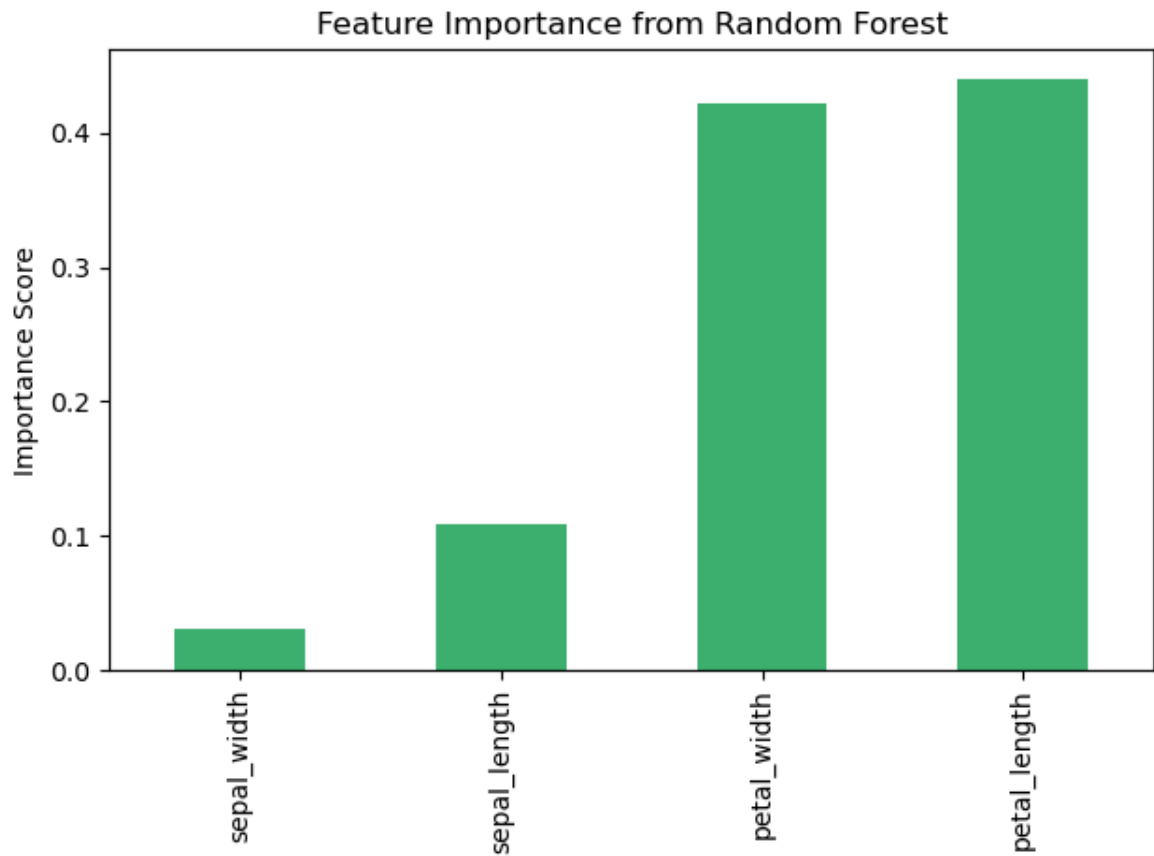
```
plt.title("Confusion Matrix")
plt.show()
```



```
In [23]: import pandas as pd
import matplotlib.pyplot as plt

feature_importance = pd.Series(model.feature_importances_, index=X.columns)

feature_importance.sort_values().plot(kind='bar', color='mediumseagreen')
plt.title("Feature Importance from Random Forest")
plt.ylabel("Importance Score")
plt.tight_layout()
plt.show()
```

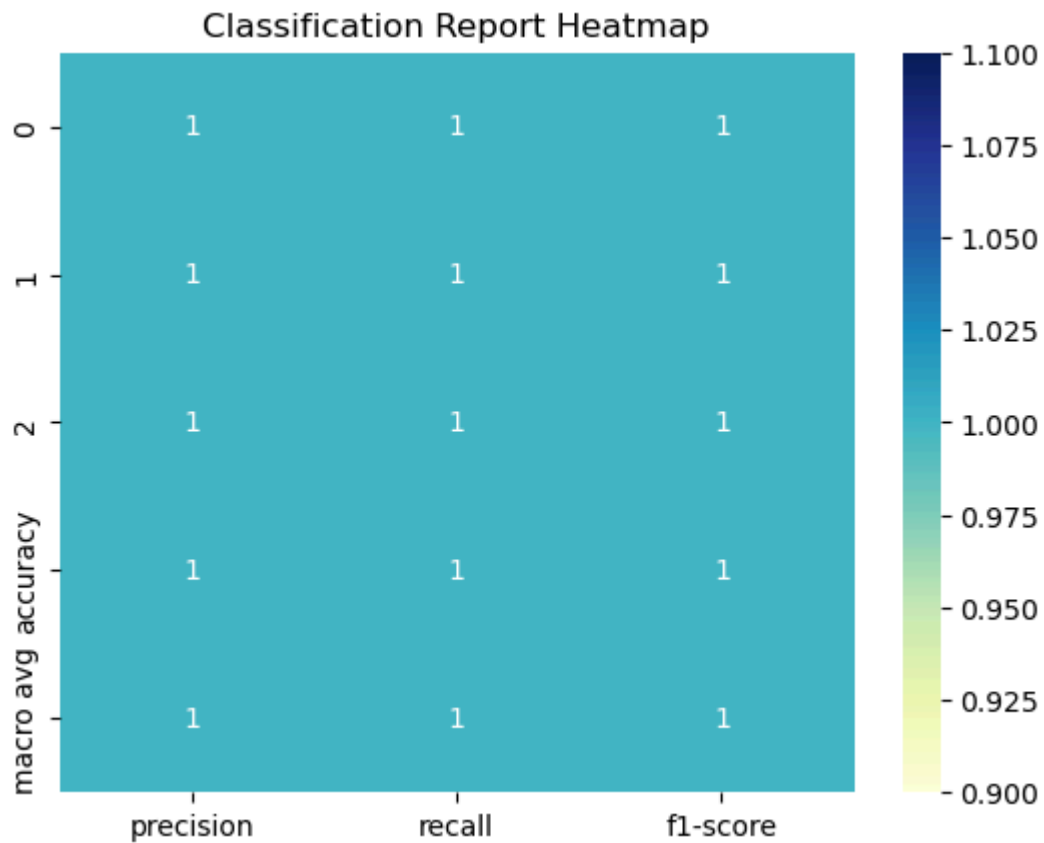


```
In [24]: from sklearn.metrics import classification_report
import pandas as pd

report = classification_report(y_test, y_pred, output_dict=True)
df_report = pd.DataFrame(report).transpose()

sns.heatmap(df_report.iloc[:-1, :-1], annot=True, cmap='YlGnBu')
plt.title("Classification Report Heatmap")
plt.show()
```





```
In [40]: from sklearn.model_selection import cross_val_score
scores = cross_val_score(RandomForestClassifier(), X, y, cv=5)
print("Cross-Validation Accuracy Scores:", scores)
```

Cross-Validation Accuracy Scores: [0.96666667 0.96666667 0.93333333 0.93333333 1.]

In [ ]: