```
# 1.    Load the basic libraries and packages

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import pydot
from sklearn.tree import export_graphviz
from sklearn import metrics
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


# 2.    Load the dataset

iris = datasets.load_iris()
class_name = iris.target_names
data = pd.DataFrame({"Sepal Length" : iris.data[:,0],
                     "Sepal Width" : iris.data[:,1],
                     "Petal Length" : iris.data[:,2],
                     "Petal Width" : iris.data[:,3] ,
                     "Species" : iris.target ,
                     "Species Name" : class_name[iris.target]});
data.head()
```

	Sepal Length	Sepal Width	Petal Length	Petal Width	Species	Species Name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

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

```
# 3.    Separate the feature and prediction value columns

X = data.drop(["Species","Species Name"],axis=1)
y = data["Species"]


# 4. Splitting the Training and Testing Data

X_Train , X_Test  , y_train , Y_Test = train_test_split(X, y , test_size=0.2 , random_state=42)


# 5. Creating a Model

forest = RandomForestClassifier(n_estimators=200 , random_state = 42)


# 6. Training a Model

forest.fit(X_Train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(n_estimators=200, random_state=42)
```

```
# 7. Predicitng the Output Using model

Y_Predicted = forest.predict(X_Test)


# 8. Measuring the Accuracy

metrics.accuracy_score(Y_Test,Y_Predicted)
```

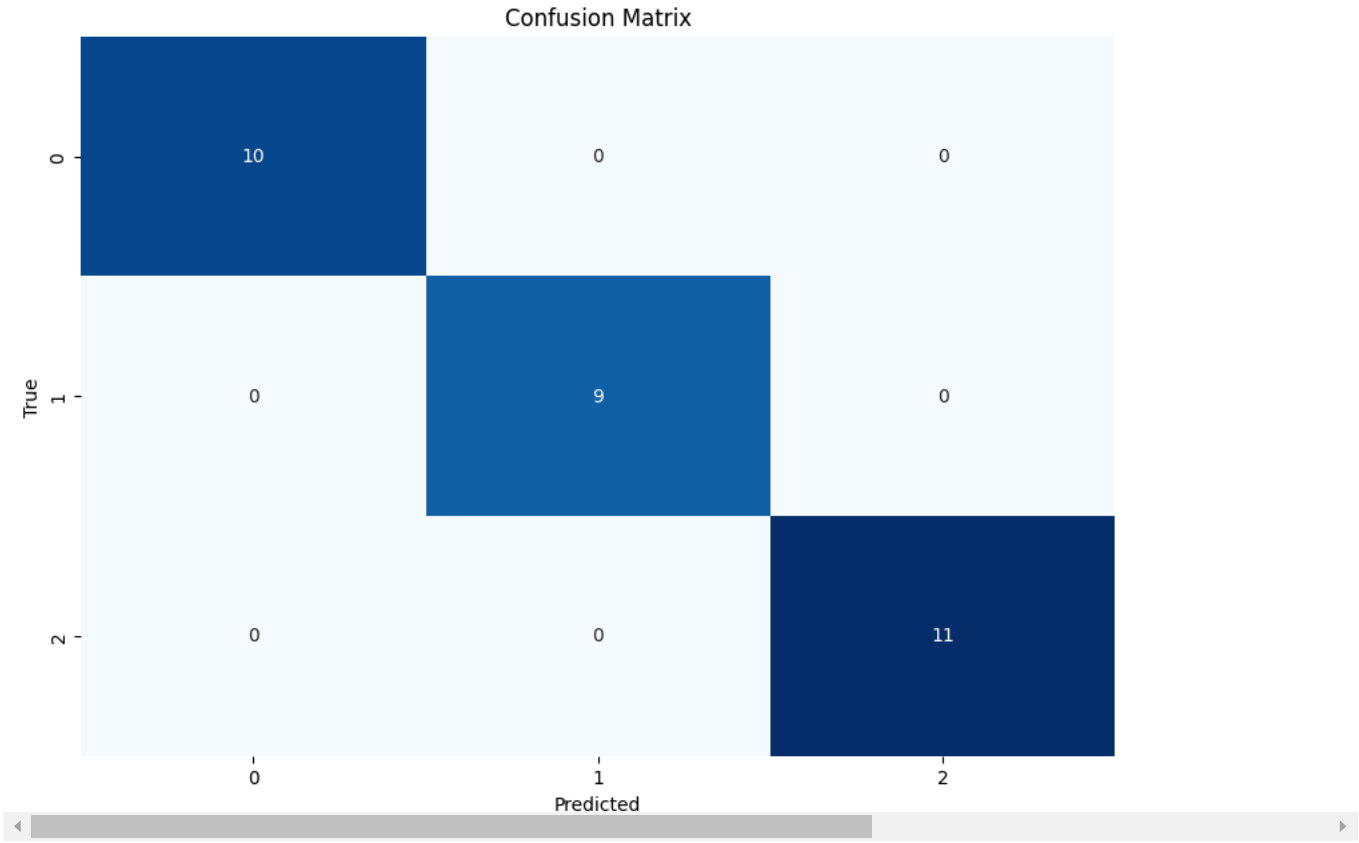
```
1.0
```

```
# 9. Creating a Confusion Matrix

confusion_matrix = confusion_matrix(y_true = Y_Test , y_pred = Y_Predicted)
confusion_matrix
```

```
array([[10,  0,  0],
       [ 0,  9,  0],
       [ 0,  0, 11]])
```

```
# 10. Plotting a Confusion Matrix

plt.figure(figsize=(10, 7))
sns.heatmap(confusion_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```




```
# 11. Accesing a Tree

tree  = forest.estimators_[5]
```

```
# 12. Exporitng a Tree

export_graphviz(tree, out_file='/content/tree.dot', feature_names=X.columns, rounded=True, precision=1)
```

```
# 13. Converting the Tree into png format

graph = pydot.graph_from_dot_file('/content/tree.dot')
graph[0].write_png('/content/tree.png')
```

```
# 14. Loading a Graph

(graph,) = pydot.graph_from_dot_file('/content/tree.dot')
```

```
# 15. Analysing the Feature Importance

forest.feature_importances_
```

```
array([0.10196014, 0.03119641, 0.45803602, 0.40880743])
```

```
# 16. Visulizing the FFeature Importance

feature_importances = pd.Series(forest.feature_importances_, index=X.columns)
```

```
# Sort the feature importances in descending order
sorted_importances = feature_importances.sort_values(ascending=True)

# Plot the sorted feature importances
sorted_importances.plot(kind='barh', figsize=(10, 8))

# Add labels and title
plt.xlabel('Importance')
plt.ylabel('Features')
plt.title('Feature Importances - Sorted')
plt.show()
```