Introduction :

The experiments aims to demonstrate the use of a Random Forest Classifier to classify Iris flowers based on their features. The Iris flower dataset is a well-known dataset in machine learning, and the goal here is to build a model that can accurately classify iris flowers into three species: setosa, versicolor, and virginica.

Problem Statement :

The challenge is to build a machine learning model to classify iris flowers according to their characteristics. The classification task is necessary to understand and classify different iris flower types based on their characteristics.

Methodology :

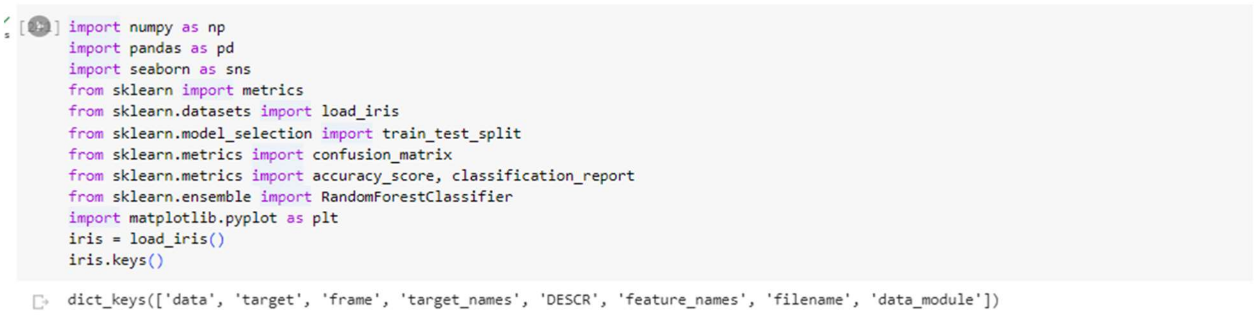
1. **Data Preparation :**
   i. The code begins by importing the necessary libraries and loading the Iris dataset.
   ii. The dataset is then converted into a Pandas DataFrame for ease of analysis.
   iii. Features (sepal length, sepal width, petal length, and petal width) are stored in `X`, and the target variable (species name) is stored in `y`.
   iv. The data is split into training and testing sets using `train_test_split`.

2. **Model Building :**
   i. A Random Forest Classifier is chosen as the machine learning algorithm for this task. The code initializes three different models with different numbers of trees (10, 15, and 20).
   ii. Each model is trained on the training data using the `fit` method.

3. **Model Training and Evaluation :**
   i. After training each model, the code evaluates its accuracy on the testing data using the `score` method and calculates predictions (`y_pred`) for the testing data.
   ii. Classification reports are generated for each model to provide detailed performance metrics, including precision, recall, F1-score, and support, for each class (Iris species).
   iii. A confusion matrix is also generated and visualized using a heatmap to understand the model's performance in detail.

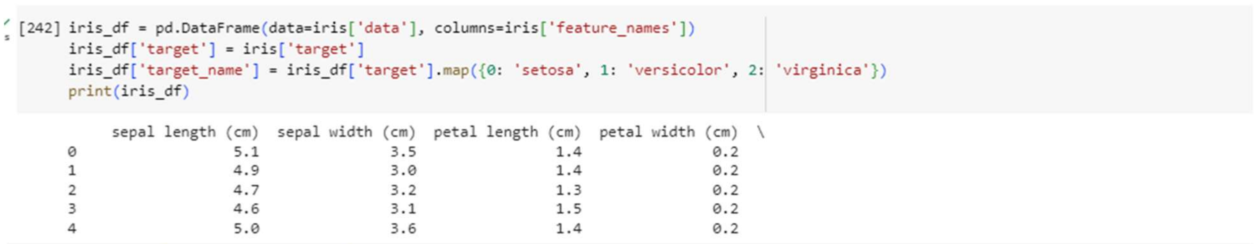
4. **Hyperparameter Tuning :**
   i. To explore the impact of the number of trees (n_estimators) on model performance, the code conducts an additional experiment.
   ii. It iterates through different values of `n_estimators` (10, 15, 20, 25, and 30) and calculates the accuracy for each.
   iii. The accuracy vs. the number of trees is plotted to visualize how the model's performance changes with the number of trees.

Source code and Result :



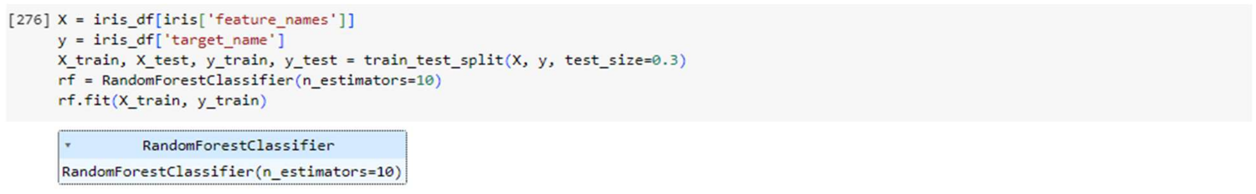
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import metrics
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, classification_report
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
iris = load_iris()
iris.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```



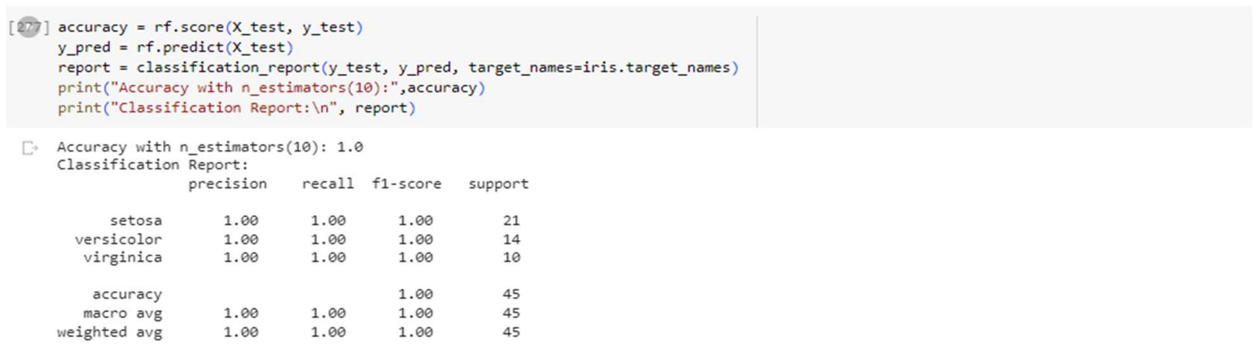
```
[242] iris_df = pd.DataFrame(data=iris['data'], columns=iris['feature_names'])
      iris_df['target'] = iris['target']
      iris_df['target_name'] = iris_df['target'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})
      print(iris_df)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
```



```
[276] X = iris_df[iris['feature_names']]
      y = iris_df['target_name']
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
      rf = RandomForestClassifier(n_estimators=10)
      rf.fit(X_train, y_train)
```

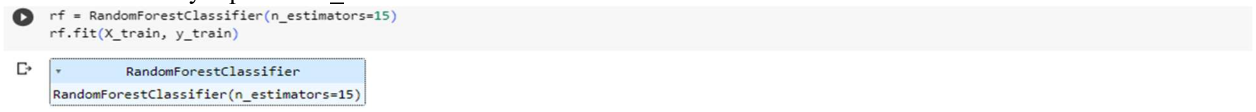
```
          RandomForestClassifier
RandomForestClassifier(n_estimators=10)
```



```
[277] accuracy = rf.score(X_test, y_test)
      y_pred = rf.predict(X_test)
      report = classification_report(y_test, y_pred, target_names=iris.target_names)
      print("Accuracy with n_estimators(10):",accuracy)
      print("Classification Report:\n", report)
```

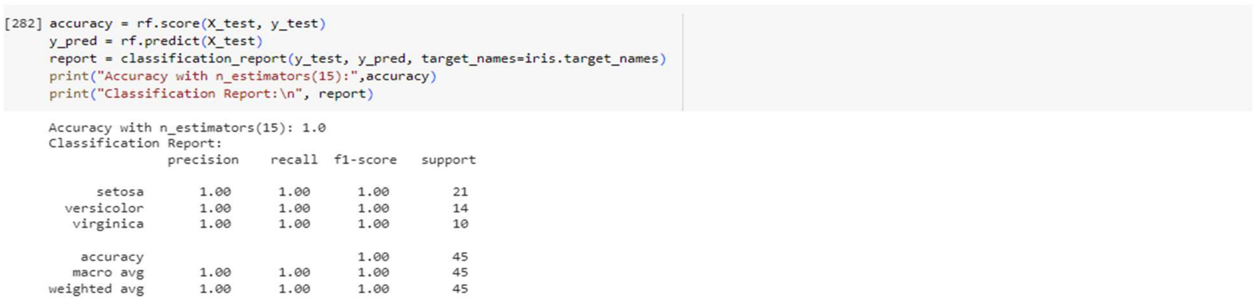
```
Accuracy with n_estimators(10): 1.0
Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        21
  versicolor       1.00      1.00      1.00        14
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

- Accuracy report for n_estimators = 10



```
rf = RandomForestClassifier(n_estimators=15)
rf.fit(X_train, y_train)
```

```
          RandomForestClassifier
RandomForestClassifier(n_estimators=15)
```



```
[282] accuracy = rf.score(X_test, y_test)
      y_pred = rf.predict(X_test)
      report = classification_report(y_test, y_pred, target_names=iris.target_names)
      print("Accuracy with n_estimators(15):",accuracy)
      print("Classification Report:\n", report)
```

```
Accuracy with n_estimators(15): 1.0
Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        21
  versicolor       1.00      1.00      1.00        14
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

- Accuracy report for n_estimators = 15

```
[283] rf = RandomForestClassifier(n_estimators=20)
      rf.fit(X_train, y_train)
```

```
▾        RandomForestClassifier
RandomForestClassifier(n_estimators=20)
```

```
[284] accuracy = rf.score(X_test, y_test)
      y_pred = rf.predict(X_test)
      report = classification_report(y_test, y_pred, target_names=iris.target_names)
      print("Accuracy with n_estimators(20):",accuracy)
      print("Classification Report:\n", report)
```

```
Accuracy with n_estimators(20): 1.0
Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        21
  versicolor       1.00      1.00      1.00        14
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

- Accuracy report for n_estimators = 20

Confusion Matrix –

```
confusion_mat = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat, annot=True, fmt="d", cmap="Greens",
            xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

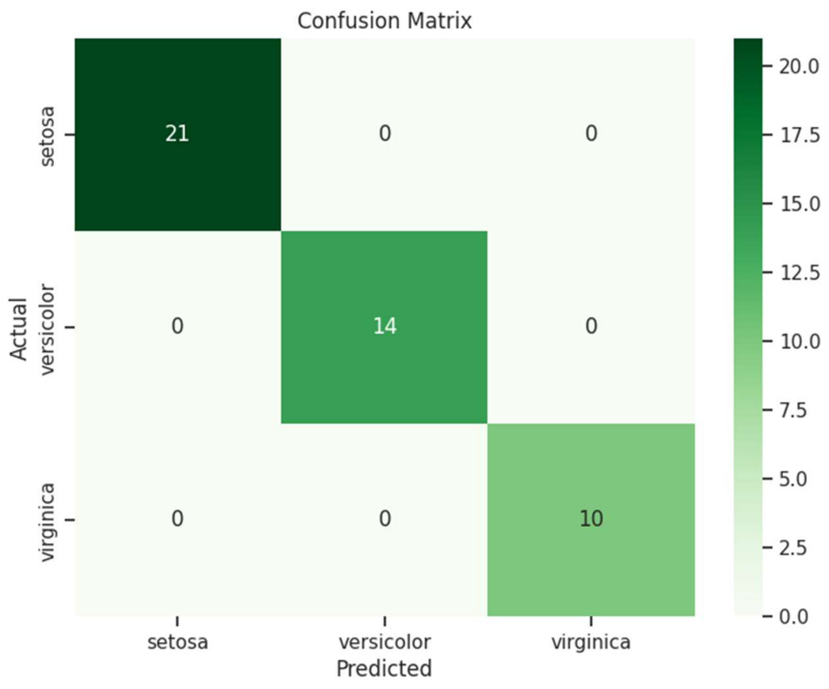
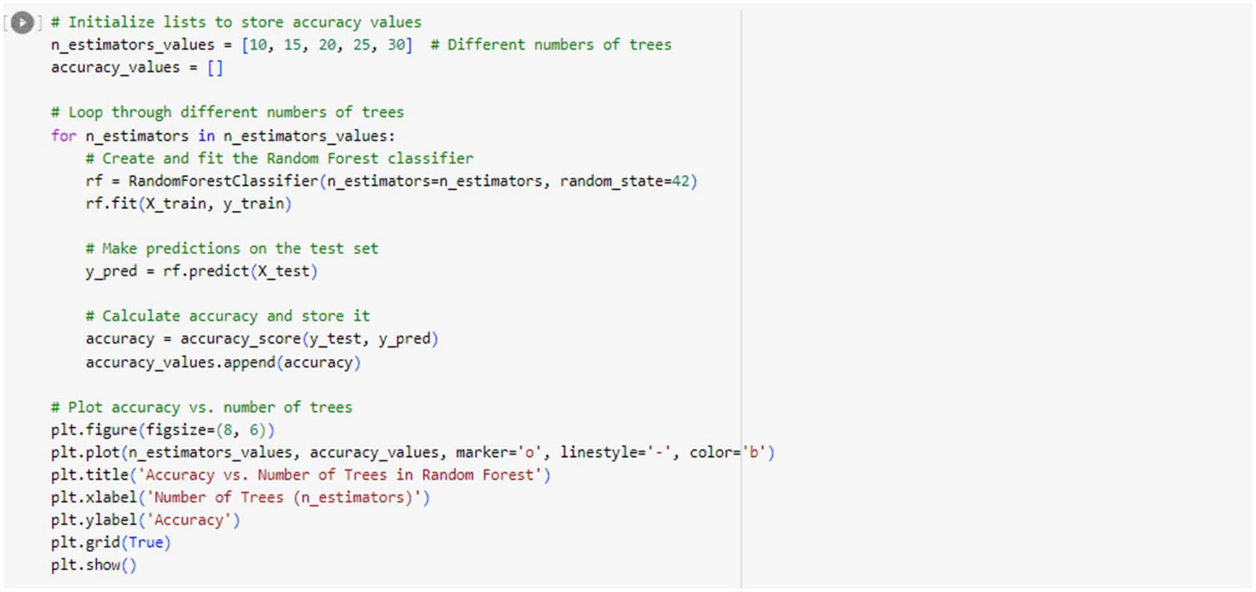
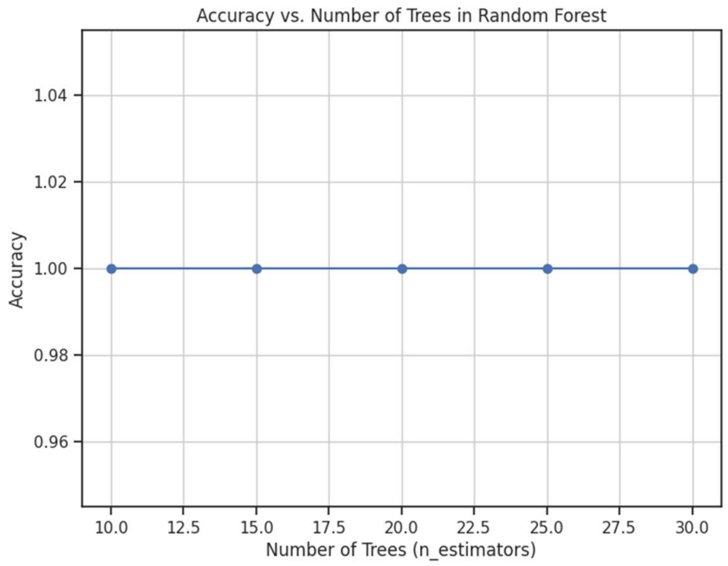


Fig – 1: Confusion Matrix

To Accuracy VS Number of Trees –



```
# Initialize lists to store accuracy values
n_estimators_values = [10, 15, 20, 25, 30]  # Different numbers of trees
accuracy_values = []

# Loop through different numbers of trees
for n_estimators in n_estimators_values:
    # Create and fit the Random Forest classifier
    rf = RandomForestClassifier(n_estimators=n_estimators, random_state=42)
    rf.fit(X_train, y_train)

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

    # Calculate accuracy and store it
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_values.append(accuracy)

# Plot accuracy vs. number of trees
plt.figure(figsize=(8, 6))
plt.plot(n_estimators_values, accuracy_values, marker='o', linestyle='-', color='b')
plt.title('Accuracy vs. Number of Trees in Random Forest')
plt.xlabel('Number of Trees (n_estimators)')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()
```



Analysis : For n_estimators = 10, 15, 20; we got the best accuracy of **1.00**.

Conclusion :

The Random Forest Classifier is an effective algorithm for the Iris flower classification task. The code demonstrates the impact of changing the number of trees (n_estimators) on model accuracy. By comparing the results of models with different numbers of trees, it is possible to determine the optimal number of trees for this specific dataset. This experiment shows that the Random Forest Classifier can achieve high accuracy in classifying iris flowers into their respective species based on their features.