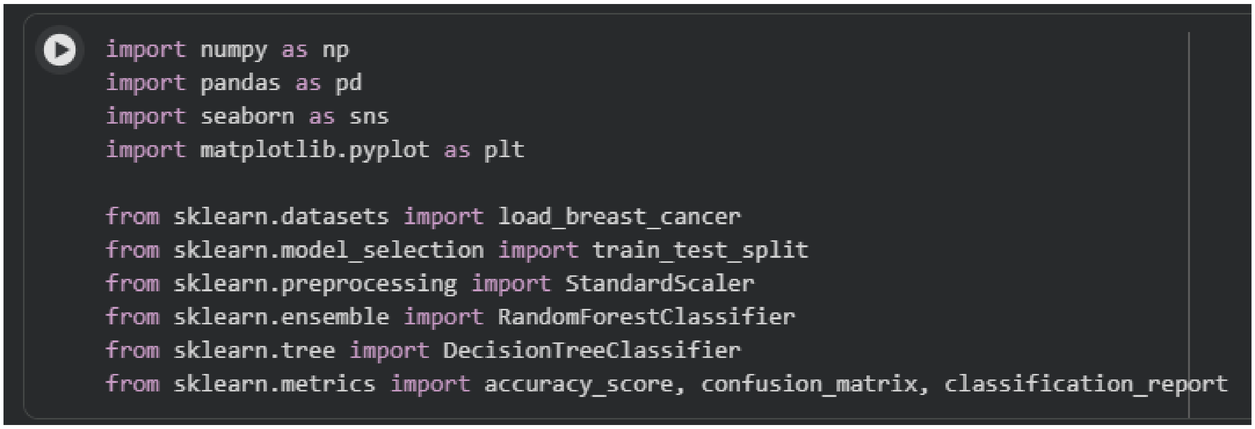
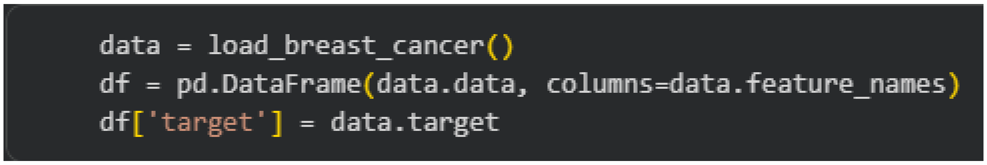
LAB NO 05

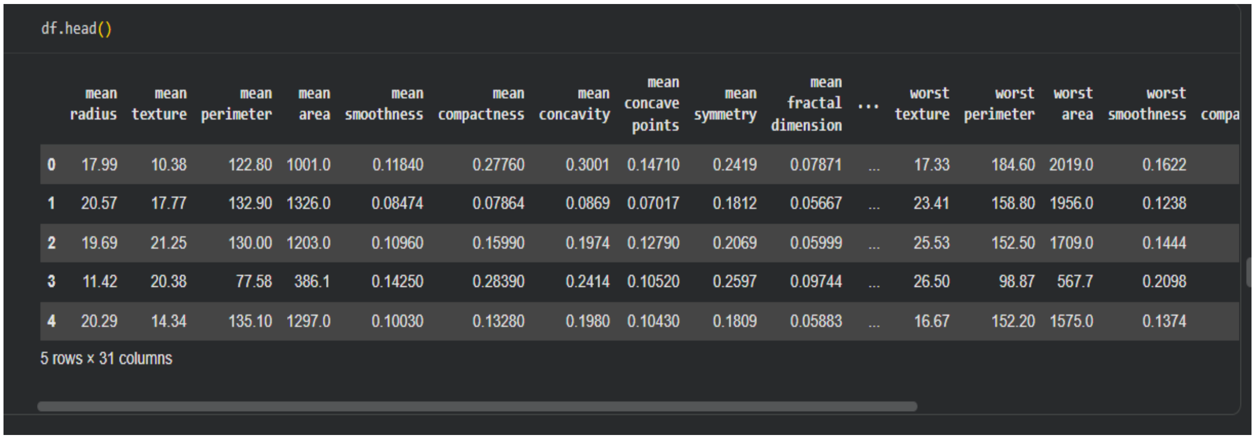
1. **Load a dataset for classification (e.g., Titanic, Breast Cancer dataset).**



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

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```



```python
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
```

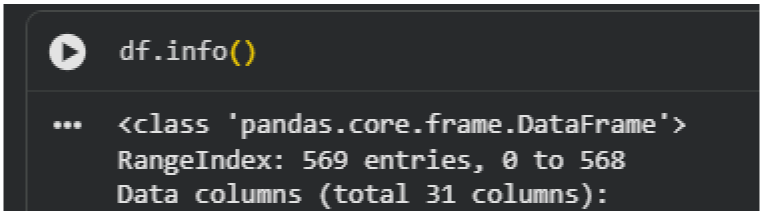


```python
df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	compa
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	

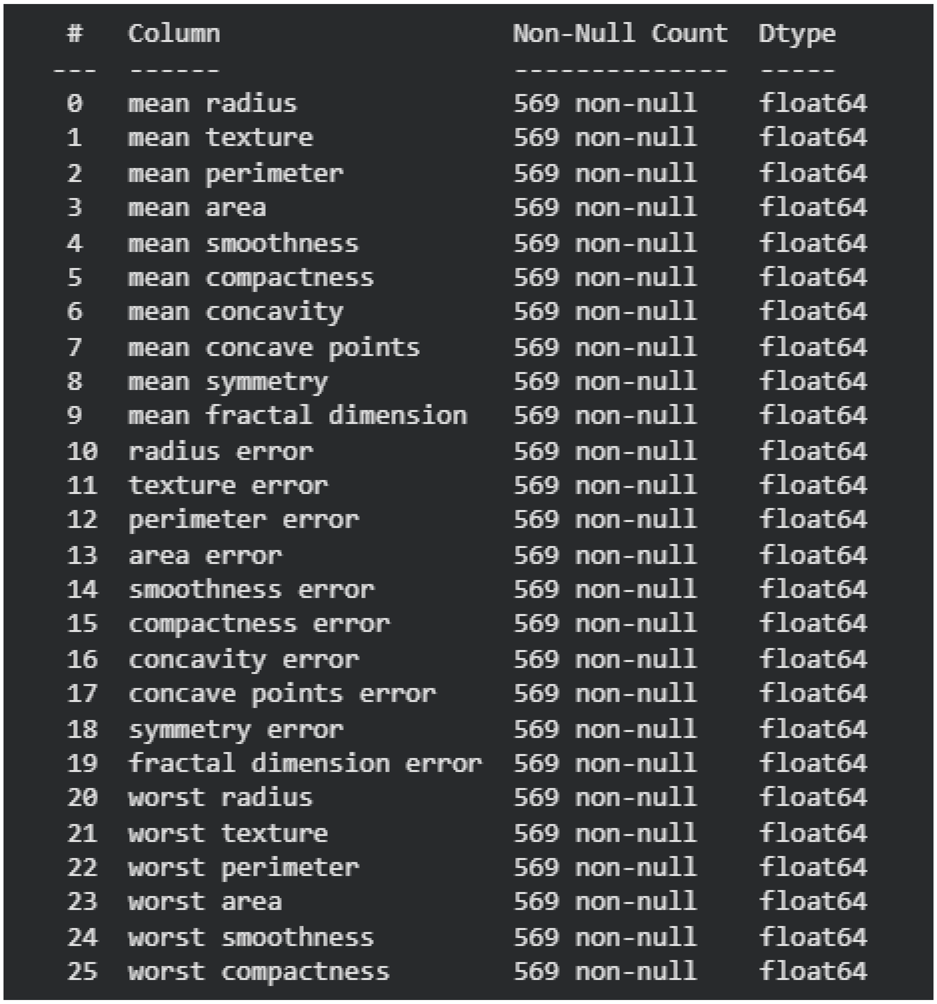
5 rows × 31 columns

2. **Apply data preprocessing (handle missing values, encode categorical data)**

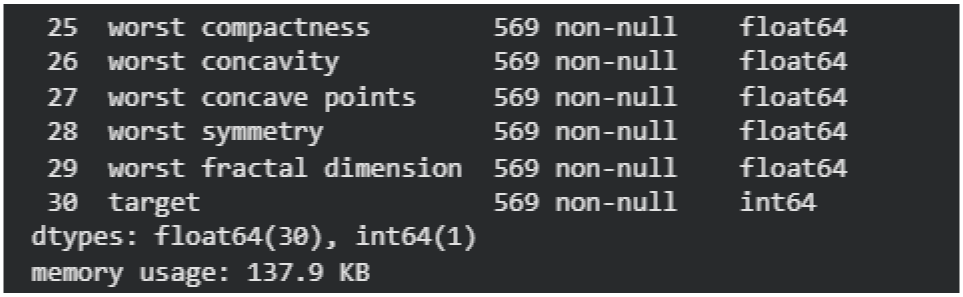


```python
df.info()
```

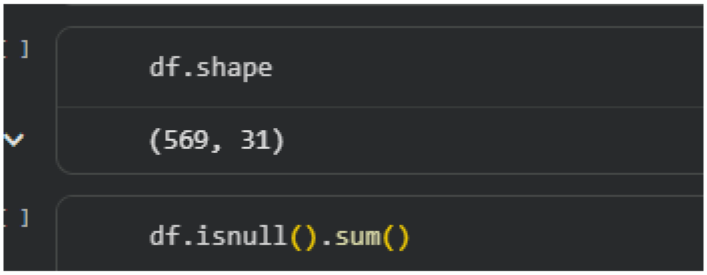
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
```



```
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
```



```
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    int64  
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

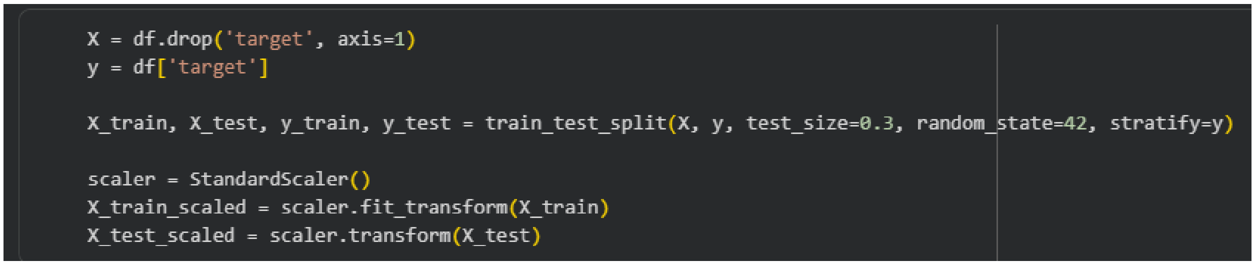


```
df.shape
```

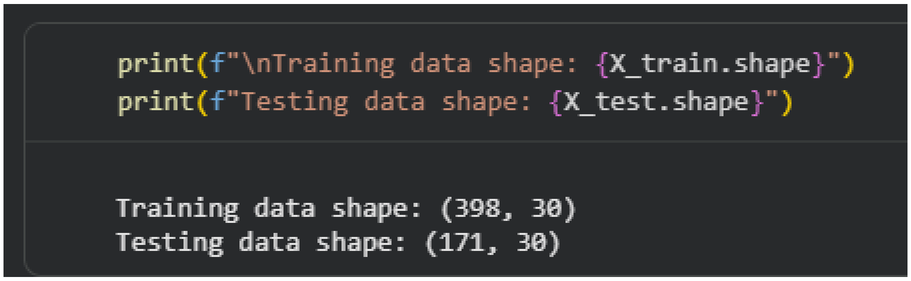
```
(569, 31)
```

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

3. **Split the dataset into training and testing sets.**



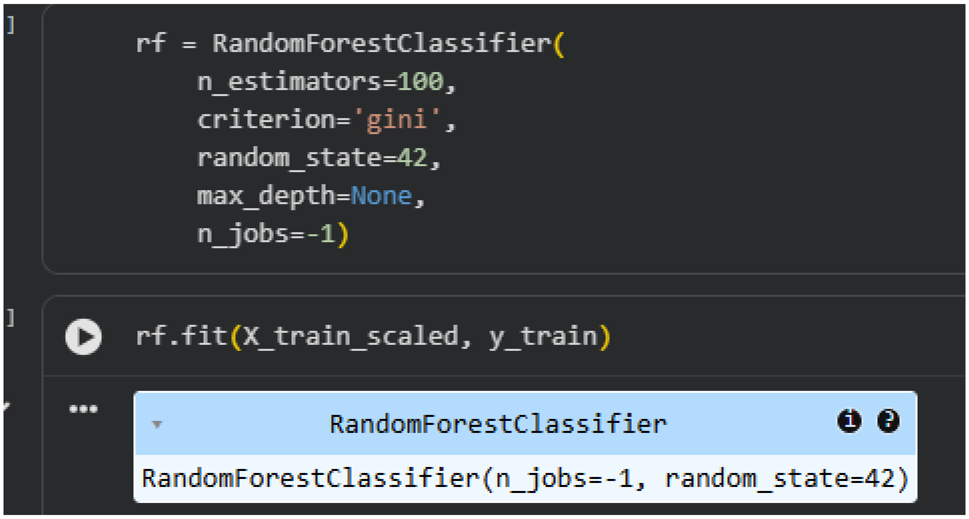
```
X = df.drop('target', axis=1)
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```



```
print(f"\nTraining data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}")
```

```

Training data shape: (398, 30)
Testing data shape: (171, 30)
```

4. **Train a Random Forest Classifier on the training data.**

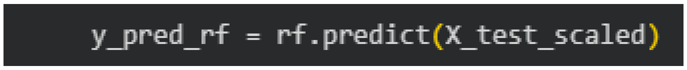


```
rf = RandomForestClassifier(
    n_estimators=100,
    criterion='gini',
    random_state=42,
    max_depth=None,
    n_jobs=-1)
```

```
rf.fit(X_train_scaled, y_train)
```

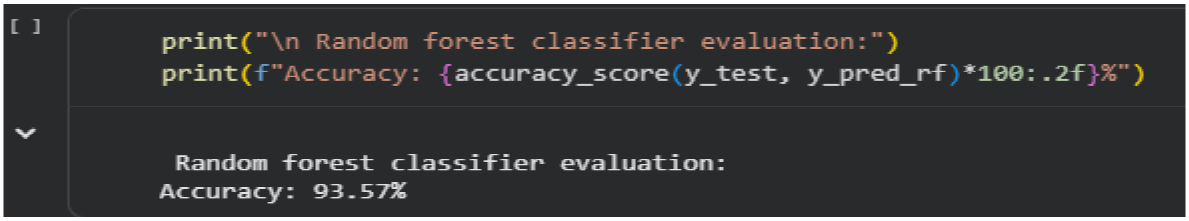
```
RandomForestClassifier
RandomForestClassifier(n_jobs=-1, random_state=42)
```

5. **Make predictions on the test set.**



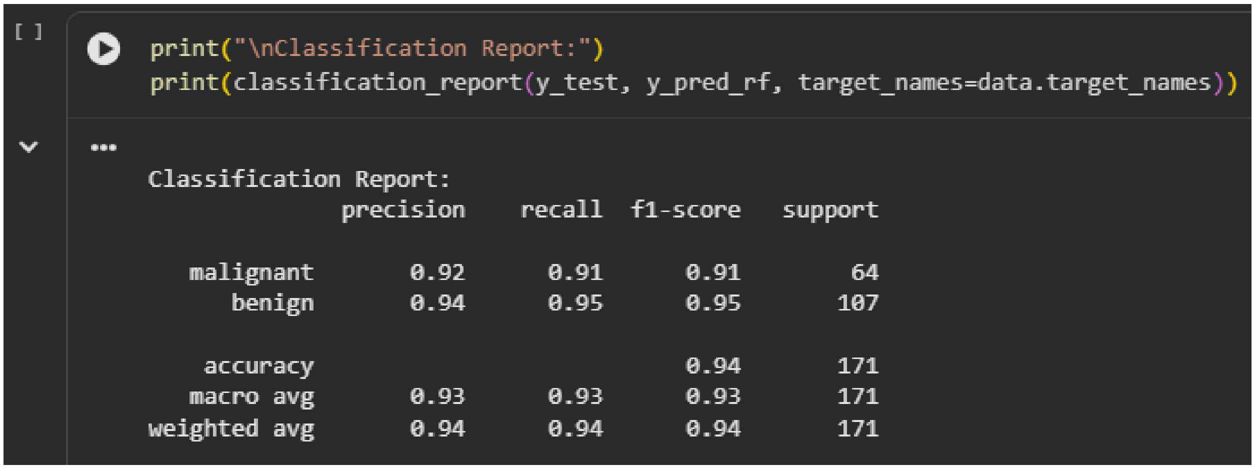
```
y_pred_rf = rf.predict(X_test_scaled)
```

6. **Evaluate performance using accuracy, precision, recall, and F1-score.**



```
print("\n Random forest classifier evaluation:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf)*100:.2f}%")
```

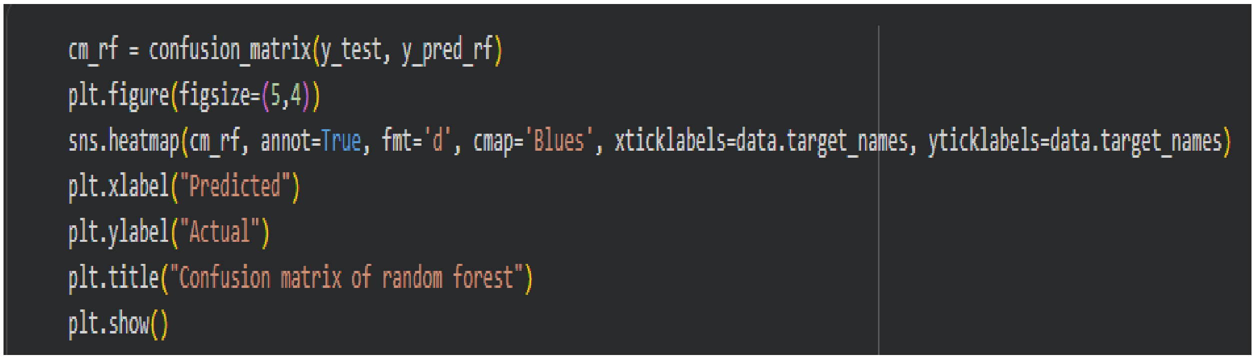
```

 Random forest classifier evaluation:
Accuracy: 93.57%
```



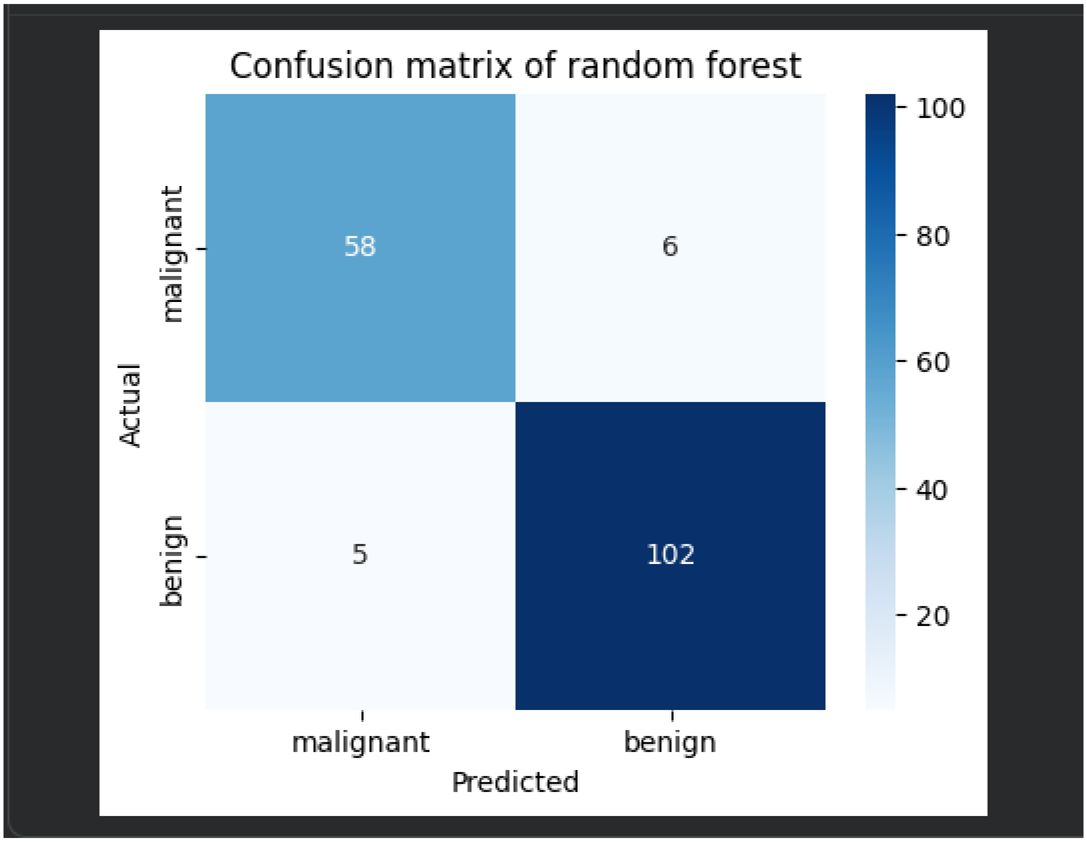
```
print("\nClassification Report:")
print(classification_report(y_test, y_pred_rf, target_names=data.target_names))
```

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

   malignant       0.92      0.91      0.91        64
      benign       0.94      0.95      0.95       107

    accuracy                           0.94       171
   macro avg       0.93      0.93      0.93       171
weighted avg       0.94      0.94      0.94       171
```

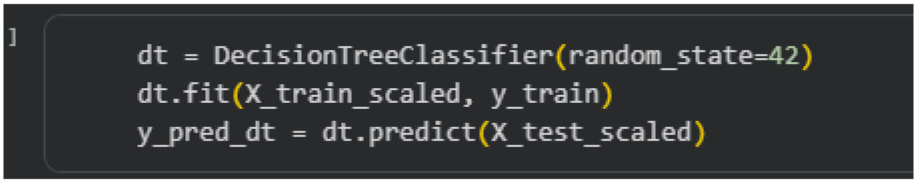
7. Visualize the Confusion Matrix.



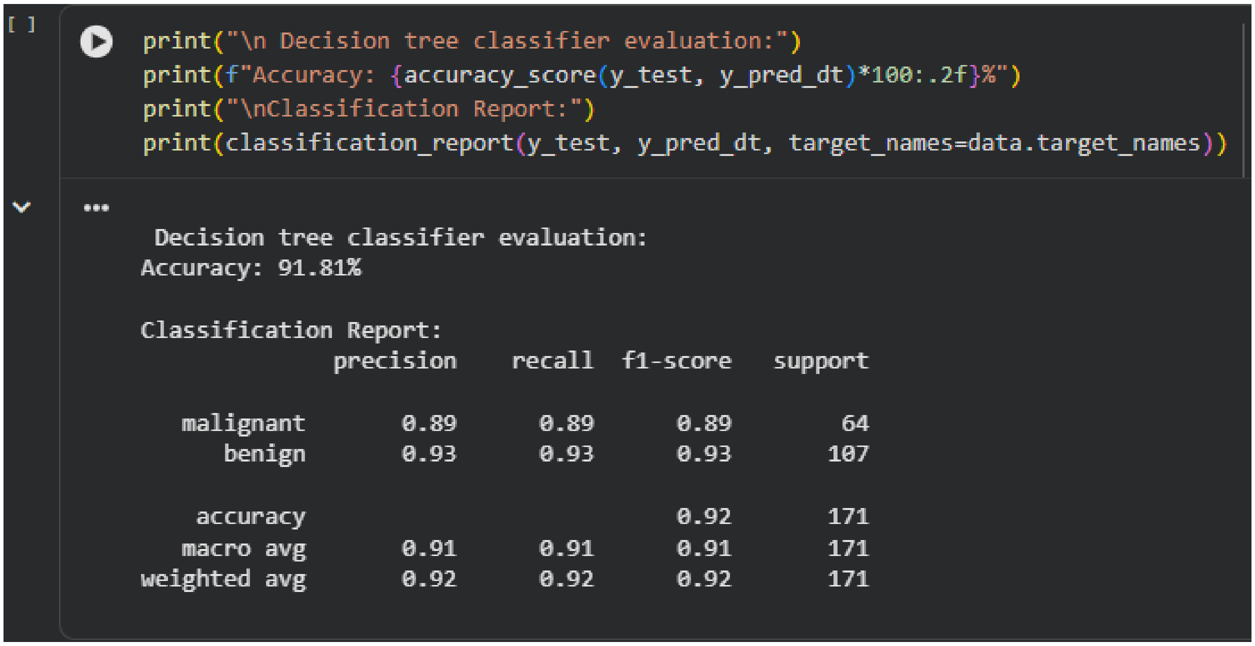
```
cm_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(5,4))
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues', xticklabels=data.target_names, yticklabels=data.target_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion matrix of random forest")
plt.show()
```



8. Compare with a Single Decision Tree

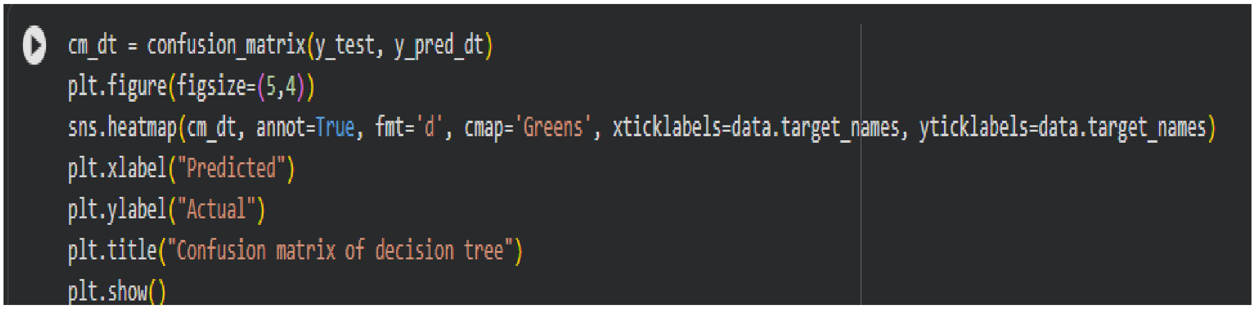


```
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train_scaled, y_train)
y_pred_dt = dt.predict(X_test_scaled)
```

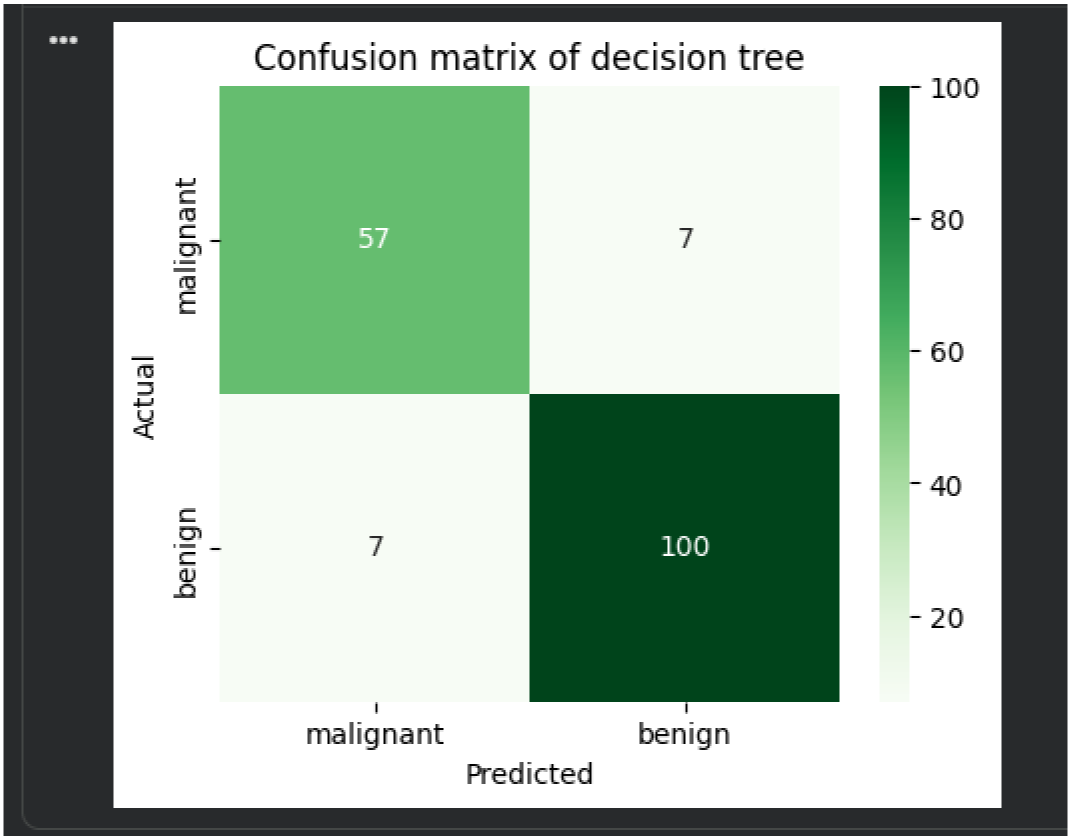


```
print("\n Decision tree classifier evaluation:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_dt)*100:.2f}%")
print("\nClassification Report:")
print(classification_report(y_test, y_pred_dt, target_names=data.target_names))
```

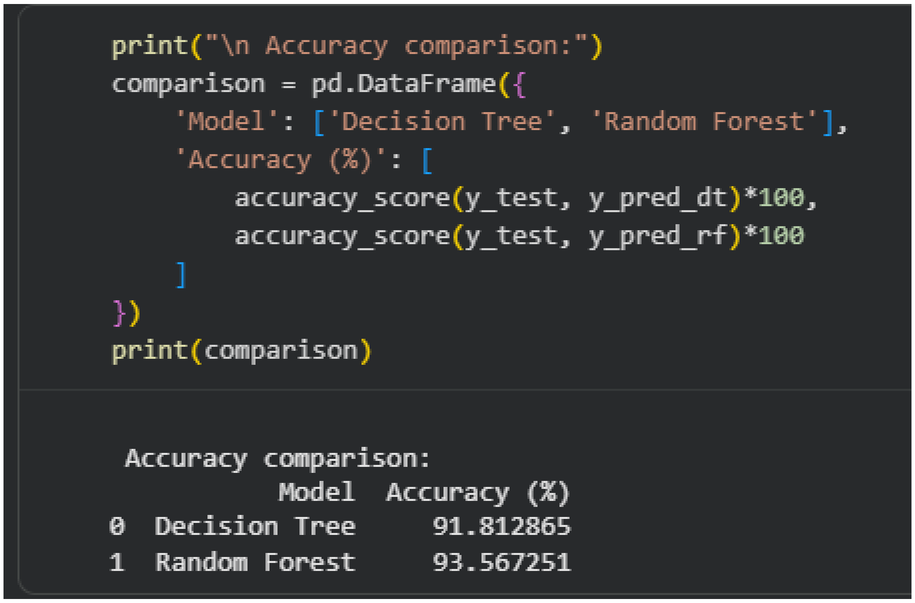
```
 Decision tree classifier evaluation:
Accuracy: 91.81%

Classification Report:
              precision    recall  f1-score   support

   malignant       0.89      0.89      0.89        64
      benign       0.93      0.93      0.93       107

    accuracy                           0.92       171
   macro avg       0.91      0.91      0.91       171
weighted avg       0.92      0.92      0.92       171
```



```
cm_dt = confusion_matrix(y_test, y_pred_dt)
plt.figure(figsize=(5,4))
sns.heatmap(cm_dt, annot=True, fmt='d', cmap='Greens', xticklabels=data.target_names, yticklabels=data.target_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion matrix of decision tree")
plt.show()
```

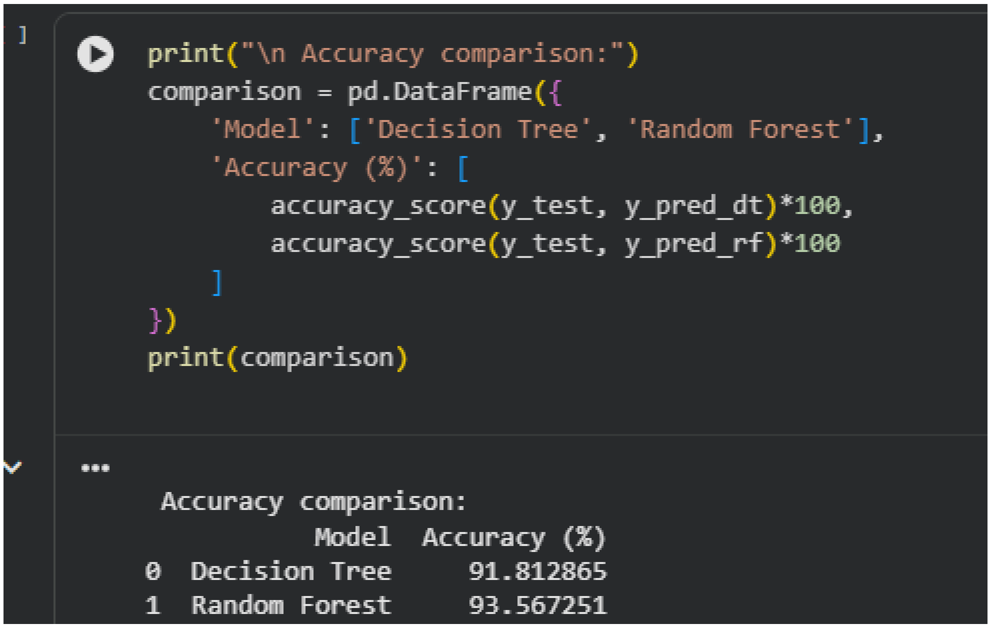


Comparison Table



```
print("\n Accuracy comparison:")
comparison = pd.DataFrame({
    'Model': ['Decision Tree', 'Random Forest'],
    'Accuracy (%)': [
        accuracy_score(y_test, y_pred_dt)*100,
        accuracy_score(y_test, y_pred_rf)*100
    ]
})
print(comparison)
```

```
 Accuracy comparison:
           Model  Accuracy (%)
0  Decision Tree     91.812865
1  Random Forest     93.567251
```

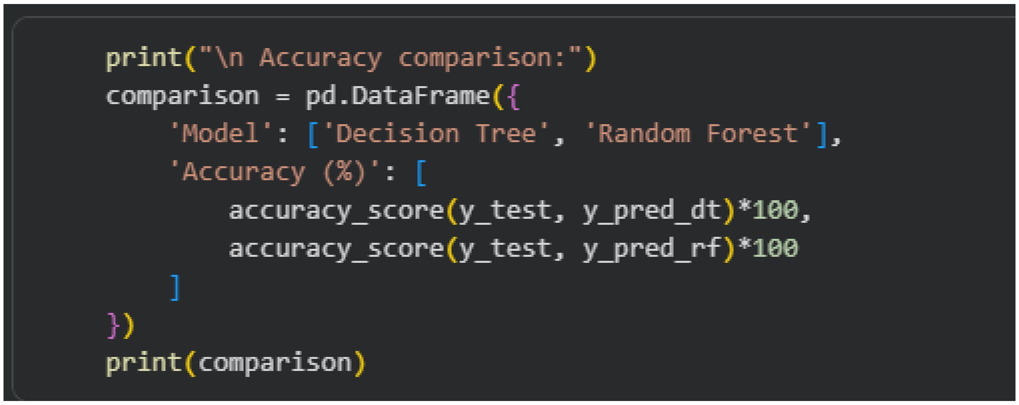


```
print("\n Accuracy comparison:")
comparison = pd.DataFrame({
    'Model': ['Decision Tree', 'Random Forest'],
    'Accuracy (%)': [
        accuracy_score(y_test, y_pred_dt)*100,
        accuracy_score(y_test, y_pred_rf)*100
    ]
})
print(comparison)
```

```
 Accuracy comparison:
           Model  Accuracy (%)
0  Decision Tree     91.812865
1  Random Forest     93.567251
```

```
print("\n Accuracy comparison:")
comparison = pd.DataFrame({
    'Model': ['Decision Tree', 'Random Forest'],
    'Accuracy (%)': [
        accuracy_score(y_test, y_pred_dt)*100,
        accuracy_score(y_test, y_pred_rf)*100
    ]
})
print(comparison)
```

```
 Accuracy comparison:
           Model  Accuracy (%)
0  Decision Tree     91.812865
1  Random Forest     93.567251
```



```
print("\n Accuracy comparison:")
comparison = pd.DataFrame({
    'Model': ['Decision Tree', 'Random Forest'],
    'Accuracy (%)': [
        accuracy_score(y_test, y_pred_dt)*100,
        accuracy_score(y_test, y_pred_rf)*100
    ]
})
print(comparison)
```