```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, f1_score
import matplotlib.pyplot as plt
import seaborn as sns

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

data = pd.read_csv("/content/credit.csv")

print(data.head())
```

```
<IPython.core.display.HTML object>

Saving credit.csv to credit (1).csv
  checking_balance  months_loan_duration credit_history
purpose  \
0           < 0 DM                     6       critical
furniture/appliances
1       1 - 200 DM                    48           good
furniture/appliances
2          unknown                    12       critical
education
3           < 0 DM                    42           good
furniture/appliances
4           < 0 DM                    24           poor
car

   amount savings_balance employment_duration  percent_of_income  \
0    1169         unknown           > 7 years                  4
1    5951        < 100 DM         1 - 4 years                  2
2    2096        < 100 DM         4 - 7 years                  2
3    7882        < 100 DM         4 - 7 years                  2
4    4870        < 100 DM         1 - 4 years                  3

   years_at_residence  age other_credit housing  existing_loans_count
\
0                   4   67         none     own                     2

1                   2   22         none     own                     1

2                   3   49         none     own                     1

3                   4   45         none   other                     1

4                   4   53         none   other                     2
```

```
          job  dependents phone default
0     skilled           1   yes      no
1     skilled           1    no     yes
2   unskilled           2    no      no
3     skilled           2    no      no
4     skilled           2    no     yes
```

```
print(data.isnull().sum())

data.fillna(method='ffill', inplace=True)

data = pd.get_dummies(data, drop_first=True)

X = data.drop("default_yes", axis=1)
y = data['default_yes']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
months_loan_duration                0
amount                              0
percent_of_income                   0
years_at_residence                  0
age                                 0
existing_loans_count                0
dependents                          0
checking_balance_< 0 DM             0
checking_balance_> 200 DM           0
checking_balance_unknown            0
credit_history_good                 0
credit_history_perfect              0
credit_history_poor                 0
credit_history_very good            0
purpose_car                         0
purpose_car0                        0
purpose_education                   0
purpose_furniture/appliances        0
purpose_renovations                 0
savings_balance_500 - 1000 DM       0
savings_balance_< 100 DM            0
savings_balance_> 1000 DM           0
savings_balance_unknown             0
employment_duration_4 - 7 years     0
employment_duration_< 1 year        0
employment_duration_> 7 years       0
employment_duration_unemployed      0
other_credit_none                   0
other_credit_store                  0
housing_own                         0
```

```
housing_rent                        0
job_skilled                         0
job_unemployed                      0
job_unskilled                       0
phone_yes                           0
default_yes                         0
dtype: int64

<ipython-input-43-6f6304d405c0>:3: FutureWarning: DataFrame.fillna
with 'method' is deprecated and will raise in a future version. Use
obj.ffill() or obj.bfill() instead.
  data.fillna(method='ffill', inplace=True)
```

```
rf = RandomForestClassifier(n_estimators=100, random_state=42,
oob_score=True)
rf.fit(X_train, y_train)

dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)

rf_pred = rf.predict(X_test)
dt_pred = dt.predict(X_test)

rf_accuracy = accuracy_score(y_test, rf_pred)
dt_accuracy = accuracy_score(y_test, dt_pred)

print(f"Random Forest Accuracy: {rf_accuracy}")
print(f"Decision Tree Accuracy: {dt_accuracy}")
```

```
Random Forest Accuracy: 0.79
Decision Tree Accuracy: 0.65
```

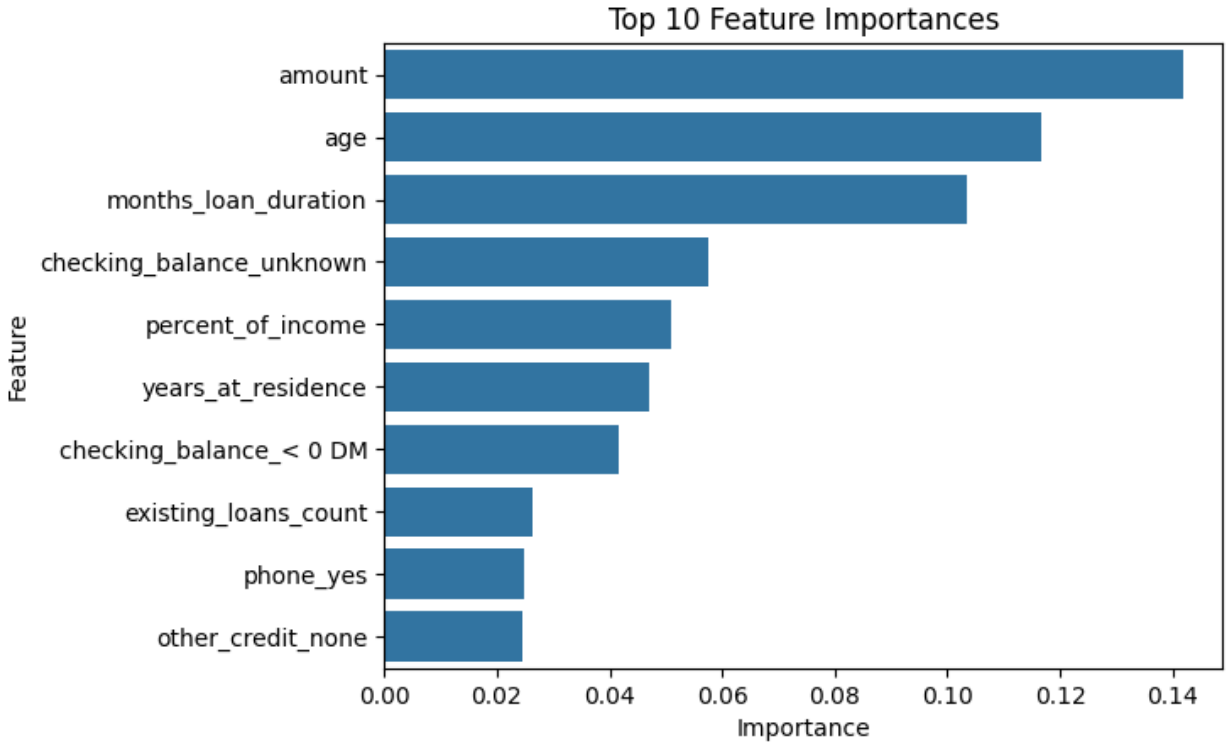
```
feature_importances = pd.DataFrame({
    'Feature': X.columns,
        'Importance': rf.feature_importances_
        }).sort_values(by='Importance', ascending=False)

print("Top predictors:")
print(feature_importances.head(10))

sns.barplot(x='Importance', y='Feature',
data=feature_importances.head(10))
plt.title('Top 10 Feature Importances')
plt.show()
```

```
Top predictors:
                     Feature  Importance
1                     amount    0.141783
4                        age    0.116724
0       months_loan_duration    0.103442
9   checking_balance_unknown    0.057422
```

```
2           percent_of_income    0.050839
3          years_at_residence    0.047147
7    checking_balance_< 0 DM     0.041699
5       existing_loans_count     0.026340
34                  phone_yes    0.024812
27         other_credit_none     0.024542
```



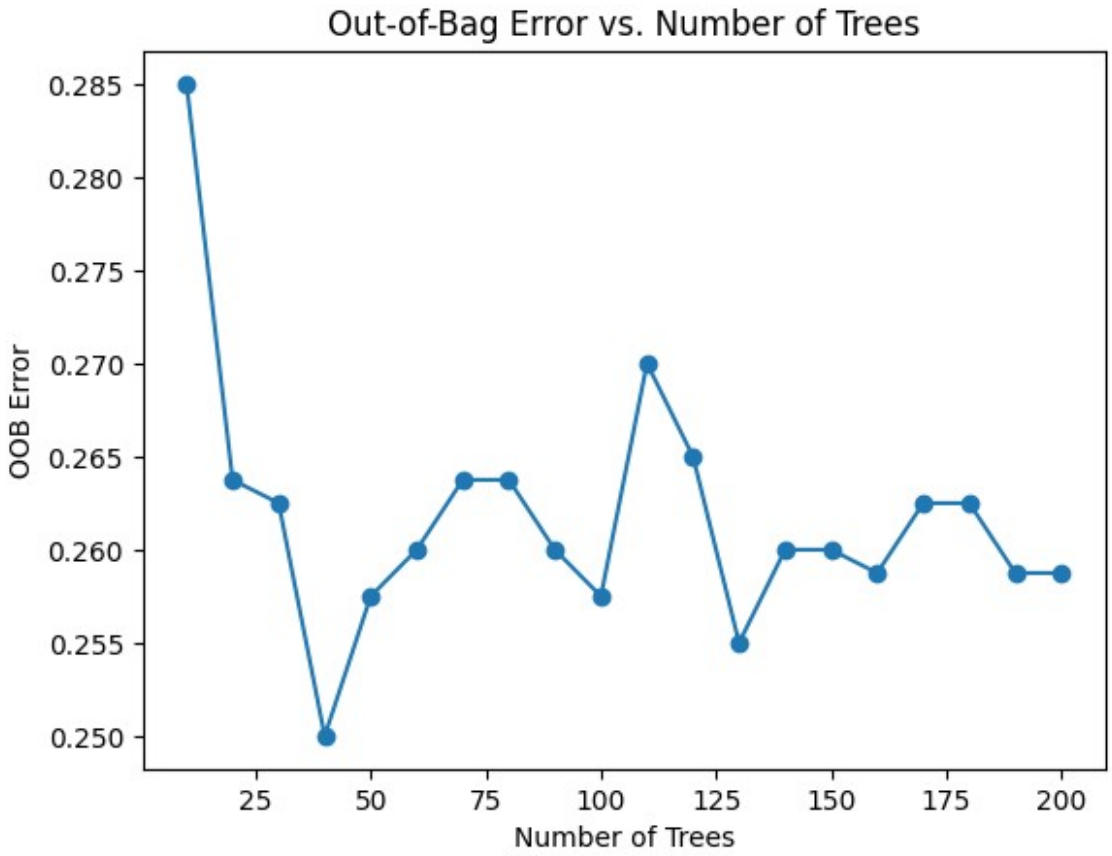

```
oob_error = 1 - rf.oob_score_
print(f"OOB Error: {oob_error}")

oob_errors = []
tree_counts = range(10, 201, 10)
for n in tree_counts:
    rf = RandomForestClassifier(n_estimators=n, random_state=42,
oob_score=True)
    rf.fit(X_train, y_train)
    oob_errors.append(1 - rf.oob_score_)

plt.plot(tree_counts, oob_errors, marker='o')
plt.title('Out-of-Bag Error vs. Number of Trees')
plt.xlabel('Number of Trees')
plt.ylabel('OOB Error')
plt.show()

OOB Error: 0.25749999999999995
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/
_forest.py:615: UserWarning: Some inputs do not have OOB scores. This
probably means too few trees were used to compute any reliable OOB
estimates.
  warn(
```



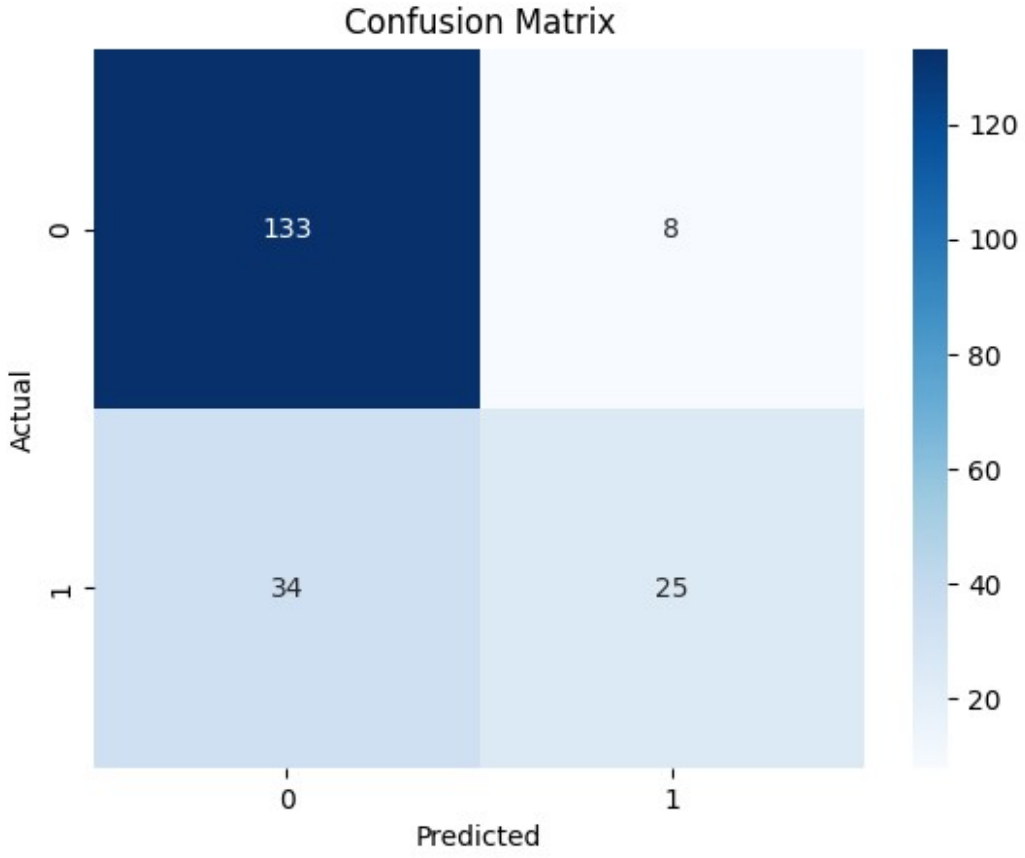

```
conf_matrix = confusion_matrix(y_test, rf_pred)
print("Confusion Matrix:")
print(conf_matrix)

sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

f1 = f1_score(y_test, rf_pred)
print(f"F1-Score: {f1}")

print("Classification Report:")
print(classification_report(y_test,rf_pred))
```

```
Confusion Matrix:
[[133   8]
 [ 34  25]]
```




```
F1-Score: 0.5434782608695652
Classification Report:
              precision    recall  f1-score   support

       False       0.80      0.94      0.86       141
        True       0.76      0.42      0.54        59

    accuracy                           0.79       200
   macro avg       0.78      0.68      0.70       200
weighted avg       0.78      0.79      0.77       200
```