```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
!pip install shap
```

```
Collecting shap
  Downloading shap-0.46.0-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from shap) (1.26.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from shap) (1.13.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from shap) (1.3.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from shap) (2.1.4)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/dist-packages (from shap) (4.66.5)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/dist-packages (from shap) (24.1)
Collecting slicer==0.0.8 (from shap)
  Downloading slicer-0.0.8-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages (from shap) (0.60.0)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from shap) (2.2.1)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba->shap) (0.43.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->shap) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->shap) (2024.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->shap) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->shap) (3.5.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
Downloading shap-0.46.0-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (540 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 540.1/540.1 kB 3.7 MB/s eta 0:00:00
Downloading slicer-0.0.8-py3-none-any.whl (15 kB)
Installing collected packages: slicer, shap
Successfully installed shap-0.46.0 slicer-0.0.8
```

```
import shap
```

```
df = pd.read_csv("/content/Churn_Modelling.csv")
print(df.head())
print(df.info())
print(df.describe())
```

```
   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   RowNumber        10000 non-null  int64 
 1   CustomerId       10000 non-null  int64 
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64 
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64 
 7   Tenure           10000 non-null  int64 
```

```
25%     2500.75000  1.562853e+07    584.000000     32.000000      3.000000
50%     5000.50000  1.569074e+07    652.000000     37.000000      5.000000
75%     7500.25000  1.575323e+07    718.000000     44.000000      7.000000
max    10000.00000  1.581569e+07    850.000000     92.000000     10.000000

             Balance  NumOfProducts    HasCrCard  IsActiveMember  \
count   10000.000000   10000.000000  10000.00000    10000.000000
mean    76485.889288       1.530200      0.70550        0.515100
std     62397.405202       0.581654      0.45584        0.499797
min         0.000000       1.000000      0.00000        0.000000
25%         0.000000       1.000000      0.00000        0.000000
50%     97198.540000       1.000000      1.00000        1.000000
75%    127644.240000       2.000000      1.00000        1.000000
max    250898.090000       4.000000      1.00000        1.000000

       EstimatedSalary        Exited
count     10000.000000  10000.000000
mean     100090.239881      0.203700
std       57510.492818      0.402769
min          11.580000      0.000000
25%       51002.110000      0.000000
50%      100193.915000      0.000000
75%      149388.247500      0.000000
max      199992.480000      1.000000
```

```
df = df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1)
df = pd.get_dummies(df, drop_first=True)
X = df.drop('Exited', axis=1)
y = df['Exited']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Train a Random Forest Classifier
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
[[2328   88]
 [ 310  274]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.96      0.92      2416
           1       0.76      0.47      0.58       584

    accuracy                           0.87      3000
   macro avg       0.82      0.72      0.75      3000
weighted avg       0.86      0.87      0.85      3000


Accuracy Score:
0.8673333333333333
```

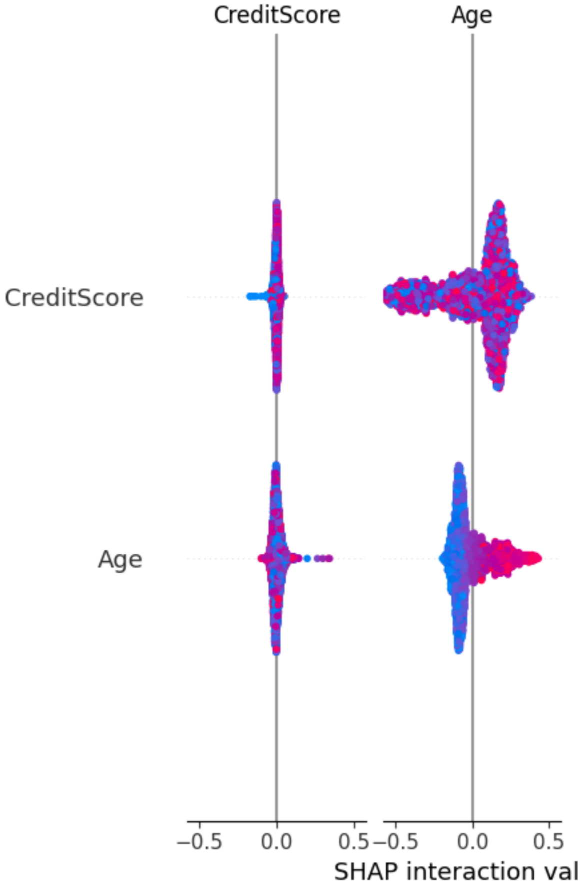
```
# Define the parameter grid for RandomSearch
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_features': ['auto', 'sqrt'],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# RandomizedSearchCV
rf_random = RandomizedSearchCV(estimator=rf_model, param_distributions=param_grid, n_iter=100, cv=3, verbose=2, random_state=42, n_jobs=-1)
```

```
rf_random.fit(X_train, y_train)

print(f"Best parameters found by RandomizedSearch: {rf_random.best_params_}")
best_rf_model = rf_random.best_estimator_

y_pred_best = best_rf_model.predict(X_test)

print("Confusion Matrix (Best Model):")
print(confusion_matrix(y_test, y_pred_best))

print("\nClassification Report (Best Model):")
print(classification_report(y_test, y_pred_best))

print("\nAccuracy Score (Best Model):")
print(accuracy_score(y_test, y_pred_best))
```

```
    estimator._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 638, in _validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validation.py", line 96, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestClassifier must be an int in the r

--------------------------------------------------------------------------------
80 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py", line 729, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 1145, in wrapper
    estimator._validate_params()
  File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 638, in _validate_params
    validate_parameter_constraints(
  File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/_param_validation.py", line 96, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestClassifier must be an int in the r

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:979: UserWarning: One or more of the test scores are non-f
 0.85842867 0.86314308 0.86157155 0.86342859 0.86485724        nan
 0.86357134        nan 0.85900018        nan        nan        nan
        nan        nan        nan        nan 0.86100016        nan
        nan        nan 0.86314308        nan 0.86400004 0.86057183
        nan        nan 0.86028559 0.86199993 0.86199975        nan
 0.86342853 0.86157142 0.86285708 0.86342853 0.85614287 0.85585712
        nan        nan 0.86214299        nan 0.85700014        nan
 0.86314308        nan        nan        nan 0.8574284         nan
        nan        nan        nan        nan        nan 0.85985671
 0.8565715         nan        nan 0.8612861  0.86228581        nan
        nan        nan 0.86300008 0.86085716        nan 0.86242899
 0.86271463        nan 0.86100016        nan        nan 0.86157197
        nan        nan 0.86399997        nan        nan        nan
        nan        nan        nan        nan        nan 0.86071477
 0.86242857        nan        nan        nan 0.85771434        nan
 0.86485724 0.86328565        nan 0.86171442]
  warnings.warn(
Best parameters found by RandomizedSearch: {'n_estimators': 50, 'min_samples_split': 2, 'min_samples_leaf': 4, 'max_features': 'sqrt'
Confusion Matrix (Best Model):
[[2333   83]
 [ 316  268]]

Classification Report (Best Model):
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      2416
           1       0.76      0.46      0.57       584

    accuracy                           0.87      3000
   macro avg       0.82      0.71      0.75      3000
weighted avg       0.86      0.87      0.85      3000


Accuracy Score (Best Model):
0.867
```

```
# Initialize SHAP explainer
explainer = shap.TreeExplainer(best_rf_model)

# Get SHAP values
shap_values = explainer.shap_values(X_test)

# If it's a binary classification, SHAP may return a list of arrays
# If it's a binary classification, SHAP may return a list of arrays
# Select shap values for class 1 (index 1) for binary classification
if isinstance(shap_values, list):
    shap_values = shap_values[1]

# Check the shape of SHAP values and X_test
print(f"Shape of SHAP values: {shap_values.shape}")
print(f"Shape of X_test: {X_test.shape}")

# Ensure the shapes match
assert shap_values.shape[0] == X_test.shape[0], "Mismatch in the number of samples"
assert shap_values.shape[1] == X_test.shape[1], "Mismatch in the number of features"

# Summary plot for the selected class
shap.summary_plot(shap_values, X_test, feature_names=X.columns)
```

```
Shape of SHAP values: (3000, 11, 2)
Shape of X_test: (3000, 11)
```



```
# Save the model as a pickle file
import pickle
model_filename = 'best_rf_model.pkl'

# Open a file in write-binary mode and save the model
with open(model_filename, 'wb') as file:
    pickle.dump(best_rf_model, file)
```