Exercise 2 Date: 08/01/2025

Random Forest Classifier - Online Shoppers Purchasing Intention

Aim: To build a Random Forest Classifier model to predict customer behavior based on the features of online shoppers, specifically predicting whether a visitor will make a purchase or not.

Algorithm:

The Random Forest Classifier is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes of the individual trees.The key idea behind Random Forest is to reduce overfitting, a common issue with individual decision trees, by averaging the results of multiple trees. Each tree in the forest is built using a random subset of the data, and when making predictions, each tree votes, and the class that receives the majority of votes is chosen as the final prediction.

At each split of a decision tree, Random Forest uses a criterion like Gini Impurity or Information Gain to choose the best feature to split the data. For a Random Forest, the decision-making process is:

$$Prediction = Mode of Predictions from All Trees$$

Where:

Each tree outputs a predicted class, and the most common class among all the trees is chosen as the final output. This technique combines the power of multiple decision trees, each performing a simple task, resulting in a more accurate and stable model.

Step 1: Import Libraries

- Import necessary Python libraries such as **pandas**, **RandomForestClassifier**, and **train_test_split** for model training and evaluation.

Step 2: Load the Dataset

- Load the dataset using pandas `read_csv` function to read the "online_shoppers_intention.csv" file into a DataFrame.

Step 3: Check for Missing Values

- Check for any missing values in the dataset using `df.isna().sum()` to identify columns with null values.

Step 4: Split the Features and Target Variable

- Separate the features (`X`) and the target variable (`y`). The target variable here is `Revenue` , which indicates whether the customer made a purchase (1) or not (0).

Step 5: Encode Categorical Variables

- Encode categorical variables (e.g., `Month` and `VisitorType`) into numerical values using **LabelEncoder.**

Step 6: Split Data into Training and Testing Sets

- Split the dataset into training and testing sets (80% train, 20% test) using `train_test_split` .

Step 7: Initialize and Train the Random Forest Model

- Initialize the **RandomForestClassifier** with 100 estimators (trees) and a random state for reproducibility.
- Train the model on the training data.

Step 8: Make Predictions

- Use the trained model to make predictions on the test dataset.

Step 9: Evaluate the Model

- Evaluate the performance of the model using various metrics like accuracy, precision, recall, F1 score, confusion matrix, and classification report.

Import the libraries

In [104…
```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings("ignore")
```

Load the Dataset

In [82]:
```
df = pd.read_csv("../online_shoppers_intention.csv")
```

In [83]:
```
df
```

Out[83]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	Spec
0	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.000000	
1	0	0.0	0	0.0	2	64.000000	0.000000	0.100000	0.000000	
2	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.000000	
3	0	0.0	0	0.0	2	2.666667	0.050000	0.140000	0.000000	
4	0	0.0	0	0.0	10	627.500000	0.020000	0.050000	0.000000	
...	...	...	...	...	...	...	...	...	...	
12325	3	145.0	0	0.0	53	1783.791667	0.007143	0.029031	12.241717	
12326	0	0.0	0	0.0	5	465.750000	0.000000	0.021333	0.000000	
12327	0	0.0	0	0.0	6	184.250000	0.083333	0.086667	0.000000	
12328	4	75.0	0	0.0	15	346.000000	0.000000	0.021053	0.000000	
12329	0	0.0	0	0.0	3	21.250000	0.000000	0.066667	0.000000	

12330 rows × 18 columns

In [84]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12330 non-null  int64
 1   Administrative_Duration  12330 non-null  float64
 2   Informational            12330 non-null  int64
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  int64
 5   ProductRelated_Duration  12330 non-null  float64
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object
 11  OperatingSystems         12330 non-null  int64
 12  Browser                  12330 non-null  int64
 13  Region                   12330 non-null  int64
 14  TrafficType              12330 non-null  int64
 15  VisitorType              12330 non-null  object
 16  Weekend                  12330 non-null  bool
 17  Revenue                  12330 non-null  bool
dtypes: bool(2), float64(7), int64(7), object(2)
memory usage: 1.5+ MB
```

Check for null values

In [85]:
```
df.isna().sum()
```

Out[85]:
```
Administrative             0
Administrative_Duration    0
Informational              0
Informational_Duration     0
ProductRelated             0
ProductRelated_Duration    0
BounceRates                0
ExitRates                  0
PageValues                 0
SpecialDay                 0
Month                      0
OperatingSystems           0
Browser                    0
Region                     0
TrafficType                0
VisitorType                0
Weekend                    0
Revenue                    0
dtype: int64
```

Check target values

In [86]:
```
df.Revenue.value_counts()
```

Out[86]:
```
Revenue
False    10422
True      1908
Name: count, dtype: int64
```

Split features and target

In [87]:
```
y = df[['Revenue']]
y
```

Out[87]:

	Revenue
0	False
1	False
2	False
3	False
4	False
...	...
12325	False
12326	False
12327	False
12328	False
12329	False

12330 rows × 1 columns

In [88]:
```
X=df.drop(columns=['Revenue'])
X
```

Out[88]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	Spec
0	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.000000	
1	0	0.0	0	0.0	2	64.000000	0.000000	0.100000	0.000000	
2	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.000000	
3	0	0.0	0	0.0	2	2.666667	0.050000	0.140000	0.000000	
4	0	0.0	0	0.0	10	627.500000	0.020000	0.050000	0.000000	
...	...	...	...	...	...	...	...	...	...	...
12325	3	145.0	0	0.0	53	1783.791667	0.007143	0.029031	12.241717	
12326	0	0.0	0	0.0	5	465.750000	0.000000	0.021333	0.000000	
12327	0	0.0	0	0.0	6	184.250000	0.083333	0.086667	0.000000	
12328	4	75.0	0	0.0	15	346.000000	0.000000	0.021053	0.000000	
12329	0	0.0	0	0.0	3	21.250000	0.000000	0.066667	0.000000	

12330 rows × 17 columns

Encode categorical variables

In [89]:
```
enc = LabelEncoder()
X['Month'] = enc.fit_transform(X['Month'])
X['VisitorType'] = enc.fit_transform(X['VisitorType'])
y['Revenue'] = enc.fit_transform(y['Revenue'])
```

Split the data into training and testing sets

In [90]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Initialize the Random Forest Classifier

In [103…
```
warnings.filterwarnings("ignore")
warnings.filterwarnings("ignore", category=FutureWarning)

y_train = np.ravel(y_train)

# Parameter grid
param_grid = {
    'n_estimators': [50, 100, 150, 200],
    'max_depth': [None, 10, 20, 30, 40],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2'],
    'bootstrap': [True, False]
}

rfc = RandomForestClassifier(random_state=42)

grid_search = GridSearchCV(estimator=rfc, param_grid=param_grid,
                           cv=5, n_jobs=-1, verbose=0, scoring='accuracy')

grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

print(f"Best Parameters: {best_params}")

y_pred = best_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the best model: {accuracy:.4f}")

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
Best Parameters: {'bootstrap': False, 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}
Accuracy of the best model: 0.8970
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      2055
           1       0.76      0.55      0.64       411

    accuracy                           0.90      2466
   macro avg       0.84      0.76      0.79      2466
weighted avg       0.89      0.90      0.89      2466
```

In [110…
```
rfc = RandomForestClassifier(n_estimators=150, max_depth=10, max_features='sqrt', min_samples_leaf=1, min_samples_split=2, bootstrap=False, random_state=42)
```

Train the model

In [111…
```
rfc.fit(X_train, y_train)
```

Out[111…
```
RandomForestClassifier(bootstrap=False, max_depth=10, n_estimators=150,
                       random_state=42)
```

In [112…
```
y_pred = rfc.predict(X_test)
```

Performance metrics

In [113…
```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

In [114…
```
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)
```

```
Accuracy: 0.8970
Precision: 0.7643
Recall: 0.5523
F1 Score: 0.6412

Confusion Matrix:
[[1985   70]
 [ 184  227]]

Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      2055
           1       0.76      0.55      0.64       411

    accuracy                           0.90      2466
   macro avg       0.84      0.76      0.79      2466
weighted avg       0.89      0.90      0.89      2466
```

Result

A Random Forest Classifier was built to predict whether online shop visitors will turn into paying customers. The model was trained with a set of hyperparameters optimized through grid search and cross-validation. The final model achieved an accuracy of 89.70%.

Key Differences between DecisionTreeClassifier and RandomForestClassifier Results

The Random Forest Classifier (RFC) outperformed the Decision Tree in accuracy, making it the better model for this task.
While Decision Trees are more prone to overfitting, RFC mitigates this risk by averaging predictions from multiple trees, leading to better generalization.
Although Decision Trees are faster to train, RFC handles larger datasets efficiently due to its ensemble approach, with the improved performance justifying the slightly longer training time.

If model interpretability and simplicity are more important, the Decision Tree Classifier could be preferred. If the focus is on accuracy and minimizing overfitting, the Random Forest Classifier might be the better choice.
