# Problem Statement:-

**Q4.** Imagine you working as a sale manager now you need to predict the Revenue and whether that particular revenue is on the weekend or not and find the Informational_Duration using the Ensemble learning algorithm

Dataset link :- https://www.kaggle.com/datasets/henrysue/online-shoppers-intention

In [1]:
```
## Import the necessary libraries:-
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [3]:
```
# Load the dataset
data = pd.read_csv(r"C:\Users\hrush\Downloads\archive (3)\online_shoppers_intention.csv")
```

In [4]:
```
data.head()
```

Out[4]:

| | Administrative | Administrative_Duration | Informational | Informational_Duration | ProductRelated | ProductRelated_Duration | BounceRates | ExitR |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0.0 | 0 | 0.0 | 1 | 0.000000 | 0.20 | |
| **1** | 0 | 0.0 | 0 | 0.0 | 2 | 64.000000 | 0.00 | |
| **2** | 0 | 0.0 | 0 | 0.0 | 1 | 0.000000 | 0.20 | |
| **3** | 0 | 0.0 | 0 | 0.0 | 2 | 2.666667 | 0.05 | |
| **4** | 0 | 0.0 | 0 | 0.0 | 10 | 627.500000 | 0.02 | |

In [5]:
```
data.shape
```

Out[5]:
```
(12330, 18)
```

In [7]:
```
data.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
```

In [8]:
```
# Convert target variable to categorical
data['Revenue'] = data['Revenue'].astype(str)
```

In [9]:
```
# Extract the relevant features for revenue prediction
features = data.drop(['Revenue'], axis=1)
```

In [10]:
```
# Convert weekend column to numerical values (0 for False, 1 for True)
features['Weekend'] = features['Weekend'].astype(int)
```

In [11]:
```
# Convert informational duration column to numerical values (0 for False, 1 for True)
features['Informational_Duration'] = features['Informational_Duration'].apply(lambda x: 1 if x > 0 else 0)
```

In [12]:
```
# Encode categorical features using one-hot encoding
features = pd.get_dummies(features)
```

In [13]:
```
# Extract the target variable (Revenue)
```

```
target = data['Revenue']
```

In [14]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

## Doing Model Building Using Random Forest Classifier

In [15]:
```
# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [16]:
```
# Train the classifier
rf_classifier.fit(X_train, y_train)
```

Out[16]:
```
▾        RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [17]:
```
# Predict the revenue on the test set
y_pred = rf_classifier.predict(X_test)
```

In [18]:
```
# Calculate accuracy and confusion matrix
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
```

In [19]:
```
# Print the accuracy and confusion matrix
print("Accuracy:", accuracy)
print("Confusion Matrix:")
print(confusion)
```

```
Accuracy: 0.8961881589618816
Confusion Matrix:
[[1985   70]
 [ 186  225]]
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js