1. Loading and Preprocessing the Dataset

In [6]:
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

# Load the dataset
df = pd.read_csv(r'C:\Users\harik\OneDrive\Documents\NWU DOCS\ML\week6\customer chu

# Drop irrelevant columns
df = df.drop(columns=['RowNumber', 'CustomerId', 'Surname'])

# Label encode the 'Gender' column
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])  # Female=0, Male=1

# One-hot encode 'Geography' column
column_transformer = ColumnTransformer(
    transformers=[('geo', OneHotEncoder(drop='first'), ['Geography'])],
    remainder='passthrough'
)

# Apply transformation to the features
X = df.drop('Exited', axis=1)  # Features
X = column_transformer.fit_transform(X)

# Target variable
y = df['Exited']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta

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

2. Implementing Machine Learning Models a. Decision Tree

In [7]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

# Initialize Decision Tree model
decision_tree = DecisionTreeClassifier(random_state=42)

# Train the model
decision_tree.fit(X_train_scaled, y_train)

# Make predictions
y_pred_tree = decision_tree.predict(X_test_scaled)
```

```
# Evaluate the model
print("Decision Tree Classification Report:\n", classification_report(y_test, y_pre
```

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

           0       0.88      0.88      0.88      2416
           1       0.52      0.53      0.52       584

    accuracy                           0.81      3000
   macro avg       0.70      0.70      0.70      3000
weighted avg       0.81      0.81      0.81      3000
```

b. K-Nearest Neighbors (KNN)

In [8]:
```
from sklearn.neighbors import KNeighborsClassifier

# Initialize KNN model
knn = KNeighborsClassifier(n_neighbors=5)

# Train the model
knn.fit(X_train_scaled, y_train)

# Make predictions
y_pred_knn = knn.predict(X_test_scaled)

# Evaluate the model
print("KNN Classification Report:\n", classification_report(y_test, y_pred_knn))
```

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

           0       0.86      0.95      0.90      2416
           1       0.63      0.36      0.46       584

    accuracy                           0.83      3000
   macro avg       0.74      0.65      0.68      3000
weighted avg       0.81      0.83      0.81      3000
```

c. Random Forest

In [9]:
```
from sklearn.ensemble import RandomForestClassifier

# Initialize Random Forest model
random_forest = RandomForestClassifier(random_state=42)

# Train the model
random_forest.fit(X_train_scaled, y_train)

# Make predictions
y_pred_rf = random_forest.predict(X_test_scaled)
```

```
# Evaluate the model
print("Random Forest Classification Report:\n", classification_report(y_test, y_pre
```

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

           0       0.88      0.96      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
```

3. Dimensionality Reduction Using PCA

In [10]:
```
from sklearn.decomposition import PCA

# Initialize PCA and reduce to 2 components
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

# Train the Decision Tree on PCA-reduced data
decision_tree.fit(X_train_pca, y_train)
y_pred_pca = decision_tree.predict(X_test_pca)

# Evaluate the model on PCA-reduced data
print("Decision Tree with PCA Classification Report:\n", classification_report(y_te
```

```
Decision Tree with PCA Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.83      0.84      2416
           1       0.34      0.36      0.35       584

    accuracy                           0.74      3000
   macro avg       0.59      0.59      0.59      3000
weighted avg       0.74      0.74      0.74      3000
```

1. How did the performance of each model compare on your dataset?

The classification reports indicate that:

Decision Tree: Accuracy: 81% Precision: High for class 0 (0.88), but relatively low for class 1 (0.52). Recall: 88% for class 0, but only 53% for class 1. This model performs well at predicting nonchurners (class 0) but struggles significantly with identifying churners (class 1).

KNearest Neighbors (KNN): Accuracy: 83% Precision: Slightly lower for class 1 (0.63), while class 0 retains good precision (0.86). Recall: 95% for class 0, but only 36% for class 1. Similar to the Decision Tree, KNN excels at predicting nonchurners but has a poor recall for churners.

Random Forest: Accuracy: 87% Precision: 0.76 for class 1 indicates a better ability to identify churners compared to the other models. Recall: 46% for class 1 shows room for improvement but is better than the previous models. This model offers the best balance between precision and recall, suggesting it could be the most reliable for identifying customers at risk of churning.

Decision Tree with PCA: Accuracy: 74% Precision: 0.34 for class 1 indicates a significant drop in ability to identify churners. Recall: 36% for class 1, reflecting the challenges of PCA in this case. Applying PCA resulted in a decrease in performance, particularly for class 1, likely due to the loss of important information during dimensionality reduction.

2. Did the application of dimensionality reduction improve or worsen the results? Why?

The application of PCA clearly worsened the results for the Decision Tree model.

Comparison with and without PCA: The Decision Tree classification report without PCA had an accuracy of 81%, while with PCA, the accuracy dropped to 74%. The precision and recall for churners (class 1) also significantly declined from 0.52 to 0.34 and from 0.53 to 0.36, respectively.

Reasoning: PCA reduces dimensionality by focusing on variance and can eliminate some features that might hold predictive power, which is crucial for identifying class 1 (churners). Since the original features contained valuable information regarding churn, the loss of this information through PCA likely contributed to the model's poorer performance.

3. Which model would you choose for deployment based on the results, and why?

Random Forest is the preferred choice for deployment based on the results.

Rationale: With an accuracy of 87%, it not only performs the best among the models tested but also shows a good balance between precision (0.76) and recall (0.46) for class 1. This balance is critical in applications like customer churn prediction, where both false negatives (failing to identify a churner) and false positives (incorrectly identifying a nonchurner as a churner) can have significant consequences for customer retention strategies.

Considerations: The Random Forest model's robustness to overfitting, its ability to handle large feature sets, and its relative immunity to noise make it a strong candidate for deployment. It also provides insights into feature importance, which can be valuable for business decisions regarding customer retention strategies.

In summary, while the Decision Tree and KNN models provided decent performance, Random Forest stands out as the most reliable option for deployment, especially considering the high stakes involved in predicting customer churn.