Customer Churn Prediction with A/B Testing Framework

Project Overview:

This project aims to analyze customer churn in a banking context, improve prediction accuracy using advanced machine learning techniques, and outline an A/B testing framework to validate retention strategies. The primary objectives include identifying factors that influence churn and testing targeted interventions.

In [20]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report
```

In [22]:
```
# Load the dataset
data = pd.read_csv(r"C:/Users/Asus/Downloads/Churn_Modelling.csv")

# Display basic information about the dataset
print(data.info())
print(data.describe())
print(data.head())
```

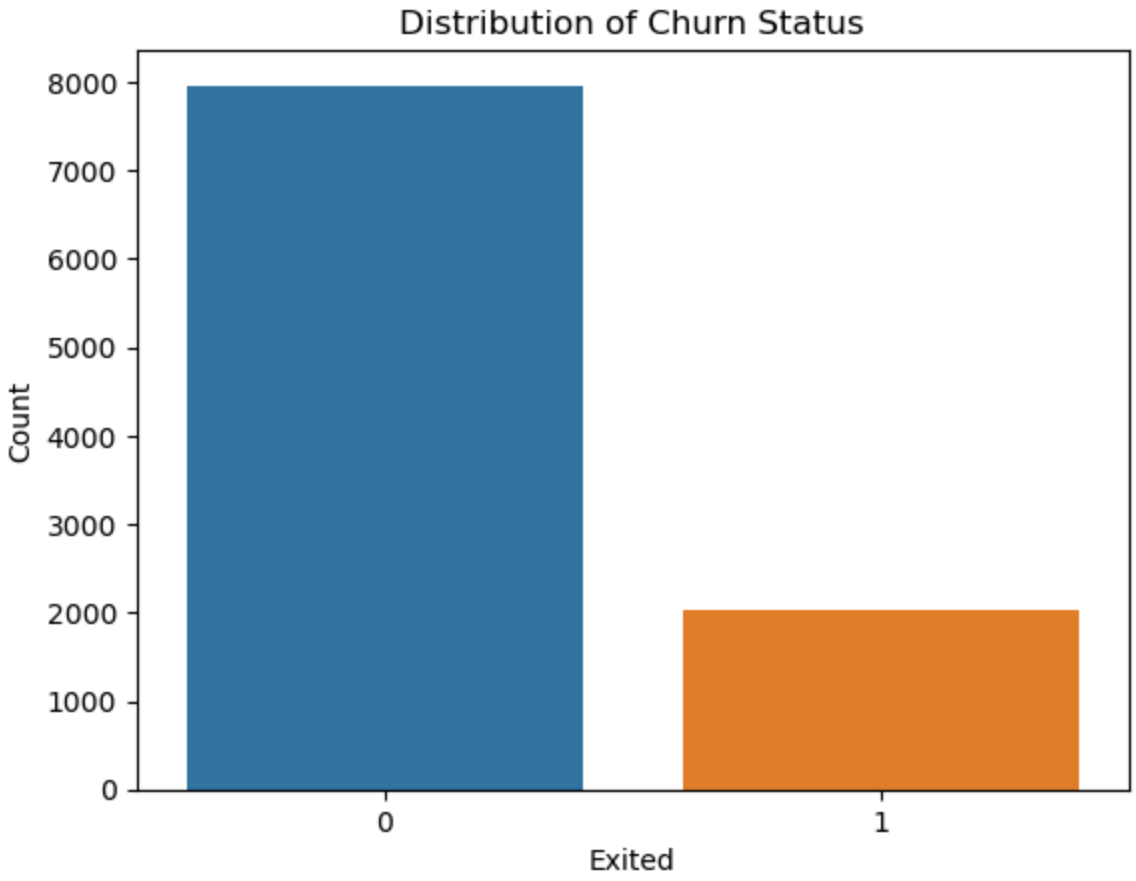
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   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  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
None
         RowNumber    CustomerId   CreditScore           Age         Tenur
e  \
count  10000.00000  1.000000e+04  10000.000000  10000.000000  10000.00000
0
mean    5000.50000  1.569094e+07    650.528800     38.921800      5.01280
0
std     2886.89568  7.193619e+04     96.653299     10.487806      2.89217
4
min        1.00000  1.556570e+07    350.000000     18.000000      0.00000
0
25%     2500.75000  1.562853e+07    584.000000     32.000000      3.00000
0
50%     5000.50000  1.569074e+07    652.000000     37.000000      5.00000
0
75%     7500.25000  1.575323e+07    718.000000     44.000000      7.00000
0
max    10000.00000  1.581569e+07    850.000000     92.000000     10.00000
0

             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
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

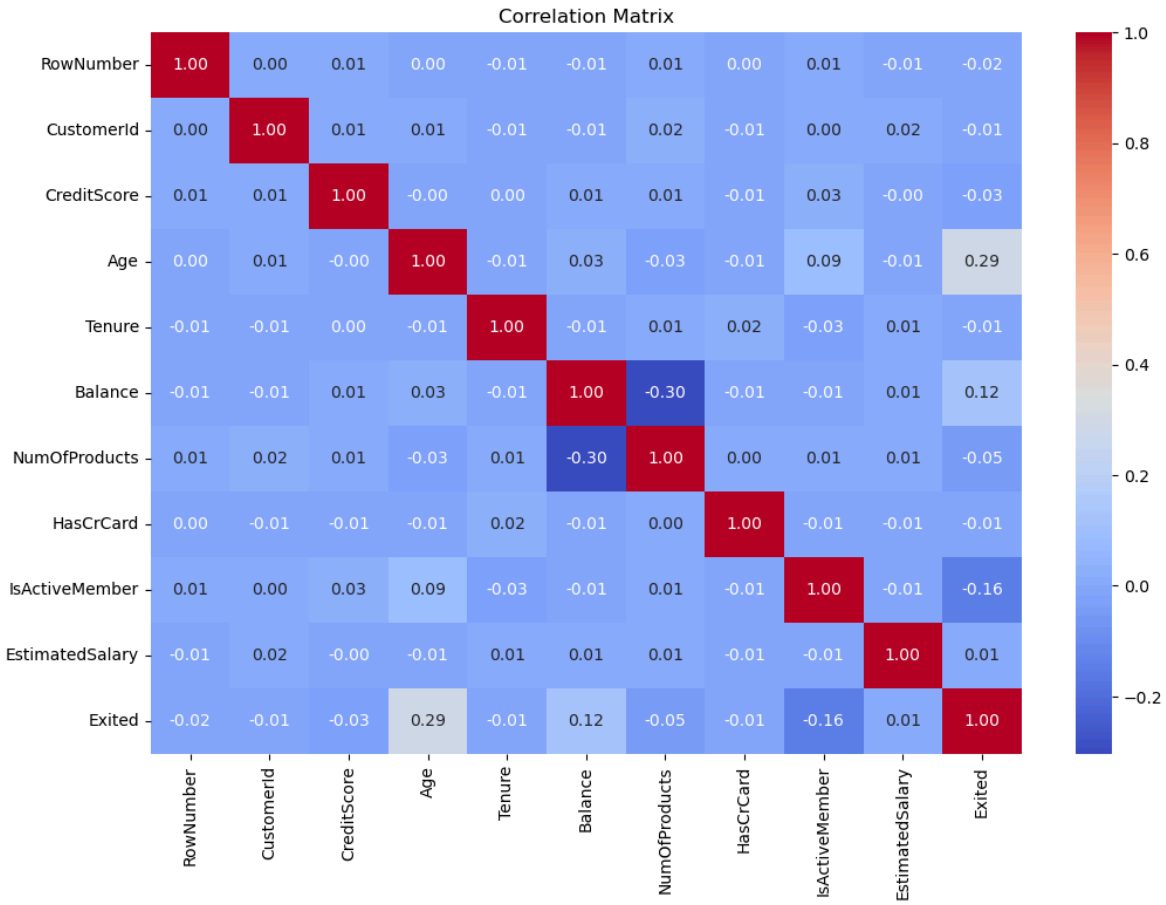
In [23]:
```
print("Missing values in each column:")
print(data.isnull().sum())
```

```
Missing values in each column:
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [24]:
```
sns.countplot(x='Exited', data=data)
plt.title('Distribution of Churn Status')
plt.xlabel('Exited')
plt.ylabel('Count')
plt.show()
```



In [25]:
```
numeric_data = data.select_dtypes(include=[np.number])  # Selecting only nu
plt.figure(figsize=(12, 8))
sns.heatmap(numeric_data.corr(), annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



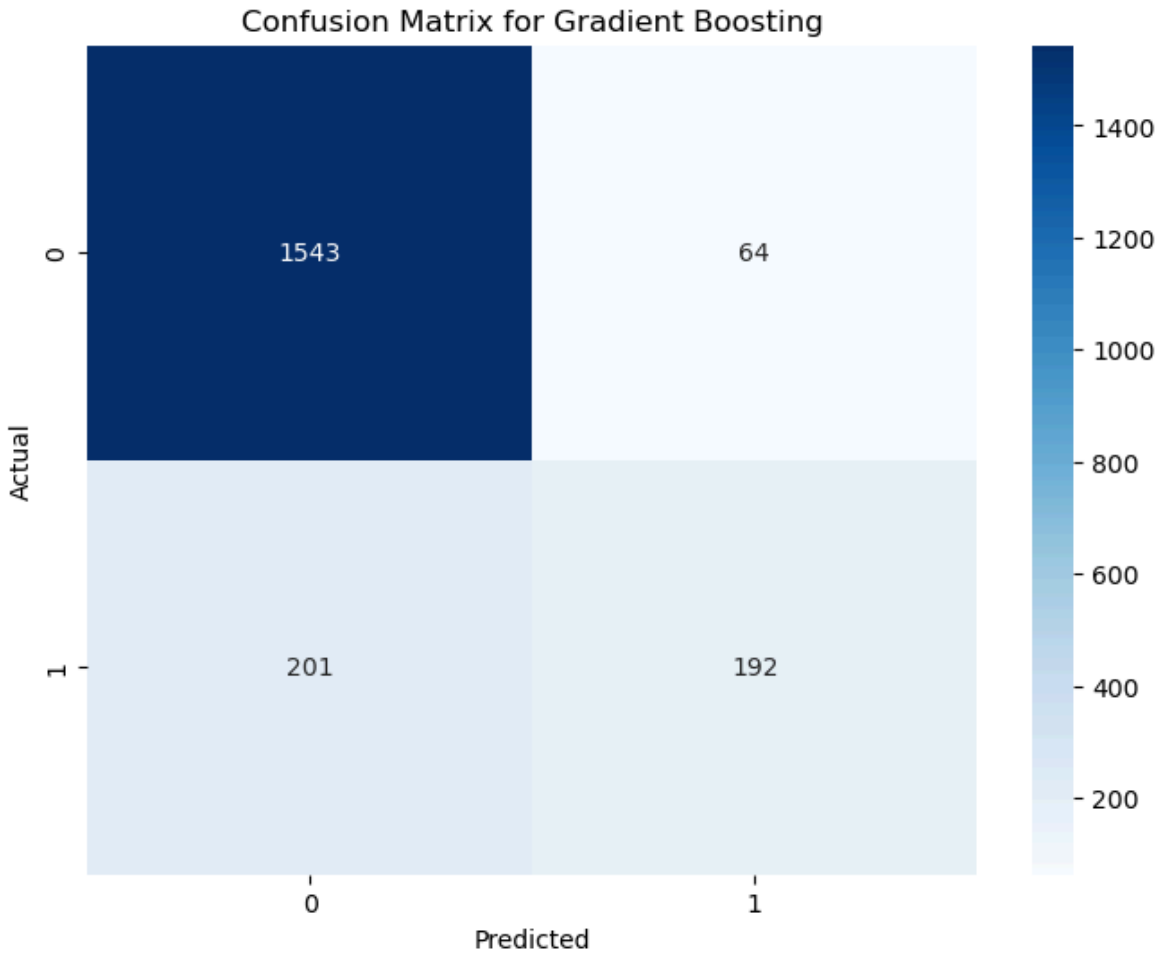
In [26]:
```
# Preparing data for modeling
# Dropping columns that are not needed for prediction
data = data.drop(columns=['RowNumber', 'CustomerId', 'Surname'])

# Encoding categorical variables
data = pd.get_dummies(data, drop_first=True)

# Split the data into features and target
X = data.drop('Exited', axis=1)
y = data['Exited']

# A/B Testing Setup
# Simulating A/B test groups
data['group'] = np.random.choice(['A', 'B'], size=len(data))

# Analyze churn based on group
churn_rates = data.groupby('group')['Exited'].mean()
print("Churn Rates by Group:\n", churn_rates)
```

```
Churn Rates by Group:
 group
A    0.200517
B    0.206924
Name: Exited, dtype: float64
```

In [27]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# Predictive Modeling with Gradient Boosting
gb_model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1,
gb_model.fit(X_train, y_train)
gb_pred = gb_model.predict(X_test)

# Evaluate Model Performance
print("Gradient Boosting Classification Report:")
print(classification_report(y_test, gb_pred))
```

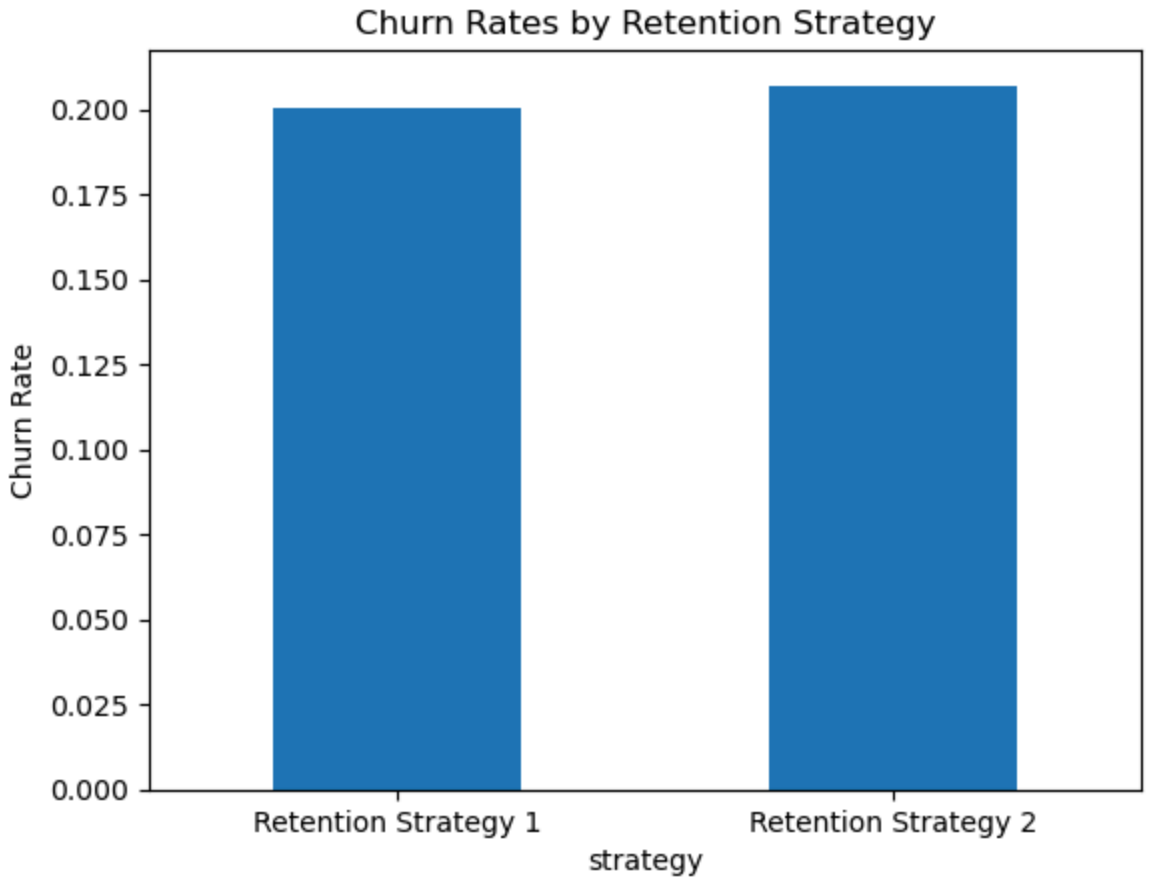
```
Gradient Boosting Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.96      0.92      1607
           1       0.75      0.49      0.59       393

    accuracy                           0.87      2000
   macro avg       0.82      0.72      0.76      2000
weighted avg       0.86      0.87      0.86      2000
```

In [28]:
```
# Confusion Matrix
confusion_mat = confusion_matrix(y_test, gb_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_mat, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix for Gradient Boosting')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

# A/B Test Evaluation
# Analyzing the churn rate of the two groups after applying a retention str
data['strategy'] = np.where(data['group'] == 'A', 'Retention Strategy 1', '
ab_results = data.groupby('strategy')['Exited'].mean()
print("Churn Rates After A/B Testing:\n", ab_results)
```




```
Churn Rates After A/B Testing:
 strategy
Retention Strategy 1    0.200517
Retention Strategy 2    0.206924
Name: Exited, dtype: float64
```

In [29]:
```
# Visualize A/B Test Results
ab_results.plot(kind='bar', title='Churn Rates by Retention Strategy', ylab
plt.xticks(rotation=0)
plt.show()
```



In [ ]: