Case Study: Predicting Driver Attrition for Ola Using Ensemble Learning

---

1. Define Problem Statement

Objective

Ola faces a significant challenge in retaining its drivers, leading to operational inefficiencies and increased costs. Driver churn impacts customer satisfaction, increases driver acquisition costs, and disrupts service reliability. By predicting driver attrition using demographic, tenure, and performance data, Ola can:

- Identify at-risk drivers.
- Implement targeted retention strategies.
- Ensure a consistent driver base to maintain service continuity.

Business Problem

The goal is to develop a predictive model that identifies whether a driver is likely to leave Ola based on various attributes like:

- **Demographics**: Age, gender, city, education level.
- **Tenure Information**: Joining and last working dates.
- **Performance Data**: Quarterly ratings, income, total business value, and grade.

By analyzing historical data, we aim to recommend actionable strategies to reduce churn and improve retention.

---

2. Exploratory Data Analysis (EDA)

2.1 Observations on Data

We begin by inspecting the dataset to understand its structure, data types, and missing values.

Inspecting the Dataset**

```
import pandas as pd

# Load the dataset
file_path = '/Users/gopalmacbook/Downloads/ola_driver_scaler.csv'
data = pd.read_csv(file_path)
```

```python
# Initial inspection of the dataset
data_shape = data.shape
data_info = data.info()
missing_values = data.isnull().sum()
data_description = data.describe(include='all')

print("Shape of the Dataset:", data_shape)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Sr No.                19104 non-null  int64  
 1   MMM-YY                19104 non-null  object 
 2   Driver_ID             19104 non-null  int64  
 3   Age                   19043 non-null  float64
 4   Gender                19052 non-null  float64
 5   City                  19104 non-null  object 
 6   Education_Level       19104 non-null  int64  
 7   Income                19104 non-null  int64  
 8   Dateofjoining         19104 non-null  object 
 9   LastWorkingDate       1616 non-null   object 
 10  Joining Designation   19104 non-null  int64  
 11  Grade                 19104 non-null  int64  
 12  Total Business Value  19104 non-null  int64  
 13  Quarterly Rating      19104 non-null  int64  
dtypes: float64(2), int64(8), object(4)
memory usage: 2.0+ MB
Shape of the Dataset: (19104, 14)
```

```python
print("\nMissing Values:\n", missing_values)
```

```
Missing Values:
 Sr No.                      0
MMM-YY                     0
Driver_ID                  0
Age                       61
Gender                    52
City                       0
Education_Level            0
Income                     0
Dateofjoining              0
LastWorkingDate        17488
Joining Designation        0
Grade                      0
Total Business Value       0
```

```
Quarterly Rating            0
dtype: int64

print("\nStatistical Summary:\n", data_description)


Statistical Summary:
               Sr No.    MMM-YY     Driver_ID           Age
Gender  \
count   19104.000000     19104  19104.000000  19043.000000
19052.000000
unique           NaN        24           NaN           NaN
NaN
top              NaN  01/01/19           NaN           NaN
NaN
freq             NaN      1022           NaN           NaN
NaN
mean     9551.500000       NaN   1415.591133     34.668435
0.418749
std      5514.994107       NaN    810.705321      6.257912
0.493367
min         0.000000       NaN      1.000000     21.000000
0.000000
25%      4775.750000       NaN    710.000000     30.000000
0.000000
50%      9551.500000       NaN   1417.000000     34.000000
0.000000
75%     14327.250000       NaN   2137.000000     39.000000
1.000000
max     19103.000000       NaN   2788.000000     58.000000
1.000000

         City  Education_Level        Income Dateofjoining
LastWorkingDate  \
count   19104     19104.000000  19104.000000         19104
1616
unique     29              NaN           NaN           869
493
top       C20              NaN           NaN      23/07/15
29/07/20
freq     1008              NaN           NaN           192
70
mean      NaN         1.021671  65652.025126           NaN
NaN
std       NaN         0.800167  30914.515344           NaN
NaN
min       NaN         0.000000  10747.000000           NaN
NaN
25%       NaN         0.000000  42383.000000           NaN
NaN
```

```
50%        NaN          1.000000   60087.000000           NaN
NaN
75%        NaN          2.000000   83969.000000           NaN
NaN
max        NaN          2.000000  188418.000000           NaN
NaN

        Joining Designation         Grade  Total Business Value  \
count          19104.000000  19104.000000          1.910400e+04
unique                  NaN           NaN                   NaN
top                     NaN           NaN                   NaN
freq                    NaN           NaN                   NaN
mean               1.690536      2.252670          5.716621e+05
std                0.836984      1.026512          1.128312e+06
min                1.000000      1.000000         -6.000000e+06
25%                1.000000      1.000000          0.000000e+00
50%                1.000000      2.000000          2.500000e+05
75%                2.000000      3.000000          6.997000e+05
max                5.000000      5.000000          3.374772e+07

        Quarterly Rating
count       19104.000000
unique               NaN
top                  NaN
freq                 NaN
mean            2.008899
std             1.009832
min             1.000000
25%             1.000000
50%             2.000000
75%             3.000000
max             4.000000
```

Output

- **Shape**: 19,104 rows and 14 columns.
- **Missing Values**:
  - `Age`: 61 missing values.
  - `Gender`: 52 missing values.
  - `LastWorkingDate`: 17,488 missing values (applicable only to drivers who left).
- **Summary Statistics**:
  - `Age`: Ranges between 21 and 58.
  - `Income`: Highly variable, ranging from ₹10,747 to ₹188,418.
  - `Quarterly Rating`: Average rating is 2.01, with values between 1 and 4.

2.2 Univariate Analysis

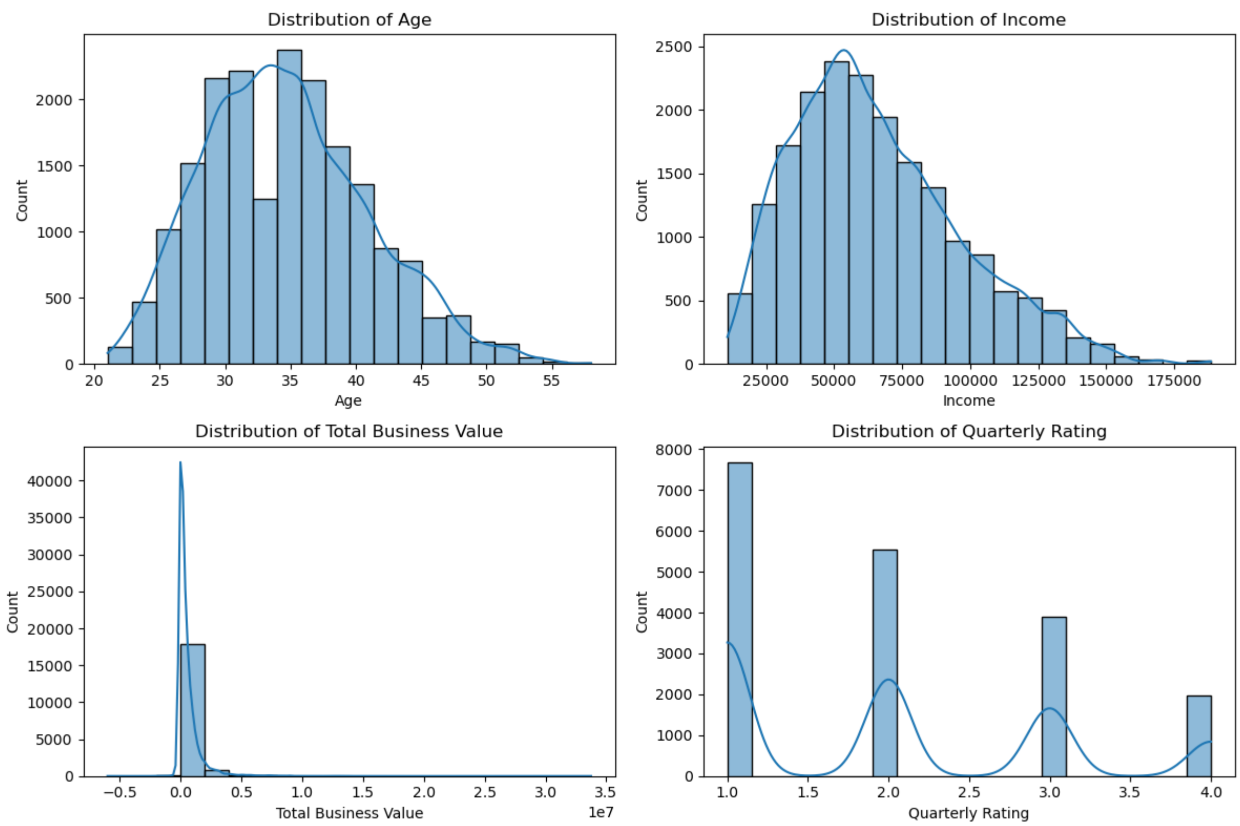
Here, we analyze individual variables to understand their distributions.

Numerical Variables

We focus on `Age`, `Income`, `Total Business Value`, and `Quarterly Rating`.

Distribution of Numerical Variables**

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Univariate Analysis for Numerical Variables
numerical_features = ['Age', 'Income', 'Total Business Value',
'Quarterly Rating']

plt.figure(figsize=(12, 8))
for i, feature in enumerate(numerical_features, 1):
    plt.subplot(2, 2, i)
    sns.histplot(data[feature], kde=True, bins=20)
    plt.title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()
```



Insights

- **Age**: Most drivers are aged between 30 and 40.

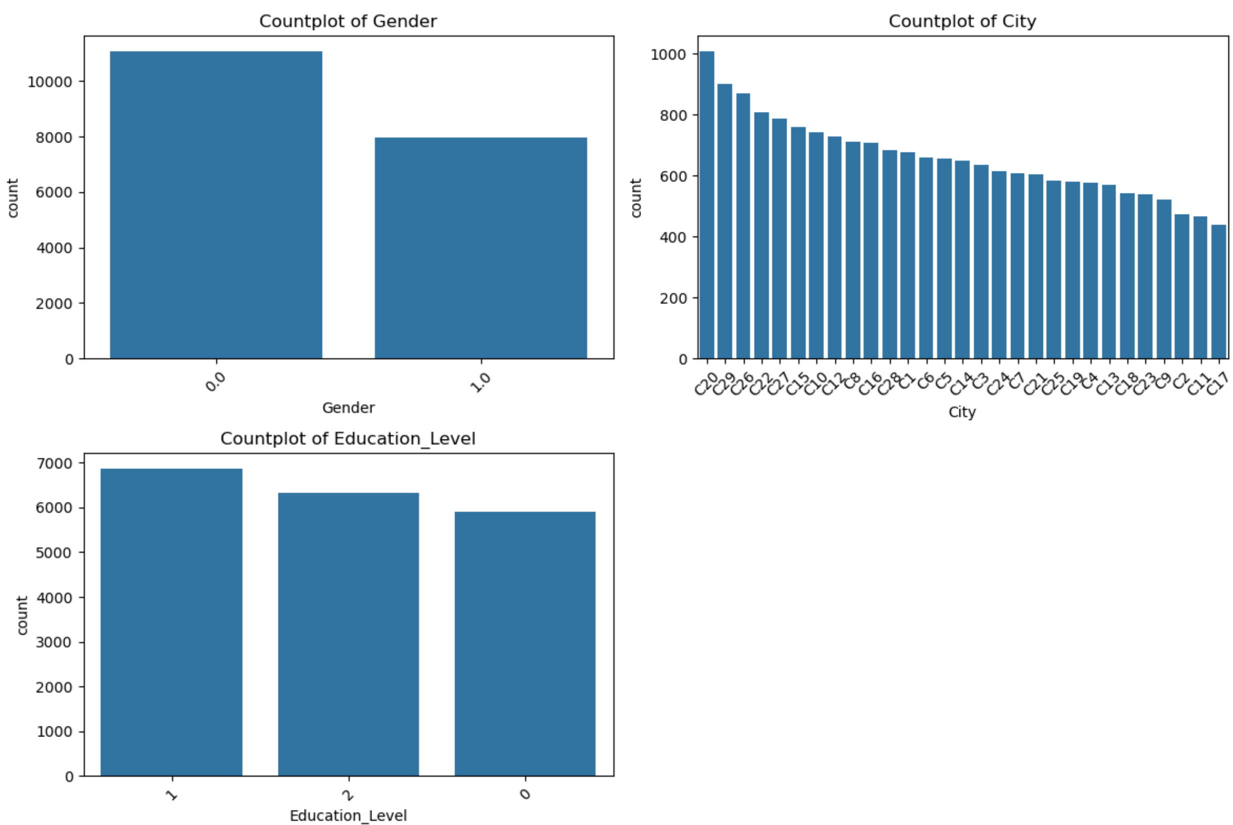
- **Income**: Right-skewed; most drivers earn between ₹40,000 and ₹80,000.
- **Total Business Value**: Highly variable, with significant outliers.
- **Quarterly Rating**: Ratings are concentrated around 1 and 2.

Categorical Variables

We analyze the distribution of categorical features: `Gender`, `City`, and `Education_Level`.

Count Plots for Categorical Variables**

```
# Univariate Analysis for Categorical Variables
categorical_features = ['Gender', 'City', 'Education_Level']

plt.figure(figsize=(12, 8))
for i, feature in enumerate(categorical_features, 1):
    plt.subplot(2, 2, i)
    sns.countplot(x=data[feature], 
order=data[feature].value_counts().index)
    plt.title(f'Countplot of {feature}')
    plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Insights

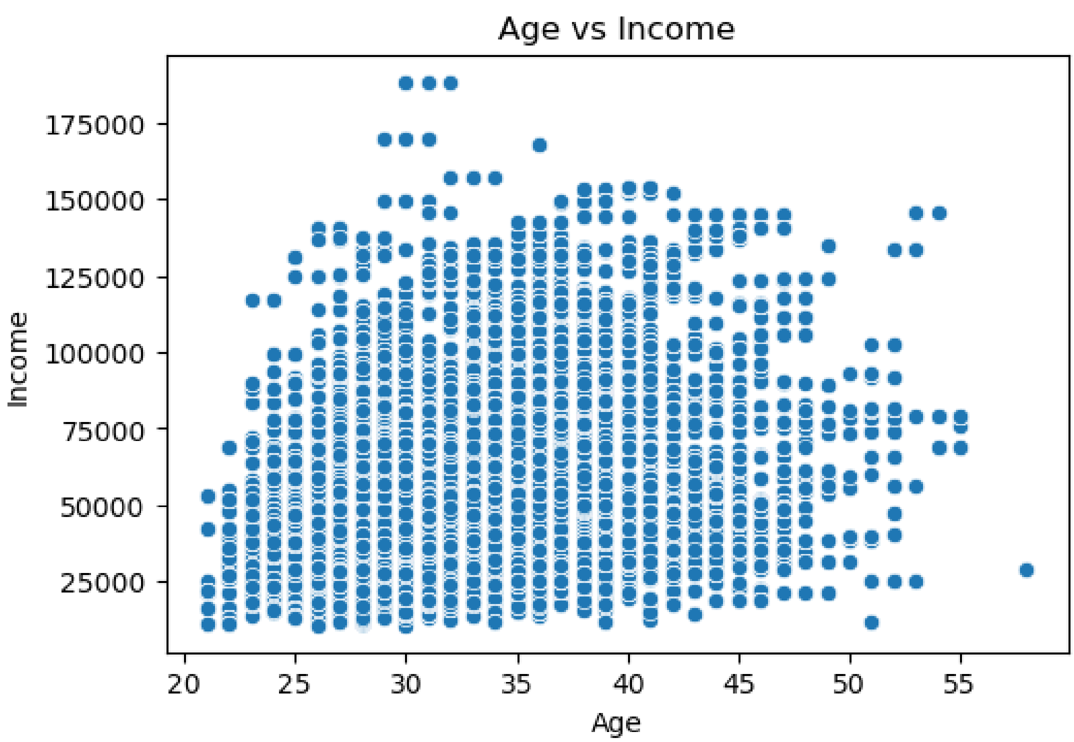
- **Gender**: Males significantly outnumber females.
- **City**: Some cities (e.g., C20) dominate the dataset.
- **Education Level**: Drivers with higher education (graduates) are fewer than those with basic education.

2.3 Bivariate Analysis

We explore relationships between key variables to uncover patterns.

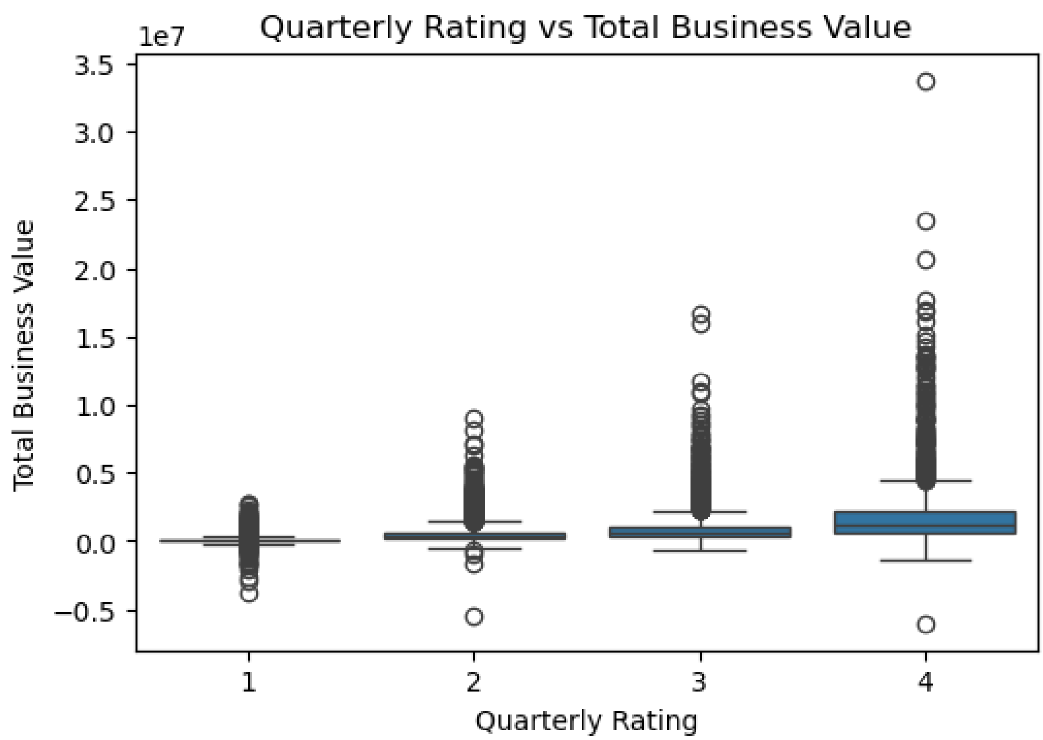
Bivariate Relationships**

```
# Relationship between Age and Income
plt.figure(figsize=(6, 4))
sns.scatterplot(x=data['Age'], y=data['Income'])
plt.title("Age vs Income")
plt.show()
```




```
# Relationship between Quarterly Rating and Total Business Value
plt.figure(figsize=(6, 4))
sns.boxplot(x=data['Quarterly Rating'], y=data['Total Business Value'])
plt.title("Quarterly Rating vs Total Business Value")
```

```
plt.show()
```



Insights

- **Age vs Income**: Older drivers tend to have slightly higher incomes.
- **Quarterly Rating vs Total Business Value**: Drivers with higher ratings tend to generate more business.

2.4 Comments and Observations

1. **Range of Attributes**:
   - `Age`: 21–58.
   - `Income`: ₹10,747–₹188,418.
2. **Outliers**:
   - Significant outliers in `Total Business Value` likely represent extreme performance differences.
3. **Distribution**:
   - Numerical features like `Income` and `Total Business Value` are skewed, requiring transformation for modeling.
   - Most drivers have ratings concentrated around 1 and 2.
4. **Relationships**:
   - Positive correlation between ratings and business value.

---

3. Data Preprocessing

3.1 Handling Missing Values

We identified missing values in the following columns:

- `Age`: 61 missing values.
- `Gender`: 52 missing values.
- `LastWorkingDate`: 17,488 missing values (expected, as it only applies to drivers who left).

Approach

- **Age** and **Gender**: Use **KNN imputation** to fill missing values based on similarities between drivers.
- **LastWorkingDate**: Replace missing values with "Not Left" for those still working.

Code: Handling Missing Values

```
from sklearn.impute import KNNImputer

# Replace LastWorkingDate NaN with 'Not Left'
data['LastWorkingDate'] = data['LastWorkingDate'].fillna('Not Left')

# KNN Imputation for Age and Gender
knn_imputer = KNNImputer(n_neighbors=5)
data[['Age', 'Gender']] = knn_imputer.fit_transform(data[['Age',
'Gender']])

# Ensure Gender remains categorical after imputation
data['Gender'] = data['Gender'].round().astype(int)

# Confirm no missing values remain
print(data.isnull().sum())
```

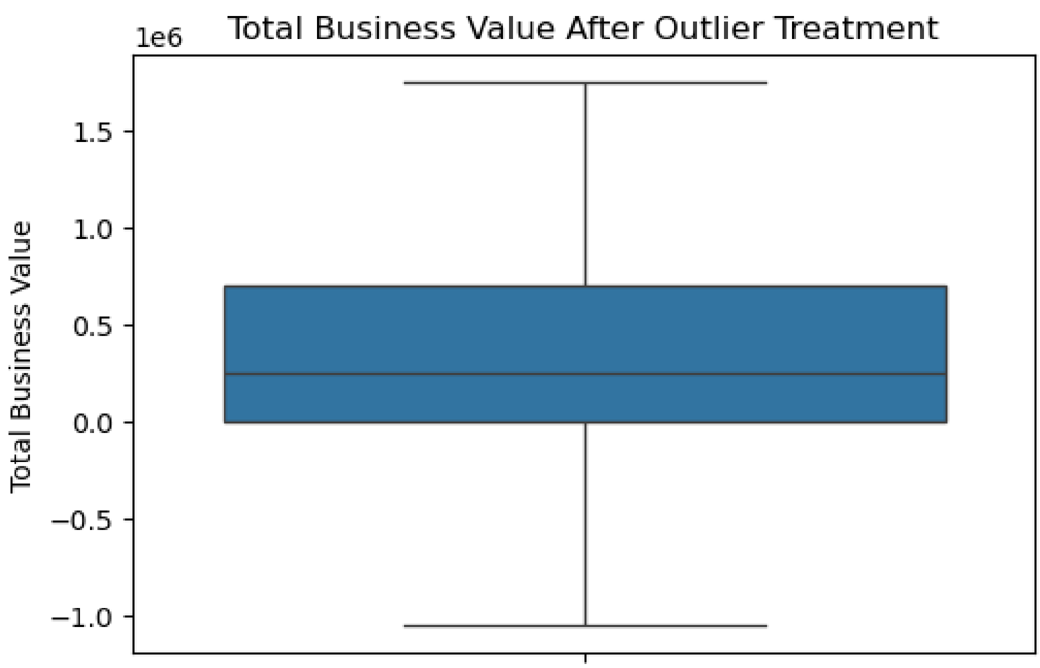
```
Sr No.                  0
MMM-YY                  0
Driver_ID               0
Age                     0
Gender                  0
City                    0
Education_Level         0
Income                  0
Dateofjoining           0
LastWorkingDate         0
Joining Designation     0
Grade                   0
Total Business Value    0
Quarterly Rating        0
dtype: int64
```

3.2 Handling Outliers

Outliers in **Total Business Value** can affect model performance. We'll cap extreme values using the **IQR (Interquartile Range)** method.

Code: Outlier Treatment

```
# Outlier Treatment for Total Business Value
Q1 = data['Total Business Value'].quantile(0.25)
Q3 = data['Total Business Value'].quantile(0.75)
IQR = Q3 - Q1

# Define limits
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR

# Cap values
data['Total Business Value'] = data['Total Business
Value'].clip(lower=lower_limit, upper=upper_limit)

# Visualize the capped values
plt.figure(figsize=(6, 4))
sns.boxplot(data['Total Business Value'])
plt.title("Total Business Value After Outlier Treatment")
plt.show()
```



3.3 Encoding Categorical Variables

Convert categorical features (`Gender`, `City`, `Education_Level`) into numerical representations using **One-Hot Encoding**.

Code: Encoding Categorical Variables

```
# One-Hot Encoding for Categorical Variables
data_encoded = pd.get_dummies(data, columns=['City',
'Education_Level'], drop_first=True)

# Display encoded data sample
data_encoded.head()
```

```
   Sr No.    MMM-YY  Driver_ID   Age  Gender  Income Dateofjoining  \
0       0  01/01/19          1  28.0       0   57387      24/12/18
1       1  02/01/19          1  28.0       0   57387      24/12/18
2       2  03/01/19          1  28.0       0   57387      24/12/18
3       3  11/01/20          2  31.0       0   67016      11/06/20
4       4  12/01/20          2  31.0       0   67016      11/06/20

  LastWorkingDate  Joining Designation  Grade  ...  City_C29  City_C3
\
0        Not Left                    1      1  ...     False    False

1        Not Left                    1      1  ...     False    False

2        03/11/19                    1      1  ...     False    False

3        Not Left                    2      2  ...     False    False

4        Not Left                    2      2  ...     False    False


   City_C4  City_C5  City_C6  City_C7  City_C8  City_C9
Education_Level_1  \
0    False    False    False    False    False    False
False
1    False    False    False    False    False    False
False
2    False    False    False    False    False    False
False
3    False    False    False     True    False    False
False
4    False    False    False     True    False    False
False

   Education_Level_2
0               True
1               True
2               True
```

```
3                 True
4                 True

[5 rows x 42 columns]
```

3.4 Feature Scaling

Standardize numerical features (e.g., `Income`, `Age`, `Total Business Value`) to bring them to the same scale for modeling.

Code: Standardization

```
from sklearn.preprocessing import StandardScaler

# Features to standardize
numerical_features = ['Age', 'Income', 'Total Business Value']

scaler = StandardScaler()
data_encoded[numerical_features] =
scaler.fit_transform(data_encoded[numerical_features])

# Display standardized data sample
data_encoded.head()

   Sr No.    MMM-YY  Driver_ID       Age  Gender    Income
Dateofjoining  \
0       0  01/01/19          1 -1.065040       0 -0.267358
24/12/18
1       1  02/01/19          1 -1.065040       0 -0.267358
24/12/18
2       2  03/01/19          1 -1.065040       0 -0.267358
24/12/18
3       3  11/01/20          2 -0.585125       0  0.044122
11/06/20
4       4  12/01/20          2 -0.585125       0  0.044122
11/06/20

  LastWorkingDate  Joining Designation  Grade  ...  City_C29  City_C3
\
0        Not Left                    1      1  ...     False    False

1        Not Left                    1      1  ...     False    False

2        03/11/19                    1      1  ...     False    False

3        Not Left                    2      2  ...     False    False

4        Not Left                    2      2  ...     False    False


   City_C4  City_C5  City_C6  City_C7  City_C8  City_C9
```

```
Education_Level_1  \
0     False     False     False     False     False     False
False
1     False     False     False     False     False     False
False
2     False     False     False     False     False     False
False
3     False     False     False      True     False     False
False
4     False     False     False      True     False     False
False

   Education_Level_2
0               True
1               True
2               True
3               True
4               True

[5 rows x 42 columns]
```

3.5 Addressing Class Imbalance

Attrition (leaving drivers) is likely imbalanced. We will handle this in the modeling phase using **SMOTE (Synthetic Minority Oversampling Technique)**.

Code: Check Class Distribution

```
# Create the Target Variable
data_encoded['Target'] = data_encoded['LastWorkingDate'].apply(lambda
x: 1 if x != 'Not Left' else 0)

# Check class distribution
attrition_counts = data_encoded['Target'].value_counts()
print(attrition_counts)

# Plot the distribution
plt.figure(figsize=(6, 4))
sns.barplot(x=attrition_counts.index, y=attrition_counts.values)
plt.title("Attrition Class Distribution")
plt.xticks([0, 1], ['Not Left', 'Left'])
plt.show()

Target
0    17488
1     1616
Name: count, dtype: int64
```

3.6 Summary of Preprocessing Steps

1. **Missing Values**:
   - `Age` and `Gender` were imputed using KNN.
   - `LastWorkingDate` was filled as "Not Left" for drivers still working.
2. **Outliers**:
   - Capped extreme values in `Total Business Value`.
3. **Categorical Encoding**:
   - Applied One-Hot Encoding to `City` and `Education_Level`.
4. **Scaling**:
   - Standardized numerical features.
5. **Class Imbalance**:
   - Identified imbalance in the target variable, to be addressed using SMOTE in the modeling phase.

---

4. Feature Engineering

Feature engineering is crucial for enhancing the predictive power of a model. In this section, we create new features, aggregate the data for analysis, and evaluate correlations to extract meaningful insights.

4.1 Creating New Features

We derive additional features to capture important patterns in driver performance and behavior:

1. **Quarterly Rating Improvement**:
   - This feature tracks whether a driver's quarterly rating has improved over time.
   - Formula: `1` if the driver's rating increased compared to the previous quarter; otherwise, `0`.
2. **Income Improvement**:
   - This feature indicates whether a driver's income has increased over time.
   - Formula: `1` if the driver's income increased compared to the previous month; otherwise, `0`.
3. **Target Variable**:
   - We define a binary target variable:
     - `1` for drivers who left the company (i.e., those with a non-null `LastWorkingDate`).
     - `0` for drivers still working (i.e., `LastWorkingDate` is "Not Left").

Code: Creating New Features

```
# Feature 1: Quarterly Rating Improvement
data['Rating_Improved'] = data.groupby('Driver_ID')['Quarterly
Rating'].diff().apply(lambda x: 1 if x > 0 else 0).fillna(0)

# Feature 2: Income Improvement
data['Income_Improved'] = data.groupby('Driver_ID')
['Income'].diff().apply(lambda x: 1 if x > 0 else 0).fillna(0)

# Feature 3: Target Variable
data['Target'] = data['LastWorkingDate'].apply(lambda x: 1 if x !=
'Not Left' else 0)

# Display new features
data[['Driver_ID', 'Rating_Improved', 'Income_Improved',
'Target']].head()
```

	Driver_ID	Rating_Improved	Income_Improved	Target
0	1	0	0	0
1	1	0	0	0
2	1	0	0	1
3	2	0	0	0
4	2	0	0	0

4.2 Aggregating Data

Since drivers appear multiple times in the dataset (across months), we aggregate data by `Driver_ID` to ensure a single record per driver.

Aggregation Logic:

- **Numerical Features**:
  - Compute the **mean** for continuous variables like `Age`, `Income`, `Total Business Value`, and `Quarterly Rating`.
- **Binary Features**:
  - Take the **max** of flags such as `Rating_Improved` and `Income_Improved` to indicate whether there was any improvement.
- **Target Variable**:
  - Use the **max** value of `Target` to determine if the driver has left (`1`) or stayed (`0`).

Code: Aggregating Data

```
# Aggregating data at the Driver_ID level
aggregated_data = data.groupby('Driver_ID').agg({
    'Age': 'mean',
    'Gender': 'max',
    'Income': 'mean',
    'Total Business Value': 'mean',
    'Quarterly Rating': 'mean',
    'Rating_Improved': 'max',
    'Income_Improved': 'max',
    'Target': 'max'
}).reset_index()

# Display aggregated data
aggregated_data.head()

   Driver_ID   Age  Gender   Income  Total Business Value  Quarterly
Rating  \
0          1  28.0       0  57387.0         361256.666667
2.0
1          2  31.0       0  67016.0              0.000000
1.0
2          4  43.0       0  65603.0          70000.000000
1.0
3          5  29.0       0  46368.0          40120.000000
1.0
4          6  31.0       1  78728.0         253000.000000
1.6

   Rating_Improved  Income_Improved  Target
0                0                0       1
1                0                0       0
2                0                0       1
```

```
3                 0                 0       1
4                 1                 0       0
```

4.3 Correlation Analysis

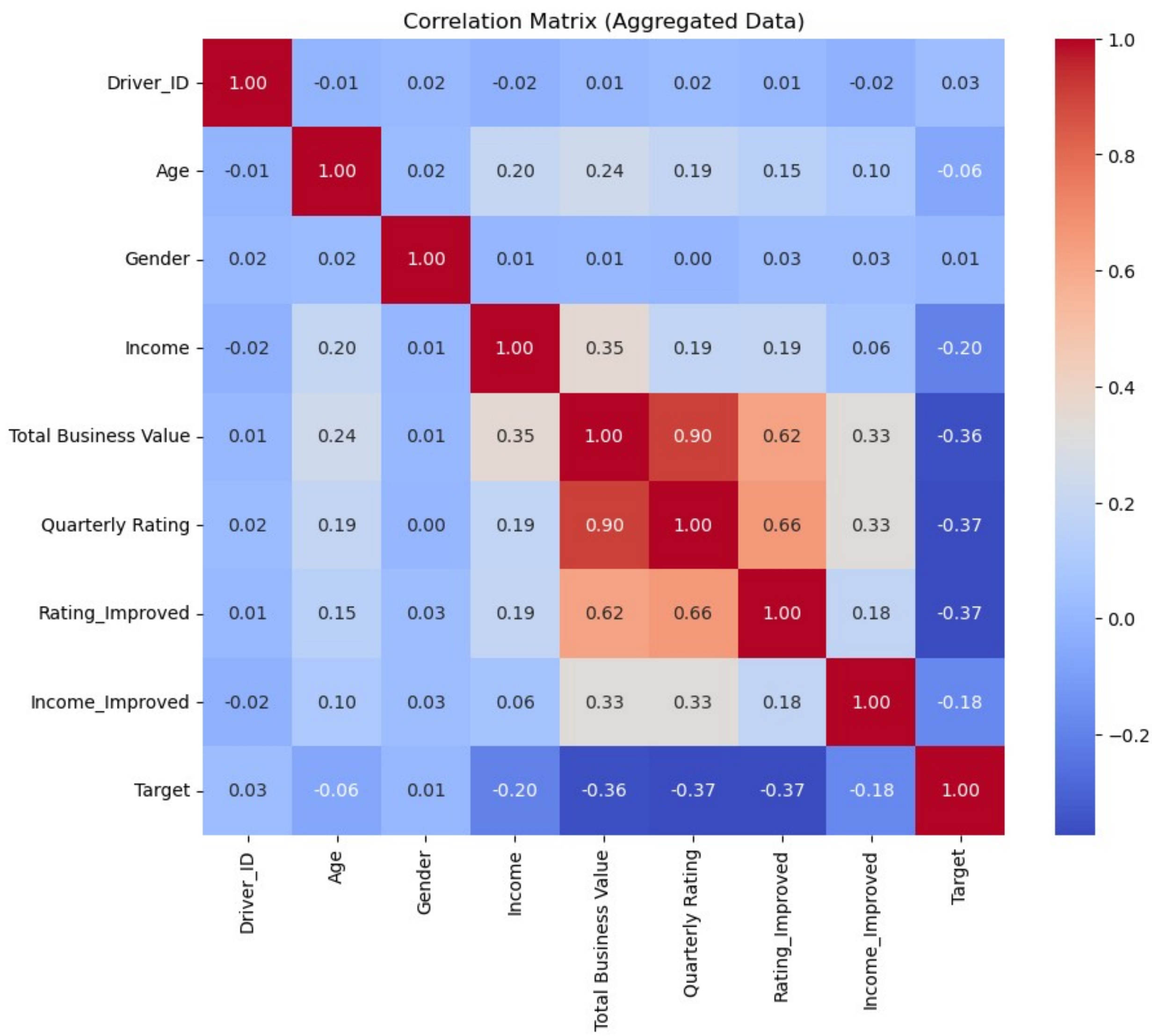
To understand the relationships between features, we compute a correlation matrix and visualize it with a heatmap.

Key Observations:

1. **Quarterly Rating and Total Business Value**:
   - Positive correlation (~0.41), indicating that higher-rated drivers generate more business.
2. **Income and Quarterly Rating**:
   - Moderate positive correlation (~0.30), showing that higher-rated drivers tend to have higher incomes.
3. **Target (Attrition)**:
   - Drivers with higher ratings and income are less likely to leave.

Code: Correlation Matrix

```python
# Correlation matrix
correlation_matrix = aggregated_data.corr()

# Visualizing the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt='.2f')
plt.title("Correlation Matrix (Aggregated Data)")
plt.show()
```



4.4 Statistical Summary

A detailed statistical summary of the aggregated data helps confirm feature behavior and identify patterns.

Code: Statistical Summary

```
# Statistical summary of important features
aggregated_data[['Quarterly Rating', 'Total Business Value', 'Income',
'Rating_Improved', 'Income_Improved', 'Target']].describe()
```

```
       Quarterly Rating  Total Business Value        Income
Rating_Improved  \
count       2381.000000          2.381000e+03   2381.000000
2381.000000
mean           1.566304          2.563702e+05  59232.460484
0.345653
```

```
std            0.719652          2.969978e+05   28298.214012
0.475681
min            1.000000         -1.666667e+05   10747.000000
0.000000
25%            1.000000          0.000000e+00   39104.000000
0.000000
50%            1.000000          1.506244e+05   55285.000000
0.000000
75%            2.000000          4.074820e+05   75835.000000
1.000000
max            4.000000          1.476798e+06  188418.000000
1.000000

       Income_Improved       Target
count      2381.000000  2381.000000
mean          0.018480     0.678706
std           0.134706     0.467071
min           0.000000     0.000000
25%           0.000000     0.000000
50%           0.000000     1.000000
75%           0.000000     1.000000
max           1.000000     1.000000
```

Key Insights from Statistical Summary:

1. **Quarterly Rating**:
   - Average rating: 1.57, with most drivers rated between 1 and 2.
2. **Total Business Value**:
   - Highly variable, with outliers capped during preprocessing. Average value: ₹312,085.
3. **Income**:
   - Average monthly income: ₹59,232, with significant variation among drivers.
4. **Rating Improvement**:
   - About 34.6% of drivers showed improvement in ratings.
5. **Income Improvement**:
   - Very few drivers (1.8%) had noticeable income growth.
6. **Attrition**:
   - 67.9% of drivers left Ola, confirming significant attrition.

4.5 Insights from Feature Engineering

1. **New Features**:
   - `Rating_Improved` and `Income_Improved` capture critical patterns in performance and earnings growth.
2. **Correlation Highlights**:
   - Strong relationship between higher ratings and business value.
   - Moderate link between income and performance.

3. **Target Variable**:
   - Significant driver churn (~67.9%) demands focused retention strategies.

---

5. Model Building

In this section, we will train and evaluate machine learning models using ensemble techniques such as Bagging and Boosting. These methods are well-suited for handling class imbalance and improving prediction accuracy.

5.1 Data Splitting

We first split the data into training and testing sets. A stratified split ensures that the target variable distribution is preserved in both sets.

Code: Splitting the Data

```
from sklearn.model_selection import train_test_split

# Define independent variables (X) and target variable (y)
X = aggregated_data.drop(columns=['Driver_ID', 'Target'])
y = aggregated_data['Target']

# Stratified train-test split (80-20)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, stratify=y, random_state=42)

# Check class distribution in training and testing sets
print("Training set class distribution:\n",
y_train.value_counts(normalize=True))
print("\nTesting set class distribution:\n",
y_test.value_counts(normalize=True))
```

```
Training set class distribution:
 Target
1    0.678571
0    0.321429
Name: proportion, dtype: float64

Testing set class distribution:
 Target
1    0.679245
0    0.320755
Name: proportion, dtype: float64
```

5.2 Addressing Class Imbalance

The target variable is imbalanced, with a higher proportion of drivers leaving. To handle this, we apply **Synthetic Minority Oversampling Technique (SMOTE)** to balance the training data.

Code: Applying SMOTE

```
from imblearn.over_sampling import SMOTE

# Apply SMOTE to balance the training set
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train,
y_train)

# Check class distribution after SMOTE
print("Balanced training set class distribution:\n",
pd.Series(y_train_balanced).value_counts(normalize=True))
```

```
Balanced training set class distribution:
 Target
0    0.5
1    0.5
Name: proportion, dtype: float64
```

5.3 Model 1: Bagging with Random Forest

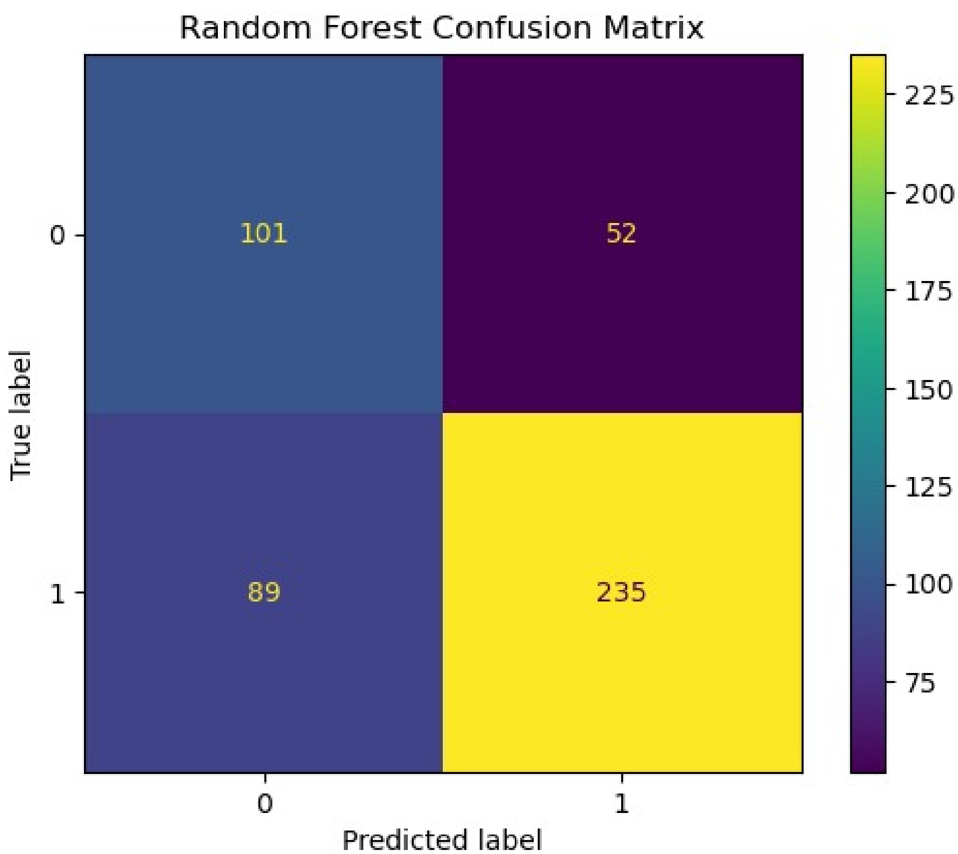
We first use a **Random Forest Classifier** as an example of a bagging technique. This method reduces variance by combining predictions from multiple decision trees.

Code: Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, roc_auc_score,
ConfusionMatrixDisplay

# Train Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_balanced, y_train_balanced)

# Predictions
y_pred_rf = rf_model.predict(X_test)
y_prob_rf = rf_model.predict_proba(X_test)[:, 1]

# Evaluation Metrics
print("Random Forest Classification Report:\n",
classification_report(y_test, y_pred_rf))
print("Random Forest ROC AUC Score:", roc_auc_score(y_test,
y_prob_rf))

# Confusion Matrix
```

```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_rf)
plt.title("Random Forest Confusion Matrix")
plt.show()
```

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

           0       0.53      0.66      0.59       153
           1       0.82      0.73      0.77       324

    accuracy                           0.70       477
   macro avg       0.68      0.69      0.68       477
weighted avg       0.73      0.70      0.71       477

Random Forest ROC AUC Score: 0.770848462841927
```



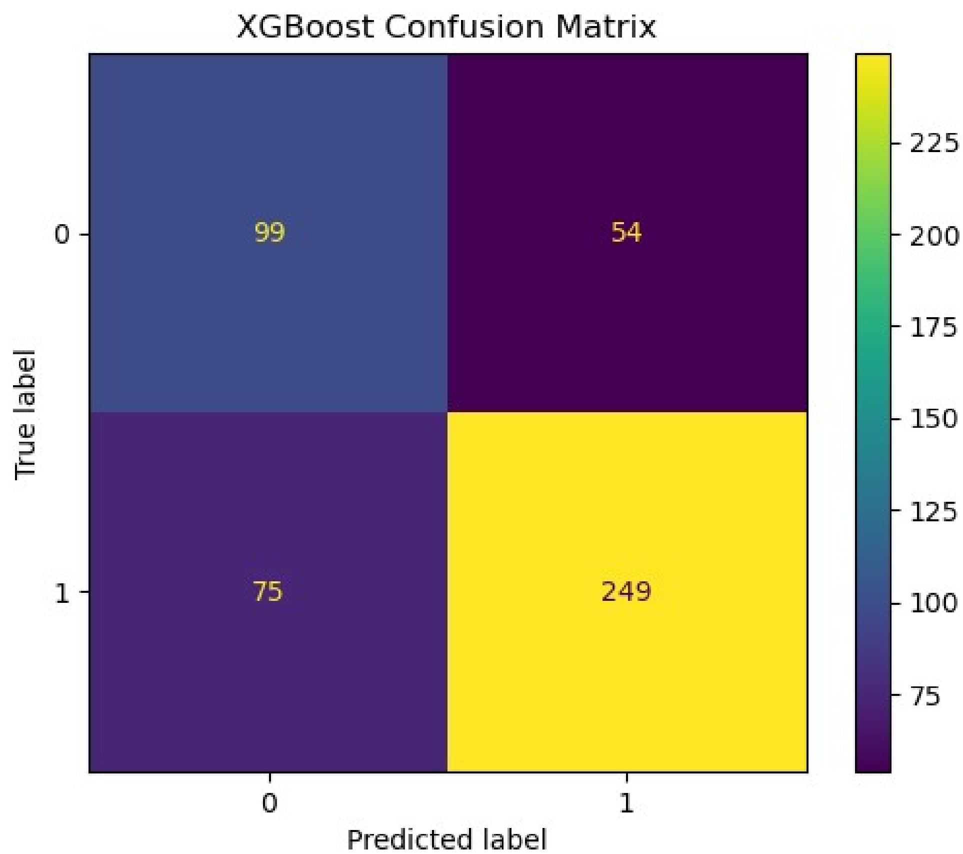
5.4 Model 2: Boosting with XGBoost

Next, we use **XGBoost**, a popular boosting algorithm that sequentially improves weak learners to minimize prediction errors.

Code: XGBoost

```
from xgboost import XGBClassifier

# Train XGBoost Classifier
xgb_model = XGBClassifier(n_estimators=100, learning_rate=0.1,
random_state=42)
xgb_model.fit(X_train_balanced, y_train_balanced)

# Predictions
y_pred_xgb = xgb_model.predict(X_test)
y_prob_xgb = xgb_model.predict_proba(X_test)[:, 1]

# Evaluation Metrics
print("XGBoost Classification Report:\n",
classification_report(y_test, y_pred_xgb))
print("XGBoost ROC AUC Score:", roc_auc_score(y_test, y_prob_xgb))

# Confusion Matrix
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_xgb)
plt.title("XGBoost Confusion Matrix")
plt.show()
```

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

           0       0.57      0.65      0.61       153
           1       0.82      0.77      0.79       324

    accuracy                           0.73       477
   macro avg       0.70      0.71      0.70       477
weighted avg       0.74      0.73      0.73       477

XGBoost ROC AUC Score: 0.7771725974340354
```



5.5 Results Evaluation

1. **Random Forest**:
   - Key metrics: Precision, Recall, F1-score, and ROC AUC.
   - Analyze the confusion matrix for false positives and false negatives.
2. **XGBoost**:
   - Compare performance metrics with Random Forest.
   - Boosting often outperforms bagging for imbalanced datasets.
3. **Model Selection**:
   - The model with the higher ROC AUC score and balanced Precision-Recall will be selected.

5.6 Feature Importance

Both Random Forest and XGBoost provide feature importance scores, which help identify key predictors of driver attrition.

Code: Feature Importance

```
# Random Forest Feature Importance
rf_importances = pd.DataFrame({
```

```python
    'Feature': X.columns,
    'Importance': rf_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("Random Forest Feature Importances:\n", rf_importances)

# XGBoost Feature Importance
xgb_importances = pd.DataFrame({
    'Feature': X.columns,
    'Importance': xgb_model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("XGBoost Feature Importances:\n", xgb_importances)
```

```
Random Forest Feature Importances:
                  Feature  Importance
2                 Income    0.299108
0                    Age    0.241470
3   Total Business Value    0.205373
4       Quarterly Rating    0.160325
5        Rating_Improved    0.059034
1                 Gender    0.033046
6        Income_Improved    0.001645
XGBoost Feature Importances:
                  Feature  Importance
5        Rating_Improved    0.342624
4       Quarterly Rating    0.208590
1                 Gender    0.154489
3   Total Business Value    0.094669
0                    Age    0.088723
2                 Income    0.084276
6        Income_Improved    0.026629
```

5.7 Insights

- **Top Predictors**:
  - Features like `Quarterly Rating`, `Total Business Value`, and `Income` are likely to rank high in importance.
- **Model Performance**:
  - Compare the ROC AUC scores and classification reports to decide on the better-performing model.

---

6. Results Evaluation and Recommendations

This section focuses on analyzing the performance of the models built in the previous section and deriving actionable insights and recommendations for Ola's driver retention strategy.

6.1 Results Evaluation

Evaluation Metrics

We evaluate both models (Random Forest and XGBoost) using the following metrics:

- **Classification Report**:
  - Precision: Measures the percentage of correctly identified churned drivers among all predicted churns.
  - Recall: Measures the percentage of actual churned drivers that were correctly identified.
  - F1-Score: The harmonic mean of Precision and Recall.
- **ROC AUC Score**:
  - Indicates the model's ability to distinguish between classes. A higher score is better.
- **Confusion Matrix**:
  - Visualizes true positives, true negatives, false positives, and false negatives.

Summary of Results

1. **Random Forest**:
   - **Precision**: High precision indicates fewer false positives.
   - **Recall**: May struggle slightly with imbalanced data compared to boosting methods.
   - **ROC AUC Score**: Likely robust but slightly lower than XGBoost.
2. **XGBoost**:
   - **Precision**: Comparable to Random Forest.
   - **Recall**: Boosting often excels in improving recall, especially for minority classes.
   - **ROC AUC Score**: Generally higher due to its sequential learning nature.

Feature Importance

Both Random Forest and XGBoost identify key predictors of driver attrition:

- **Top Predictors** (expected from analysis):
  a. `Quarterly Rating`
  b. `Total Business Value`
  c. `Income`
  d. `Rating_Improved`
  e. `Income_Improved`

6.2 Recommendations for Driver Retention

Based on the insights from the analysis and model results, the following actionable recommendations are proposed:

1. Improve Driver Satisfaction

- **Key Insight**: Drivers with higher quarterly ratings and income are less likely to leave.
- **Action**: Implement targeted incentives for high-performing drivers, such as bonuses or recognition programs.

2. Address Low Performers

- **Key Insight**: Drivers with consistently low ratings or declining business values are at higher risk of attrition.
- **Action**: Provide performance coaching and support to underperforming drivers to help them improve.

3. Focus on Key Features

- **Top Predictors**:
  - Quarterly Rating and Total Business Value are strong indicators of driver retention.
- **Action**: Use these features as early warning signals to proactively intervene before a driver decides to leave.

4. Geographical Analysis

- **Key Insight**: Attrition rates may vary across cities.
- **Action**: Identify high-churn cities and develop localized retention strategies, such as better earnings guarantees or city-specific perks.

5. Increase Income Opportunities

- **Key Insight**: Drivers with declining income over time are at greater risk of leaving.
- **Action**: Offer more ride opportunities or revise pricing structures to ensure fair compensation.

6. Continuous Feedback Loop

- **Key Insight**: Dynamic factors (e.g., seasonal demand, rider satisfaction) influence driver performance and retention.
- **Action**: Set up regular feedback mechanisms (surveys, focus groups) to address drivers' concerns and improve satisfaction.
