Ola_business_case_study

May 28, 2025

[1]:
```
# Problem Definition:

# Ola is facing high driver attrition, which not only raises recruitment costs␣
↪but also affects organizational morale and service continuity.
# The goal is to build a predictive model using historical driver data-
including demographics, tenure, and performance metrics-to identify
# drivers at risk of leaving the company. Early identification will allow␣
↪proactive retention efforts, helping reduce churn and associated costs.
# Additionally, analyzing key drivers of attrition can support strategic policy␣
↪decisions to improve driver engagement and satisfaction.
```

[2]:
```
import pandas as pd

df = pd.read_csv("ola_driver_scaler.csv")
```

[3]:
```
df.shape
```

[3]:
```
(19104, 14)
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            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
```

[5]:
```
df.head()
```

```
/usr/local/lib/python3.11/dist-
packages/google/colab/_dataframe_summarizer.py:88: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to `dateutil`.
To ensure parsing is consistent and as-expected, please specify a format.
  cast_date_col = pd.to_datetime(column, errors="coerce")
```

[5]:
```
   Unnamed: 0    MMM-YY  Driver_ID   Age  Gender City  Education_Level  \
0           0  01/01/19          1  28.0     0.0  C23                2
1           1  02/01/19          1  28.0     0.0  C23                2
2           2  03/01/19          1  28.0     0.0  C23                2
3           3  11/01/20          2  31.0     0.0   C7                2
4           4  12/01/20          2  31.0     0.0   C7                2

   Income Dateofjoining LastWorkingDate  Joining Designation  Grade  \
0   57387      24/12/18             NaN                    1      1
1   57387      24/12/18             NaN                    1      1
2   57387      24/12/18        03/11/19                    1      1
3   67016      11/06/20             NaN                    2      2
4   67016      11/06/20             NaN                    2      2

   Total Business Value  Quarterly Rating
0               2381060                 2
1               -665480                 2
2                     0                 2
3                     0                 1
4                     0                 1
```

[6]:
```
df.tail()
```

[6]:
```
       Unnamed: 0    MMM-YY  Driver_ID   Age  Gender City  Education_Level  \
19099       19099  08/01/20       2788  30.0     0.0  C27                2
19100       19100  09/01/20       2788  30.0     0.0  C27                2
19101       19101  10/01/20       2788  30.0     0.0  C27                2
19102       19102  11/01/20       2788  30.0     0.0  C27                2
19103       19103  12/01/20       2788  30.0     0.0  C27                2

       Income Dateofjoining LastWorkingDate  Joining Designation  Grade  \
19099   70254      06/08/20             NaN                    2      2
19100   70254      06/08/20             NaN                    2      2
19101   70254      06/08/20             NaN                    2      2
19102   70254      06/08/20             NaN                    2      2
```

```
19103   70254      06/08/20             NaN                    2      2

       Total Business Value  Quarterly Rating
19099                740280                 3
19100                448370                 3
19101                     0                 2
19102                200420                 2
19103                411480                 2
```

[7]:
```python
df.columns
```

[7]:
```
Index(['Unnamed: 0', 'MMM-YY', 'Driver_ID', 'Age', 'Gender', 'City',
       'Education_Level', 'Income', 'Dateofjoining', 'LastWorkingDate',
       'Joining Designation', 'Grade', 'Total Business Value',
       'Quarterly Rating'],
      dtype='object')
```

[8]:
```python
df.dtypes
```

[8]:
```
Unnamed: 0                int64
MMM-YY                   object
Driver_ID                 int64
Age                     float64
Gender                  float64
City                     object
Education_Level           int64
Income                    int64
Dateofjoining            object
LastWorkingDate          object
Joining Designation       int64
Grade                     int64
Total Business Value      int64
Quarterly Rating          int64
dtype: object
```

[9]:
```python
# From the above we can see that the data has 19104 rows and 14 columns.
# We see that the data types of the columns are mixed with the presence of␣
 ↪numerical(int and float)
# and categorical(objects).
# Now one thing that stands out is that this data has multiple rows for the␣
 ↪same Driver_ID.
# This means we will have to tune down the data to one row per ID by␣
 ↪aggregating other columns accordingly.
```

[10]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            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
```

[11]:

```
#Converting some columns to categorical type to reduce memory usage
df['Gender'] = df['Gender'].astype('category')
df['City'] = df['City'].astype('category')
df['Education_Level'] = df['Education_Level'].astype('category')
df['Joining Designation'] = df['Joining Designation'].astype('category')
df['Grade'] = df['Grade'].astype('category')
```

[12]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            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  category
 5   City                  19104 non-null  category
 6   Education_Level       19104 non-null  category
 7   Income                19104 non-null  int64
 8   Dateofjoining         19104 non-null  object
 9   LastWorkingDate       1616 non-null   object
 10  Joining Designation   19104 non-null  category
 11  Grade                 19104 non-null  category
 12  Total Business Value  19104 non-null  int64
```

```
 13  Quarterly Rating      19104 non-null  int64
dtypes: category(5), float64(1), int64(5), object(3)
memory usage: 1.4+ MB
```

[13]:
```
df.head()
```

```
/usr/local/lib/python3.11/dist-
packages/google/colab/_dataframe_summarizer.py:88: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to `dateutil`.
To ensure parsing is consistent and as-expected, please specify a format.
  cast_date_col = pd.to_datetime(column, errors="coerce")
```

[13]:
```
   Unnamed: 0    MMM-YY  Driver_ID   Age Gender City Education_Level  Income  \
0           0  01/01/19          1  28.0    0.0  C23               2   57387
1           1  02/01/19          1  28.0    0.0  C23               2   57387
2           2  03/01/19          1  28.0    0.0  C23               2   57387
3           3  11/01/20          2  31.0    0.0   C7               2   67016
4           4  12/01/20          2  31.0    0.0   C7               2   67016

  Dateofjoining LastWorkingDate Joining Designation Grade  \
0      24/12/18             NaN                   1     1
1      24/12/18             NaN                   1     1
2      24/12/18        03/11/19                   1     1
3      11/06/20             NaN                   2     2
4      11/06/20             NaN                   2     2

   Total Business Value  Quarterly Rating
0               2381060                 2
1               -665480                 2
2                     0                 2
3                     0                 1
4                     0                 1
```

[14]:
```
#From the above we can see that converting the columns to type category helped
 ↪by reducing the memory usage.
```

[15]:
```
df.isnull().sum()
```

[15]:
```
Unnamed: 0                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
```

[16]:
```python
df.isnull().mean()
```

[16]:
```
Unnamed: 0              0.000000
MMM-YY                  0.000000
Driver_ID               0.000000
Age                     0.003193
Gender                  0.002722
City                    0.000000
Education_Level         0.000000
Income                  0.000000
Dateofjoining           0.000000
LastWorkingDate         0.915410
Joining Designation     0.000000
Grade                   0.000000
Total Business Value    0.000000
Quarterly Rating        0.000000
dtype: float64
```

[17]:
```python
#From the above we see that Age, Gender and LastWOrkingDate columns have some␣
↪missing values.
#But these can be easily be imputed using driver aggregation.
#We can also try using KNN imputation, but aggregation using driver ID seems␣
↪better here.
```

[18]:
```python
df.describe()
```

[18]:
```
         Unnamed: 0     Driver_ID           Age         Income  \
count  19104.000000  19104.000000  19043.000000   19104.000000
mean    9551.500000   1415.591133     34.668435   65652.025126
std     5514.994107    810.705321      6.257912   30914.515344
min        0.000000      1.000000     21.000000   10747.000000
25%     4775.750000    710.000000     30.000000   42383.000000
50%     9551.500000   1417.000000     34.000000   60087.000000
75%    14327.250000   2137.000000     39.000000   83969.000000
max    19103.000000   2788.000000     58.000000  188418.000000

       Total Business Value  Quarterly Rating
count          1.910400e+04      19104.000000
mean           5.716621e+05          2.008899
std            1.128312e+06          1.009832
min           -6.000000e+06          1.000000
```

```
25%          0.000000e+00         1.000000
50%          2.500000e+05         2.000000
75%          6.997000e+05         3.000000
max          3.374772e+07         4.000000
```

[19]:
```
#Ignoring the columns where stats summary doesn't make sense, let us focus on
 ↪the ones where
#it does make sense.
#We can see that the average age of drivers is roughly 35, indicating that
 ↪youngsters are
#more likely to use this platform and register themselves as drivers.
#The 75% of the age of 39 also shows that there are so many less that 39years
 ↪of age.
#Also, the mean and median age are close to each other indicating there are no
 ↪outliers in driver age.
#The average income of the driver of 65K shows that the median income shows 60K
#This agin shows that there are no outliers. This also shows that Ola drivers
 ↪make a decent amount
#of money, especially considering a country such as India, where many office
 ↪desk jobs pay less than this.
```

[20]:
```
df.describe(include='category')
```

[20]:

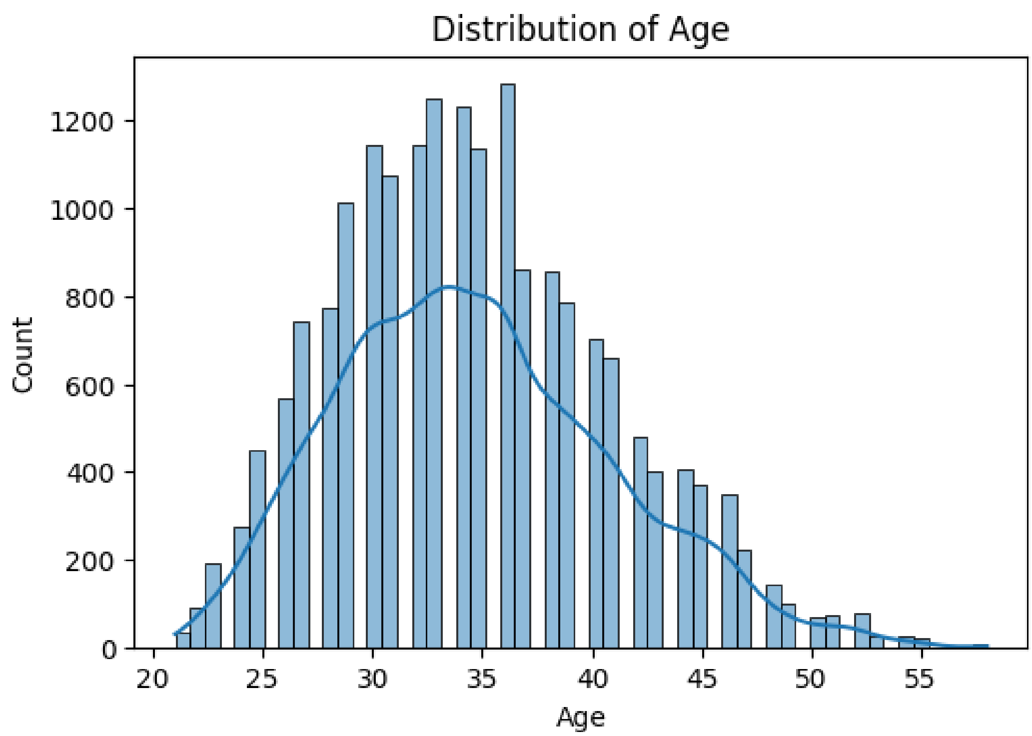
	Gender	City	Education_Level	Joining Designation	Grade
count	19052.0	19104	19104	19104	19104
unique	2.0	29	3	5	5
top	0.0	C20	1	1	2
freq	11074.0	1008	6864	9831	6627

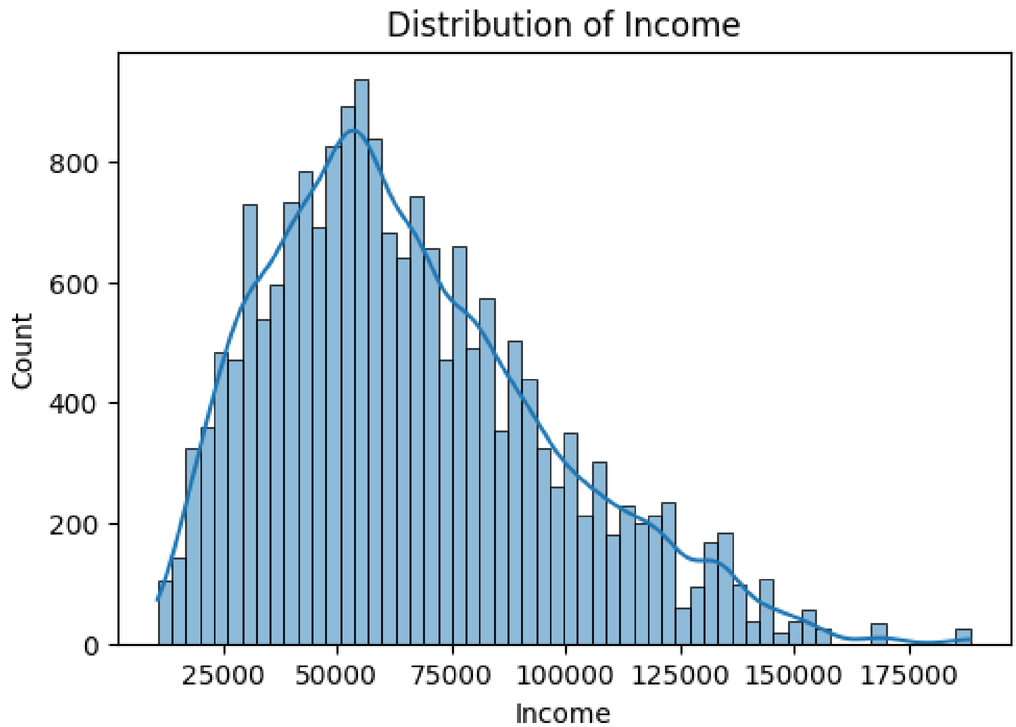
[21]:
```
#The above shows that there are more male drivers compared to female drivers
#And C20 is the city where we have the max number of drivers.
```

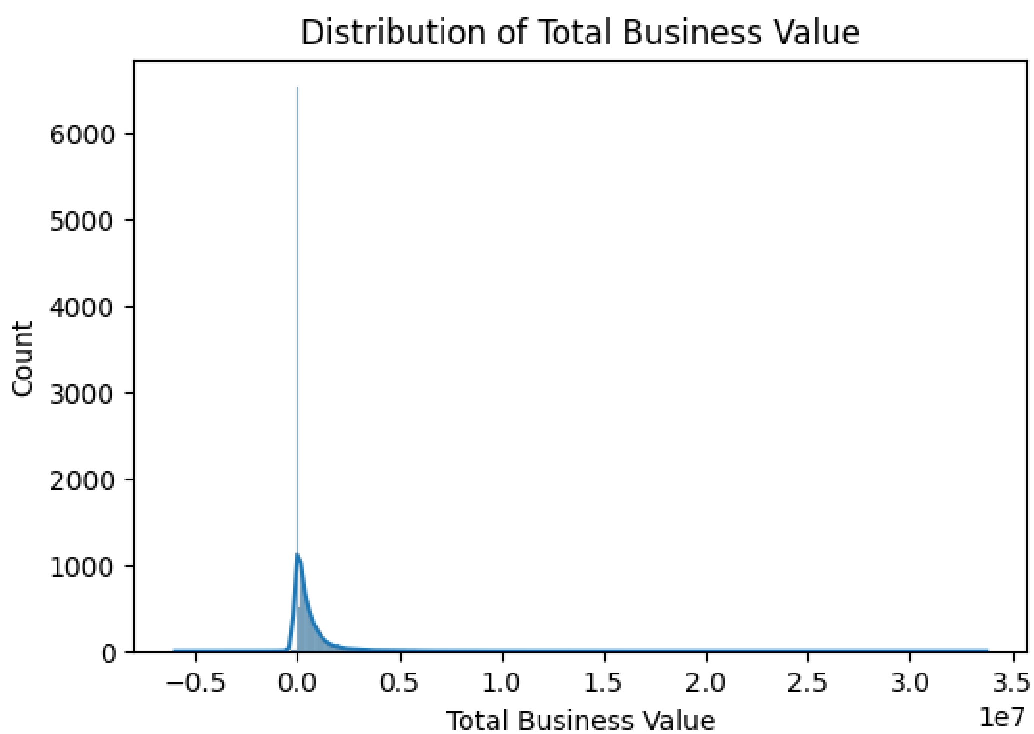
[22]:
```
#Univariate analysis
```

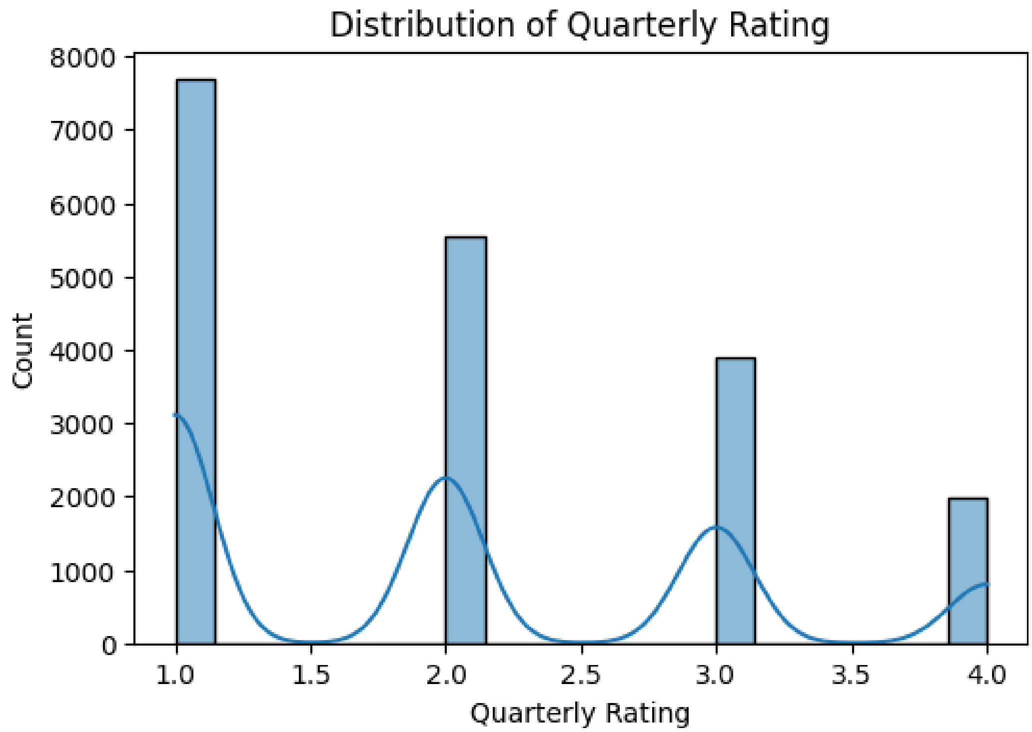
[23]:
```
import matplotlib.pyplot as plt
import seaborn as sns

continuous_vars = ['Age', 'Income', 'Total Business Value', 'Quarterly Rating']
categorical_vars = ['Gender', 'City', 'Education_Level', 'Joining Designation',
 ↪'Grade']

for col in continuous_vars:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
    plt.show()
```

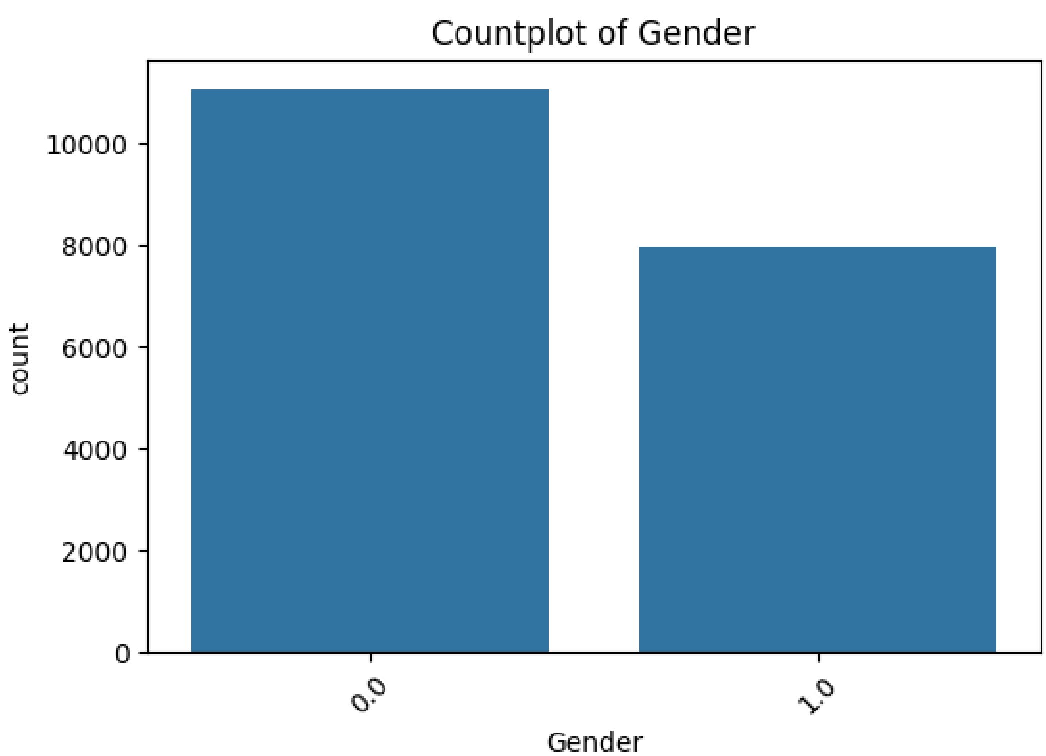
```python
for col in categorical_vars:
    plt.figure(figsize=(6, 4))
    sns.countplot(x=col, data=df)
    plt.title(f'Countplot of {col}')
    plt.xticks(rotation=45)
    plt.show()
```

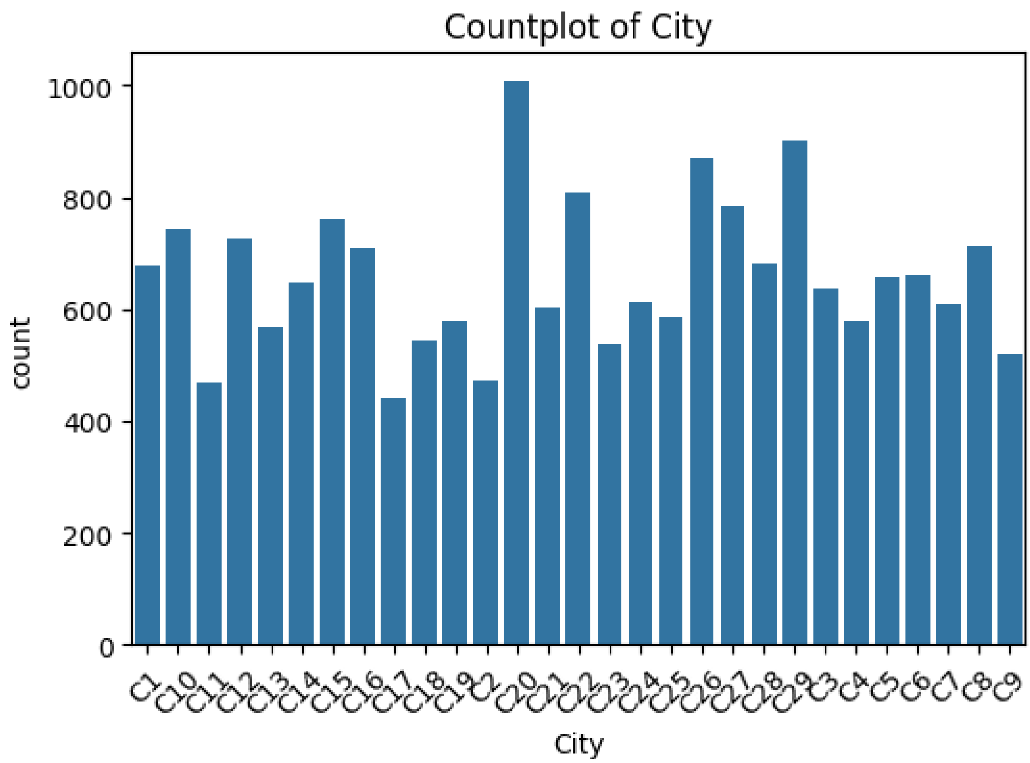


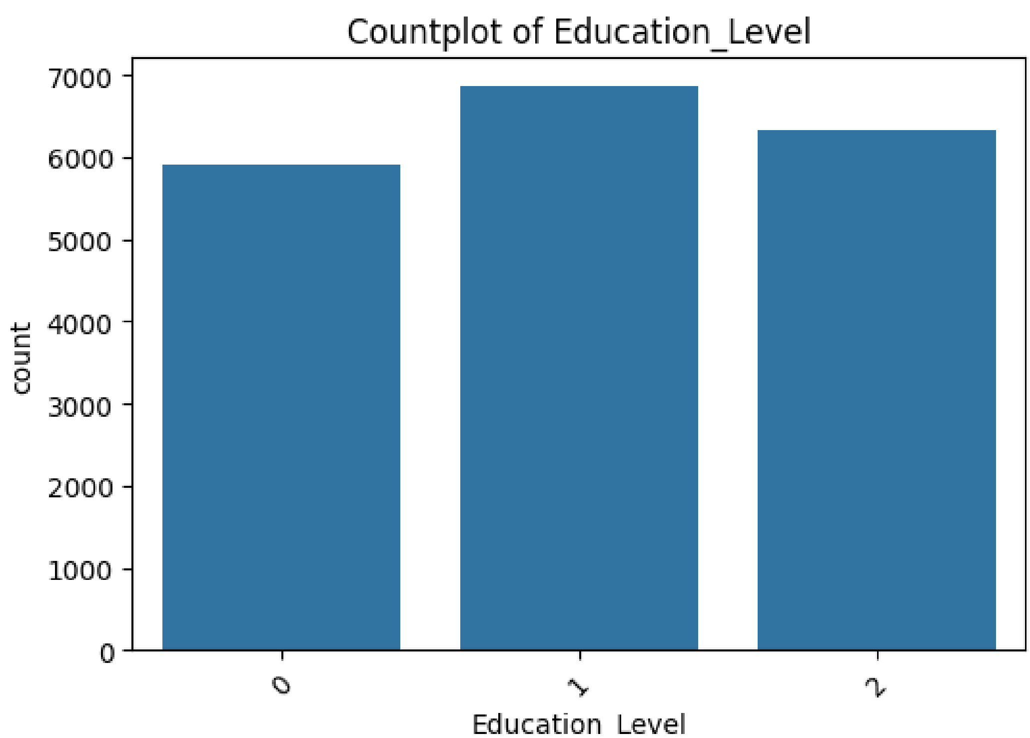

Distribution of Income
Count
Income
0
200
400
600
800
25000
50000
75000
100000
125000
150000
175000

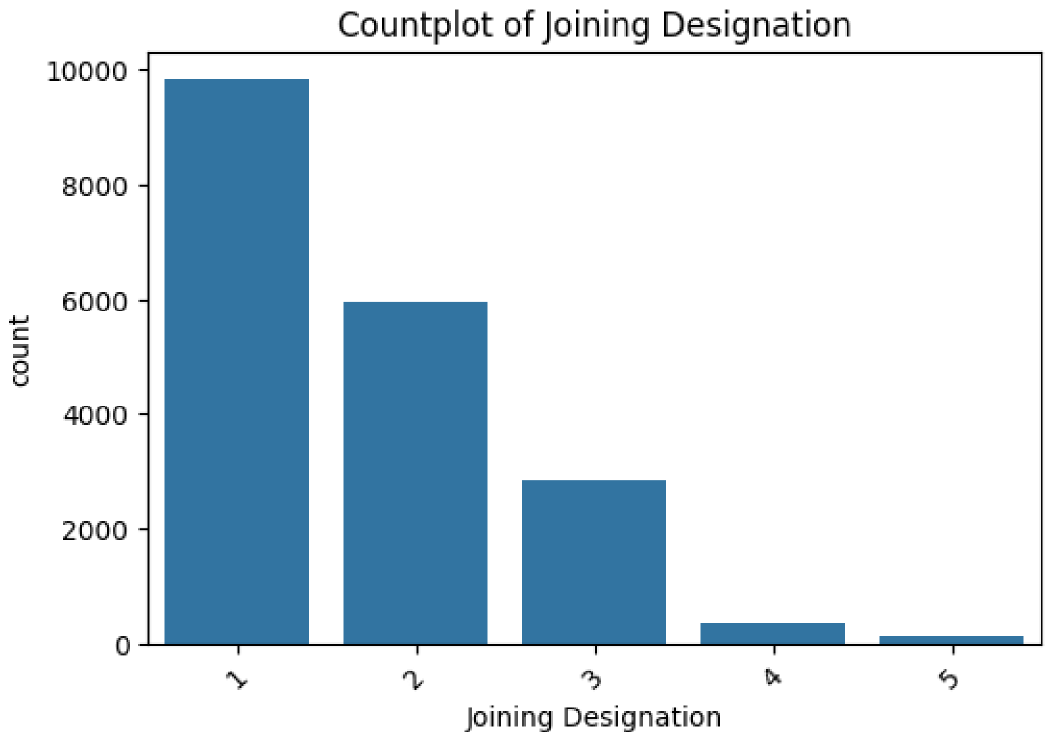

Distribution of Total Business Value
Count
Total Business Value
1e7
0
1000
2000
3000
4000
5000
6000
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

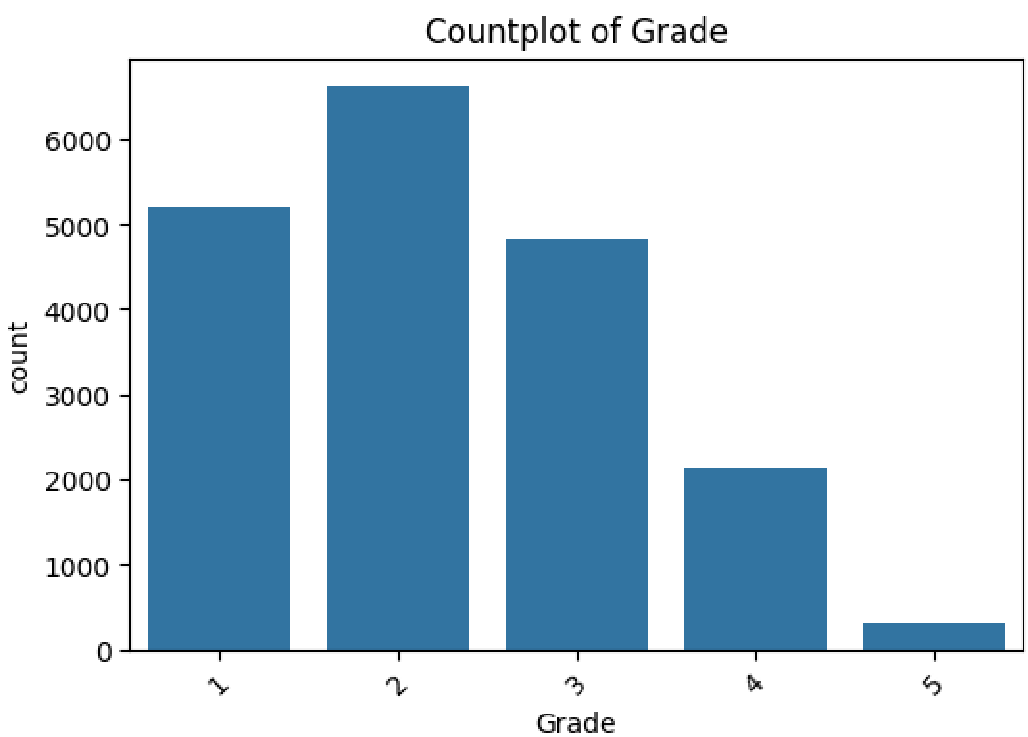

Distribution of Quarterly Rating
Count
0
1000
2000
3000
4000
5000
6000
7000
8000
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Quarterly Rating


Countplot of Gender
count
0
2000
4000
6000
8000
10000
0.0
1.0
Gender


Countplot of City
count
1000
800
600
400
200
0
C1
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
C2
C20
C21
C22
C23
C24
C25
C26
C27
C28
C29
C3
C4
C5
C6
C7
C8
C9
City


Countplot of Education_Level
count
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
Education_Level

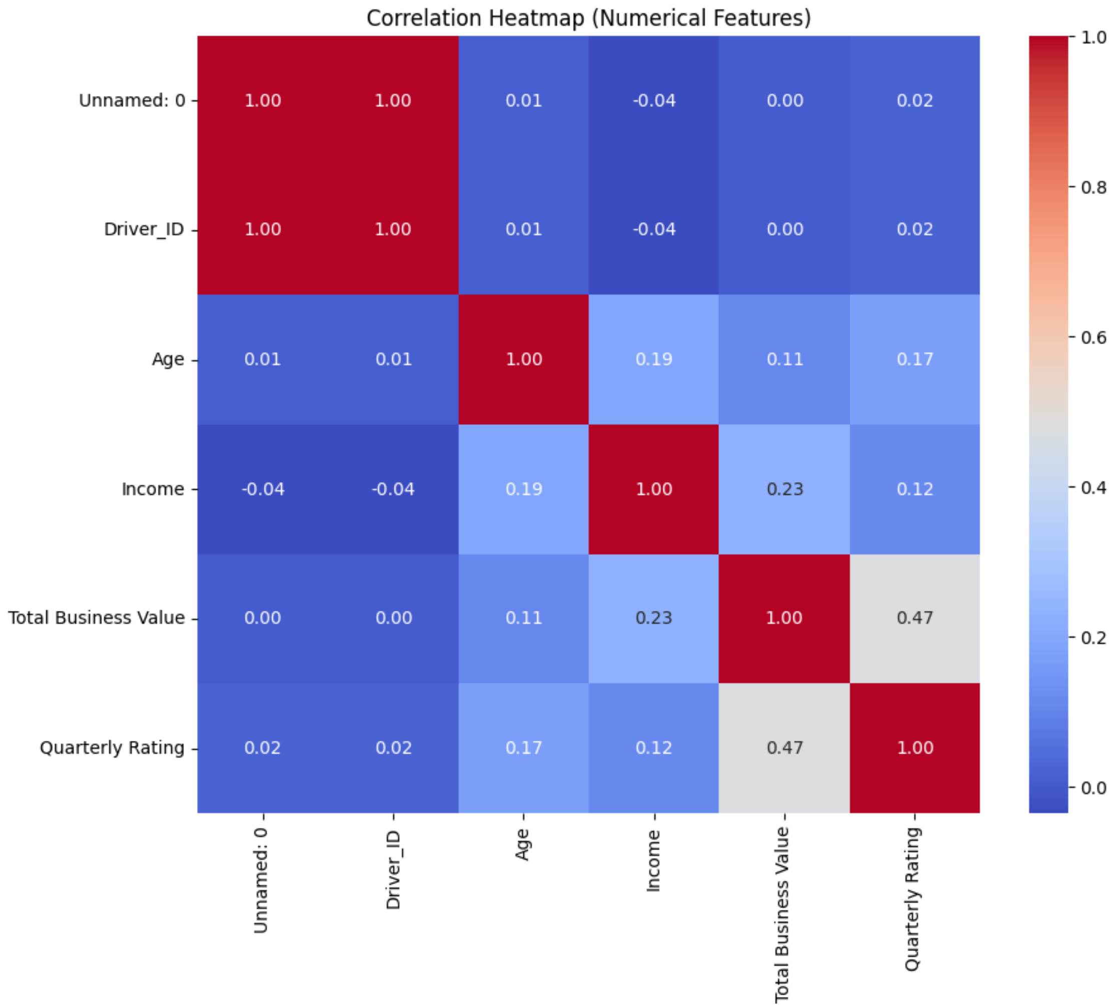

Countplot of Joining Designation
10000
8000
6000
4000
2000
0
count
1
2
3
4
5
Joining Designation

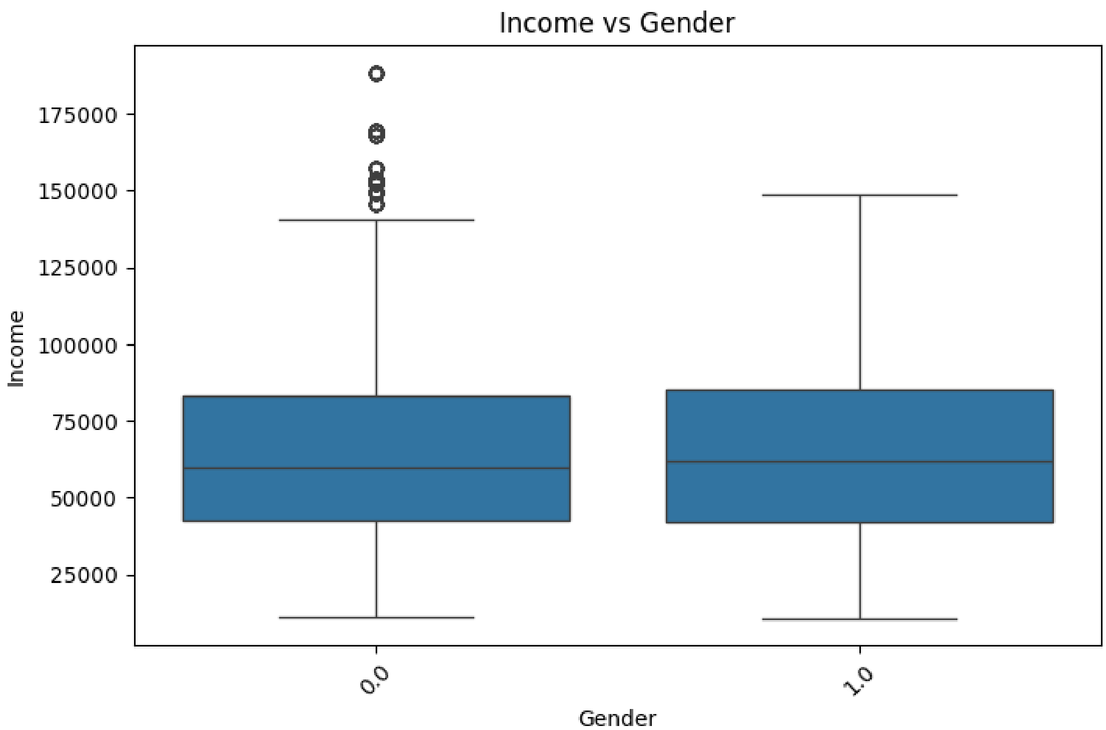

Countplot of Grade
6000
5000
4000
3000
2000
1000
0
count
1
2
3
4
5
Grade

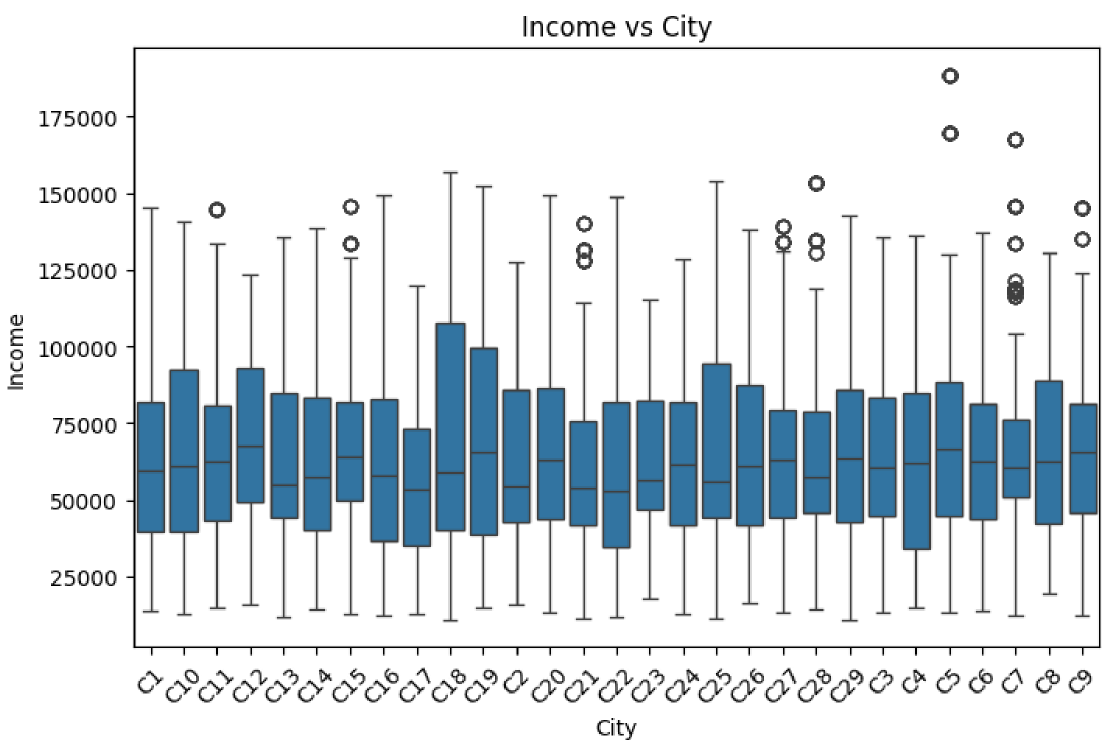
[24]:
```
# The count vs age shows a near normal distribution curve with a slight right␣
↪skewness.
# This is expected as the max number of drivers(or rather number of rides)␣
↪would be youngsters less than the age of 40.

# The count vs Income shows a right skewed distribution indicating that there␣
↪are many people
# who earn less than 100000 and few who earn above this.

# The count vs business value is again a strict right skewed data showing that␣
↪there are very few who
# create a lot of business value.

# Quaterly rating is a discrete graph, showing that as the rating increases,␣
↪the number of drivers
# decreases.

# The plot for gender reveals the expected stat about majority of the␣
↪drivers(or rather number of rides) being males,
# but it also shows a surprising insight that the count of female drivers(or␣
↪rather number of rides) are not so far behind too.

# The plot for city shows that the number of drivers(or rather number of rides)␣
↪in city C20 is max as already seen earlier in
# the stats summary, followed by C29.

# The education level plot shows that the max number of drivers(or rather␣
↪number of rides) are 12+, followed by graduates
# and then 10th.

# Joining designation plot shows that the majority of them have the designation␣
↪as 1 and this decreases gradually as the designation increases.

# Grade is a discrete plot but it shows a bit of right skewness indicating that␣
↪there are few drivers with high grades.
```

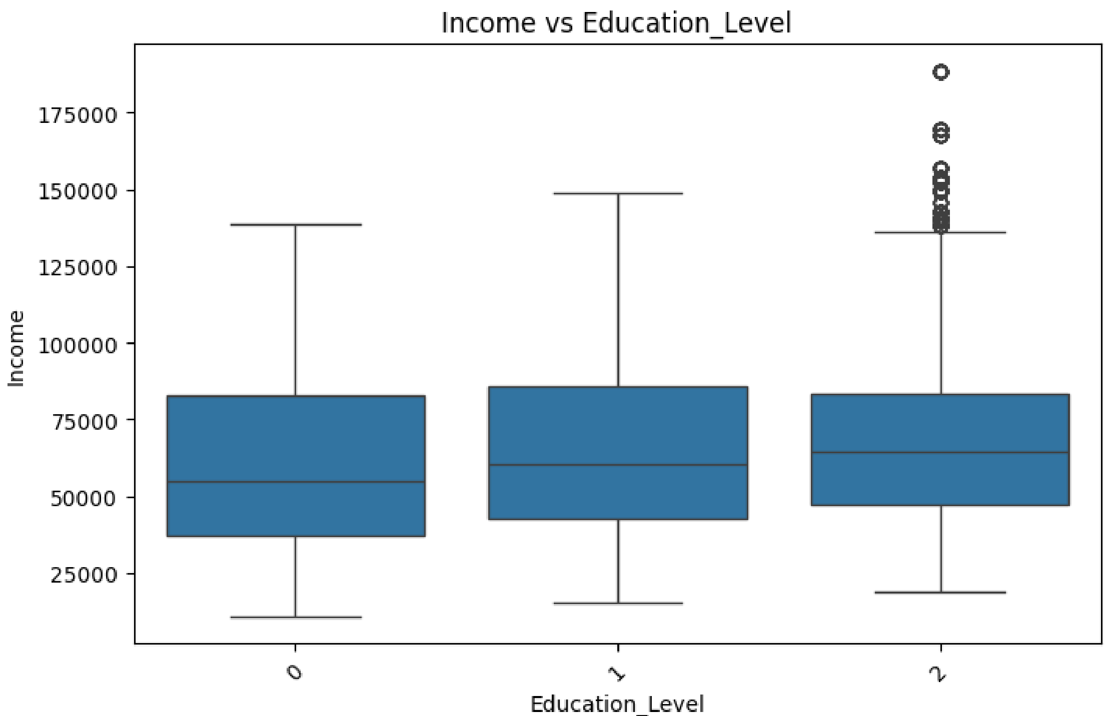
[25]:
```
import numpy as np

# Correlation heatmap for numerical variables
plt.figure(figsize=(10, 8))
sns.heatmap(df.select_dtypes(include=np.number).corr(), annot=True,␣
↪cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap (Numerical Features)')
plt.show()
```

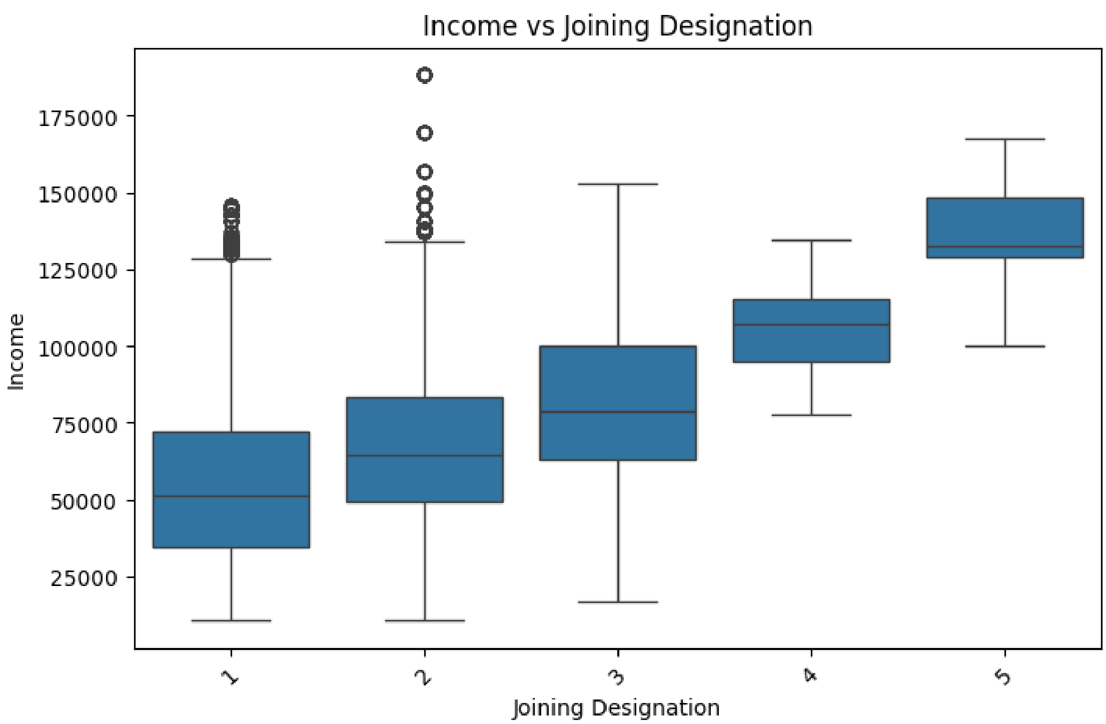
```python
# Boxplots of Income vs Categorical Variables
for col in ['Gender', 'City', 'Education_Level', 'Joining Designation',␣
 ↪'Grade']:
    plt.figure(figsize=(8, 5))
    sns.boxplot(x=col, y='Income', data=df)
    plt.title(f'Income vs {col}')
    plt.xticks(rotation=45)
    plt.show()

# Scatterplots of important continuous variable pairs
sns.pairplot(df[["Age", "Income", "Total Business Value", "Quarterly Rating"]])
plt.show()

# Barplots of mean Income/Total Business Value by categorical variables
for col in ['Gender', 'City', 'Education_Level', 'Joining Designation',␣
 ↪'Grade']:
    plt.figure(figsize=(8, 4))
    sns.barplot(x=col, y='Total Business Value', data=df, estimator=np.mean)
    plt.title(f'Mean Total Business Value by {col}')
    plt.xticks(rotation=45)
    plt.show()

# Violin plot of Quarterly Rating by Grade
plt.figure(figsize=(8, 5))
sns.violinplot(x='Grade', y='Quarterly Rating', data=df)
plt.title('Quarterly Rating Distribution by Grade')
plt.show()
```

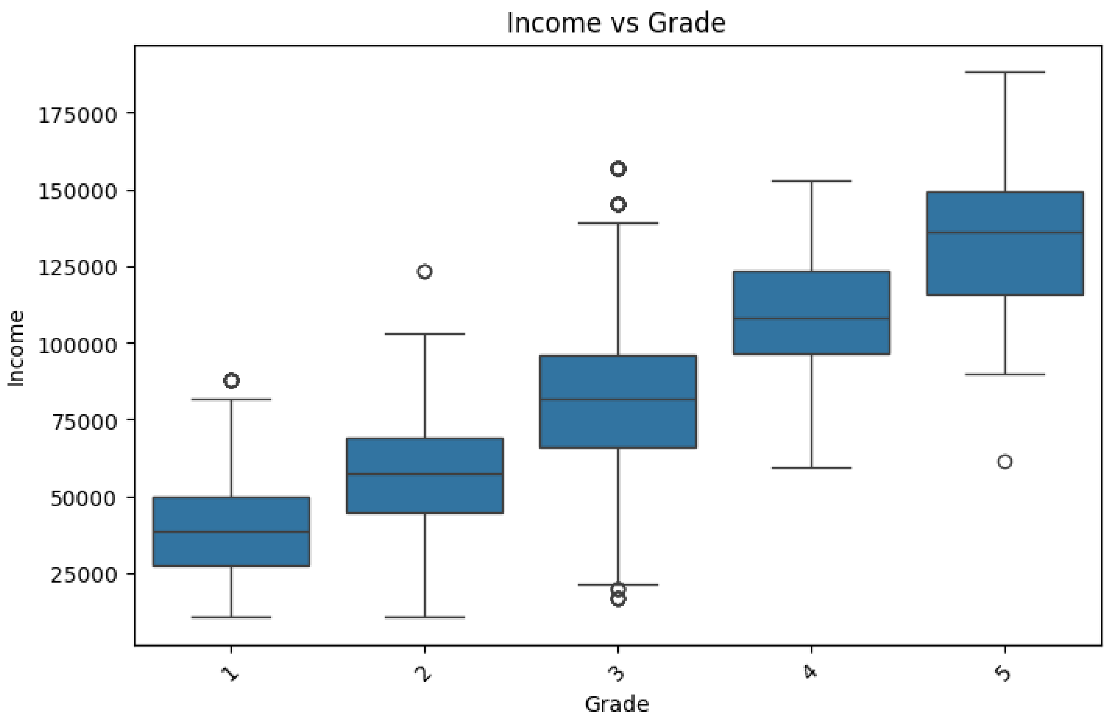

Correlation Heatmap (Numerical Features)
Unnamed: 0
Driver_ID
Age
Income
Total Business Value
Quarterly Rating
1.00
0.01
-0.04
0.00
0.02
0.19
0.11
0.17
0.23
0.12
0.47
1.0
0.8
0.6
0.4
0.2
0.0

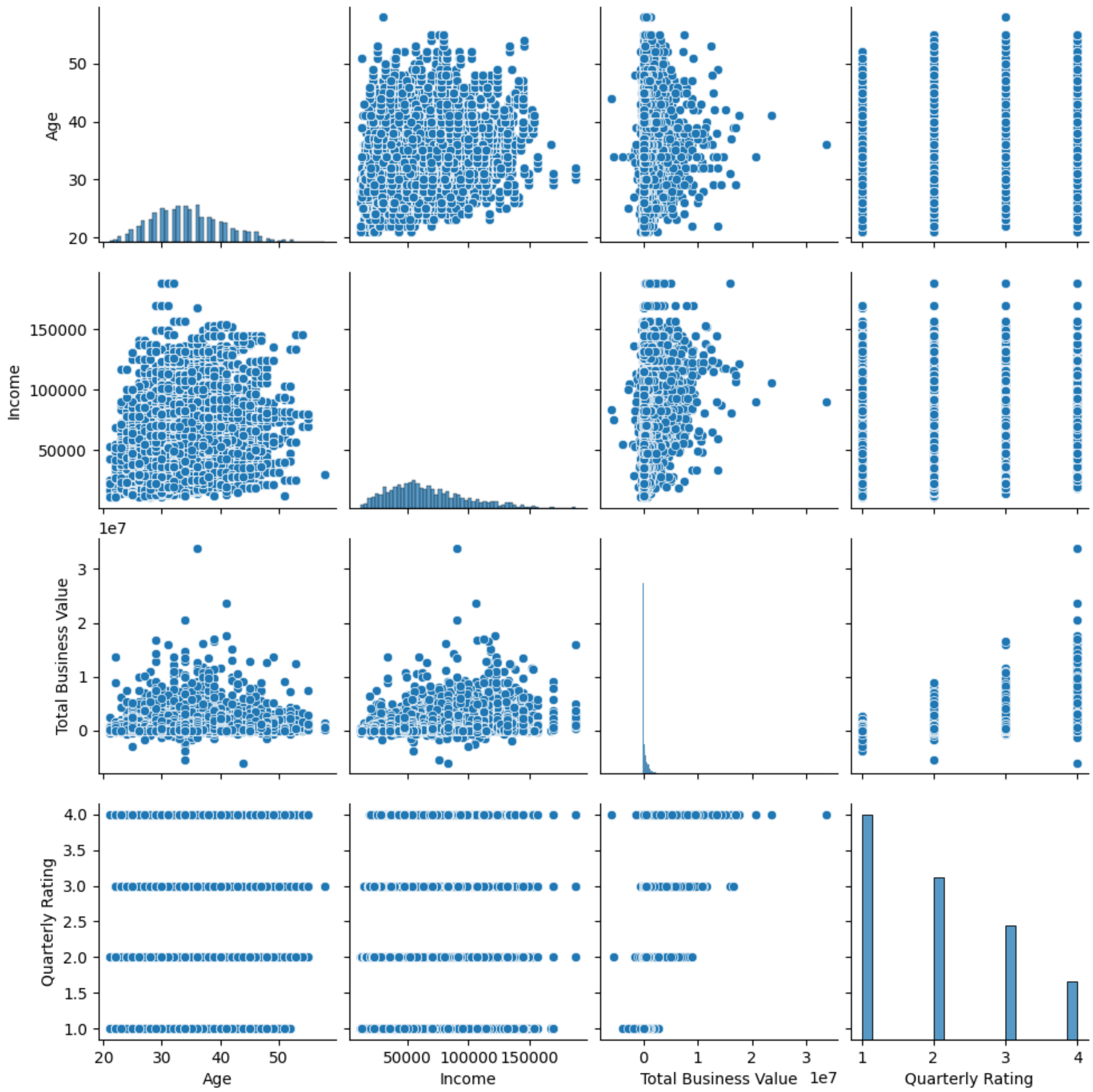

Income vs Gender
175000
150000
125000
100000
75000
50000
25000
Income
0.0
1.0
Gender

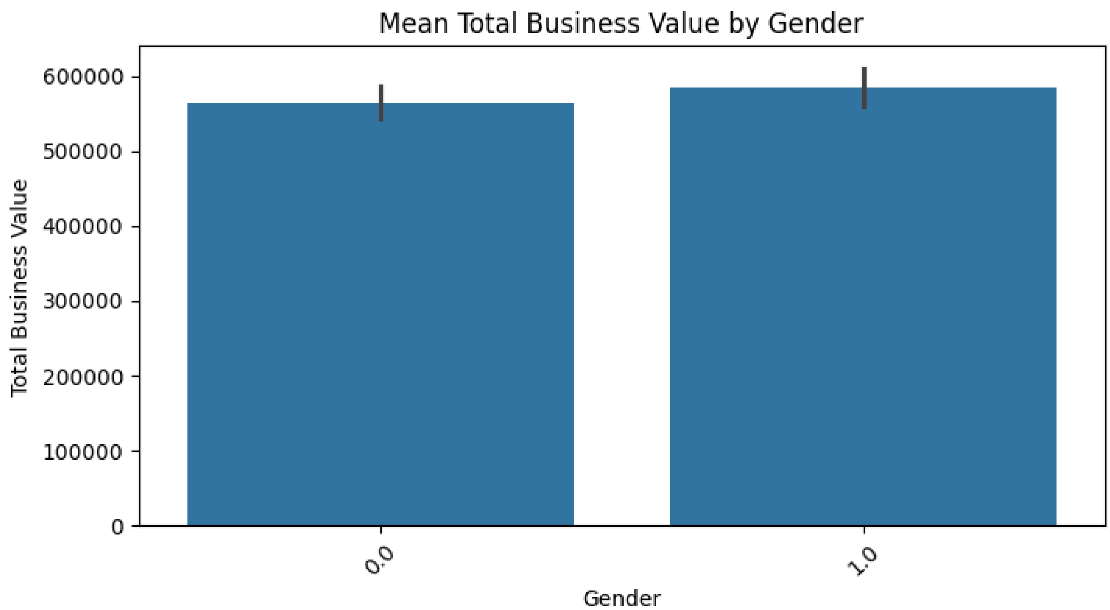

Income vs City
175000
150000
125000
100000
75000
50000
25000
Income
C1 C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C2 C20 C21 C22 C23 C24 C25 C26 C27 C28 C29 C3 C4 C5 C6 C7 C8 C9
City

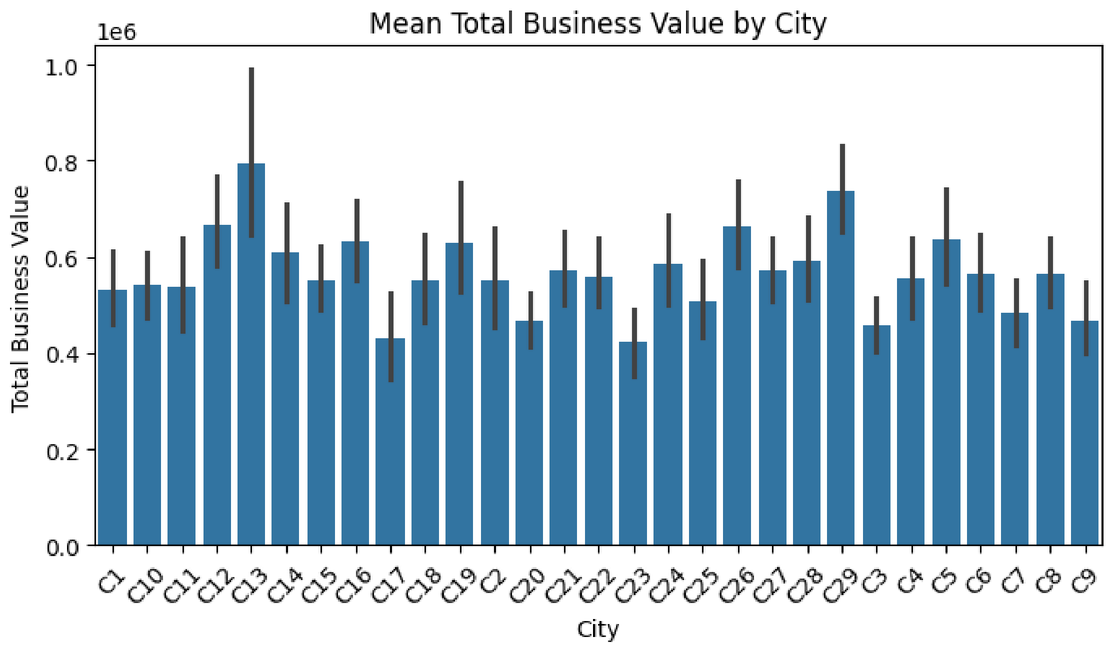

Income vs Education_Level
Income
175000
150000
125000
100000
75000
50000
25000
0
1
2
Education_Level

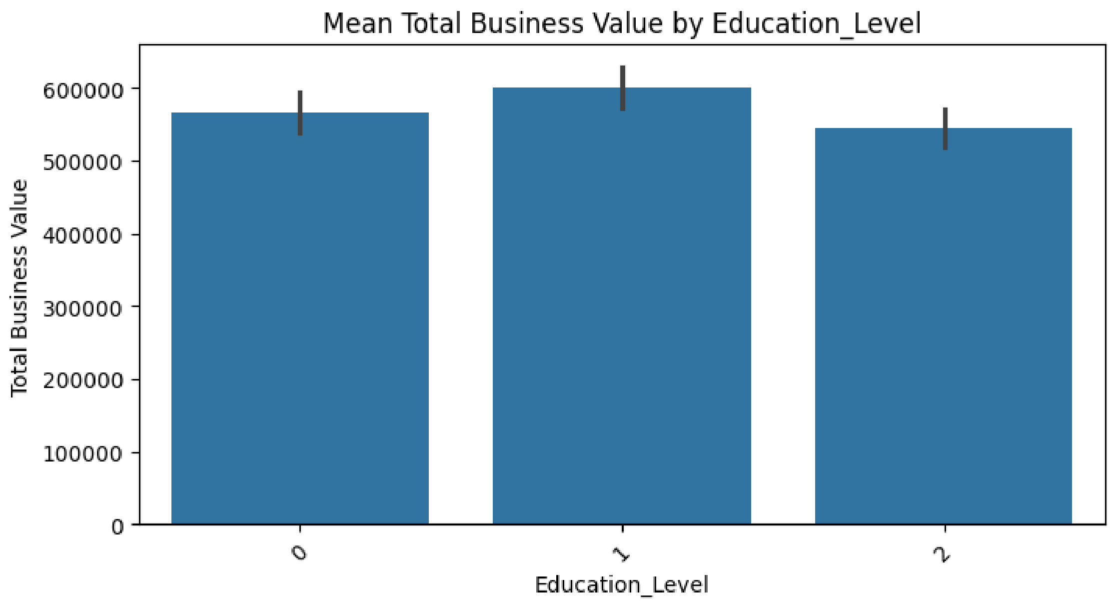

Income vs Joining Designation
Income
175000
150000
125000
100000
75000
50000
25000
1
2
3
4
5
Joining Designation

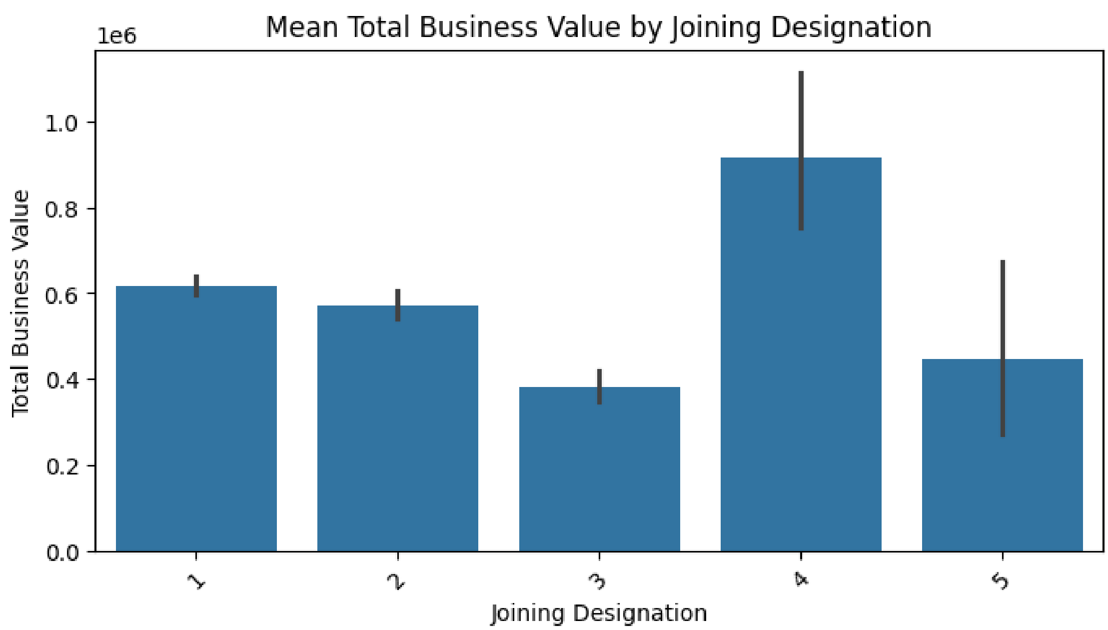

Income vs Grade
175000
150000
125000
100000
75000
50000
25000
Income
1
2
3
4
5
Grade

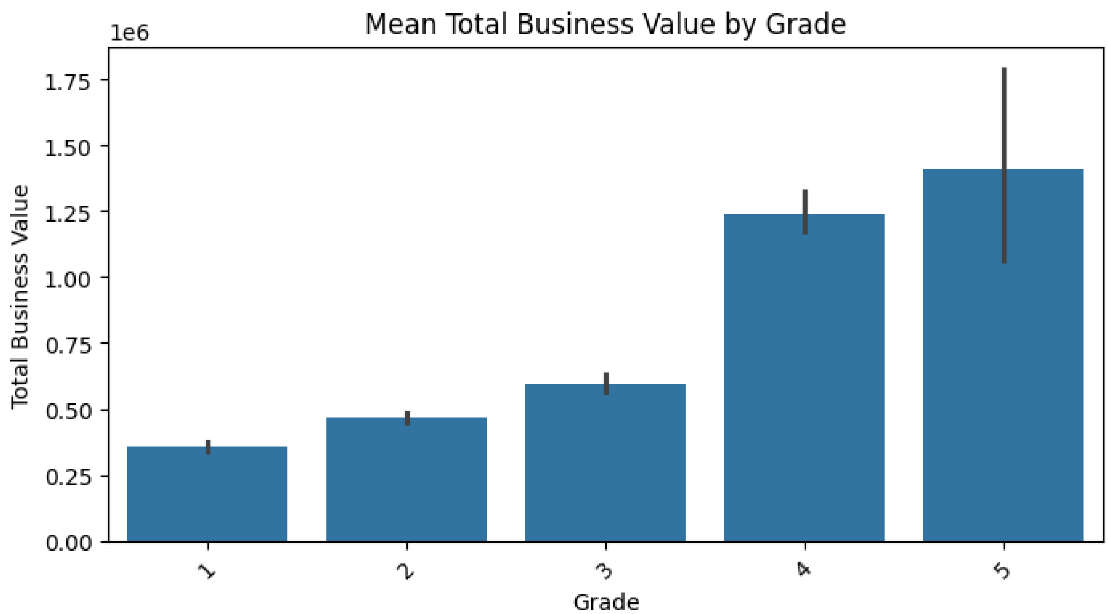

Age
Income
Total Business Value
Quarterly Rating
1e7
20
30
40
50
50000
100000
150000
0
1
2
3
-1
4
1.0
1.5
2.0
2.5
3.0
3.5
4.0

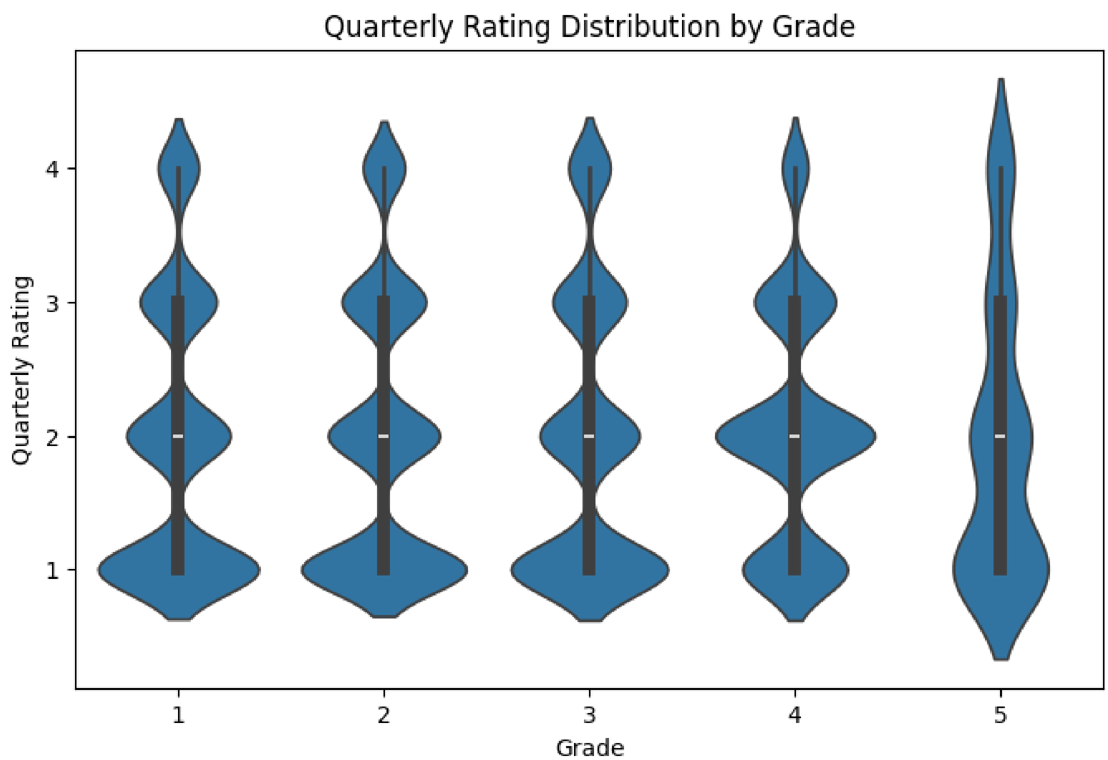

Mean Total Business Value by Gender
Total Business Value
600000
500000
400000
300000
200000
100000
0
0.0
1.0
Gender


Mean Total Business Value by City
1e6
Total Business Value
1.0
0.8
0.6
0.4
0.2
0.0
C1
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
C2
C20
C21
C22
C23
C24
C25
C26
C27
C28
C29
C3
C4
C5
C6
C7
C8
C9
City


Mean Total Business Value by Education_Level
Total Business Value
600000
500000
400000
300000
200000
100000
0
0
1
2
Education_Level


Mean Total Business Value by Joining Designation
1e6
Total Business Value
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
Joining Designation


Mean Total Business Value by Grade
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Total Business Value
1
2
3
4
5
Grade


Quarterly Rating Distribution by Grade
4
3
2
1
Quarterly Rating
1
2
3
4
5
Grade

[26]:
```
# The heat map correctly shows that Driver_ID is not correlated with with any␣
↪other features which is expected.
# Age has a slight positive correlation with income, business value and␣
↪Quaterly rating.
# Total business value has a slight higher correlation with Quaterly rating.

# The box plots for income and gender shows that income is almost the same for␣
↪both the genders
# however, there are a few outliers on the higher end for males showing that␣
↪some males are
# willing to work overtime and make more bucks.

# Income vs city shows that the income is more or less equally distributed with␣
↪all the cities
# with a few outliers

# Education level also does not impact the income too much, except that
# drivers with higher education have a slight higher income and there are␣
↪outliers
# in the higher end for eductaion level 2


# The joining designation of the driver is directly affecting the income.
# Higher the designation, higher the income.

# The grade of the driver is directly affecting the income.
# Higher the grade, higher the income.

# The total business value also seem similar for both the genders

# The mean total business value for different cities are almost same
# but C13 and C29 leads this by a slight margin

# The mean total business value by education level also is almost the same.

# Joining designation 4 has the highest total business value

# Grade 5 has the highest total business value
```

0.0.1 Exploratory Data Analysis Insights

Range of Attributes and Outliers

- **Age**:
  - Average: ~35 years

- 75th percentile: 39 years

- Mean Median → No significant outliers

- **Income**:
  - Average: 65,000

  - Median: 60,000

  - Mean Median → Stable income levels, no major outliers
- **Total Business Value**:
  - Highly right-skewed

  - Few drivers generate high business value

  - Majority on lower end

Distribution of Variables (Univariate Analysis)

- **Age**:
  - Near-normal distribution

  - Slight right skew

  - Most drivers are under 40
- **Income**:
  - Right-skewed

  - Many earn $<$ 100,000

  - Few high earners
- **Total Business Value**:
  - Strictly right-skewed

  - Majority have low business value
- **Quarterly Rating**:
  - Discrete distribution

  - Count decreases as rating increases
- **Gender**:
  - More males than females

  - Female count is not far behind
- **City**:
  - C20 has highest count

  - Followed by C29
- **Education Level**:
  - Most are 12+
  - Followed by Graduates and 10+
- **Joining Designation**:
  - Majority at Designation 1
  - Count decreases with higher designation
- **Grade**:
  - Discrete and right-skewed
  - Fewer high-grade drivers

Relationships Between Variables (Bivariate Analysis)

- **Correlation Heatmap**:
  - `Driver_ID`: No correlation (expected)
  - **Age**: Positive correlation with Income, Business Value, Quarterly Rating
  - **Total Business Value**: Correlated with Quarterly Rating
- **Box Plots**:
  - **Income vs Gender**:
    * Similar across genders
    * Few high-end male outliers
  - **Income vs City**:
    * Uniform across cities
    * Few high-income outliers
  - **Income vs Education Level**:
    * Slight increase with education
    * Outliers for graduates
  - **Income vs Joining Designation**:
    * Higher designation → Higher income
  - **Income vs Grade**:
    * Higher grade → Higher income
- **Total Business Value**:
  - **Gender**: Similar between male and female drivers

- **City**: C13 and C29 have slightly higher average values
- **Education Level**: Uniform across levels
- **Joining Designation**: Designation 4 has highest value
- **Grade**: Grade 5 has highest business value

---

[27]:
```
#Data Preprocessing
```

[28]:
```
df.head()
```

```
/usr/local/lib/python3.11/dist-
packages/google/colab/_dataframe_summarizer.py:88: UserWarning: Could not infer
format, so each element will be parsed individually, falling back to `dateutil`.
To ensure parsing is consistent and as-expected, please specify a format.
  cast_date_col = pd.to_datetime(column, errors="coerce")
```

[28]:
```
   Unnamed: 0    MMM-YY  Driver_ID   Age  Gender City  Education_Level  Income  \
0           0  01/01/19          1  28.0     0.0  C23                2   57387
1           1  02/01/19          1  28.0     0.0  C23                2   57387
2           2  03/01/19          1  28.0     0.0  C23                2   57387
3           3  11/01/20          2  31.0     0.0   C7                2   67016
4           4  12/01/20          2  31.0     0.0   C7                2   67016

  Dateofjoining LastWorkingDate  Joining Designation  Grade  \
0      24/12/18             NaN                    1      1
1      24/12/18             NaN                    1      1
2      24/12/18        03/11/19                    1      1
3      11/06/20             NaN                    2      2
4      11/06/20             NaN                    2      2

   Total Business Value  Quarterly Rating
0               2381060                 2
1               -665480                 2
2                     0                 2
3                     0                 1
4                     0                 1
```

[29]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            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  category
 5   City                  19104 non-null  category
 6   Education_Level       19104 non-null  category
 7   Income                19104 non-null  int64
 8   Dateofjoining         19104 non-null  object
 9   LastWorkingDate       1616 non-null   object
 10  Joining Designation   19104 non-null  category
 11  Grade                 19104 non-null  category
 12  Total Business Value  19104 non-null  int64
 13  Quarterly Rating      19104 non-null  int64
dtypes: category(5), float64(1), int64(5), object(3)
memory usage: 1.4+ MB
```

[30]:
```python
date_cols = ['MMM-YY', 'Dateofjoining', 'LastWorkingDate']
for col in date_cols:
    df[col] = pd.to_datetime(df[col], format='%d/%m/%y', errors='coerce')
```

[31]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            19104 non-null  int64
 1   MMM-YY                19104 non-null  datetime64[ns]
 2   Driver_ID             19104 non-null  int64
 3   Age                   19043 non-null  float64
 4   Gender                19052 non-null  category
 5   City                  19104 non-null  category
 6   Education_Level       19104 non-null  category
 7   Income                19104 non-null  int64
 8   Dateofjoining         19104 non-null  datetime64[ns]
 9   LastWorkingDate       1616 non-null   datetime64[ns]
 10  Joining Designation   19104 non-null  category
 11  Grade                 19104 non-null  category
 12  Total Business Value  19104 non-null  int64
 13  Quarterly Rating      19104 non-null  int64
dtypes: category(5), datetime64[ns](3), float64(1), int64(5)
memory usage: 1.4 MB
```

[32]:
```python
df.head()
```

[32]:
```
   Unnamed: 0     MMM-YY  Driver_ID   Age Gender City Education_Level  Income  \
0           0 2019-01-01          1  28.0    0.0  C23               2   57387
1           1 2019-01-02          1  28.0    0.0  C23               2   57387
2           2 2019-01-03          1  28.0    0.0  C23               2   57387
3           3 2020-01-11          2  31.0    0.0   C7               2   67016
4           4 2020-01-12          2  31.0    0.0   C7               2   67016

  Dateofjoining LastWorkingDate Joining Designation Grade  \
0    2018-12-24             NaT                   1     1
1    2018-12-24             NaT                   1     1
2    2018-12-24      2019-11-03                   1     1
3    2020-06-11             NaT                   2     2
4    2020-06-11             NaT                   2     2

   Total Business Value  Quarterly Rating
0               2381060                 2
1               -665480                 2
2                     0                 2
3                     0                 1
4                     0                 1
```

[33]:
```python
df.shape
```

[33]:
```
(19104, 14)
```

[34]:
```python
import pandas as pd


unique_age_gender_per_driver = df.groupby('Driver_ID').agg(
    unique_age=('Age', lambda x: x.unique().tolist()),
    unique_gender=('Gender', lambda x: x.unique().tolist())
).reset_index()
```

[35]:
```python
unique_age_gender_per_driver.head(50)
```

[35]:
```
    Driver_ID              unique_age unique_gender
0           1                  [28.0]         [0.0]
1           2                  [31.0]         [0.0]
2           4                  [43.0]         [0.0]
3           5                  [29.0]         [0.0]
4           6                  [31.0]         [1.0]
5           8                  [34.0]         [0.0]
6          11                  [28.0]         [1.0]
7          12                  [35.0]         [0.0]
8          13      [29.0, 30.0, 31.0]         [0.0]
9          14                  [39.0]         [1.0]
10         16                  [30.0]         [1.0]
```

```
11          17               [42.0, 43.0]          [0.0]
12          18                     [27.0]          [1.0]
13          20                [26.0, nan]          [1.0]
14          21               [33.0, 34.0]          [1.0]
15          22           [40.0, nan, 41.0]         [0.0]
16          24           [30.0, 31.0, nan]         [0.0]
17          25          [29.0, 30.0, 31.0]         [0.0]
18          26          [41.0, 42.0, 43.0]         [0.0]
19          29                     [30.0]          [0.0]
20          30                     [31.0]          [0.0]
21          31                     [32.0]          [1.0]
22          34                     [28.0]          [1.0]
23          35                     [32.0]          [0.0]
24          36               [40.0, 41.0]          [1.0]
25          37                     [33.0]          [1.0]
26          38                     [22.0]          [0.0]
27          39                     [32.0]          [0.0]
28          40                [nan, 32.0]          [0.0]
29          41          [33.0, 34.0, 35.0]         [0.0]
30          42                     [44.0]          [1.0]
31          43                     [27.0]     [1.0, nan]
32          44                     [28.0]          [0.0]
33          45                     [35.0]          [0.0]
34          46                     [36.0]          [1.0]
35          47                     [30.0]          [0.0]
36          49           [21.0, nan, 22.0]    [0.0, nan]
37          50                     [49.0]          [1.0]
38          51               [33.0, 34.0]          [0.0]
39          52               [36.0, 37.0]          [0.0]
40          54          [33.0, 34.0, 35.0]         [0.0]
41          55                     [30.0]          [1.0]
42          56          [34.0, 35.0, 36.0]         [1.0]
43          57          [36.0, 37.0, 38.0]         [1.0]
44          58                     [41.0]          [0.0]
45          59                     [35.0]          [1.0]
46          60          [46.0, 47.0, 48.0]         [1.0]
47          61                     [32.0]          [0.0]
48          62                     [27.0]          [0.0]
49          63    [26.0, 27.0, nan, 28.0]          [0.0]
```

[36]:

```
#Imputing the missing values for age and gender using KNN imputation.
#This could have been done very easily using aggregation by driver_ID.
#But as the question asks KNN imputation to be used I am using this.
#I am selecting a few columns only to do this. I am leaving out the Driver_ID␣
↪as that column is too obvious
#and would be able to easily fill it off
```

[37]:
```python
from sklearn.impute import KNNImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

df['Dateofjoining'] = pd.to_datetime(df['Dateofjoining'])
df['Joining_Year'] = df['Dateofjoining'].dt.year
df['Joining_Month'] = df['Dateofjoining'].dt.month

categorical_cols = ['City', 'Education_Level']
numerical_cols = ['Income', 'Joining_Year', 'Joining_Month']
features = categorical_cols + numerical_cols

impute_data = df[features + ['Age', 'Gender']].copy()

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore', sparse_output=False),␣
 ↪categorical_cols)
    ],
    remainder='passthrough'
)

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('imputer', KNNImputer(n_neighbors=1)) #After few trials I settled on 1 as␣
 ↪this way I would not get fractions for age and gender.
])

imputed_array = pipeline.fit_transform(impute_data)

age_index = -2
gender_index = -1

df['Age'] = imputed_array[:, age_index]
df['Gender'] = imputed_array[:, gender_index]
```

[38]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            19104 non-null  int64
 1   MMM-YY                19104 non-null  datetime64[ns]
 2   Driver_ID             19104 non-null  int64
```

```
 3   Age                   19104 non-null  float64
 4   Gender                19104 non-null  float64
 5   City                  19104 non-null  category
 6   Education_Level       19104 non-null  category
 7   Income                19104 non-null  int64
 8   Dateofjoining         19104 non-null  datetime64[ns]
 9   LastWorkingDate       1616 non-null   datetime64[ns]
 10  Joining Designation   19104 non-null  category
 11  Grade                 19104 non-null  category
 12  Total Business Value  19104 non-null  int64
 13  Quarterly Rating      19104 non-null  int64
 14  Joining_Year          19104 non-null  int32
 15  Joining_Month         19104 non-null  int32
dtypes: category(4), datetime64[ns](3), float64(2), int32(2), int64(5)
memory usage: 1.7 MB
```

[39]:
```
df.shape
```

[39]:
```
(19104, 16)
```

[40]:
```
import pandas as pd


unique_age_gender_per_driver = df.groupby('Driver_ID').agg(
    unique_age=('Age', lambda x: x.unique().tolist()),
    unique_gender=('Gender', lambda x: x.unique().tolist())
).reset_index()
```

[41]:
```
unique_age_gender_per_driver.head(50)
```

[41]:
```
    Driver_ID          unique_age unique_gender
0           1              [28.0]         [0.0]
1           2              [31.0]         [0.0]
2           4              [43.0]         [0.0]
3           5              [29.0]         [0.0]
4           6              [31.0]         [1.0]
5           8              [34.0]         [0.0]
6          11              [28.0]         [1.0]
7          12              [35.0]         [0.0]
8          13  [29.0, 30.0, 31.0]         [0.0]
9          14              [39.0]         [1.0]
10         16              [30.0]         [1.0]
11         17        [42.0, 43.0]         [0.0]
12         18              [27.0]         [1.0]
13         20              [26.0]         [1.0]
14         21        [33.0, 34.0]         [1.0]
15         22        [40.0, 41.0]         [0.0]
```

```
16          24          [30.0, 31.0]          [0.0]
17          25    [29.0, 30.0, 31.0]          [0.0]
18          26    [41.0, 42.0, 43.0]          [0.0]
19          29                [30.0]          [0.0]
20          30                [31.0]          [0.0]
21          31                [32.0]          [1.0]
22          34                [28.0]          [1.0]
23          35                [32.0]          [0.0]
24          36          [40.0, 41.0]          [1.0]
25          37                [33.0]          [1.0]
26          38                [22.0]          [0.0]
27          39                [32.0]          [0.0]
28          40                [32.0]          [0.0]
29          41    [33.0, 34.0, 35.0]          [0.0]
30          42                [44.0]          [1.0]
31          43                [27.0]          [1.0]
32          44                [28.0]          [0.0]
33          45                [35.0]          [0.0]
34          46                [36.0]          [1.0]
35          47                [30.0]          [0.0]
36          49          [21.0, 22.0]          [0.0]
37          50                [49.0]          [1.0]
38          51          [33.0, 34.0]          [0.0]
39          52          [36.0, 37.0]          [0.0]
40          54    [33.0, 34.0, 35.0]          [0.0]
41          55                [30.0]          [1.0]
42          56    [34.0, 35.0, 36.0]          [1.0]
43          57    [36.0, 37.0, 38.0]          [1.0]
44          58                [41.0]          [0.0]
45          59                [35.0]          [1.0]
46          60    [46.0, 47.0, 48.0]          [1.0]
47          61                [32.0]          [0.0]
48          62                [27.0]          [0.0]
49          63    [26.0, 27.0, 28.0]          [0.0]
```

[42]:
```
#From the before and after checks of the unique_age_gender_per_driver,
#we can see that KNN imputation has helped with replacing missing values with␣
↪appropriate values.
```

[43]:
```
#  Joining_Year
#  Joining_Month
#Dropping the above two temporary columns created
df.drop('Joining_Year', axis=1, inplace=True)
df.drop('Joining_Month', axis=1, inplace=True)
```

[44]:
```
df.head()
```

[44]:
```
   Unnamed: 0     MMM-YY  Driver_ID   Age  Gender City Education_Level  \
0           0 2019-01-01          1  28.0     0.0  C23               2
1           1 2019-01-02          1  28.0     0.0  C23               2
2           2 2019-01-03          1  28.0     0.0  C23               2
3           3 2020-01-11          2  31.0     0.0   C7               2
4           4 2020-01-12          2  31.0     0.0   C7               2

   Income Dateofjoining LastWorkingDate Joining Designation Grade  \
0   57387    2018-12-24             NaT                   1     1
1   57387    2018-12-24             NaT                   1     1
2   57387    2018-12-24      2019-11-03                   1     1
3   67016    2020-06-11             NaT                   2     2
4   67016    2020-06-11             NaT                   2     2

   Total Business Value  Quarterly Rating
0               2381060                 2
1               -665480                 2
2                     0                 2
3                     0                 1
4                     0                 1
```

[45]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            19104 non-null  int64
 1   MMM-YY                19104 non-null  datetime64[ns]
 2   Driver_ID             19104 non-null  int64
 3   Age                   19104 non-null  float64
 4   Gender                19104 non-null  float64
 5   City                  19104 non-null  category
 6   Education_Level       19104 non-null  category
 7   Income                19104 non-null  int64
 8   Dateofjoining         19104 non-null  datetime64[ns]
 9   LastWorkingDate       1616 non-null   datetime64[ns]
 10  Joining Designation   19104 non-null  category
 11  Grade                 19104 non-null  category
 12  Total Business Value  19104 non-null  int64
 13  Quarterly Rating      19104 non-null  int64
dtypes: category(4), datetime64[ns](3), float64(2), int64(5)
memory usage: 1.5 MB
```

[46]:
```
#Creating a column to check whether the QuaterlyRating has increased for the
↪driver.
```

```
df['MMM-YY'] = pd.to_datetime(df['MMM-YY'], format='%b-%y')
df = df.sort_values(by=['Driver_ID', 'MMM-YY'])
df['QRIncrease'] = df.groupby('Driver_ID')['Quarterly Rating'].diff().gt(0).
 ↪astype(int)
```

[47]:
```
df.head(50)
```

[47]:
```
    Unnamed: 0     MMM-YY  Driver_ID   Age  Gender City  Education_Level  \
0            0 2019-01-01          1  28.0     0.0  C23                2
1            1 2019-01-02          1  28.0     0.0  C23                2
2            2 2019-01-03          1  28.0     0.0  C23                2
3            3 2020-01-11          2  31.0     0.0   C7                2
4            4 2020-01-12          2  31.0     0.0   C7                2
5            5 2019-01-12          4  43.0     0.0  C13                2
6            6 2020-01-01          4  43.0     0.0  C13                2
7            7 2020-01-02          4  43.0     0.0  C13                2
8            8 2020-01-03          4  43.0     0.0  C13                2
9            9 2020-01-04          4  43.0     0.0  C13                2
10          10 2019-01-01          5  29.0     0.0   C9                0
11          11 2019-01-02          5  29.0     0.0   C9                0
12          12 2019-01-03          5  29.0     0.0   C9                0
13          13 2020-01-08          6  31.0     1.0  C11                1
14          14 2020-01-09          6  31.0     1.0  C11                1
15          15 2020-01-10          6  31.0     1.0  C11                1
16          16 2020-01-11          6  31.0     1.0  C11                1
17          17 2020-01-12          6  31.0     1.0  C11                1
18          18 2020-01-09          8  34.0     0.0   C2                0
19          19 2020-01-10          8  34.0     0.0   C2                0
20          20 2020-01-11          8  34.0     0.0   C2                0
21          21 2020-01-12         11  28.0     1.0  C19                2
22          22 2019-01-07         12  35.0     0.0  C23                2
23          23 2019-01-08         12  35.0     0.0  C23                2
24          24 2019-01-09         12  35.0     0.0  C23                2
25          25 2019-01-10         12  35.0     0.0  C23                2
26          26 2019-01-11         12  35.0     0.0  C23                2
27          27 2019-01-12         12  35.0     0.0  C23                2
28          28 2019-01-01         13  29.0     0.0  C19                2
29          29 2019-01-02         13  29.0     0.0  C19                2
30          30 2019-01-03         13  29.0     0.0  C19                2
31          31 2019-01-04         13  29.0     0.0  C19                2
32          32 2019-01-05         13  29.0     0.0  C19                2
33          33 2019-01-06         13  29.0     0.0  C19                2
34          34 2019-01-07         13  29.0     0.0  C19                2
35          35 2019-01-08         13  29.0     0.0  C19                2
36          36 2019-01-09         13  29.0     0.0  C19                2
37          37 2019-01-10         13  29.0     0.0  C19                2
```

38	38	2019-01-11	13	30.0	0.0	C19	2
39	39	2019-01-12	13	30.0	0.0	C19	2
40	40	2020-01-01	13	30.0	0.0	C19	2
41	41	2020-01-02	13	30.0	0.0	C19	2
42	42	2020-01-03	13	30.0	0.0	C19	2
43	43	2020-01-04	13	30.0	0.0	C19	2
44	44	2020-01-05	13	30.0	0.0	C19	2
45	45	2020-01-06	13	30.0	0.0	C19	2
46	46	2020-01-07	13	30.0	0.0	C19	2
47	47	2020-01-08	13	30.0	0.0	C19	2
48	48	2020-01-09	13	30.0	0.0	C19	2
49	49	2020-01-10	13	30.0	0.0	C19	2

	Income	Dateofjoining	LastWorkingDate	Joining Designation	Grade	\
0	57387	2018-12-24	NaT	1	1	
1	57387	2018-12-24	NaT	1	1	
2	57387	2018-12-24	2019-11-03	1	1	
3	67016	2020-06-11	NaT	2	2	
4	67016	2020-06-11	NaT	2	2	
5	65603	2019-07-12	NaT	2	2	
6	65603	2019-07-12	NaT	2	2	
7	65603	2019-07-12	NaT	2	2	
8	65603	2019-07-12	NaT	2	2	
9	65603	2019-07-12	2020-04-27	2	2	
10	46368	2019-09-01	NaT	1	1	
11	46368	2019-09-01	NaT	1	1	
12	46368	2019-09-01	2019-07-03	1	1	
13	78728	2020-07-31	NaT	3	3	
14	78728	2020-07-31	NaT	3	3	
15	78728	2020-07-31	NaT	3	3	
16	78728	2020-07-31	NaT	3	3	
17	78728	2020-07-31	NaT	3	3	
18	70656	2020-09-19	NaT	3	3	
19	70656	2020-09-19	NaT	3	3	
20	70656	2020-09-19	2020-11-15	3	3	
21	42172	2020-07-12	NaT	1	1	
22	28116	2019-06-29	NaT	1	1	
23	28116	2019-06-29	NaT	1	1	
24	28116	2019-06-29	NaT	1	1	
25	28116	2019-06-29	NaT	1	1	
26	28116	2019-06-29	NaT	1	1	
27	28116	2019-06-29	2019-12-21	1	1	
28	119227	2015-05-28	NaT	1	4	
29	119227	2015-05-28	NaT	1	4	
30	119227	2015-05-28	NaT	1	4	
31	119227	2015-05-28	NaT	1	4	
32	119227	2015-05-28	NaT	1	4	

33	119227	2015-05-28	NaT	1	4
34	119227	2015-05-28	NaT	1	4
35	119227	2015-05-28	NaT	1	4
36	119227	2015-05-28	NaT	1	4
37	119227	2015-05-28	NaT	1	4
38	119227	2015-05-28	NaT	1	4
39	119227	2015-05-28	NaT	1	4
40	119227	2015-05-28	NaT	1	4
41	119227	2015-05-28	NaT	1	4
42	119227	2015-05-28	NaT	1	4
43	119227	2015-05-28	NaT	1	4
44	119227	2015-05-28	NaT	1	4
45	119227	2015-05-28	NaT	1	4
46	119227	2015-05-28	NaT	1	4
47	119227	2015-05-28	NaT	1	4
48	119227	2015-05-28	NaT	1	4
49	119227	2015-05-28	NaT	1	4

	Total Business Value	Quarterly Rating	QRIncrease
0	2381060	2	0
1	-665480	2	0
2	0	2	0
3	0	1	0
4	0	1	0
5	0	1	0
6	0	1	0
7	0	1	0
8	350000	1	0
9	0	1	0
10	0	1	0
11	120360	1	0
12	0	1	0
13	0	1	0
14	0	1	0
15	0	2	1
16	1265000	2	0
17	0	2	0
18	0	1	0
19	0	1	0
20	0	1	0
21	0	1	0
22	500000	4	0
23	1707180	4	0
24	400000	4	0
25	0	1	0
26	0	1	0
27	0	1	0

28	250000	1	0
29	1719680	1	0
30	545240	1	0
31	250000	2	1
32	895510	2	0
33	0	2	0
34	350650	2	0
35	708360	2	0
36	1190290	2	0
37	0	1	0
38	200000	1	0
39	300000	1	0
40	151110	1	0
41	200000	1	0
42	1593590	1	0
43	0	1	0
44	400000	1	0
45	258610	1	0
46	150000	1	0
47	500000	1	0
48	200000	1	0
49	350000	1	0

[48]:
```
#We can see that this column has values as 1 when the Quaterly rating has␣
↪increased.
```

[49]:
```
#Creating a column to check whether the Income has increased for the driver.

df = df.sort_values(by=['Driver_ID', 'MMM-YY'])
df['IncomeIncreased'] = df.groupby('Driver_ID')['Income'].diff().gt(0).
↪astype(int)
```

[50]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            19104 non-null  int64
 1   MMM-YY                19104 non-null  datetime64[ns]
 2   Driver_ID             19104 non-null  int64
 3   Age                   19104 non-null  float64
 4   Gender                19104 non-null  float64
 5   City                  19104 non-null  category
 6   Education_Level       19104 non-null  category
 7   Income                19104 non-null  int64
```

```
 8   Dateofjoining         19104 non-null  datetime64[ns]
 9   LastWorkingDate       1616 non-null   datetime64[ns]
 10  Joining Designation   19104 non-null  category
 11  Grade                 19104 non-null  category
 12  Total Business Value  19104 non-null  int64
 13  Quarterly Rating      19104 non-null  int64
 14  QRIncrease            19104 non-null  int64
 15  IncomeIncreased       19104 non-null  int64
dtypes: category(4), datetime64[ns](3), float64(2), int64(7)
memory usage: 1.8 MB
```

[51]:
```python
df.head()
```

[51]:
```
   Unnamed: 0     MMM-YY  Driver_ID   Age  Gender City Education_Level  \
0           0 2019-01-01          1  28.0     0.0  C23               2
1           1 2019-01-02          1  28.0     0.0  C23               2
2           2 2019-01-03          1  28.0     0.0  C23               2
3           3 2020-01-11          2  31.0     0.0   C7               2
4           4 2020-01-12          2  31.0     0.0   C7               2

   Income Dateofjoining LastWorkingDate Joining Designation Grade  \
0   57387    2018-12-24             NaT                   1     1
1   57387    2018-12-24             NaT                   1     1
2   57387    2018-12-24      2019-11-03                   1     1
3   67016    2020-06-11             NaT                   2     2
4   67016    2020-06-11             NaT                   2     2

   Total Business Value  Quarterly Rating  QRIncrease  IncomeIncreased
0               2381060                 2           0                0
1               -665480                 2           0                0
2                     0                 2           0                0
3                     0                 1           0                0
4                     0                 1           0                0
```

[52]:
```python
df.drop('Unnamed: 0', axis=1, inplace=True)
```

[53]:
```python
df.head()
```

[53]:
```
      MMM-YY  Driver_ID   Age  Gender City Education_Level  Income  \
0 2019-01-01          1  28.0     0.0  C23               2   57387
1 2019-01-02          1  28.0     0.0  C23               2   57387
2 2019-01-03          1  28.0     0.0  C23               2   57387
3 2020-01-11          2  31.0     0.0   C7               2   67016
4 2020-01-12          2  31.0     0.0   C7               2   67016

  Dateofjoining LastWorkingDate Joining Designation Grade  \
0    2018-12-24             NaT                   1     1
```

```
1    2018-12-24          NaT              1     1
2    2018-12-24   2019-11-03              1     1
3    2020-06-11          NaT              2     2
4    2020-06-11          NaT              2     2

   Total Business Value  Quarterly Rating  QRIncrease  IncomeIncreased
0               2381060                 2           0                0
1               -665480                 2           0                0
2                     0                 2           0                0
3                     0                 1           0                0
4                     0                 1           0                0
```

[54]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   MMM-YY                19104 non-null  datetime64[ns]
 1   Driver_ID             19104 non-null  int64
 2   Age                   19104 non-null  float64
 3   Gender                19104 non-null  float64
 4   City                  19104 non-null  category
 5   Education_Level       19104 non-null  category
 6   Income                19104 non-null  int64
 7   Dateofjoining         19104 non-null  datetime64[ns]
 8   LastWorkingDate       1616 non-null   datetime64[ns]
 9   Joining Designation   19104 non-null  category
 10  Grade                 19104 non-null  category
 11  Total Business Value  19104 non-null  int64
 12  Quarterly Rating      19104 non-null  int64
 13  QRIncrease            19104 non-null  int64
 14  IncomeIncreased       19104 non-null  int64
dtypes: category(4), datetime64[ns](3), float64(2), int64(6)
memory usage: 1.7 MB
```

[55]:
```python
df['Education_Level'] = pd.to_numeric(df['Education_Level'], errors='coerce')
df['Joining Designation'] = pd.to_numeric(df['Joining Designation'],␣
 ↪errors='coerce')
df['Grade'] = pd.to_numeric(df['Grade'], errors='coerce')
```

[56]:
```python
df.head()
```

[56]:
```
      MMM-YY  Driver_ID   Age  Gender City  Education_Level  Income  \
0 2019-01-01          1  28.0     0.0  C23                2   57387
1 2019-01-02          1  28.0     0.0  C23                2   57387
```

```
2 2019-01-03          1  28.0     0.0  C23               2   57387
3 2020-01-11          2  31.0     0.0   C7               2   67016
4 2020-01-12          2  31.0     0.0   C7               2   67016

  Dateofjoining LastWorkingDate  Joining Designation  Grade  \
0    2018-12-24             NaT                    1      1
1    2018-12-24             NaT                    1      1
2    2018-12-24      2019-11-03                    1      1
3    2020-06-11             NaT                    2      2
4    2020-06-11             NaT                    2      2

   Total Business Value  Quarterly Rating  QRIncrease  IncomeIncreased
0               2381060                 2           0                0
1               -665480                 2           0                0
2                     0                 2           0                0
3                     0                 1           0                0
4                     0                 1           0                0
```

[57]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19104 entries, 0 to 19103
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   MMM-YY                19104 non-null  datetime64[ns]
 1   Driver_ID             19104 non-null  int64
 2   Age                   19104 non-null  float64
 3   Gender                19104 non-null  float64
 4   City                  19104 non-null  category
 5   Education_Level       19104 non-null  int64
 6   Income                19104 non-null  int64
 7   Dateofjoining         19104 non-null  datetime64[ns]
 8   LastWorkingDate       1616 non-null   datetime64[ns]
 9   Joining Designation   19104 non-null  int64
 10  Grade                 19104 non-null  int64
 11  Total Business Value  19104 non-null  int64
 12  Quarterly Rating      19104 non-null  int64
 13  QRIncrease            19104 non-null  int64
 14  IncomeIncreased       19104 non-null  int64
dtypes: category(1), datetime64[ns](3), float64(2), int64(9)
memory usage: 2.1 MB
```

[58]:

```
#Now I am aggregating all the columns based on Driver_ID with the below rules
# MMM-YY : Get the number of times the driver has reported in 2019 and in 2020
↪and create 2 columns. After that drop the MMM-YY column.
```

```
# Driver_ID the column needed for aggregation. Nothing to be done.

# Age : get the mean age based on Driver_ID
# Gender : get the mode value based on Driver_ID
# City : get the mode value based on Driver_ID
# Education_Level : get the highest value based on Driver_ID
# Income : get the mean value based on Driver_ID
# Dateofjoining : get the mode value based on Driver_ID.
# LastWorkingDate : if at least one cell has the date, then have it present for␣
↪this Driver_ID, if not leave it blank
# Joining Designation : get the mode value based on Driver_ID
# Grade : get the mode value based on Driver_ID
# Total Business Value : get the mean value based on Driver_ID
# Quarterly Rating : get the mode value based on Driver_ID
# QRIncrease : if the Driver_ID has at least one 1, then have it as 1, else as 0
# IncomeIncreased : if the Driver_ID has at least one 1, then have it as 1,␣
↪else as 0


df['Reports_2019'] = (df['MMM-YY'].dt.year == 2019).astype(int)
df['Reports_2020'] = (df['MMM-YY'].dt.year == 2020).astype(int)

agg_df = df.groupby('Driver_ID').agg({
    'Reports_2019': 'sum',
    'Reports_2020': 'sum',
    'Age': 'mean',
    'Gender': lambda x: x.mode().iloc[0],
    'City': lambda x: x.mode().iloc[0],
    'Education_Level': 'max',
    'Income': 'mean',
    'Dateofjoining': lambda x: x.mode().iloc[0],
    'LastWorkingDate': lambda x: x.dropna().iloc[0] if x.dropna().any() else pd.
↪NaT,
    'Joining Designation': lambda x: x.mode().iloc[0],
    'Grade': lambda x: x.mode().iloc[0],
    'Total Business Value': 'mean',
    'Quarterly Rating': lambda x: x.mode().iloc[0],
    'QRIncrease': 'max',
    'IncomeIncreased': 'max'
}).reset_index()
```

```
<ipython-input-58-82858c4a6d6d>:33: FutureWarning: 'any' with datetime64 dtypes
is deprecated and will raise in a future version. Use (obj !=
pd.Timestamp(0)).any() instead.
  'LastWorkingDate': lambda x: x.dropna().iloc[0] if x.dropna().any() else
pd.NaT,
```

[59]:
```
df.head()
```

[59]:
```
      MMM-YY  Driver_ID   Age  Gender City  Education_Level  Income  \
0 2019-01-01          1  28.0     0.0  C23                2   57387
1 2019-01-02          1  28.0     0.0  C23                2   57387
2 2019-01-03          1  28.0     0.0  C23                2   57387
3 2020-01-11          2  31.0     0.0   C7                2   67016
4 2020-01-12          2  31.0     0.0   C7                2   67016

  Dateofjoining LastWorkingDate  Joining Designation  Grade  \
0    2018-12-24             NaT                    1      1
1    2018-12-24             NaT                    1      1
2    2018-12-24      2019-11-03                    1      1
3    2020-06-11             NaT                    2      2
4    2020-06-11             NaT                    2      2

   Total Business Value  Quarterly Rating  QRIncrease  IncomeIncreased  \
0               2381060                 2           0                0
1               -665480                 2           0                0
2                     0                 2           0                0
3                     0                 1           0                0
4                     0                 1           0                0

   Reports_2019  Reports_2020
0             1             0
1             1             0
2             1             0
3             0             1
4             0             1
```

[60]:
```
agg_df.head()
```

[60]:
```
   Driver_ID  Reports_2019  Reports_2020   Age  Gender City  Education_Level  \
0          1             3             0  28.0     0.0  C23                2
1          2             0             2  31.0     0.0   C7                2
2          4             1             4  43.0     0.0  C13                2
3          5             3             0  29.0     0.0   C9                0
4          6             0             5  31.0     1.0  C11                1

    Income Dateofjoining LastWorkingDate  Joining Designation  Grade  \
0  57387.0    2018-12-24      2019-11-03                    1      1
1  67016.0    2020-06-11             NaT                    2      2
2  65603.0    2019-07-12      2020-04-27                    2      2
3  46368.0    2019-09-01      2019-07-03                    1      1
4  78728.0    2020-07-31             NaT                    3      3

   Total Business Value  Quarterly Rating  QRIncrease  IncomeIncreased
```

0	571860.0	2	0	0
1	0.0	1	0	0
2	70000.0	1	0	0
3	40120.0	1	0	0
4	253000.0	2	1	0

[61]:
```
agg_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2381 entries, 0 to 2380
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Driver_ID             2381 non-null   int64
 1   Reports_2019          2381 non-null   int64
 2   Reports_2020          2381 non-null   int64
 3   Age                   2381 non-null   float64
 4   Gender                2381 non-null   float64
 5   City                  2381 non-null   category
 6   Education_Level       2381 non-null   int64
 7   Income                2381 non-null   float64
 8   Dateofjoining         2381 non-null   datetime64[ns]
 9   LastWorkingDate       1616 non-null   datetime64[ns]
 10  Joining Designation   2381 non-null   int64
 11  Grade                 2381 non-null   int64
 12  Total Business Value  2381 non-null   float64
 13  Quarterly Rating      2381 non-null   int64
 14  QRIncrease            2381 non-null   int64
 15  IncomeIncreased       2381 non-null   int64
dtypes: category(1), datetime64[ns](2), float64(4), int64(9)
memory usage: 282.8 KB
```

[62]:
```
agg_df.shape
```

[62]: (2381, 16)

[63]:
```
#One thing we can see from here is that 1616 out of 2381 drivers have quit
#which is 1616/2381 = 0.678 i.e. about 67% of the drivers have quit.
#This is alarming because this represents a huge chunk of drivers.
#And for a business that derives it's revenue from drivers, this is a huge red␣
↪flag.
```

[64]:
```
#Creating a new column for target
agg_df['didDriverQuit'] = agg_df['LastWorkingDate'].notna().astype(int)
```

[65]:
```
agg_df.shape
```

[65]: (2381, 17)

[66]:
```
agg_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2381 entries, 0 to 2380
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Driver_ID             2381 non-null   int64
 1   Reports_2019          2381 non-null   int64
 2   Reports_2020          2381 non-null   int64
 3   Age                   2381 non-null   float64
 4   Gender                2381 non-null   float64
 5   City                  2381 non-null   category
 6   Education_Level       2381 non-null   int64
 7   Income                2381 non-null   float64
 8   Dateofjoining         2381 non-null   datetime64[ns]
 9   LastWorkingDate       1616 non-null   datetime64[ns]
 10  Joining Designation   2381 non-null   int64
 11  Grade                 2381 non-null   int64
 12  Total Business Value  2381 non-null   float64
 13  Quarterly Rating      2381 non-null   int64
 14  QRIncrease            2381 non-null   int64
 15  IncomeIncreased       2381 non-null   int64
 16  didDriverQuit         2381 non-null   int64
dtypes: category(1), datetime64[ns](2), float64(4), int64(10)
memory usage: 301.4 KB
```

[67]:
```
agg_df.head()
```

[67]:
```
   Driver_ID  Reports_2019  Reports_2020   Age  Gender City  Education_Level  \
0          1             3             0  28.0     0.0  C23                2
1          2             0             2  31.0     0.0   C7                2
2          4             1             4  43.0     0.0  C13                2
3          5             3             0  29.0     0.0   C9                0
4          6             0             5  31.0     1.0  C11                1

    Income Dateofjoining LastWorkingDate  Joining Designation  Grade  \
0  57387.0    2018-12-24      2019-11-03                    1      1
1  67016.0    2020-06-11             NaT                    2      2
2  65603.0    2019-07-12      2020-04-27                    2      2
3  46368.0    2019-09-01      2019-07-03                    1      1
4  78728.0    2020-07-31             NaT                    3      3

   Total Business Value  Quarterly Rating  QRIncrease  IncomeIncreased  \
0              571860.0                 2           0                0
```

```
1            0.0              1          0               0
2        70000.0              1          0               0
3        40120.0              1          0               0
4       253000.0              2          1               0

   didDriverQuit
0              1
1              0
2              1
3              1
4              0
```

[68]:

```
#Now I am calculating the tenure of each driver and then dropping the columns␣
 ↪Dateofjoining and LastWorkingDate
#This was I get the info which both these columns give me without having to␣
 ↪keep both these.
#Also, the format of the data is more ML friendly after doing this.
agg_df['LastWorkingDate'] = agg_df['LastWorkingDate'].fillna(pd.
 ↪Timestamp('2020-12-31'))
agg_df['tenure'] = (agg_df['LastWorkingDate'] - agg_df['Dateofjoining']).dt.
 ↪days + 1
agg_df = agg_df.drop(columns=['Dateofjoining', 'LastWorkingDate'])
```

[69]:

```
agg_df.head()
```

[69]:

```
   Driver_ID  Reports_2019  Reports_2020   Age  Gender City  Education_Level  \
0          1             3             0  28.0     0.0  C23                2
1          2             0             2  31.0     0.0   C7                2
2          4             1             4  43.0     0.0  C13                2
3          5             3             0  29.0     0.0   C9                0
4          6             0             5  31.0     1.0  C11                1

    Income  Joining Designation  Grade  Total Business Value  \
0  57387.0                    1      1              571860.0
1  67016.0                    2      2                   0.0
2  65603.0                    2      2               70000.0
3  46368.0                    1      1               40120.0
4  78728.0                    3      3              253000.0

   Quarterly Rating  QRIncrease  IncomeIncreased  didDriverQuit  tenure
0                 2           0                0              1     315
1                 1           0                0              0     204
2                 1           0                0              1     291
3                 1           0                0              1     -59
4                 2           1                0              0     154
```

[70]:

```
agg_df.shape
```

[70]: (2381, 16)

[71]:
```
agg_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2381 entries, 0 to 2380
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Driver_ID              2381 non-null   int64
 1   Reports_2019           2381 non-null   int64
 2   Reports_2020           2381 non-null   int64
 3   Age                    2381 non-null   float64
 4   Gender                 2381 non-null   float64
 5   City                   2381 non-null   category
 6   Education_Level        2381 non-null   int64
 7   Income                 2381 non-null   float64
 8   Joining Designation    2381 non-null   int64
 9   Grade                  2381 non-null   int64
 10  Total Business Value   2381 non-null   float64
 11  Quarterly Rating       2381 non-null   int64
 12  QRIncrease             2381 non-null   int64
 13  IncomeIncreased        2381 non-null   int64
 14  didDriverQuit          2381 non-null   int64
 15  tenure                 2381 non-null   int64
dtypes: category(1), float64(4), int64(11)
memory usage: 282.8 KB
```

[72]:
```
#Now with this we have a very clean data set.
#But we see that the column city is categorical and needs to be converted to␣
↪numerical format
#We can do that using one-hot encoding
```

[73]:
```
agg_df['City'].nunique(dropna=False)
```

[73]: 29

[74]:
```
agg_df['City'].value_counts(dropna=False)
```

[74]:
```
City
C20    152
C15    101
C29     96
C26     93
C8      89
C27     89
C10     86
```

```
C16     84
C22     82
C3      82
C28     82
C12     81
C1      80
C5      80
C21     79
C14     79
C6      78
C4      77
C7      76
C9      75
C23     74
C25     74
C24     73
C19     72
C2      72
C13     71
C17     71
C18     69
C11     64
Name: count, dtype: int64
```

[75]:
```
#We can see that none of the cities can be dropped as the count of each is not␣
↪too less.
#Also, as seen earlier, the income and business values from each of the cities␣
↪are comparable and
#dropping any city may lead to data loss.
```

[76]:
```
#agg_df = pd.get_dummies(agg_df, columns=['City'], drop_first=True)
agg_df = pd.get_dummies(agg_df, columns=['City'], drop_first=True, dtype=int)
```

[77]:
```
agg_df.head()
```

[77]:
```
   Driver_ID  Reports_2019  Reports_2020   Age  Gender  Education_Level  \
0          1             3             0  28.0     0.0                2
1          2             0             2  31.0     0.0                2
2          4             1             4  43.0     0.0                2
3          5             3             0  29.0     0.0                0
4          6             0             5  31.0     1.0                1

    Income  Joining Designation  Grade  Total Business Value  …  City_C27  \
0  57387.0                    1      1              571860.0  …         0
1  67016.0                    2      2                   0.0  …         0
2  65603.0                    2      2               70000.0  …         0
3  46368.0                    1      1               40120.0  …         0
```

```
4  78728.0                     3      3              253000.0  …          0

   City_C28  City_C29  City_C3  City_C4  City_C5  City_C6  City_C7  City_C8  \
0         0         0        0        0        0        0        0        0
1         0         0        0        0        0        0        1        0
2         0         0        0        0        0        0        0        0
3         0         0        0        0        0        0        0        0
4         0         0        0        0        0        0        0        0

   City_C9
0        0
1        0
2        0
3        1
4        0

[5 rows x 43 columns]
```

[78]:
```
agg_df.shape
```

[78]:
```
(2381, 43)
```

[79]:
```
agg_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2381 entries, 0 to 2380
Data columns (total 43 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Driver_ID             2381 non-null   int64
 1   Reports_2019          2381 non-null   int64
 2   Reports_2020          2381 non-null   int64
 3   Age                   2381 non-null   float64
 4   Gender                2381 non-null   float64
 5   Education_Level       2381 non-null   int64
 6   Income                2381 non-null   float64
 7   Joining Designation   2381 non-null   int64
 8   Grade                 2381 non-null   int64
 9   Total Business Value  2381 non-null   float64
 10  Quarterly Rating      2381 non-null   int64
 11  QRIncrease            2381 non-null   int64
 12  IncomeIncreased       2381 non-null   int64
 13  didDriverQuit         2381 non-null   int64
 14  tenure                2381 non-null   int64
 15  City_C10              2381 non-null   int64
 16  City_C11              2381 non-null   int64
 17  City_C12              2381 non-null   int64
```

```
 18  City_C13              2381 non-null   int64
 19  City_C14              2381 non-null   int64
 20  City_C15              2381 non-null   int64
 21  City_C16              2381 non-null   int64
 22  City_C17              2381 non-null   int64
 23  City_C18              2381 non-null   int64
 24  City_C19              2381 non-null   int64
 25  City_C2               2381 non-null   int64
 26  City_C20              2381 non-null   int64
 27  City_C21              2381 non-null   int64
 28  City_C22              2381 non-null   int64
 29  City_C23              2381 non-null   int64
 30  City_C24              2381 non-null   int64
 31  City_C25              2381 non-null   int64
 32  City_C26              2381 non-null   int64
 33  City_C27              2381 non-null   int64
 34  City_C28              2381 non-null   int64
 35  City_C29              2381 non-null   int64
 36  City_C3               2381 non-null   int64
 37  City_C4               2381 non-null   int64
 38  City_C5               2381 non-null   int64
 39  City_C6               2381 non-null   int64
 40  City_C7               2381 non-null   int64
 41  City_C8               2381 non-null   int64
 42  City_C9               2381 non-null   int64
dtypes: float64(4), int64(39)
memory usage: 800.0 KB
```

[80]:

```
#Standardizing the data with the exception of yes/no columns and on hot encoded␣
↪columns
from sklearn.preprocessing import StandardScaler

cols_to_exclude = ['didDriverQuit', 'QRIncrease', 'IncomeIncreased', 'Gender',␣
↪'Driver_ID'] + [col for col in agg_df.columns if col.startswith('City_')]
cols_to_standardize = [col for col in agg_df.columns if col not in␣
↪cols_to_exclude and agg_df[col].dtype in ['int64', 'float64']]

scaler = StandardScaler()
agg_df[cols_to_standardize] = scaler.fit_transform(agg_df[cols_to_standardize])
```

[81]:

```
agg_df.head()
```

[81]:

```
   Driver_ID  Reports_2019  Reports_2020       Age  Gender  Education_Level  \
0          1     -0.244434     -0.933705 -0.911769     0.0         1.216049
1          2     -0.925134     -0.460493 -0.402355     0.0         1.216049
2          4     -0.698234      0.012720  1.635304     0.0         1.216049
3          5     -0.244434     -0.933705 -0.741964     0.0        -1.234575
```

```
4          6     -0.925134      0.249326 -0.402355     1.0         -0.009263

     Income  Joining Designation     Grade  Total Business Value  …  \
0 -0.065228            -0.975022 -1.158317              0.577950  …
1  0.275112             0.213676 -0.084348             -0.694331  …
2  0.225169             0.213676 -0.084348             -0.538595  …
3 -0.454699            -0.975022 -1.158317             -0.605072  …
4  0.689077             1.402374  0.989621             -0.131454  …

   City_C27  City_C28  City_C29  City_C3  City_C4  City_C5  City_C6  City_C7  \
0         0         0         0        0        0        0        0        0
1         0         0         0        0        0        0        0        1
2         0         0         0        0        0        0        0        0
3         0         0         0        0        0        0        0        0
4         0         0         0        0        0        0        0        0

   City_C8  City_C9
0        0        0
1        0        0
2        0        0
3        0        1
4        0        0

[5 rows x 43 columns]
```

[82]:
```python
agg_df.shape
```

[82]:
```
(2381, 43)
```

[83]:
```python
agg_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2381 entries, 0 to 2380
Data columns (total 43 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Driver_ID             2381 non-null   int64
 1   Reports_2019          2381 non-null   float64
 2   Reports_2020          2381 non-null   float64
 3   Age                   2381 non-null   float64
 4   Gender                2381 non-null   float64
 5   Education_Level       2381 non-null   float64
 6   Income                2381 non-null   float64
 7   Joining Designation   2381 non-null   float64
 8   Grade                 2381 non-null   float64
 9   Total Business Value  2381 non-null   float64
 10  Quarterly Rating      2381 non-null   float64
```

```
 11  QRIncrease          2381 non-null   int64
 12  IncomeIncreased     2381 non-null   int64
 13  didDriverQuit       2381 non-null   int64
 14  tenure              2381 non-null   float64
 15  City_C10            2381 non-null   int64
 16  City_C11            2381 non-null   int64
 17  City_C12            2381 non-null   int64
 18  City_C13            2381 non-null   int64
 19  City_C14            2381 non-null   int64
 20  City_C15            2381 non-null   int64
 21  City_C16            2381 non-null   int64
 22  City_C17            2381 non-null   int64
 23  City_C18            2381 non-null   int64
 24  City_C19            2381 non-null   int64
 25  City_C2             2381 non-null   int64
 26  City_C20            2381 non-null   int64
 27  City_C21            2381 non-null   int64
 28  City_C22            2381 non-null   int64
 29  City_C23            2381 non-null   int64
 30  City_C24            2381 non-null   int64
 31  City_C25            2381 non-null   int64
 32  City_C26            2381 non-null   int64
 33  City_C27            2381 non-null   int64
 34  City_C28            2381 non-null   int64
 35  City_C29            2381 non-null   int64
 36  City_C3             2381 non-null   int64
 37  City_C4             2381 non-null   int64
 38  City_C5             2381 non-null   int64
 39  City_C6             2381 non-null   int64
 40  City_C7             2381 non-null   int64
 41  City_C8             2381 non-null   int64
 42  City_C9             2381 non-null   int64
dtypes: float64(11), int64(32)
memory usage: 800.0 KB
```

[84]:
```python
#Separating the feature and target variables
```

[85]:
```python
Y_all = agg_df['didDriverQuit']
X_all = agg_df.drop(columns=['didDriverQuit'])
```

[86]:
```python
X_all.shape
```

[86]:
```
(2381, 42)
```

[87]:
```python
X_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2381 entries, 0 to 2380
```

```
Data columns (total 42 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Driver_ID             2381 non-null   int64
 1   Reports_2019          2381 non-null   float64
 2   Reports_2020          2381 non-null   float64
 3   Age                   2381 non-null   float64
 4   Gender                2381 non-null   float64
 5   Education_Level       2381 non-null   float64
 6   Income                2381 non-null   float64
 7   Joining Designation   2381 non-null   float64
 8   Grade                 2381 non-null   float64
 9   Total Business Value  2381 non-null   float64
 10  Quarterly Rating      2381 non-null   float64
 11  QRIncrease            2381 non-null   int64
 12  IncomeIncreased       2381 non-null   int64
 13  tenure                2381 non-null   float64
 14  City_C10              2381 non-null   int64
 15  City_C11              2381 non-null   int64
 16  City_C12              2381 non-null   int64
 17  City_C13              2381 non-null   int64
 18  City_C14              2381 non-null   int64
 19  City_C15              2381 non-null   int64
 20  City_C16              2381 non-null   int64
 21  City_C17              2381 non-null   int64
 22  City_C18              2381 non-null   int64
 23  City_C19              2381 non-null   int64
 24  City_C2               2381 non-null   int64
 25  City_C20              2381 non-null   int64
 26  City_C21              2381 non-null   int64
 27  City_C22              2381 non-null   int64
 28  City_C23              2381 non-null   int64
 29  City_C24              2381 non-null   int64
 30  City_C25              2381 non-null   int64
 31  City_C26              2381 non-null   int64
 32  City_C27              2381 non-null   int64
 33  City_C28              2381 non-null   int64
 34  City_C29              2381 non-null   int64
 35  City_C3               2381 non-null   int64
 36  City_C4               2381 non-null   int64
 37  City_C5               2381 non-null   int64
 38  City_C6               2381 non-null   int64
 39  City_C7               2381 non-null   int64
 40  City_C8               2381 non-null   int64
 41  City_C9               2381 non-null   int64
dtypes: float64(11), int64(31)
memory usage: 781.4 KB
```

[88]:
```
X_all.head()
```

[88]:
```
   Driver_ID  Reports_2019  Reports_2020       Age  Gender  Education_Level  \
0          1     -0.244434     -0.933705 -0.911769     0.0         1.216049
1          2     -0.925134     -0.460493 -0.402355     0.0         1.216049
2          4     -0.698234      0.012720  1.635304     0.0         1.216049
3          5     -0.244434     -0.933705 -0.741964     0.0        -1.234575
4          6     -0.925134      0.249326 -0.402355     1.0        -0.009263

     Income  Joining Designation     Grade  Total Business Value  …  \
0 -0.065228            -0.975022 -1.158317              0.577950  …
1  0.275112             0.213676 -0.084348             -0.694331  …
2  0.225169             0.213676 -0.084348             -0.538595  …
3 -0.454699            -0.975022 -1.158317             -0.605072  …
4  0.689077             1.402374  0.989621             -0.131454  …

   City_C27  City_C28  City_C29  City_C3  City_C4  City_C5  City_C6  City_C7  \
0         0         0         0        0        0        0        0        0
1         0         0         0        0        0        0        0        1
2         0         0         0        0        0        0        0        0
3         0         0         0        0        0        0        0        0
4         0         0         0        0        0        0        0        0

   City_C8  City_C9
0        0        0
1        0        0
2        0        0
3        0        1
4        0        0

[5 rows x 42 columns]
```

[89]:
```
Y_all.info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2381 entries, 0 to 2380
Series name: didDriverQuit
Non-Null Count  Dtype
--------------  -----
2381 non-null   int64
dtypes: int64(1)
memory usage: 18.7 KB
```

[90]:
```
Y_all.shape
```

[90]:
```
(2381,)
```

[91]:
```
Y_all.head()
```

[91]:
```
0    1
1    0
2    1
3    1
4    0
Name: didDriverQuit, dtype: int64
```

[92]:
```
X_all.shape,Y_all.shape
```

[92]:
```
((2381, 42), (2381,))
```

[93]:
```
#Taking a stratified sample for training and testing data
#now, even though we have aggregated the data with full, it was on a driver_ID␣
↪level.
#This ensures that there is no data leakage after the split.
#Also, the stratification ensures a similar split.

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(
    X_all, Y_all,
    test_size=0.1,
    stratify=Y_all
)
```

[94]:
```
x_train.shape,y_train.shape,x_test.shape,y_test.shape
```

[94]:
```
((2142, 42), (2142,), (239, 42), (239,))
```

[95]:
```
x_train.head()
```

[95]:
```
      Driver_ID  Reports_2019  Reports_2020       Age  Gender  \
1029       1210     -0.925134     -0.697099  0.616475     0.0
1206       1418      0.890065     -0.933705 -0.402355     0.0
1077       1267     -0.471334      1.905570 -0.620675     0.0
1566       1842     -0.925134     -0.460493 -0.911769     1.0
526         609      0.209365     -0.933705 -0.062745     1.0

      Education_Level    Income  Joining Designation     Grade  \
1029        -1.234575  0.014864             0.213676 -0.084348
1206         1.216049  0.298546            -0.975022 -1.158317
1077         1.216049 -0.833247            -0.975022 -1.158317
1566        -1.234575  0.339476             0.213676 -0.084348
526          1.216049 -0.461132            -0.975022 -1.158317
```

```
      Total Business Value  …  City_C27  City_C28  City_C29  City_C3  \
1029             -0.694331  …         0         0         0        0
1206             -0.251513  …         0         0         0        0
1077              0.488023  …         0         0         0        0
1566             -0.694331  …         0         0         0        0
526              -0.315989  …         0         0         0        0

      City_C4  City_C5  City_C6  City_C7  City_C8  City_C9
1029        0        0        0        0        0        0
1206        0        0        0        0        0        0
1077        0        0        0        0        0        0
1566        0        0        0        0        0        0
526         0        0        0        0        0        1

[5 rows x 42 columns]
```

[96]:
```
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2142 entries, 1029 to 40
Data columns (total 42 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Driver_ID             2142 non-null   int64
 1   Reports_2019          2142 non-null   float64
 2   Reports_2020          2142 non-null   float64
 3   Age                   2142 non-null   float64
 4   Gender                2142 non-null   float64
 5   Education_Level       2142 non-null   float64
 6   Income                2142 non-null   float64
 7   Joining Designation   2142 non-null   float64
 8   Grade                 2142 non-null   float64
 9   Total Business Value  2142 non-null   float64
 10  Quarterly Rating      2142 non-null   float64
 11  QRIncrease            2142 non-null   int64
 12  IncomeIncreased       2142 non-null   int64
 13  tenure                2142 non-null   float64
 14  City_C10              2142 non-null   int64
 15  City_C11              2142 non-null   int64
 16  City_C12              2142 non-null   int64
 17  City_C13              2142 non-null   int64
 18  City_C14              2142 non-null   int64
 19  City_C15              2142 non-null   int64
 20  City_C16              2142 non-null   int64
 21  City_C17              2142 non-null   int64
 22  City_C18              2142 non-null   int64
 23  City_C19              2142 non-null   int64
```

```
 24  City_C2               2142 non-null   int64
 25  City_C20              2142 non-null   int64
 26  City_C21              2142 non-null   int64
 27  City_C22              2142 non-null   int64
 28  City_C23              2142 non-null   int64
 29  City_C24              2142 non-null   int64
 30  City_C25              2142 non-null   int64
 31  City_C26              2142 non-null   int64
 32  City_C27              2142 non-null   int64
 33  City_C28              2142 non-null   int64
 34  City_C29              2142 non-null   int64
 35  City_C3               2142 non-null   int64
 36  City_C4               2142 non-null   int64
 37  City_C5               2142 non-null   int64
 38  City_C6               2142 non-null   int64
 39  City_C7               2142 non-null   int64
 40  City_C8               2142 non-null   int64
 41  City_C9               2142 non-null   int64
dtypes: float64(11), int64(31)
memory usage: 719.6 KB
```

[97]:
```
y_train.head()
```

[97]:
```
1029    0
1206    1
1077    0
1566    1
526     1
Name: didDriverQuit, dtype: int64
```

[98]:
```
y_train.info()
```

```
<class 'pandas.core.series.Series'>
Index: 2142 entries, 1029 to 40
Series name: didDriverQuit
Non-Null Count  Dtype
--------------  -----
2142 non-null   int64
dtypes: int64(1)
memory usage: 33.5 KB
```

[99]:
```
x_test.head()
```

[99]:
```
      Driver_ID  Reports_2019  Reports_2020       Age  Gender  \
2192       2570     -0.698234     -0.933705 -1.760794     0.0
2293       2690      1.797664      1.905570 -1.378733     0.0
912        1076     -0.698234      0.722539 -0.741964     0.0
1024       1205      1.797664      0.249326 -0.971700     0.0
```

```
778          915      0.890065     -0.933705  2.144719     0.0

      Education_Level    Income  Joining Designation     Grade  \
2192         1.216049 -1.147538            -0.975022 -1.158317
2293         1.216049  0.685710            -0.975022 -0.084348
912         -0.009263 -0.261395            -0.975022 -1.158317
1024        -0.009263  1.280652             0.213676 -0.084348
778         -1.234575 -1.439279            -0.975022 -1.158317

      Total Business Value  …  City_C27  City_C28  City_C29  City_C3  \
2192             -0.694331  …         0         0         0        0
2293              2.038039  …         0         0         0        0
912              -0.331328  …         0         0         0        0
1024              1.705952  …         0         0         0        0
778              -0.106102  …         0         0         0        0

      City_C4  City_C5  City_C6  City_C7  City_C8  City_C9
2192        0        0        0        0        0        1
2293        0        0        0        0        0        0
912         0        0        0        0        0        0
1024        0        0        0        0        0        0
778         0        0        0        0        0        0

[5 rows x 42 columns]
```

[100]:
```
x_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 239 entries, 2192 to 2314
Data columns (total 42 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Driver_ID             239 non-null    int64
 1   Reports_2019          239 non-null    float64
 2   Reports_2020          239 non-null    float64
 3   Age                   239 non-null    float64
 4   Gender                239 non-null    float64
 5   Education_Level       239 non-null    float64
 6   Income                239 non-null    float64
 7   Joining Designation   239 non-null    float64
 8   Grade                 239 non-null    float64
 9   Total Business Value  239 non-null    float64
 10  Quarterly Rating      239 non-null    float64
 11  QRIncrease            239 non-null    int64
 12  IncomeIncreased       239 non-null    int64
 13  tenure                239 non-null    float64
 14  City_C10              239 non-null    int64
```

```
 15  City_C11              239 non-null    int64
 16  City_C12              239 non-null    int64
 17  City_C13              239 non-null    int64
 18  City_C14              239 non-null    int64
 19  City_C15              239 non-null    int64
 20  City_C16              239 non-null    int64
 21  City_C17              239 non-null    int64
 22  City_C18              239 non-null    int64
 23  City_C19              239 non-null    int64
 24  City_C2               239 non-null    int64
 25  City_C20              239 non-null    int64
 26  City_C21              239 non-null    int64
 27  City_C22              239 non-null    int64
 28  City_C23              239 non-null    int64
 29  City_C24              239 non-null    int64
 30  City_C25              239 non-null    int64
 31  City_C26              239 non-null    int64
 32  City_C27              239 non-null    int64
 33  City_C28              239 non-null    int64
 34  City_C29              239 non-null    int64
 35  City_C3               239 non-null    int64
 36  City_C4               239 non-null    int64
 37  City_C5               239 non-null    int64
 38  City_C6               239 non-null    int64
 39  City_C7               239 non-null    int64
 40  City_C8               239 non-null    int64
 41  City_C9               239 non-null    int64
dtypes: float64(11), int64(31)
memory usage: 80.3 KB
```

[101]:
```
y_test.info()
```

```
<class 'pandas.core.series.Series'>
Index: 239 entries, 2192 to 2314
Series name: didDriverQuit
Non-Null Count  Dtype
--------------  -----
239 non-null    int64
dtypes: int64(1)
memory usage: 3.7 KB
```

[102]:
```
y_test.head()
```

[102]:
```
2192    1
2293    0
912     1
1024    1
```

```
778     1
Name: didDriverQuit, dtype: int64
```

[103]:
```
x_train.drop('Driver_ID', axis=1, inplace=True)
```

[104]:
```
x_train.head()
```

[104]:
```
      Reports_2019  Reports_2020       Age  Gender  Education_Level    Income  \
1029     -0.925134     -0.697099  0.616475     0.0        -1.234575  0.014864
1206      0.890065     -0.933705 -0.402355     0.0         1.216049  0.298546
1077     -0.471334      1.905570 -0.620675     0.0         1.216049 -0.833247
1566     -0.925134     -0.460493 -0.911769     1.0        -1.234575  0.339476
526       0.209365     -0.933705 -0.062745     1.0         1.216049 -0.461132

      Joining Designation     Grade  Total Business Value  Quarterly Rating  \
1029             0.213676 -0.084348             -0.694331         -0.642003
1206            -0.975022 -1.158317             -0.251513          0.591741
1077            -0.975022 -1.158317              0.488023          1.825485
1566             0.213676 -0.084348             -0.694331         -0.642003
526             -0.975022 -1.158317             -0.315989          0.591741

      …  City_C27  City_C28  City_C29  City_C3  City_C4  City_C5  City_C6  \
1029  …         0         0         0        0        0        0        0
1206  …         0         0         0        0        0        0        0
1077  …         0         0         0        0        0        0        0
1566  …         0         0         0        0        0        0        0
526   …         0         0         0        0        0        0        0

      City_C7  City_C8  City_C9
1029        0        0        0
1206        0        0        0
1077        0        0        0
1566        0        0        0
526         0        0        1

[5 rows x 41 columns]
```

[105]:
```
x_train.shape
```

[105]: (2142, 41)

[106]:
```
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2142 entries, 1029 to 40
Data columns (total 41 columns):
 #   Column                Non-Null Count  Dtype
```

```
---  ------                --------------  -----
 0   Reports_2019          2142 non-null   float64
 1   Reports_2020          2142 non-null   float64
 2   Age                   2142 non-null   float64
 3   Gender                2142 non-null   float64
 4   Education_Level       2142 non-null   float64
 5   Income                2142 non-null   float64
 6   Joining Designation   2142 non-null   float64
 7   Grade                 2142 non-null   float64
 8   Total Business Value  2142 non-null   float64
 9   Quarterly Rating      2142 non-null   float64
 10  QRIncrease            2142 non-null   int64
 11  IncomeIncreased       2142 non-null   int64
 12  tenure                2142 non-null   float64
 13  City_C10              2142 non-null   int64
 14  City_C11              2142 non-null   int64
 15  City_C12              2142 non-null   int64
 16  City_C13              2142 non-null   int64
 17  City_C14              2142 non-null   int64
 18  City_C15              2142 non-null   int64
 19  City_C16              2142 non-null   int64
 20  City_C17              2142 non-null   int64
 21  City_C18              2142 non-null   int64
 22  City_C19              2142 non-null   int64
 23  City_C2               2142 non-null   int64
 24  City_C20              2142 non-null   int64
 25  City_C21              2142 non-null   int64
 26  City_C22              2142 non-null   int64
 27  City_C23              2142 non-null   int64
 28  City_C24              2142 non-null   int64
 29  City_C25              2142 non-null   int64
 30  City_C26              2142 non-null   int64
 31  City_C27              2142 non-null   int64
 32  City_C28              2142 non-null   int64
 33  City_C29              2142 non-null   int64
 34  City_C3               2142 non-null   int64
 35  City_C4               2142 non-null   int64
 36  City_C5               2142 non-null   int64
 37  City_C6               2142 non-null   int64
 38  City_C7               2142 non-null   int64
 39  City_C8               2142 non-null   int64
 40  City_C9               2142 non-null   int64
dtypes: float64(11), int64(30)
memory usage: 702.8 KB
```

[107]:

```
#Checking for multicollinearity using VIF
```

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
import pandas as pd

# Function to check VIFs and drop features iteratively
def calculate_vif(X, thresh=5.0):
    while True:
        vif_data = pd.DataFrame()
        vif_data["Feature"] = X.columns
        vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in␣
↪range(X.shape[1])]

        print("\nCurrent VIFs:")
        print(vif_data)

        max_vif = vif_data['VIF'].max()
        if max_vif > thresh:
            drop_feature = vif_data.loc[vif_data['VIF'].idxmax(), 'Feature']
            print(f"\nDropping '{drop_feature}' due to high VIF ({max_vif:.
↪2f})\n")
            X = X.drop(columns=[drop_feature])
        else:
            break

    return X
```

[108]:
```python
# Applying VIF reduction declared above
X_train_vif_reduced = calculate_vif(x_train)
```

```
Current VIFs:
                Feature       VIF
0          Reports_2019  3.116397
1          Reports_2020  2.271841
2                   Age  1.175059
3                Gender  1.685918
4       Education_Level  1.068248
5                Income  2.493669
6   Joining Designation  4.517862
7                 Grade  6.500778
8   Total Business Value  3.615131
9      Quarterly Rating  2.698174
10           QRIncrease  3.218906
11       IncomeIncreased  1.263640
12                tenure  3.729469
13              City_C10  1.069675
14              City_C11  1.046639
15              City_C12  1.067197
```

```
16              City_C13  1.045566
17              City_C14  1.066276
18              City_C15  1.066324
19              City_C16  1.082613
20              City_C17  1.057973
21              City_C18  1.073273
22              City_C19  1.048624
23               City_C2  1.059420
24              City_C20  1.117155
25              City_C21  1.053422
26              City_C22  1.067729
27              City_C23  1.087982
28              City_C24  1.055902
29              City_C25  1.050126
30              City_C26  1.069674
31              City_C27  1.065472
32              City_C28  1.042512
33              City_C29  1.079905
34               City_C3  1.083730
35               City_C4  1.053397
36               City_C5  1.052278
37               City_C6  1.099921
38               City_C7  1.058187
39               City_C8  1.069520
40               City_C9  1.073338

Dropping 'Grade' due to high VIF (6.50)


Current VIFs:
                 Feature       VIF
0           Reports_2019  3.116322
1           Reports_2020  2.264691
2                    Age  1.172966
3                 Gender  1.684259
4        Education_Level  1.050138
5                 Income  1.987131
6    Joining Designation  1.840287
7   Total Business Value  3.182603
8       Quarterly Rating  2.483924
9             QRIncrease  3.218139
10        IncomeIncreased  1.258346
11                tenure  3.114008
12              City_C10  1.069629
13              City_C11  1.046484
14              City_C12  1.067184
15              City_C13  1.045182
16              City_C14  1.066199
```

```
17              City_C15  1.066324
18              City_C16  1.082612
19              City_C17  1.057723
20              City_C18  1.073213
21              City_C19  1.048569
22               City_C2  1.058910
23              City_C20  1.117095
24              City_C21  1.053266
25              City_C22  1.067436
26              City_C23  1.087675
27              City_C24  1.055487
28              City_C25  1.048890
29              City_C26  1.069642
30              City_C27  1.061923
31              City_C28  1.042362
32              City_C29  1.079827
33               City_C3  1.082978
34               City_C4  1.052621
35               City_C5  1.052211
36               City_C6  1.099919
37               City_C7  1.057549
38               City_C8  1.069489
39               City_C9  1.073219
```

[109]:
```
x_train=X_train_vif_reduced
```

[110]:
```
x_train.shape,y_train.shape,x_test.shape,y_test.shape
```

[110]:
```
((2142, 40), (2142,), (239, 42), (239,))
```

[111]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from collections import Counter


print("Original training set class distribution:")
print(y_train.value_counts())
print("\nPercentage distribution:")
print(y_train.value_counts(normalize=True) * 100)
```

```
Original training set class distribution:
didDriverQuit
1    1454
0     688
Name: count, dtype: int64

Percentage distribution:
didDriverQuit
```

```
1    67.880486
0    32.119514
Name: proportion, dtype: float64
```

[112]:
```
#We can see that the minority class is 32%, which is not that low to be␣
↪considered an issue
#However, let us increase the data and make the classes more balanced with the␣
↪minority having atleast 40% to 45%
#of the values
```

[113]:
```
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2142 entries, 1029 to 40
Data columns (total 40 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Reports_2019          2142 non-null   float64
 1   Reports_2020          2142 non-null   float64
 2   Age                   2142 non-null   float64
 3   Gender                2142 non-null   float64
 4   Education_Level       2142 non-null   float64
 5   Income                2142 non-null   float64
 6   Joining Designation   2142 non-null   float64
 7   Total Business Value  2142 non-null   float64
 8   Quarterly Rating      2142 non-null   float64
 9   QRIncrease            2142 non-null   int64
 10  IncomeIncreased       2142 non-null   int64
 11  tenure                2142 non-null   float64
 12  City_C10              2142 non-null   int64
 13  City_C11              2142 non-null   int64
 14  City_C12              2142 non-null   int64
 15  City_C13              2142 non-null   int64
 16  City_C14              2142 non-null   int64
 17  City_C15              2142 non-null   int64
 18  City_C16              2142 non-null   int64
 19  City_C17              2142 non-null   int64
 20  City_C18              2142 non-null   int64
 21  City_C19              2142 non-null   int64
 22  City_C2               2142 non-null   int64
 23  City_C20              2142 non-null   int64
 24  City_C21              2142 non-null   int64
 25  City_C22              2142 non-null   int64
 26  City_C23              2142 non-null   int64
 27  City_C24              2142 non-null   int64
 28  City_C25              2142 non-null   int64
 29  City_C26              2142 non-null   int64
 30  City_C27              2142 non-null   int64
```

```
 31  City_C28               2142 non-null   int64
 32  City_C29               2142 non-null   int64
 33  City_C3                2142 non-null   int64
 34  City_C4                2142 non-null   int64
 35  City_C5                2142 non-null   int64
 36  City_C6                2142 non-null   int64
 37  City_C7                2142 non-null   int64
 38  City_C8                2142 non-null   int64
 39  City_C9                2142 non-null   int64
dtypes: float64(10), int64(30)
memory usage: 686.1 KB
```

[114]:
```
y_train.info()
```

```
<class 'pandas.core.series.Series'>
Index: 2142 entries, 1029 to 40
Series name: didDriverQuit
Non-Null Count  Dtype
--------------  -----
2142 non-null   int64
dtypes: int64(1)
memory usage: 33.5 KB
```

[115]:
```
from imblearn.over_sampling import SMOTE

desired_ratio = 0.45 / 0.55
majority_class = y_train.value_counts().idxmax()
minority_class = y_train.value_counts().idxmin()
majority_count = y_train.value_counts()[majority_class]
minority_target = int(majority_count * desired_ratio)

smote = SMOTE(sampling_strategy={minority_class: minority_target},␣
 ↪random_state=42)
x_train_res, y_train_res = smote.fit_resample(x_train, y_train)
```

[116]:
```
x_train.shape,y_train.shape,x_train_res.shape,y_train_res.shape
```

[116]:
```
((2142, 40), (2142,), (2643, 40), (2643,))
```

[117]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from collections import Counter


print("Original training set class distribution:")
print(y_train_res.value_counts())
print("\nPercentage distribution:")
```

```python
print(y_train_res.value_counts(normalize=True) * 100)
```

```
Original training set class distribution:
didDriverQuit
1    1454
0    1189
Name: count, dtype: int64

Percentage distribution:
didDriverQuit
1    55.013243
0    44.986757
Name: proportion, dtype: float64
```

[118]:
```python
x_train,y_train = x_train_res,y_train_res
```

[119]:
```python
x_train.shape,y_train.shape,x_train_res.shape,y_train_res.shape
```

[119]: ((2643, 40), (2643,), (2643, 40), (2643,))

[120]:
```python
x_train.head()
```

[120]:

```
   Reports_2019  Reports_2020       Age  Gender  Education_Level    Income  \
0     -0.925134     -0.697099  0.616475     0.0        -1.234575  0.014864
1      0.890065     -0.933705 -0.402355     0.0         1.216049  0.298546
2     -0.471334      1.905570 -0.620675     0.0         1.216049 -0.833247
3     -0.925134     -0.460493 -0.911769     1.0        -1.234575  0.339476
4      0.209365     -0.933705 -0.062745     1.0         1.216049 -0.461132

   Joining Designation  Total Business Value  Quarterly Rating  QRIncrease  \
0             0.213676             -0.694331         -0.642003           0
1            -0.975022             -0.251513          0.591741           0
2            -0.975022              0.488023          1.825485           1
3             0.213676             -0.694331         -0.642003           0
4            -0.975022             -0.315989          0.591741           0

   …  City_C27  City_C28  City_C29  City_C3  City_C4  City_C5  City_C6  \
0  …         0         0         0        0        0        0        0
1  …         0         0         0        0        0        0        0
2  …         0         0         0        0        0        0        0
3  …         0         0         0        0        0        0        0
4  …         0         0         0        0        0        0        0

   City_C7  City_C8  City_C9
0        0        0        0
1        0        0        0
2        0        0        0
```

```
3        0        0        0
4        0        0        1

[5 rows x 40 columns]
```

[121]:
```
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2643 entries, 0 to 2642
Data columns (total 40 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Reports_2019          2643 non-null   float64
 1   Reports_2020          2643 non-null   float64
 2   Age                   2643 non-null   float64
 3   Gender                2643 non-null   float64
 4   Education_Level       2643 non-null   float64
 5   Income                2643 non-null   float64
 6   Joining Designation   2643 non-null   float64
 7   Total Business Value  2643 non-null   float64
 8   Quarterly Rating      2643 non-null   float64
 9   QRIncrease            2643 non-null   int64
 10  IncomeIncreased       2643 non-null   int64
 11  tenure                2643 non-null   float64
 12  City_C10              2643 non-null   int64
 13  City_C11              2643 non-null   int64
 14  City_C12              2643 non-null   int64
 15  City_C13              2643 non-null   int64
 16  City_C14              2643 non-null   int64
 17  City_C15              2643 non-null   int64
 18  City_C16              2643 non-null   int64
 19  City_C17              2643 non-null   int64
 20  City_C18              2643 non-null   int64
 21  City_C19              2643 non-null   int64
 22  City_C2               2643 non-null   int64
 23  City_C20              2643 non-null   int64
 24  City_C21              2643 non-null   int64
 25  City_C22              2643 non-null   int64
 26  City_C23              2643 non-null   int64
 27  City_C24              2643 non-null   int64
 28  City_C25              2643 non-null   int64
 29  City_C26              2643 non-null   int64
 30  City_C27              2643 non-null   int64
 31  City_C28              2643 non-null   int64
 32  City_C29              2643 non-null   int64
 33  City_C3               2643 non-null   int64
 34  City_C4               2643 non-null   int64
```

```
 35   City_C5                2643 non-null   int64
 36   City_C6                2643 non-null   int64
 37   City_C7                2643 non-null   int64
 38   City_C8                2643 non-null   int64
 39   City_C9                2643 non-null   int64
dtypes: float64(10), int64(30)
memory usage: 826.1 KB
```

[122]:
```
y_train.info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2643 entries, 0 to 2642
Series name: didDriverQuit
Non-Null Count  Dtype
--------------  -----
2643 non-null   int64
dtypes: int64(1)
memory usage: 20.8 KB
```

[123]:
```
y_train.head(50)
```

[123]:
```
0     0
1     1
2     0
3     1
4     1
5     1
6     1
7     1
8     1
9     0
10    1
11    1
12    0
13    1
14    0
15    1
16    1
17    1
18    0
19    0
20    1
21    0
22    1
23    1
24    1
25    1
```

```
26    1
27    1
28    1
29    1
30    1
31    1
32    1
33    1
34    1
35    1
36    0
37    1
38    1
39    1
40    1
41    1
42    1
43    1
44    0
45    1
46    0
47    0
48    1
49    0
Name: didDriverQuit, dtype: int64
```

[ ]:
```
#Model building.
#I am using RandomForest for the bagging method.
```

[124]:
```
def report(results, n_top=3):
    for i in range(1, n_top + 1):
        candidates = np.flatnonzero(results['rank_test_score'] == i)
        for candidate in candidates:
            print("Model with rank: {0}".format(i))
            print("Mean validation score: {0:.5f} (std: {1:.5f})".format(
                  results['mean_test_score'][candidate],
                  results['std_test_score'][candidate]))
            print("Parameters: {0}".format(results['params'][candidate]))
            print("")
```

[125]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
```

[126]:
```
param_dist = {
    "n_estimators": [100, 200, 300, 500, 700, 1000],
    "max_features": [5, 10, 20, 25, 30, 35],
    "bootstrap": [True, False],
```

```
    "class_weight": [None, 'balanced'],
    "criterion": ['entropy', 'gini'],
    "max_depth": [None, 5, 10, 15, 20, 30, 50, 70],
    "min_samples_leaf": [1, 2, 5, 10, 15, 20],
    "min_samples_split": [2, 5, 10, 15, 20]
}

rf = RandomForestClassifier(random_state=42)
random_search = RandomizedSearchCV(rf, param_distributions=param_dist,␣
 ↪n_iter=10, cv=7, scoring='f1', verbose=1, n_jobs=-1)
random_search.fit(x_train, y_train)
```

```
Fitting 7 folds for each of 10 candidates, totalling 70 fits
```

[126]:
```
RandomizedSearchCV(cv=7, estimator=RandomForestClassifier(random_state=42),
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'class_weight': [None, 'balanced'],
                                        'criterion': ['entropy', 'gini'],
                                        'max_depth': [None, 5, 10, 15, 20, 30,
                                                      50, 70],
                                        'max_features': [5, 10, 20, 25, 30, 35],
                                        'min_samples_leaf': [1, 2, 5, 10, 15,
                                                             20],
                                        'min_samples_split': [2, 5, 10, 15, 20],
                                        'n_estimators': [100, 200, 300, 500,
                                                         700, 1000]},
                   scoring='f1', verbose=1)
```

[127]:
```
report(random_search.cv_results_, 5)
```

```
Model with rank: 1
Mean validation score: 0.93341 (std: 0.01313)
Parameters: {'n_estimators': 700, 'min_samples_split': 5, 'min_samples_leaf': 1,
'max_features': 30, 'max_depth': 50, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 2
Mean validation score: 0.93180 (std: 0.01031)
Parameters: {'n_estimators': 200, 'min_samples_split': 10, 'min_samples_leaf':
1, 'max_features': 30, 'max_depth': 50, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 3
Mean validation score: 0.92793 (std: 0.01158)
Parameters: {'n_estimators': 700, 'min_samples_split': 15, 'min_samples_leaf':
5, 'max_features': 30, 'max_depth': 20, 'criterion': 'entropy', 'class_weight':
```

```
'balanced', 'bootstrap': True}

Model with rank: 4
Mean validation score: 0.92600 (std: 0.01469)
Parameters: {'n_estimators': 1000, 'min_samples_split': 20, 'min_samples_leaf':
5, 'max_features': 30, 'max_depth': 10, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 5
Mean validation score: 0.92501 (std: 0.01388)
Parameters: {'n_estimators': 500, 'min_samples_split': 15, 'min_samples_leaf':
5, 'max_features': 20, 'max_depth': 15, 'criterion': 'entropy', 'class_weight':
None, 'bootstrap': True}
```

[128]:

```
param_dist = {
    "n_estimators": [150,200,250,650,700,750],
    "max_features": [28,30,32],
    "bootstrap": [False],
    "class_weight": ['balanced'],
    "criterion": ['gini'],
    "max_depth": [48,50,52],
    "min_samples_leaf": [1, 2],
    "min_samples_split": [4,5,6,9,10,11]
}

rf = RandomForestClassifier(random_state=42)
random_search = RandomizedSearchCV(rf, param_distributions=param_dist,␣
 ↪n_iter=10, cv=7, scoring='f1', verbose=1, n_jobs=-1)
random_search.fit(x_train, y_train)
```

```
Fitting 7 folds for each of 10 candidates, totalling 70 fits
```

[128]:

```
RandomizedSearchCV(cv=7, estimator=RandomForestClassifier(random_state=42),
                   n_jobs=-1,
                   param_distributions={'bootstrap': [False],
                                        'class_weight': ['balanced'],
                                        'criterion': ['gini'],
                                        'max_depth': [48, 50, 52],
                                        'max_features': [28, 30, 32],
                                        'min_samples_leaf': [1, 2],
                                        'min_samples_split': [4, 5, 6, 9, 10,
                                                              11],
                                        'n_estimators': [150, 200, 250, 650,
                                                         700, 750]},
                   scoring='f1', verbose=1)
```

[129]:
```
report(random_search.cv_results_, 5)
```

```
Model with rank: 1
Mean validation score: 0.93462 (std: 0.01150)
Parameters: {'n_estimators': 650, 'min_samples_split': 10, 'min_samples_leaf':
2, 'max_features': 28, 'max_depth': 50, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 2
Mean validation score: 0.93379 (std: 0.01284)
Parameters: {'n_estimators': 250, 'min_samples_split': 5, 'min_samples_leaf': 1,
'max_features': 28, 'max_depth': 50, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 3
Mean validation score: 0.93374 (std: 0.01253)
Parameters: {'n_estimators': 700, 'min_samples_split': 4, 'min_samples_leaf': 1,
'max_features': 30, 'max_depth': 52, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 4
Mean validation score: 0.93282 (std: 0.01271)
Parameters: {'n_estimators': 650, 'min_samples_split': 4, 'min_samples_leaf': 2,
'max_features': 30, 'max_depth': 50, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 5
Mean validation score: 0.93256 (std: 0.01023)
Parameters: {'n_estimators': 750, 'min_samples_split': 10, 'min_samples_leaf':
2, 'max_features': 32, 'max_depth': 52, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}
```

[130]:
```
param_dist = {
    "n_estimators": [640,650,660,240,250,260],
    "max_features": [25,26,28,30],
    "bootstrap": [False],
    "class_weight": ['balanced'],
    "criterion": ['gini'],
    "max_depth": [48,50,52],
    "min_samples_leaf": [1, 2, 3],
    "min_samples_split": [4,5,6,9,10,11]
}

rf = RandomForestClassifier(random_state=42)
random_search = RandomizedSearchCV(rf, param_distributions=param_dist,␣
 ↪n_iter=10, cv=7, scoring='f1', verbose=1, n_jobs=-1)
```

```
random_search.fit(x_train, y_train)
```

```
Fitting 7 folds for each of 10 candidates, totalling 70 fits
```

[130]:
```
RandomizedSearchCV(cv=7, estimator=RandomForestClassifier(random_state=42),
                   n_jobs=-1,
                   param_distributions={'bootstrap': [False],
                                        'class_weight': ['balanced'],
                                        'criterion': ['gini'],
                                        'max_depth': [48, 50, 52],
                                        'max_features': [25, 26, 28, 30],
                                        'min_samples_leaf': [1, 2, 3],
                                        'min_samples_split': [4, 5, 6, 9, 10,
                                                              11],
                                        'n_estimators': [640, 650, 660, 240,
                                                         250, 260]},
                   scoring='f1', verbose=1)
```

[131]:
```
report(random_search.cv_results_, 5)
```

```
Model with rank: 1
Mean validation score: 0.93549 (std: 0.01174)
Parameters: {'n_estimators': 250, 'min_samples_split': 4, 'min_samples_leaf': 1,
'max_features': 28, 'max_depth': 50, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 2
Mean validation score: 0.93445 (std: 0.01272)
Parameters: {'n_estimators': 250, 'min_samples_split': 4, 'min_samples_leaf': 3,
'max_features': 28, 'max_depth': 52, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 3
Mean validation score: 0.93439 (std: 0.01264)
Parameters: {'n_estimators': 260, 'min_samples_split': 6, 'min_samples_leaf': 3,
'max_features': 25, 'max_depth': 50, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 4
Mean validation score: 0.93433 (std: 0.01237)
Parameters: {'n_estimators': 660, 'min_samples_split': 10, 'min_samples_leaf':
3, 'max_features': 28, 'max_depth': 50, 'criterion': 'gini', 'class_weight':
'balanced', 'bootstrap': False}

Model with rank: 5
Mean validation score: 0.93406 (std: 0.01114)
Parameters: {'n_estimators': 650, 'min_samples_split': 5, 'min_samples_leaf': 2,
'max_features': 26, 'max_depth': 50, 'criterion': 'gini', 'class_weight':
```

```
'balanced', 'bootstrap': False}
```

[ ]:
```
#After a few runs, I am selecting the below

# Model with rank: 1
# Mean validation score: 0.93549 (std: 0.01174)
# Parameters: {'n_estimators': 250, 'min_samples_split': 4, 'min_samples_leaf':
↪1, 'max_features': 28, 'max_depth': 50, 'criterion': 'gini', 'class_weight':
↪'balanced', 'bootstrap': False}
```

[132]:
```
rf_best_model=RandomForestClassifier(**{'n_estimators': 250,
↪'min_samples_split': 4, 'min_samples_leaf': 1, 'max_features': 28,
↪'max_depth': 50, 'criterion': 'gini', 'class_weight': 'balanced',
↪'bootstrap': False})
```

[133]:
```
rf_best_model.fit(x_train,y_train)
```

[133]:
```
RandomForestClassifier(bootstrap=False, class_weight='balanced', max_depth=50,
                       max_features=28, min_samples_split=4, n_estimators=250)
```

[134]:
```
#Feature importance
feat_imp_df=pd.DataFrame({'features':x_train.columns,
                          'importance':rf_best_model.feature_importances_})

feat_imp_df=feat_imp_df.sort_values('importance',ascending=False)
feat_imp_df['normalised_imp']=feat_imp_df['importance']/np.
↪sum(feat_imp_df['importance'])
feat_imp_df['cum_imp']=np.cumsum(feat_imp_df['normalised_imp'])
```

[135]:
```
feat_imp_df
```

[135]:

	features	importance	normalised_imp	cum_imp
1	Reports_2020	0.459513	0.459513	0.459513
0	Reports_2019	0.168336	0.168336	0.627850
11	tenure	0.130033	0.130033	0.757883
7	Total Business Value	0.057008	0.057008	0.814891
5	Income	0.028960	0.028960	0.843851
8	Quarterly Rating	0.028202	0.028202	0.872053
9	QRIncrease	0.026720	0.026720	0.898773
2	Age	0.026529	0.026529	0.925302
6	Joining Designation	0.010085	0.010085	0.935387
4	Education_Level	0.009459	0.009459	0.944846
22	City_C2	0.004910	0.004910	0.949756
39	City_C9	0.003785	0.003785	0.953541
23	City_C20	0.003595	0.003595	0.957136
3	Gender	0.003229	0.003229	0.960365

31	City_C28	0.003160	0.003160	0.963525
28	City_C25	0.003097	0.003097	0.966622
30	City_C27	0.002694	0.002694	0.969316
35	City_C5	0.002530	0.002530	0.971846
32	City_C29	0.002388	0.002388	0.974234
16	City_C14	0.002387	0.002387	0.976620
29	City_C26	0.002252	0.002252	0.978873
12	City_C10	0.002116	0.002116	0.980989
33	City_C3	0.001939	0.001939	0.982928
20	City_C18	0.001537	0.001537	0.984465
24	City_C21	0.001393	0.001393	0.985857
14	City_C12	0.001340	0.001340	0.987197
36	City_C6	0.001328	0.001328	0.988526
27	City_C24	0.001266	0.001266	0.989792
26	City_C23	0.001241	0.001241	0.991033
18	City_C16	0.001206	0.001206	0.992239
34	City_C4	0.001171	0.001171	0.993410
15	City_C13	0.001159	0.001159	0.994569
25	City_C22	0.001133	0.001133	0.995702
19	City_C17	0.001048	0.001048	0.996749
38	City_C8	0.000944	0.000944	0.997693
17	City_C15	0.000843	0.000843	0.998537
21	City_C19	0.000800	0.000800	0.999336
13	City_C11	0.000487	0.000487	0.999824
37	City_C7	0.000128	0.000128	0.999952
10	IncomeIncreased	0.000048	0.000048	1.000000

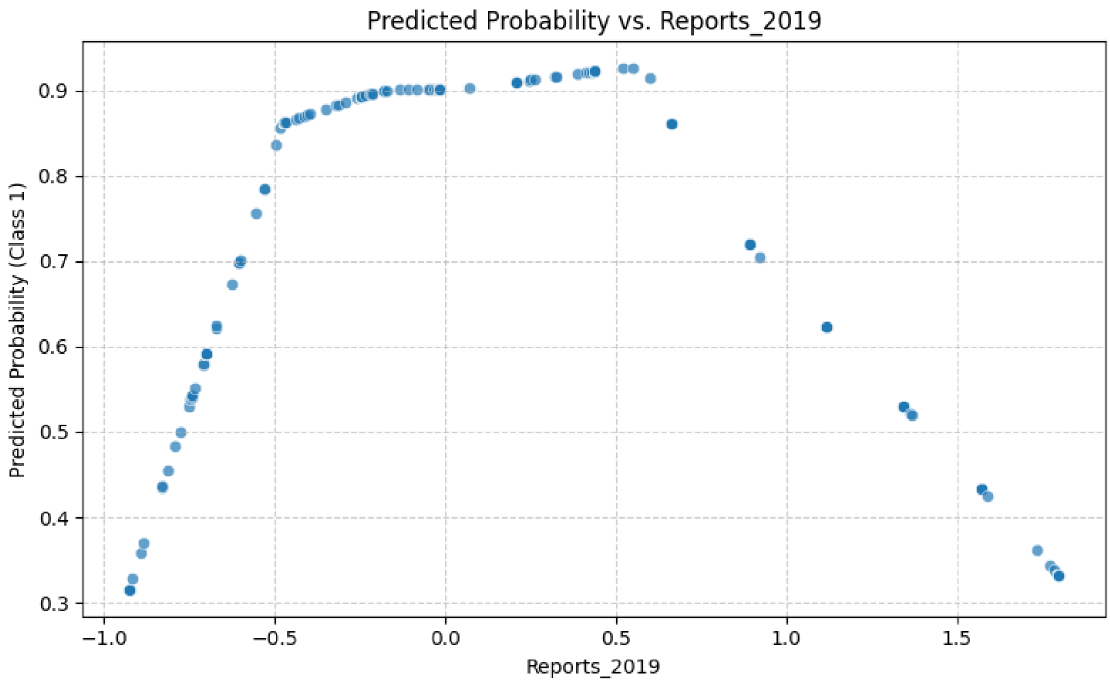
[ ]:
```
# From the above feature importance we see that Reports_2020(number of times␣
↪the driver reported in 2020) is the highest decider
# with 45% deciding rate.
# This is followed by Reports_2019(number of times the driver reported in 2019)␣
↪with 16%.
# This shows that the number of reporting dates are the most important feature
# The next feature is tenure which has about 13% importance.
# After the above features we have the remaining features with less than 10%␣
↪importance.
```

[ ]:
```
#Now using the below plots to check which feature has positive and which has␣
↪negative relation
```

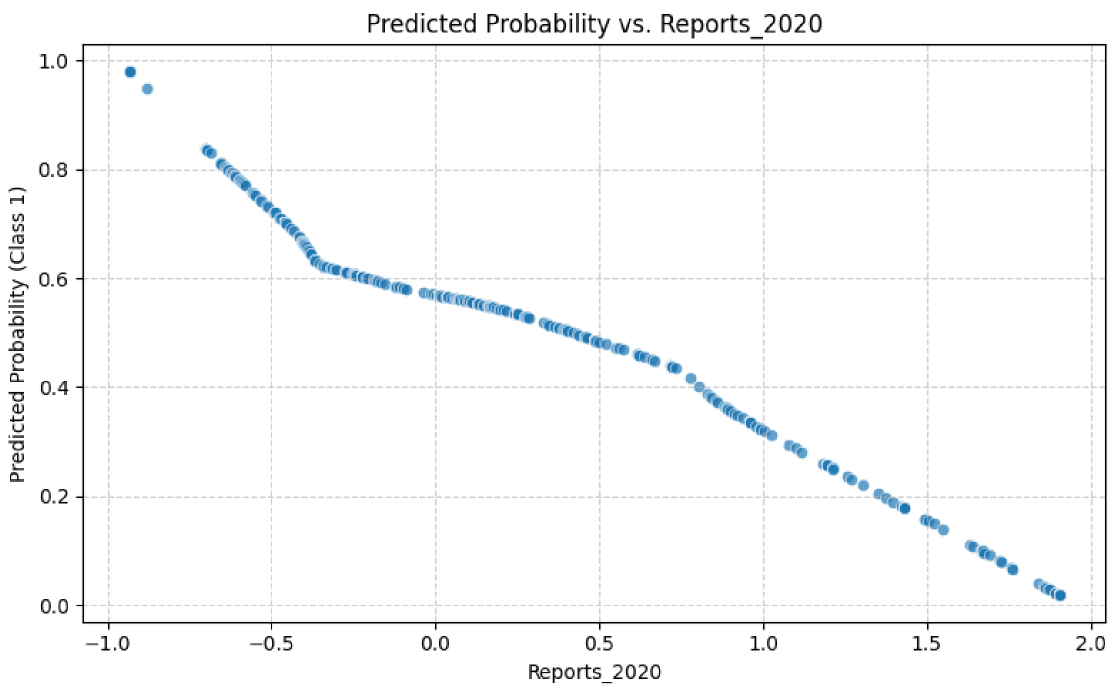
[136]:
```
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
import matplotlib.pyplot as plt

preds = rf_best_model.predict_proba(x_train)[:, 1]
```

```python
feature_names = x_train.columns

for var_name in feature_names:
    print(f"Plotting for feature: {var_name}")

    var_data = pd.DataFrame({'var': x_train[var_name], 'response': preds})

    try:
        smooth_data = sm.nonparametric.lowess(var_data['response'],␣
↪var_data['var'], frac=0.6) # frac can be adjusted
    except ValueError as e:
        print(f"Could not apply LOESS smoothing for {var_name}: {e}. Skipping␣
↪plot for this feature.")
        continue

    df_smoothed = pd.DataFrame({'response': smooth_data[:, 1], var_name:␣
↪smooth_data[:, 0]})

    plt.figure(figsize=(8, 5))
    sns.scatterplot(x=var_name, y='response', data=df_smoothed, alpha=0.7)

    plt.title(f'Predicted Probability vs. {var_name}')
    plt.xlabel(var_name)
    plt.ylabel('Predicted Probability (Class 1)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()
    print("-" * 50)
```

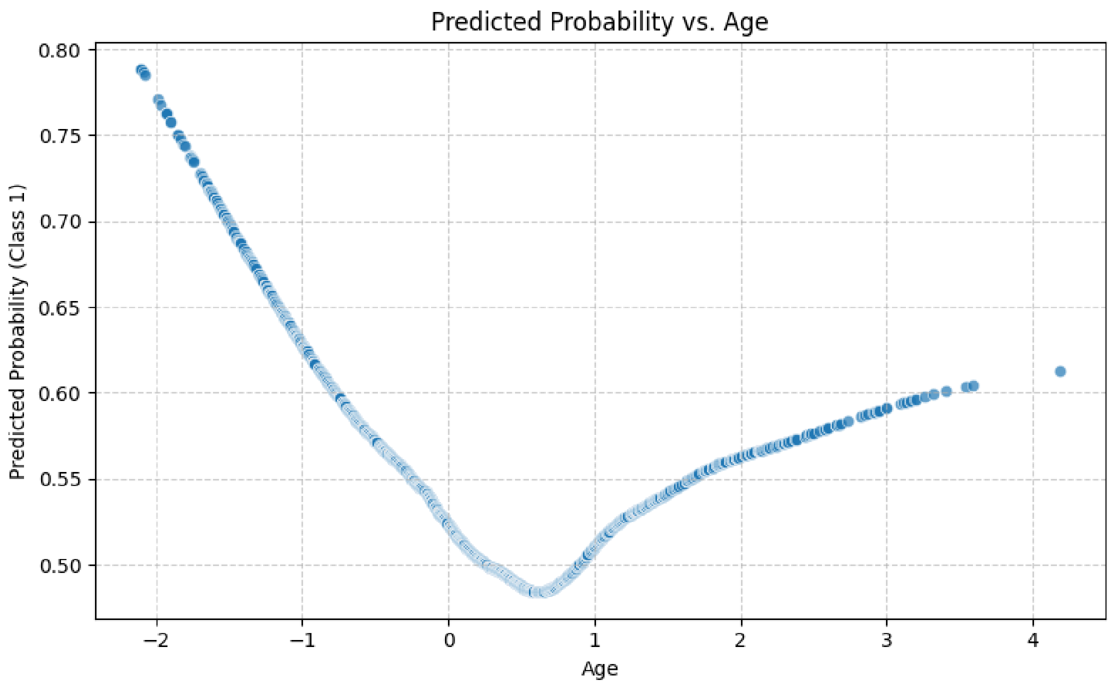
```
Plotting for feature: Reports_2019
```



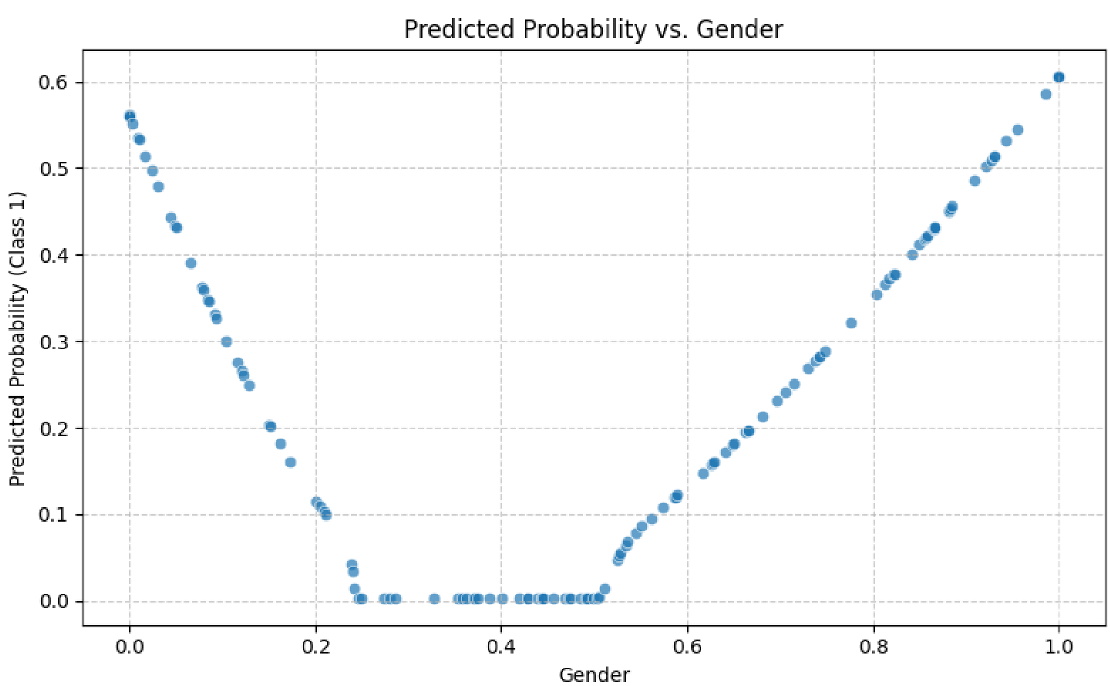

```
--------------------------------------------------
Plotting for feature: Reports_2020
```



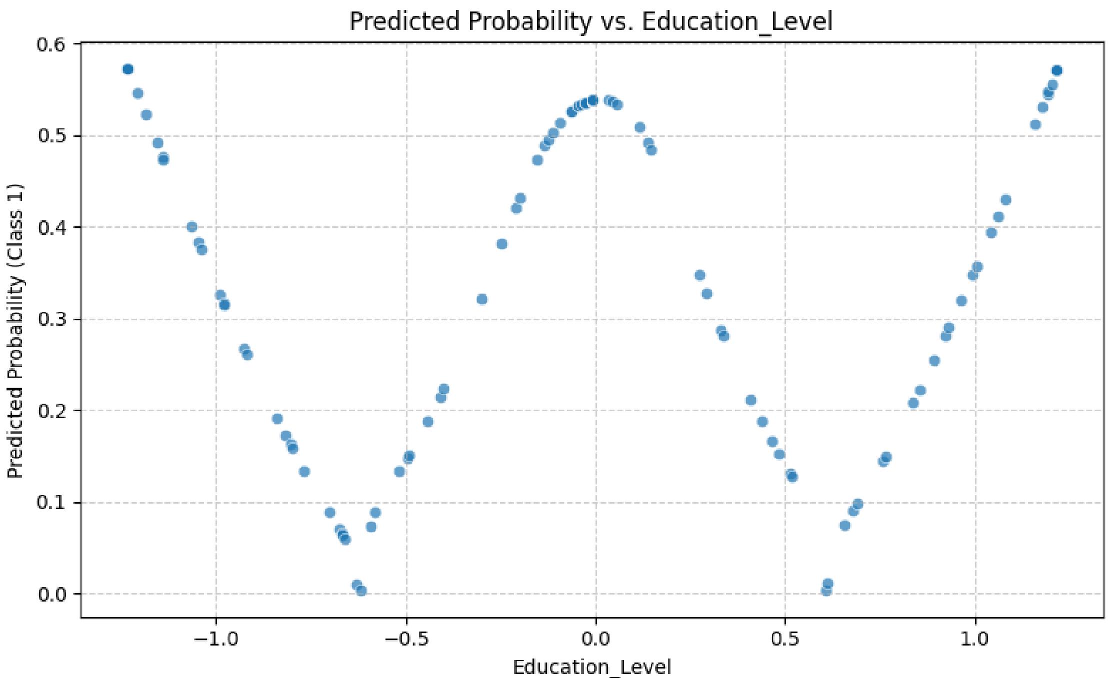
```
--------------------------------------------------
Plotting for feature: Age
```



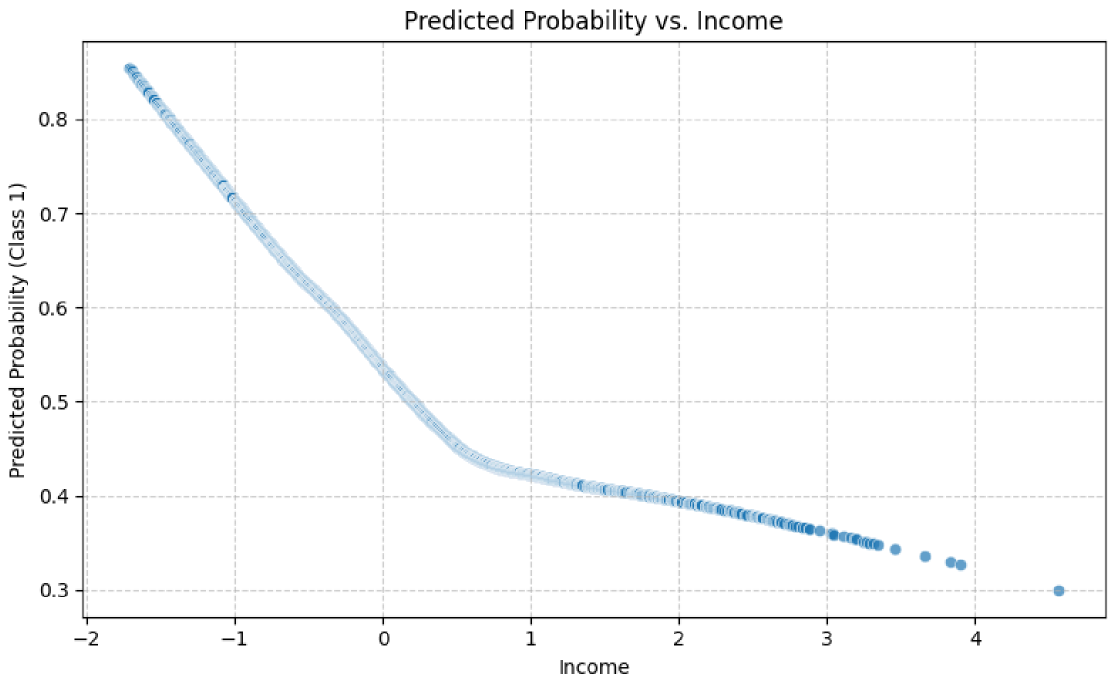

```
--------------------------------------------------
Plotting for feature: Gender
```



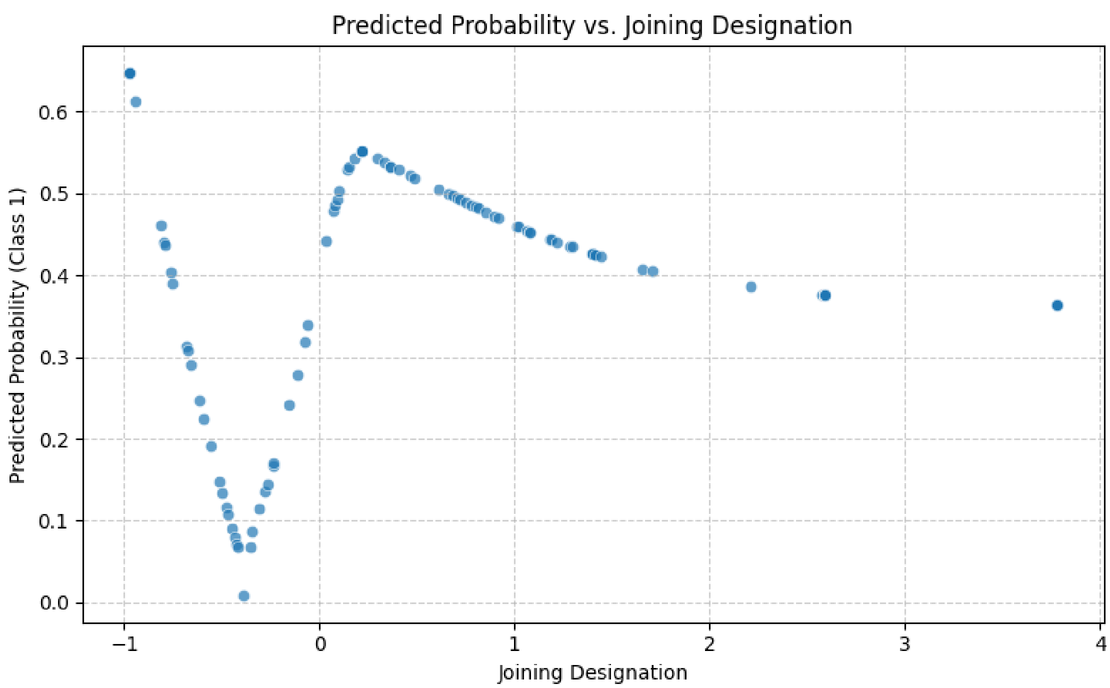
Plotting for feature: Education_Level



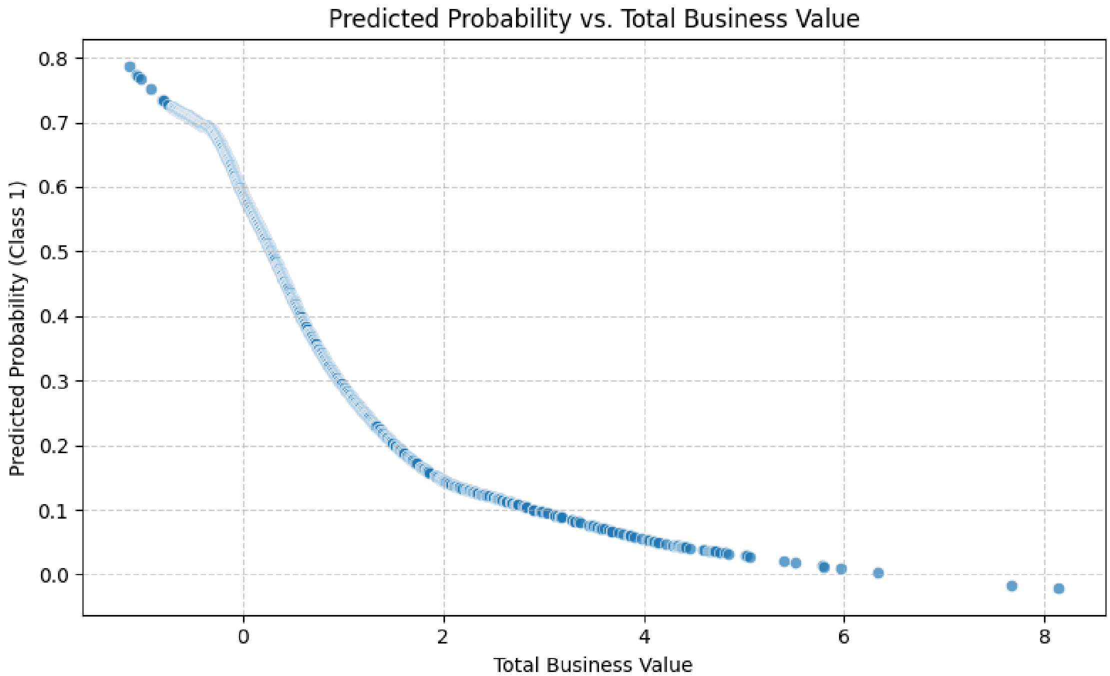
Plotting for feature: Income




```
--------------------------------------------------
Plotting for feature: Joining Designation
```

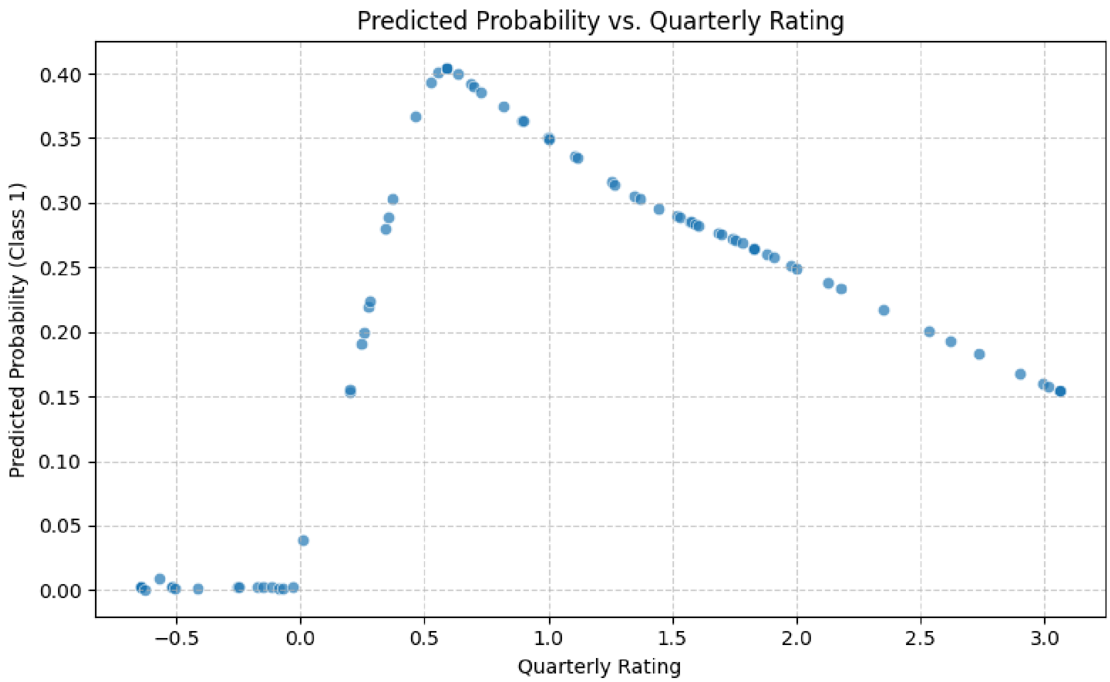


```
--------------------------------------------------
Plotting for feature: Total Business Value
```

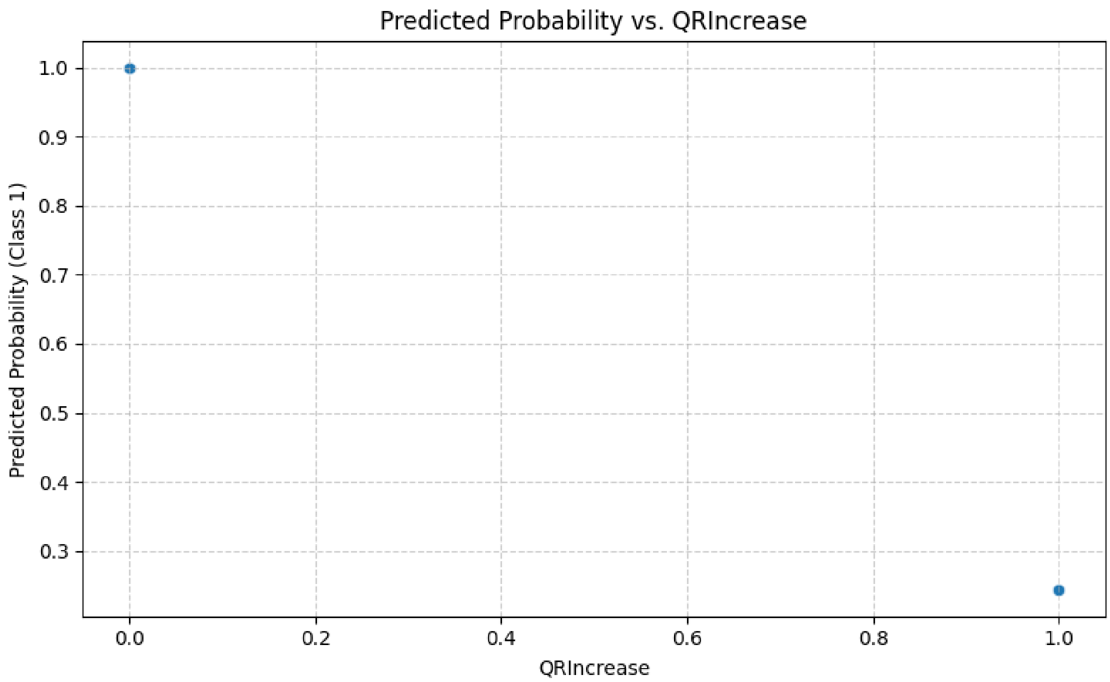



```
--------------------------------------------------
Plotting for feature: Quarterly Rating
```

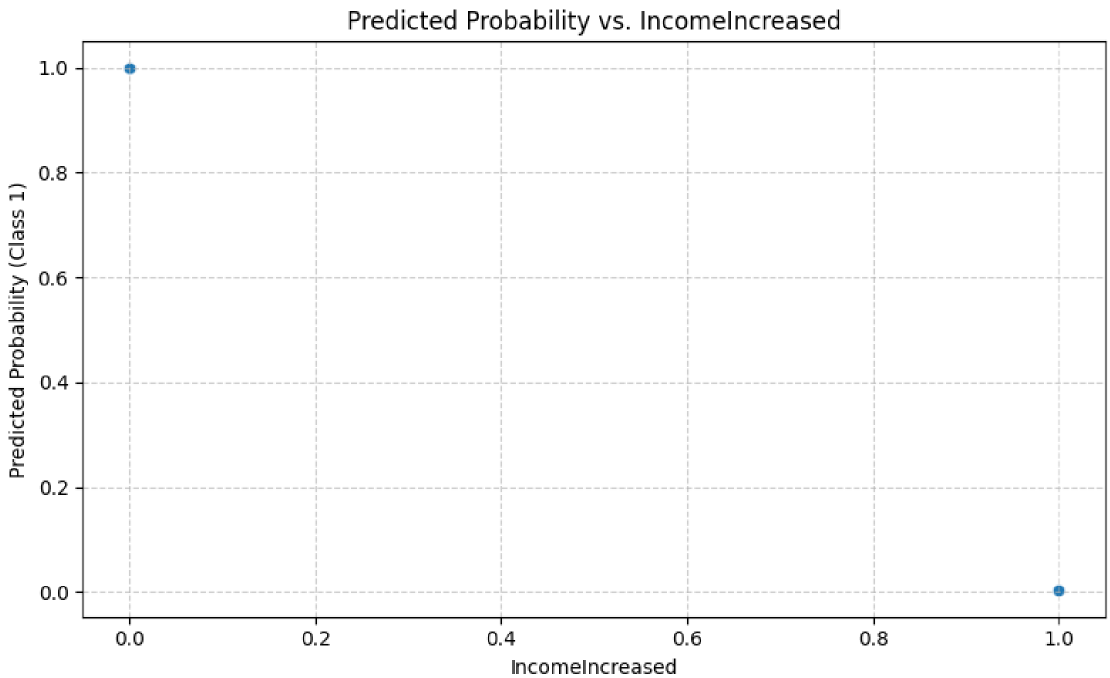
```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



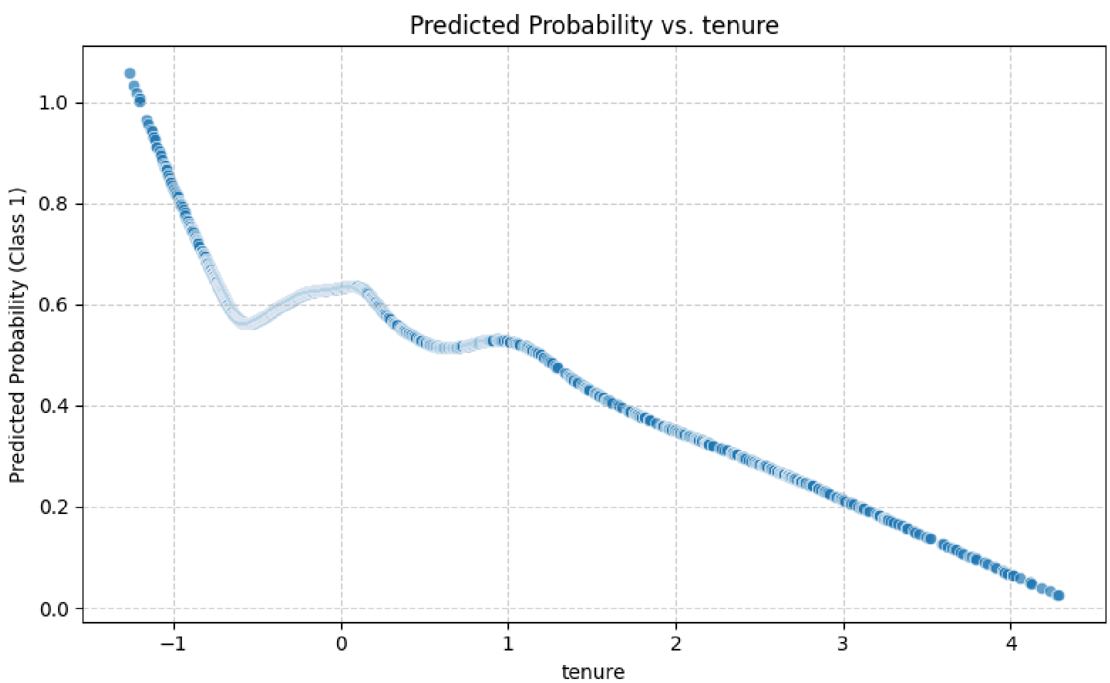

```
--------------------------------------------------
Plotting for feature: QRIncrease

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```




```
--------------------------------------------------
Plotting for feature: IncomeIncreased

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

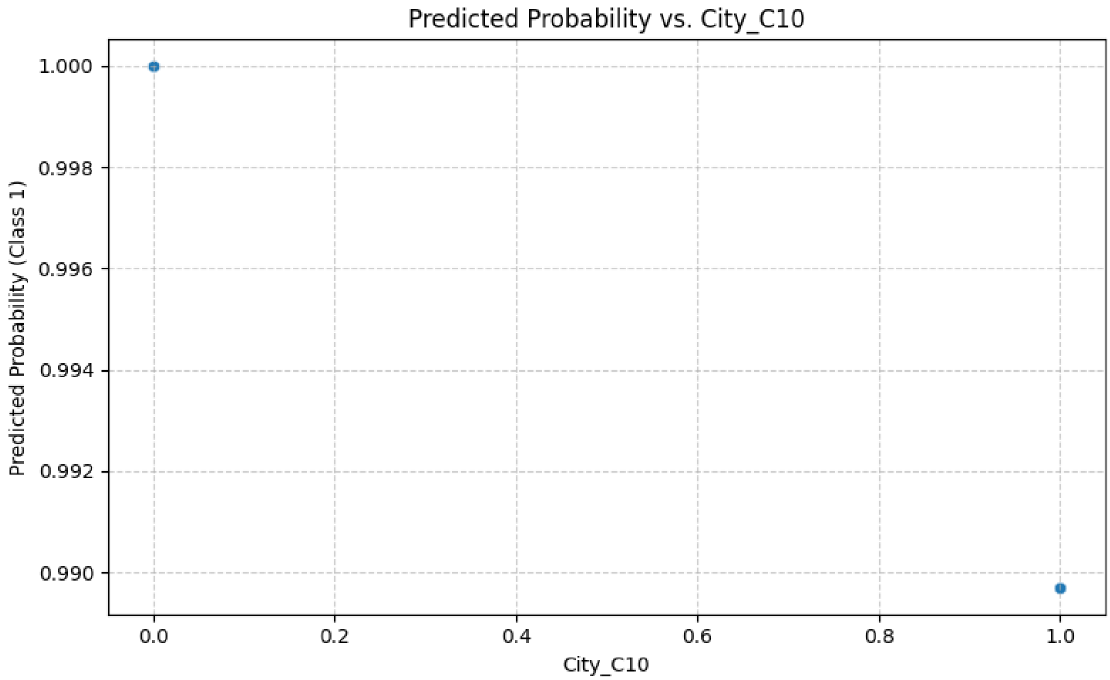



```
--------------------------------------------------
Plotting for feature: tenure
```



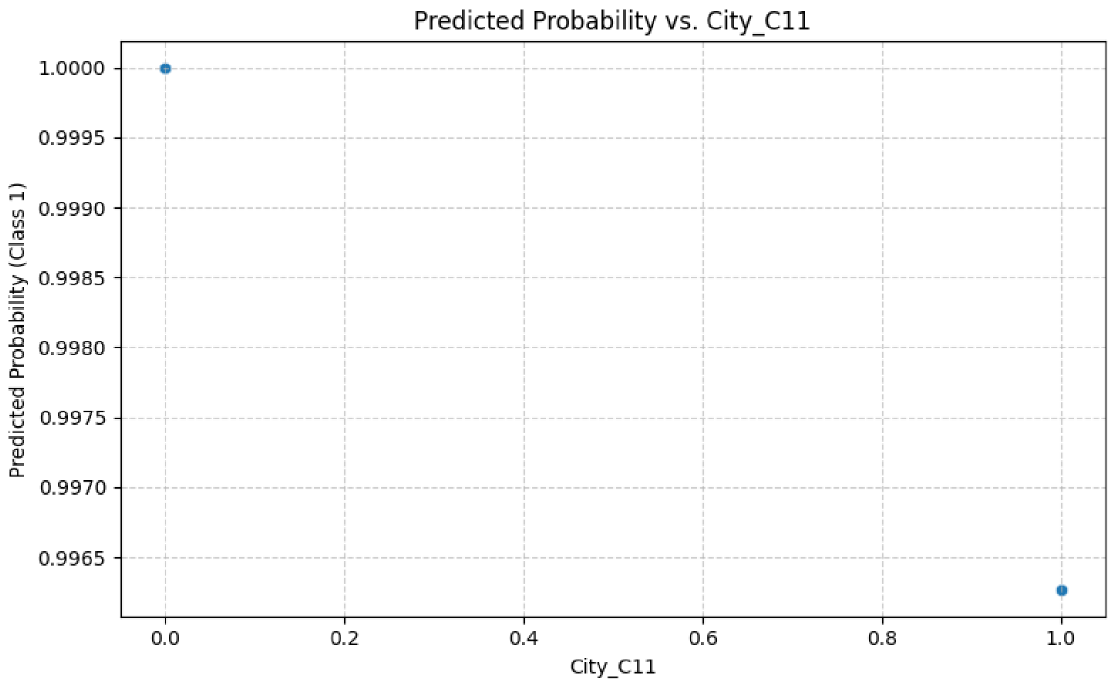
```
--------------------------------------------------
Plotting for feature: City_C10
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

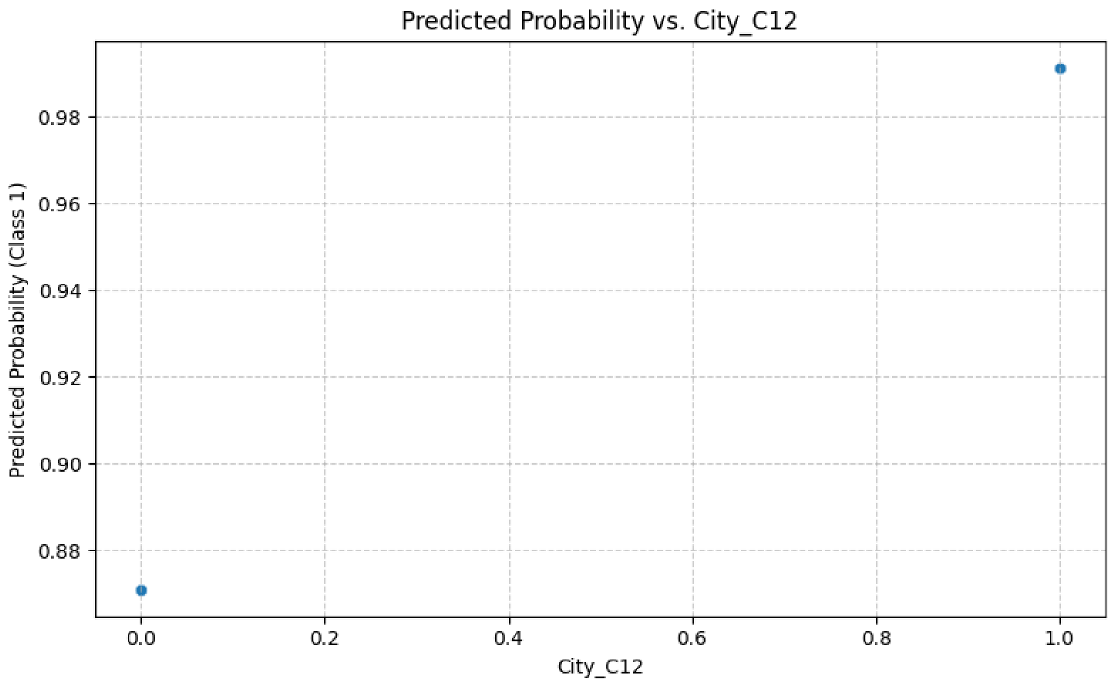



```
--------------------------------------------------
Plotting for feature: City_C11
```

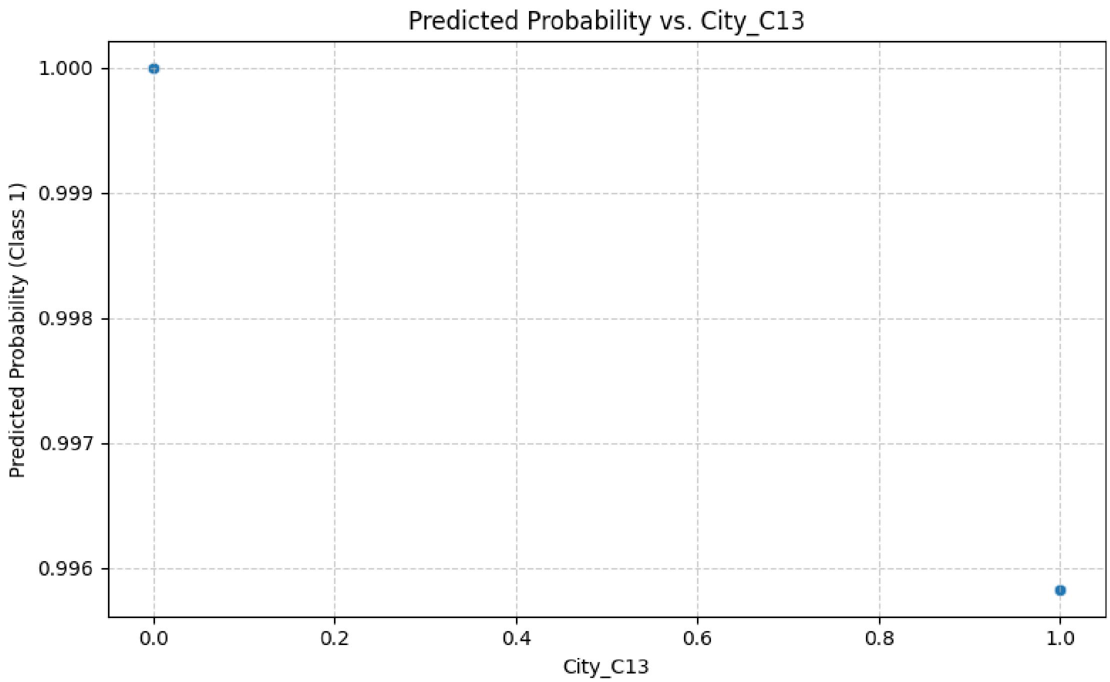
```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```




```
--------------------------------------------------
Plotting for feature: City_C12

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

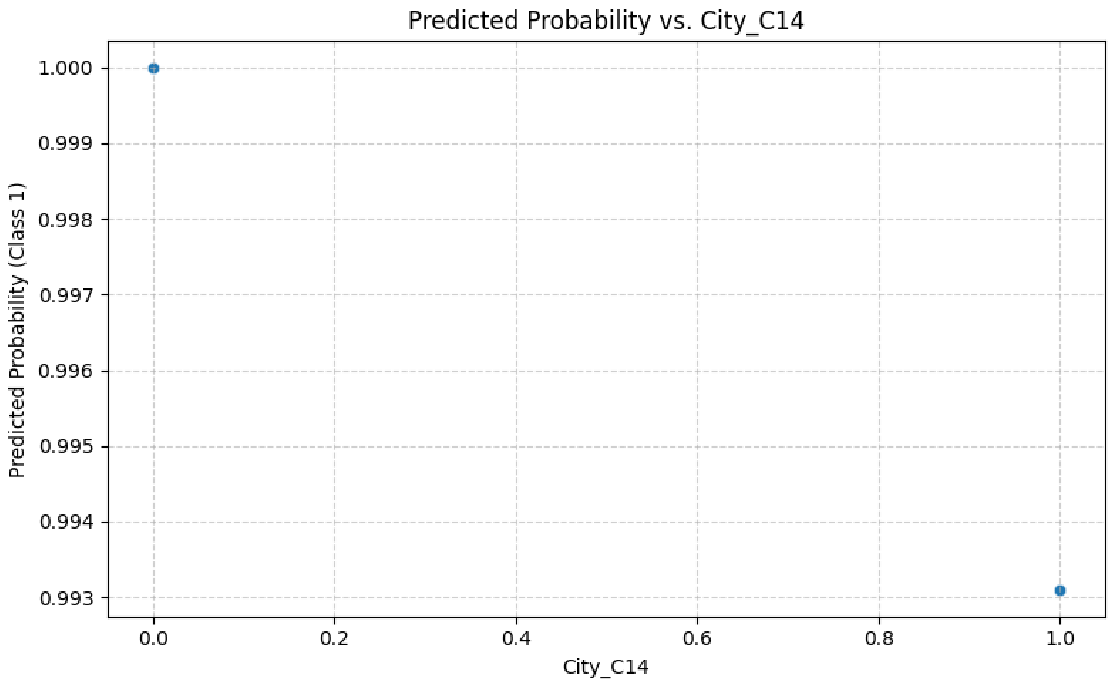



```
--------------------------------------------------
Plotting for feature: City_C13

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

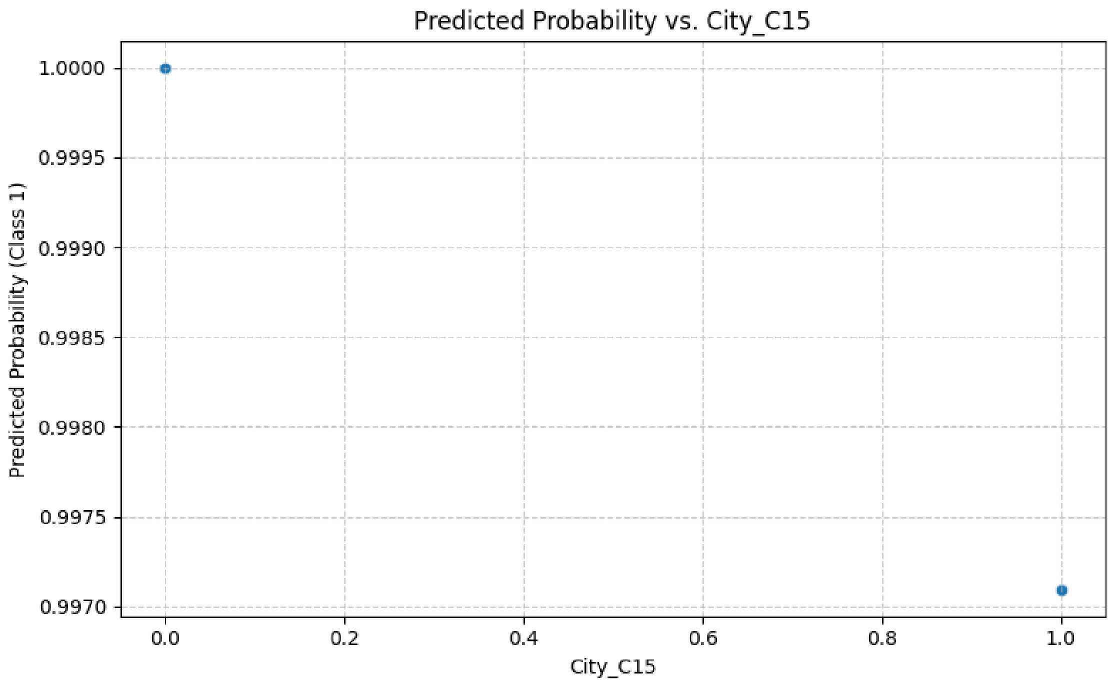



```
--------------------------------------------------
Plotting for feature: City_C14
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

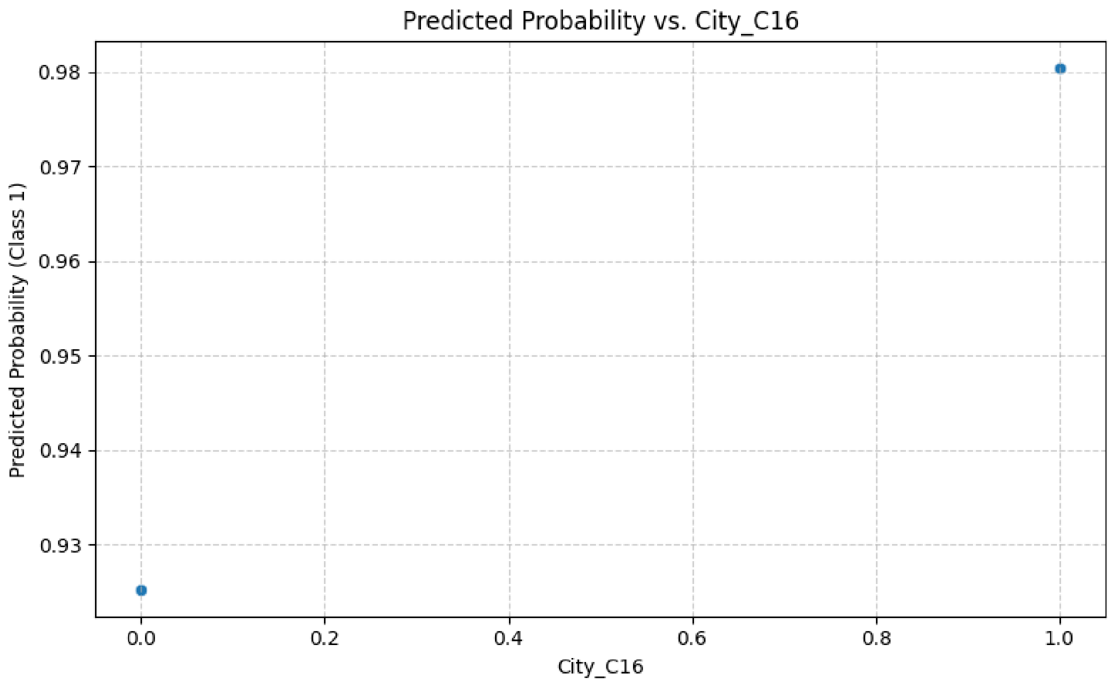



```
--------------------------------------------------
Plotting for feature: City_C15

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



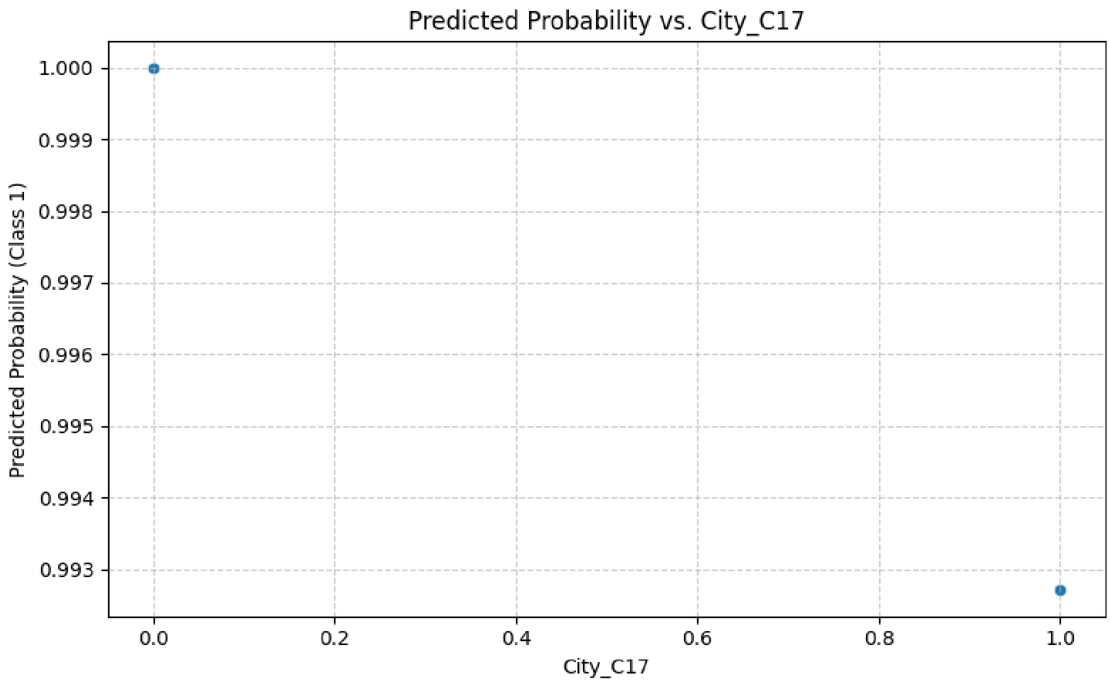

```
--------------------------------------------------
Plotting for feature: City_C16
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

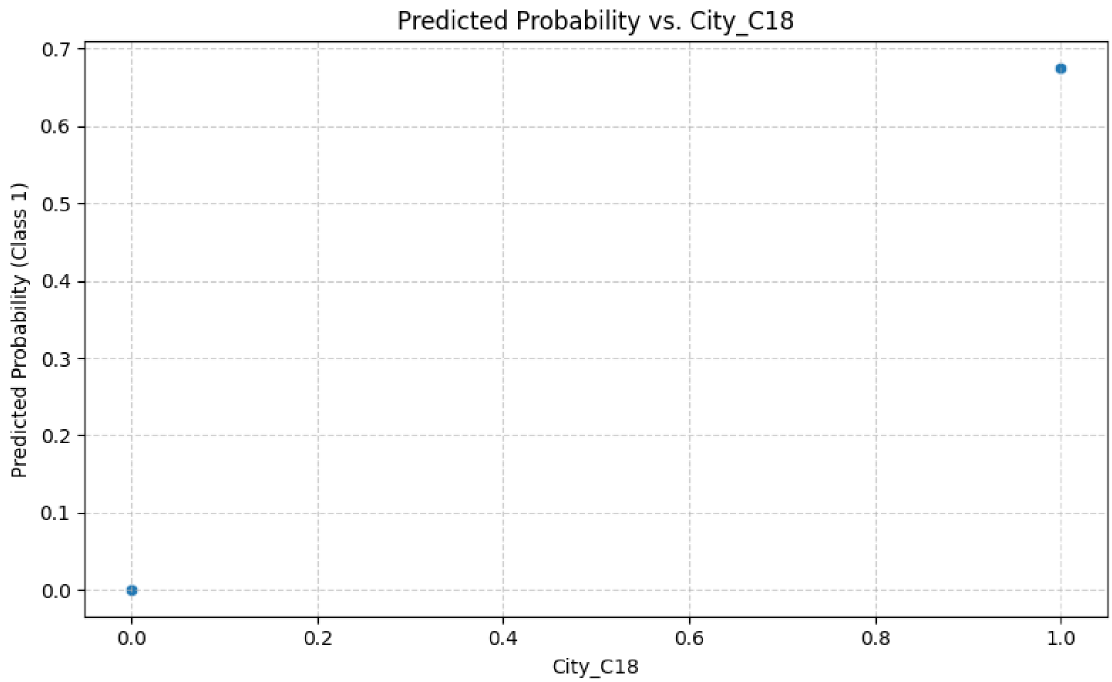



```
--------------------------------------------------
Plotting for feature: City_C17
```

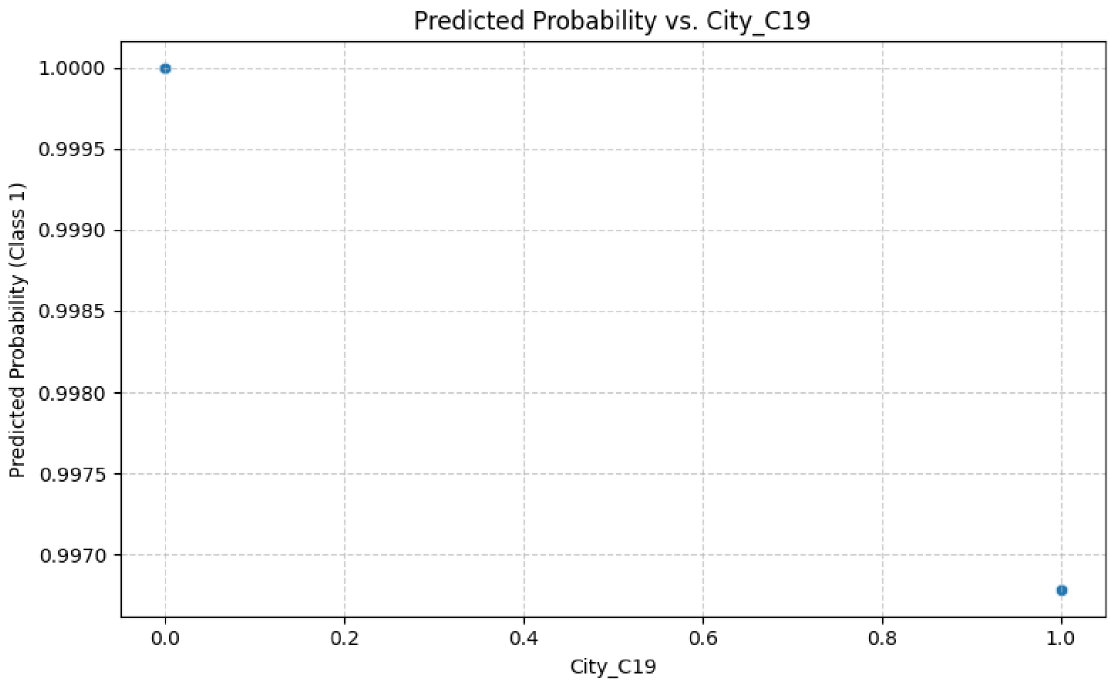
```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



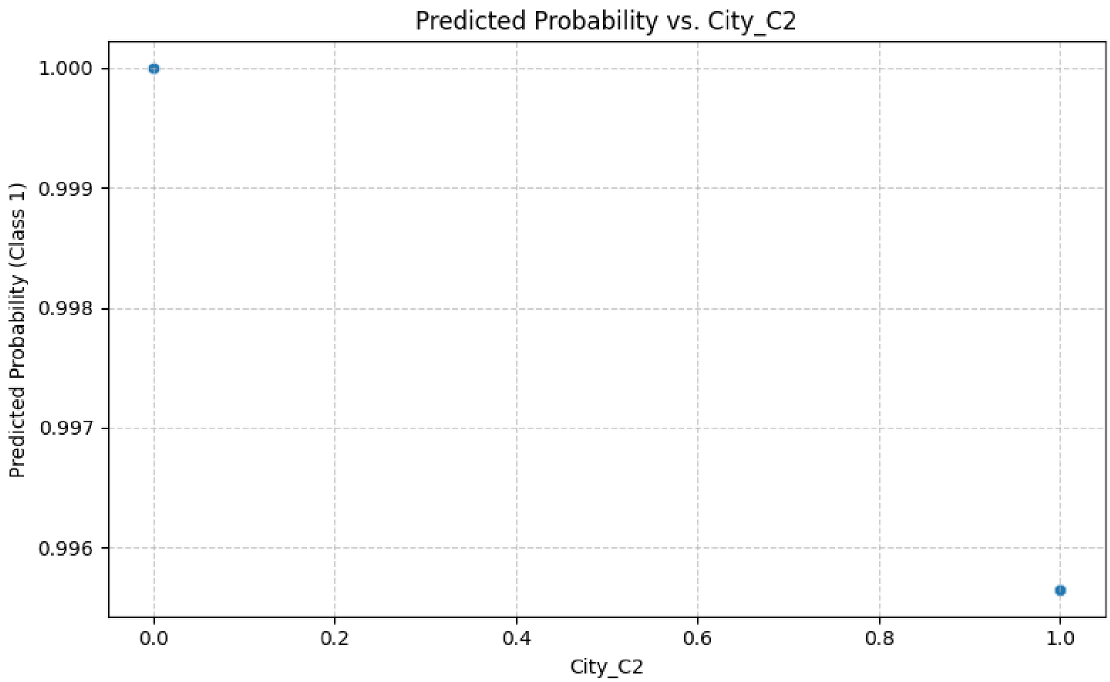

```
--------------------------------------------------
Plotting for feature: City_C18

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



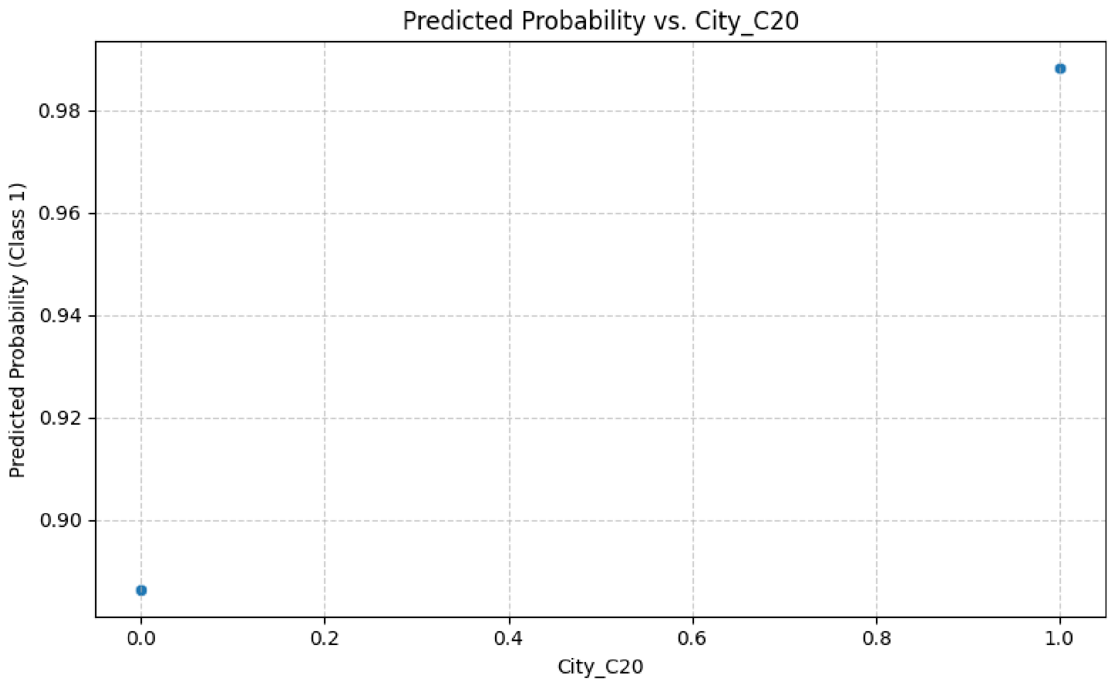

```
--------------------------------------------------
Plotting for feature: City_C19

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



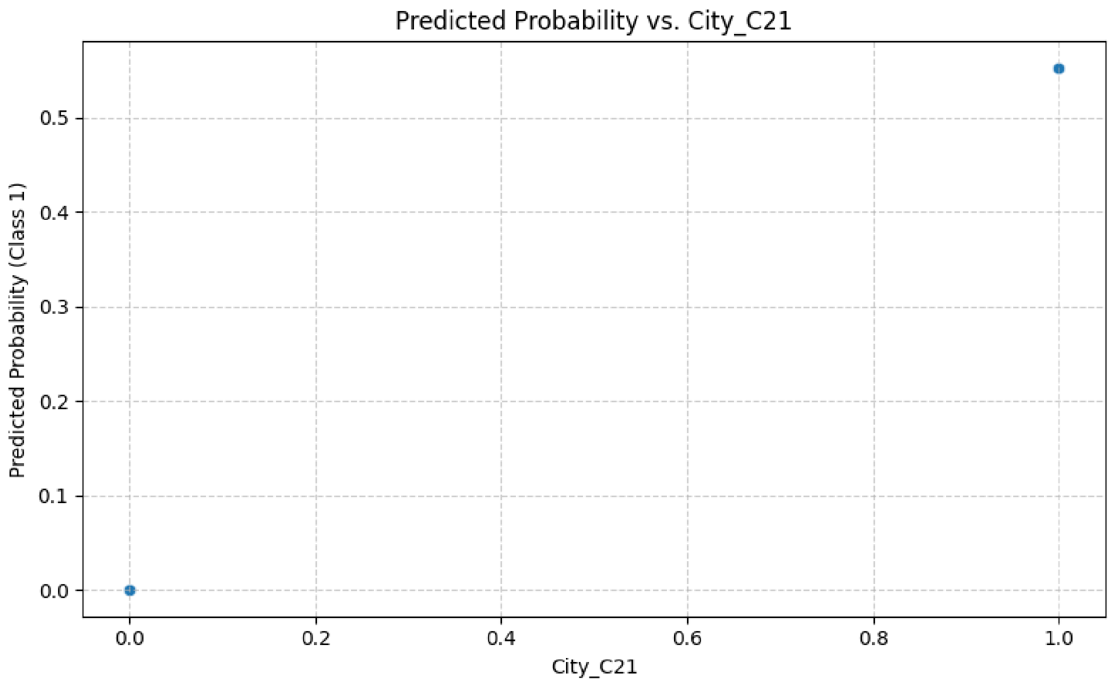

```
--------------------------------------------------
Plotting for feature: City_C2

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```




```
--------------------------------------------------
Plotting for feature: City_C20

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

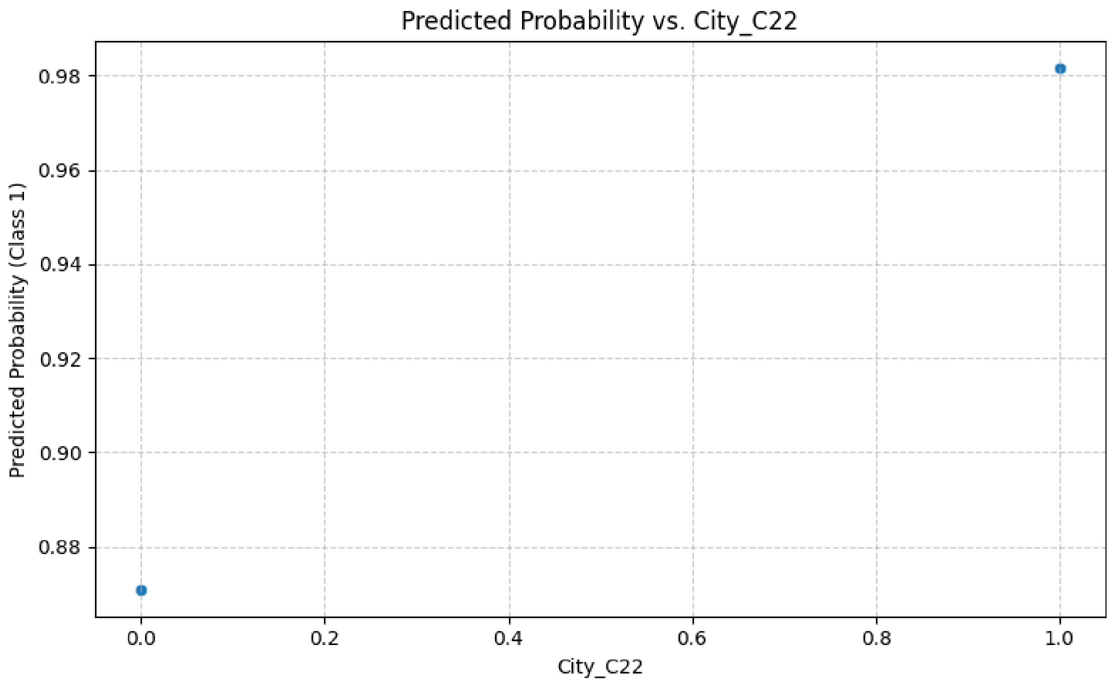



```
--------------------------------------------------
Plotting for feature: City_C21

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



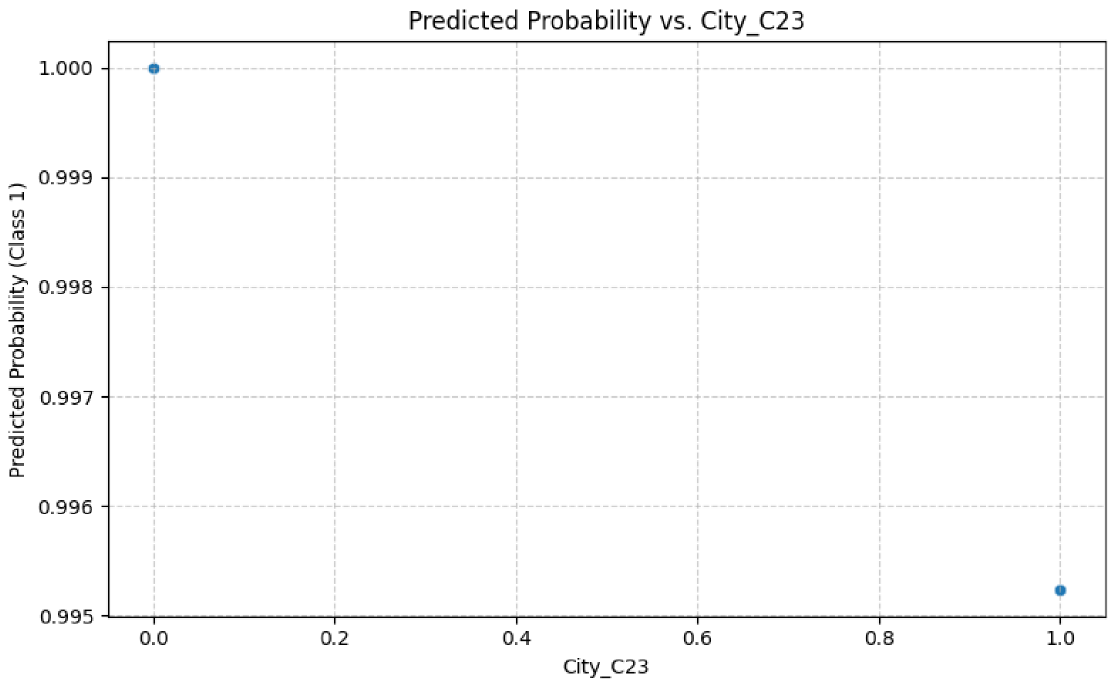

```
--------------------------------------------------
Plotting for feature: City_C22
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



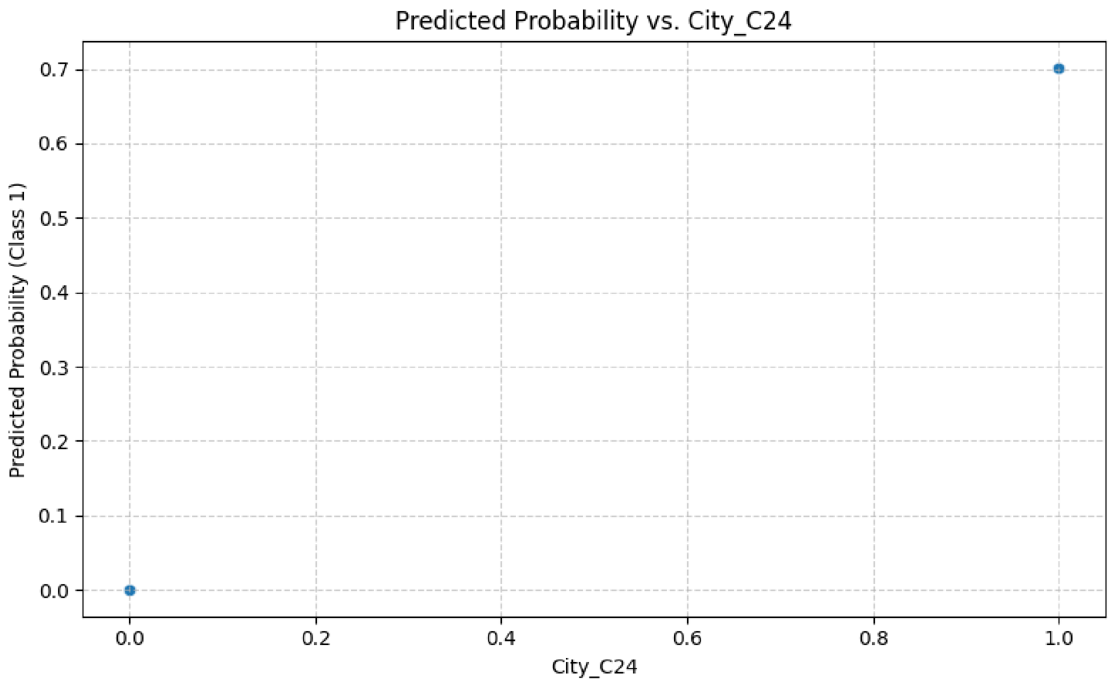

```
--------------------------------------------------
Plotting for feature: City_C23
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

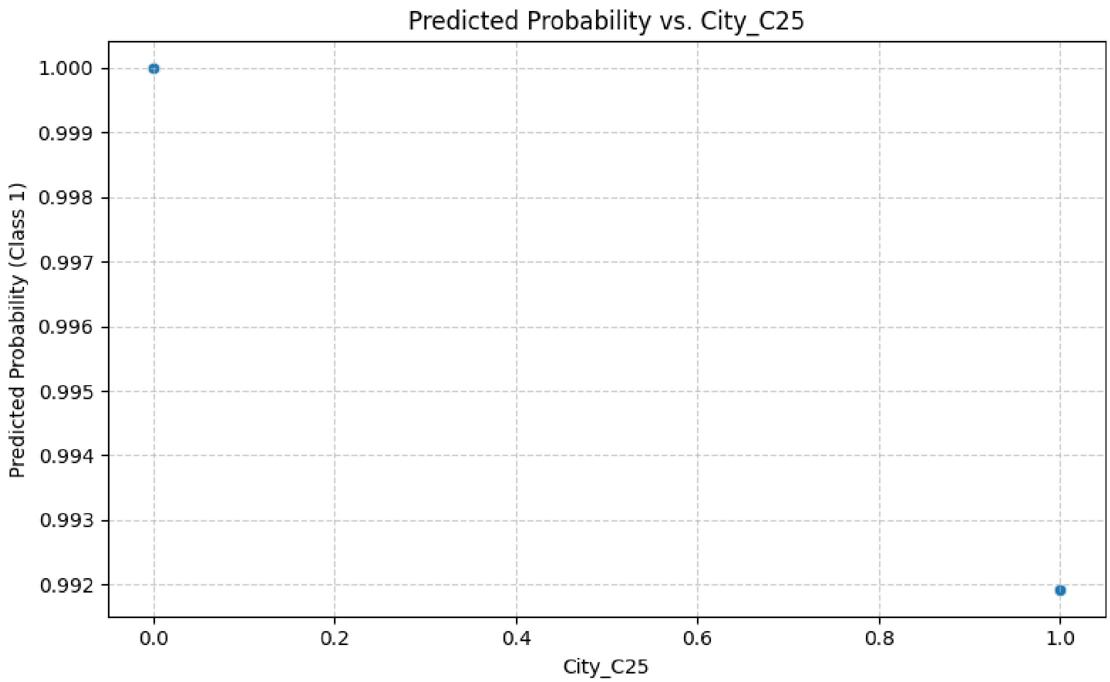



```
--------------------------------------------------
Plotting for feature: City_C24
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

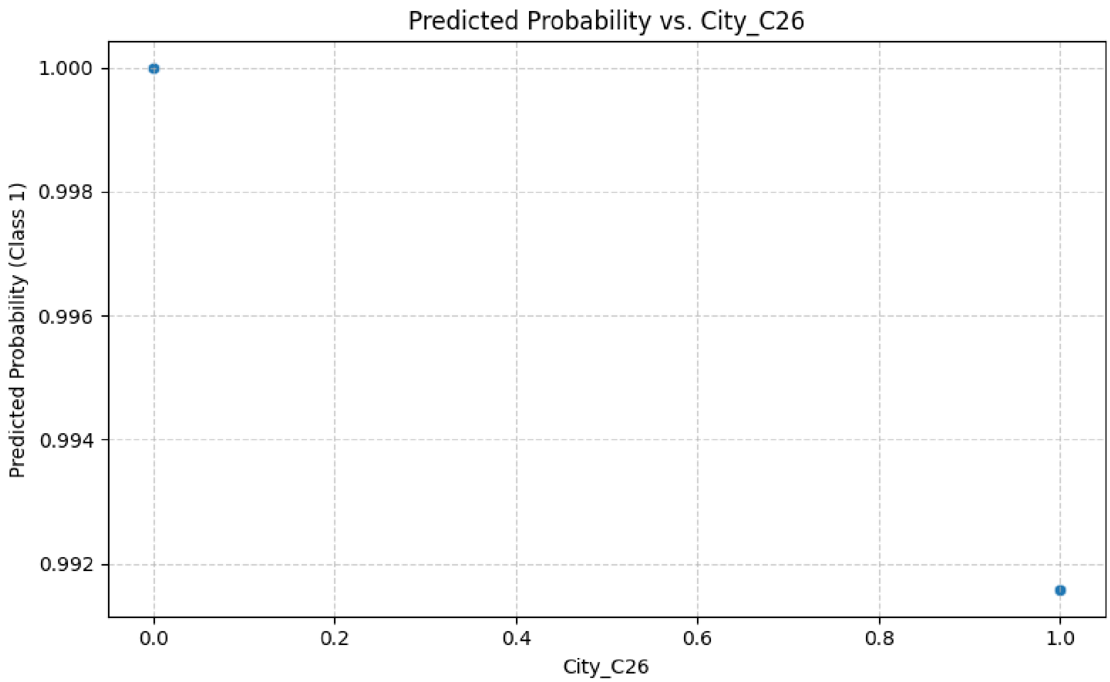



```
--------------------------------------------------
Plotting for feature: City_C25

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



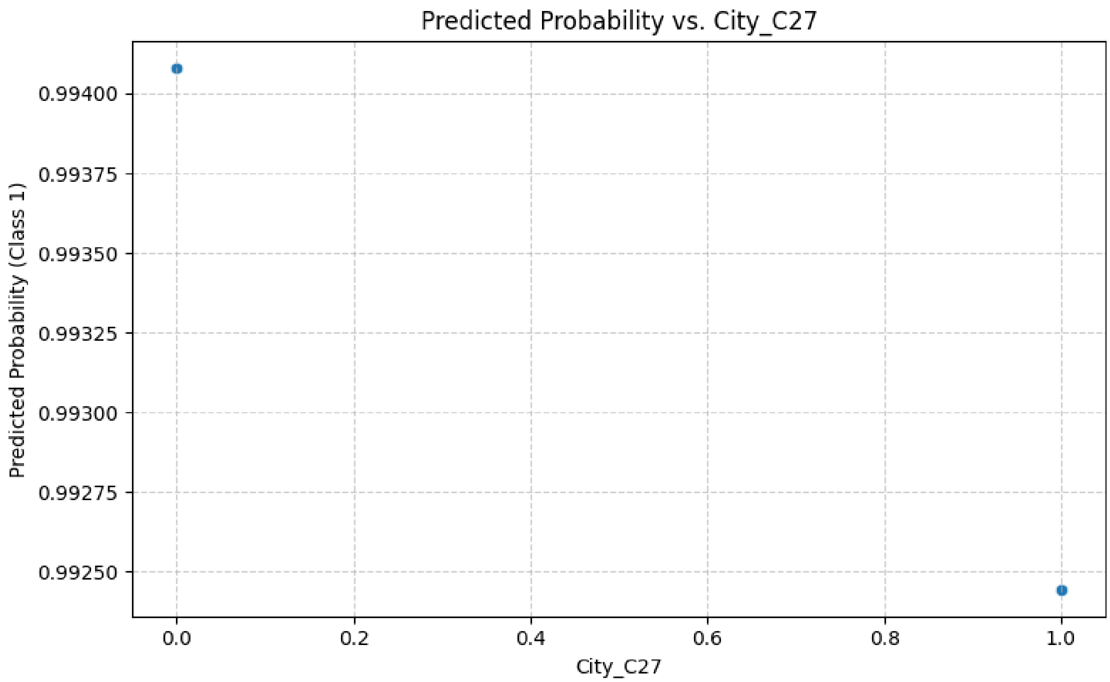

```
--------------------------------------------------
Plotting for feature: City_C26
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

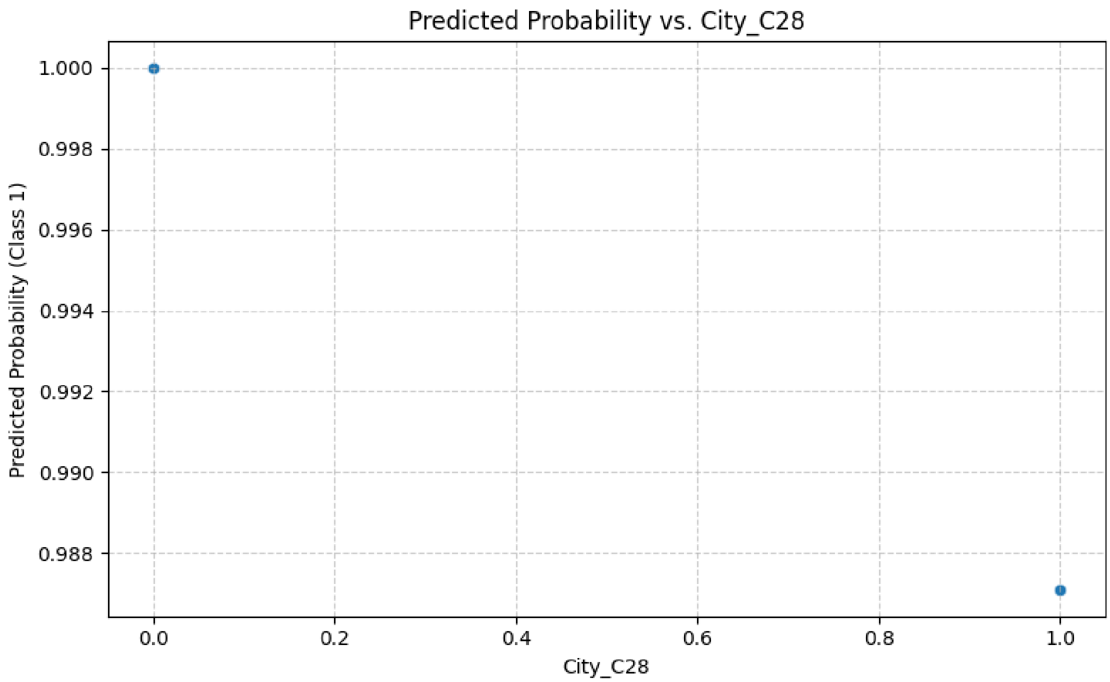



```
--------------------------------------------------
Plotting for feature: City_C27
```

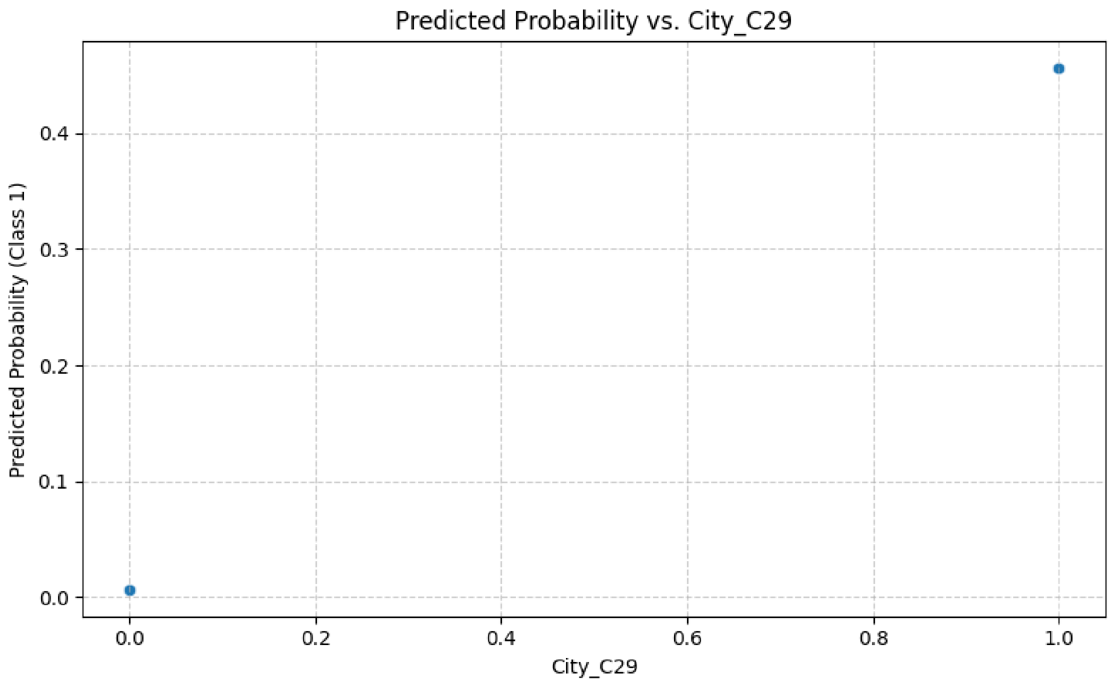
```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```




```
--------------------------------------------------
Plotting for feature: City_C28

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```

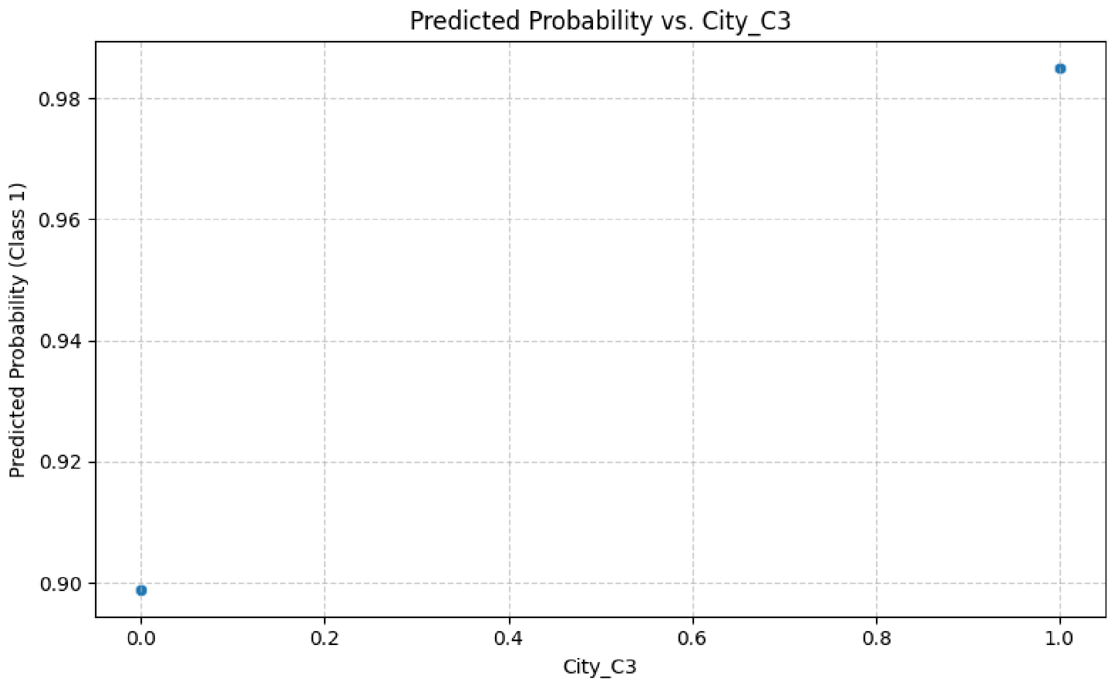



```
--------------------------------------------------
Plotting for feature: City_C29

/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



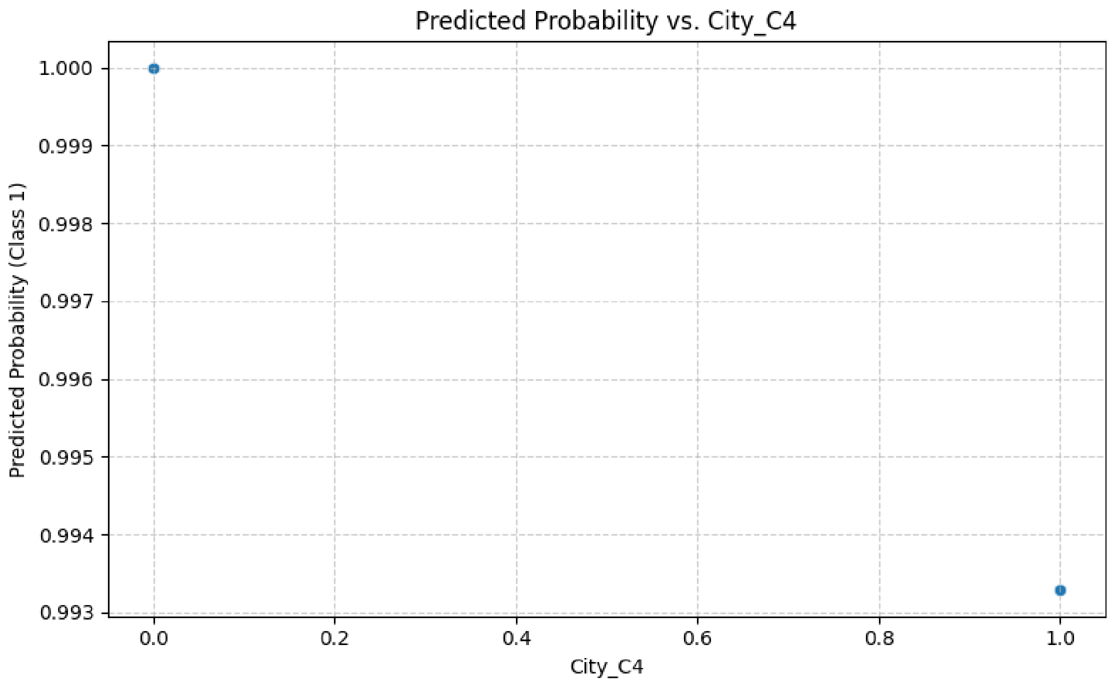

```
--------------------------------------------------
Plotting for feature: City_C3
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



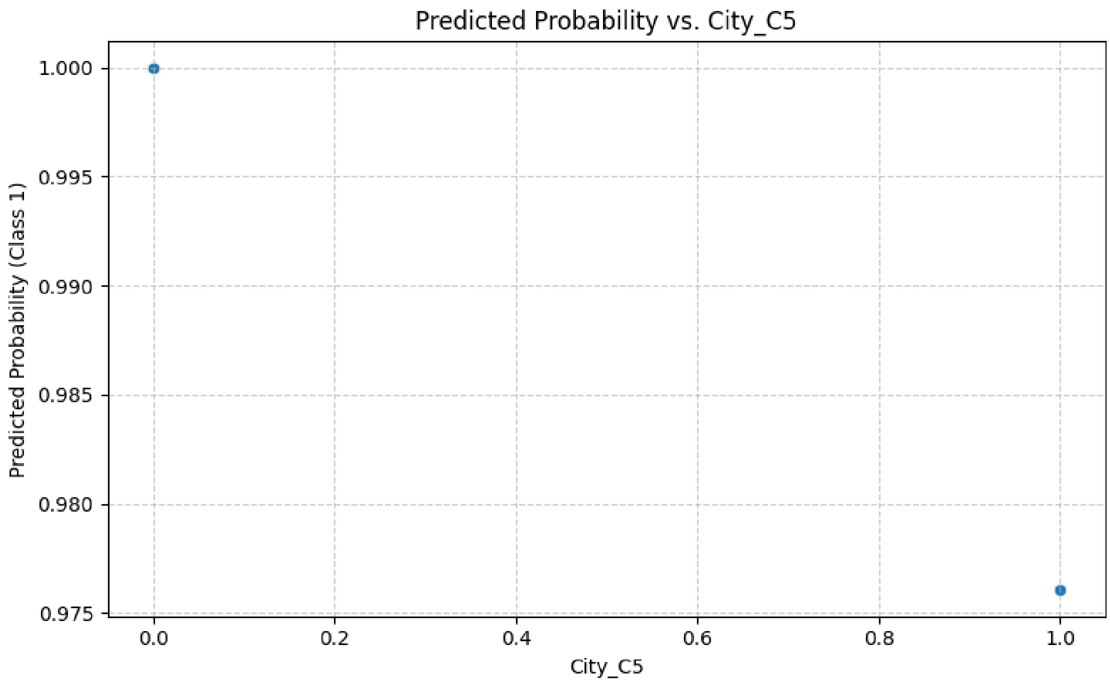

```
--------------------------------------------------
Plotting for feature: City_C4
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



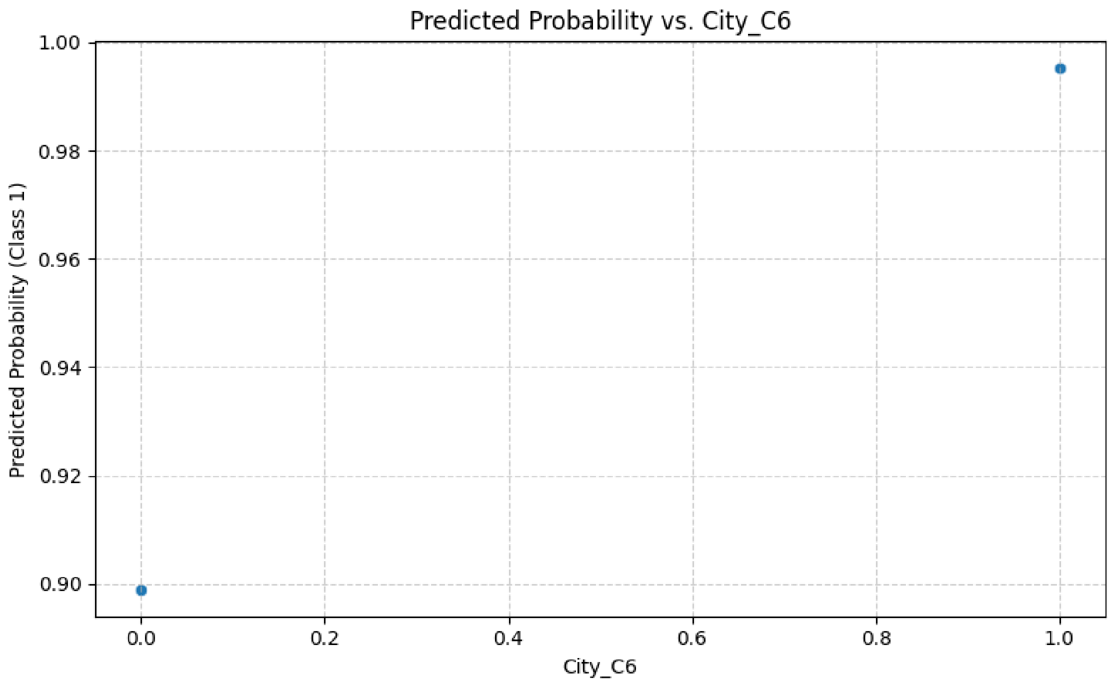

```
--------------------------------------------------
Plotting for feature: City_C5
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



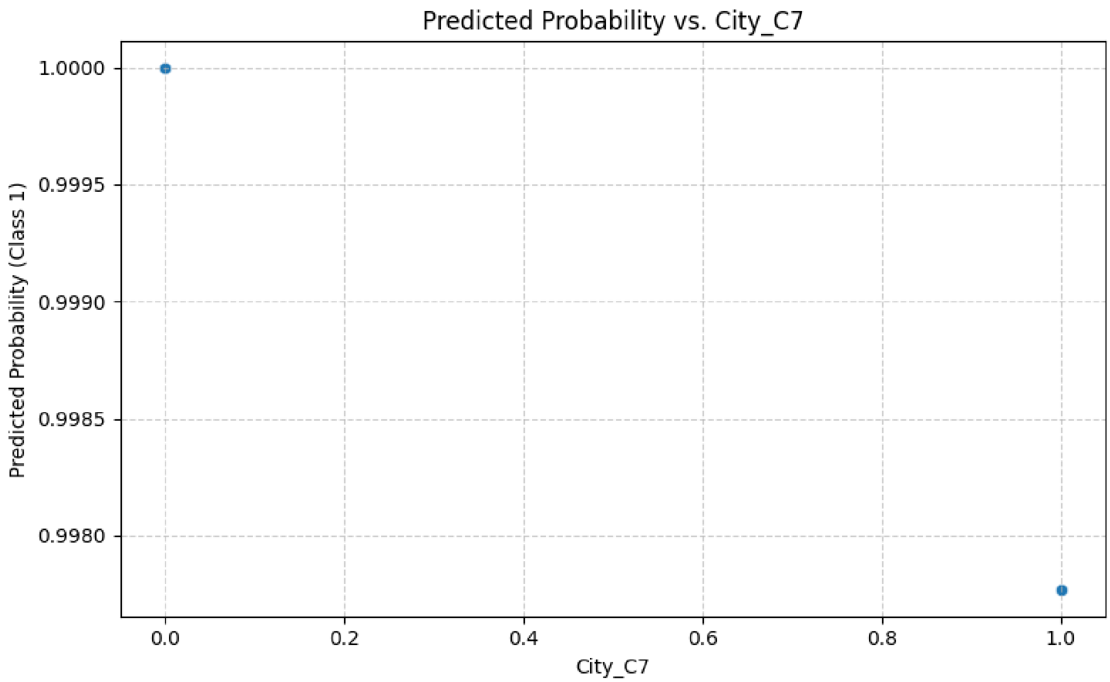

```
--------------------------------------------------
Plotting for feature: City_C6
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



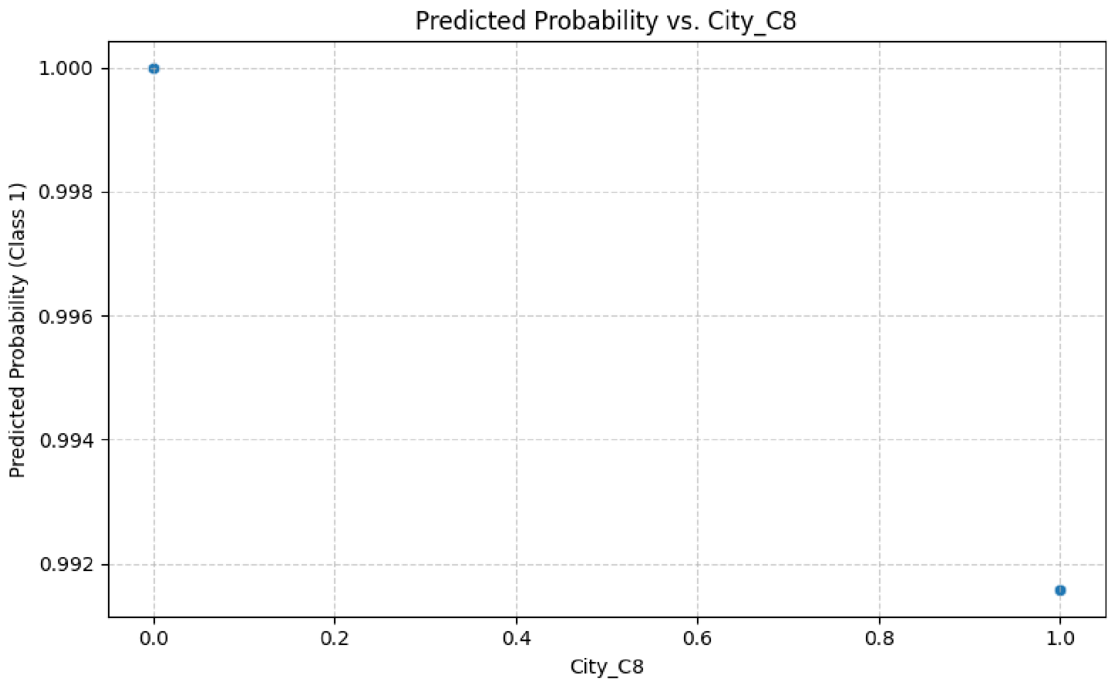

```
--------------------------------------------------
Plotting for feature: City_C7
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



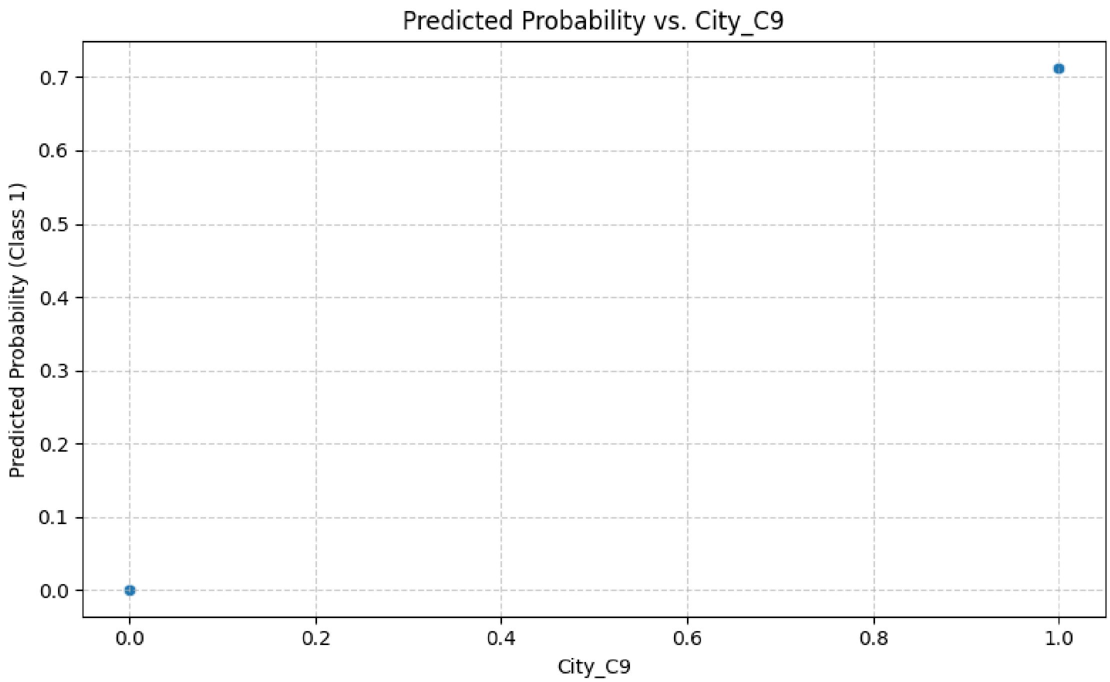

```
--------------------------------------------------
Plotting for feature: City_C8
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```




```
--------------------------------------------------
Plotting for feature: City_C9
```

```
/usr/local/lib/python3.11/dist-
packages/statsmodels/nonparametric/smoothers_lowess.py:226: RuntimeWarning:
invalid value encountered in divide
  res, _ = _lowess(y, x, x, np.ones_like(x),
```



[ ]:
```
# From the plots we can see that with the increase in reports(number of time
↪the driver reported), the probability of quitting drops to 0
# As age increase he probability if quitting gradually decreases ut then again
↪rises. The minimum probability
# of quitting is 0.5 w.r.t age. This means age is not a good param to derive a
↪conclusion
# As income increases the probability of quitting decreases.
# as business value increases the probability of quitting decreases.
# As the tenure increases the probability of quitting decreases.
# The other plots do not make much sense here and considering that they do not
↪account for a huge percentage,
# we can ignore them.
```

[ ]:
```
#Now let me predict the results on x_test using the model.
```

[ ]:
```
#For this I need to find the cut-off. Let me use the KS method to find this
```

[137]:
```
cutoffs=np.linspace(0.01,0.99,99)

train_score=rf_best_model.predict_proba(x_train)[:,1]
real=y_train
```

[138]:
```
KS_all=[]

for cutoff in cutoffs:

    predicted=(train_score>cutoff).astype(int)

    TP=((predicted==1) & (real==1)).sum()
    TN=((predicted==0) & (real==0)).sum()
    FP=((predicted==1) & (real==0)).sum()
    FN=((predicted==0) & (real==1)).sum()

    P=TP+FN
    N=TN+FP


    KS=(TP/P)-(FP/N)

    KS_all.append(KS)
```

[139]:
```
mycutoff=cutoffs[KS_all==max(KS_all)]
mycutoff
```

[139]:
```
array([0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43, 0.44, 0.45, 0.46,
       0.47, 0.48, 0.49, 0.5 , 0.51])
```

[ ]:
```
#Seeing that I have multiple cutoffs, I am picking 0.5 as the cutoff
```

[140]:
```
mycutoff=0.5
```

[ ]:
```
# Steps to be done before using test data to predict

# Remove driver_ID
# Remove Grade column as this was eliminated by VIF
#This is because the above was done to the training data
```

[142]:
```
x_test.drop('Driver_ID', axis=1, inplace=True)
x_test.drop('Grade', axis=1, inplace=True)
```

[147]:
```
x_test.head()
```

[147]:

	Reports_2019	Reports_2020	Age	Gender	Education_Level	Income \
2192	-0.698234	-0.933705	-1.760794	0.0	1.216049	-1.147538
2293	1.797664	1.905570	-1.378733	0.0	1.216049	0.685710
912	-0.698234	0.722539	-0.741964	0.0	-0.009263	-0.261395
1024	1.797664	0.249326	-0.971700	0.0	-0.009263	1.280652
778	0.890065	-0.933705	2.144719	0.0	-1.234575	-1.439279

```
      Joining Designation  Total Business Value  Quarterly Rating  QRIncrease  \
2192            -0.975022             -0.694331         -0.642003           0
2293            -0.975022              2.038039          3.059228           1
912             -0.975022             -0.331328         -0.642003           1
1024             0.213676              1.705952          1.825485           0
778             -0.975022             -0.106102          0.591741           0

      …  City_C27  City_C28  City_C29  City_C3  City_C4  City_C5  City_C6  \
2192  …         0         0         0        0        0        0        0
2293  …         0         0         0        0        0        0        0
912   …         0         0         0        0        0        0        0
1024  …         0         0         0        0        0        0        0
778   …         0         0         0        0        0        0        0

      City_C7  City_C8  City_C9
2192        0        0        1
2293        0        0        0
912         0        0        0
1024        0        0        0
778         0        0        0

[5 rows x 40 columns]
```

[148]:
```
x_test.shape
```

[148]:
```
(239, 40)
```

[149]:
```
x_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 239 entries, 2192 to 2314
Data columns (total 40 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Reports_2019          239 non-null    float64
 1   Reports_2020          239 non-null    float64
 2   Age                   239 non-null    float64
 3   Gender                239 non-null    float64
 4   Education_Level       239 non-null    float64
 5   Income                239 non-null    float64
 6   Joining Designation   239 non-null    float64
 7   Total Business Value  239 non-null    float64
 8   Quarterly Rating      239 non-null    float64
 9   QRIncrease            239 non-null    int64
 10  IncomeIncreased       239 non-null    int64
 11  tenure                239 non-null    float64
 12  City_C10              239 non-null    int64
```

```
 13  City_C11             239 non-null    int64
 14  City_C12             239 non-null    int64
 15  City_C13             239 non-null    int64
 16  City_C14             239 non-null    int64
 17  City_C15             239 non-null    int64
 18  City_C16             239 non-null    int64
 19  City_C17             239 non-null    int64
 20  City_C18             239 non-null    int64
 21  City_C19             239 non-null    int64
 22  City_C2              239 non-null    int64
 23  City_C20             239 non-null    int64
 24  City_C21             239 non-null    int64
 25  City_C22             239 non-null    int64
 26  City_C23             239 non-null    int64
 27  City_C24             239 non-null    int64
 28  City_C25             239 non-null    int64
 29  City_C26             239 non-null    int64
 30  City_C27             239 non-null    int64
 31  City_C28             239 non-null    int64
 32  City_C29             239 non-null    int64
 33  City_C3              239 non-null    int64
 34  City_C4              239 non-null    int64
 35  City_C5              239 non-null    int64
 36  City_C6              239 non-null    int64
 37  City_C7              239 non-null    int64
 38  City_C8              239 non-null    int64
 39  City_C9              239 non-null    int64
dtypes: float64(10), int64(30)
memory usage: 76.6 KB
```

[146]:
```
x_train.shape,y_train.shape,x_test.shape,y_test.shape
```

[146]:
```
((2643, 40), (2643,), (239, 40), (239,), (239, 40))
```

[150]:
```
test_score=rf_best_model.predict_proba(x_test)[:,1]
test_classes=(test_score>mycutoff).astype(int)
```

[152]:
```
test_classes.shape,y_test.shape
```

[152]:
```
((239,), (239,))
```

[154]:
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, test_classes)
print(f"Accuracy of the model: {accuracy:.4f}")
```

```
Accuracy of the model: 0.9121
```

[ ]:
```
#From this we can see that the model has generalized well.
#Earlier with hyperparameter tuning on the train data we had got the accuracy␣
↪of 0.935.
#Now on the testing data we are getting 0.9121, which is a good generalization
```

[ ]:
```
#let me now check the other performance metrics
```

[155]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, confusion_matrix
# Calculate metrics
accuracy = accuracy_score(y_test, test_classes)
precision = precision_score(y_test, test_classes, zero_division=0)
recall = recall_score(y_test, test_classes, zero_division=0)
f1 = f1_score(y_test, test_classes, zero_division=0)
conf_matrix = confusion_matrix(y_test, test_classes)

print(f"\nAccuracy: {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("\nConfusion Matrix Interpretation:")
print("True Negatives (TN):", conf_matrix[0, 0])
print("False Positives (FP):", conf_matrix[0, 1])
print("False Negatives (FN):", conf_matrix[1, 0])
print("True Positives (TP):", conf_matrix[1, 1])
```

```
Accuracy: 0.9121
Precision: 0.9490
Recall: 0.9198
F1-Score: 0.9342

Confusion Matrix:
[[ 69   8]
 [ 13 149]]

Confusion Matrix Interpretation:
True Negatives (TN): 69
False Positives (FP): 8
False Negatives (FN): 13
True Positives (TP): 149
```

[ ]:
```
#Now let me do the model training using a boosting algorithm
#XGBoost in this case and see what the result is
```

[156]:
```
#sklearn and XGBoost are having some compatibility issues, so doing this, i.e.␣
↪changing the version to the right one
!pip uninstall -y scikit-learn
!pip install scikit-learn==1.5.2
```

```
Found existing installation: scikit-learn 1.6.1
Uninstalling scikit-learn-1.6.1:
  Successfully uninstalled scikit-learn-1.6.1
Collecting scikit-learn==1.5.2
  Downloading scikit_learn-1.5.2-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn==1.5.2) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn==1.5.2) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn==1.5.2) (1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn==1.5.2) (3.6.0)
Downloading
scikit_learn-1.5.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(13.3 MB)
                         13.3/13.3 MB
85.7 MB/s eta 0:00:00
Installing collected packages: scikit-learn
Successfully installed scikit-learn-1.5.2
```

[157]:
```
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV
```

[158]:
```
xgb_params = {
    "learning_rate": [0.01, 0.05, 0.1, 0.3, 0.5],
    "gamma": [i / 10.0 for i in range(0, 5)],
    "max_depth": [2, 3, 4, 5, 6, 7, 8],
    "min_child_weight": [1, 2, 5, 10],
    "max_delta_step": [0, 1, 2, 5, 10],
    "subsample": [i / 10.0 for i in range(5, 10)],
    "colsample_bytree": [i / 10.0 for i in range(5, 10)],
    "colsample_bylevel": [i / 10.0 for i in range(5, 10)],
    "reg_lambda": [1e-5, 1e-2, 0.1, 1, 100],
    "reg_alpha": [1e-5, 1e-2, 0.1, 1, 100],
    "scale_pos_weight": [1, 2, 3, 4, 5, 6, 7, 8, 9],
    "n_estimators": [100, 500, 700, 1000]
}

xgb = XGBClassifier(objective='binary:logistic', use_label_encoder=False,␣
↪eval_metric='logloss', random_state=42)
```

```
random_search_xgb = RandomizedSearchCV(xgb, param_distributions=xgb_params,␣
↪n_iter=10, cv=7, scoring='f1', verbose=1, n_jobs=-1, random_state=42)
random_search_xgb.fit(x_train, y_train)
```

```
Fitting 7 folds for each of 10 candidates, totalling 70 fits
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning:
[10:56:16] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

[158]:
```
RandomizedSearchCV(cv=7,
                   estimator=XGBClassifier(base_score=None, booster=None,
                                           callbacks=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None, device=None,
                                           early_stopping_rounds=None,
                                           enable_categorical=False,
                                           eval_metric='logloss',
                                           feature_types=None, gamma=None,
                                           grow_policy=None,
                                           importance_type=None,
                                           interaction_constraints=None,
                                           learning…
                                        'gamma': [0.0, 0.1, 0.2, 0.3, 0.4],
                                        'learning_rate': [0.01, 0.05, 0.1, 0.3,
                                                          0.5],
                                        'max_delta_step': [0, 1, 2, 5, 10],
                                        'max_depth': [2, 3, 4, 5, 6, 7, 8],
                                        'min_child_weight': [1, 2, 5, 10],
                                        'n_estimators': [100, 500, 700, 1000],
                                        'reg_alpha': [1e-05, 0.01, 0.1, 1, 100],
                                        'reg_lambda': [1e-05, 0.01, 0.1, 1,
                                                       100],
                                        'scale_pos_weight': [1, 2, 3, 4, 5, 6,
                                                             7, 8, 9],
                                        'subsample': [0.5, 0.6, 0.7, 0.8, 0.9]},
                   random_state=42, scoring='f1', verbose=1)
```

[159]:
```
report(random_search_xgb.cv_results_, 5)
```

```
Model with rank: 1
Mean validation score: 0.93684 (std: 0.01791)
Parameters: {'subsample': 0.7, 'scale_pos_weight': 4, 'reg_lambda': 1e-05,
'reg_alpha': 0.01, 'n_estimators': 700, 'min_child_weight': 1, 'max_depth': 8,
'max_delta_step': 0, 'learning_rate': 0.05, 'gamma': 0.3, 'colsample_bytree':
```

```
0.5, 'colsample_bylevel': 0.9}

Model with rank: 2
Mean validation score: 0.93498 (std: 0.01748)
Parameters: {'subsample': 0.7, 'scale_pos_weight': 7, 'reg_lambda': 0.01,
'reg_alpha': 1, 'n_estimators': 500, 'min_child_weight': 2, 'max_depth': 5,
'max_delta_step': 10, 'learning_rate': 0.1, 'gamma': 0.1, 'colsample_bytree':
0.9, 'colsample_bylevel': 0.8}

Model with rank: 3
Mean validation score: 0.93224 (std: 0.02191)
Parameters: {'subsample': 0.5, 'scale_pos_weight': 5, 'reg_lambda': 1e-05,
'reg_alpha': 0.1, 'n_estimators': 500, 'min_child_weight': 5, 'max_depth': 5,
'max_delta_step': 1, 'learning_rate': 0.1, 'gamma': 0.3, 'colsample_bytree':
0.7, 'colsample_bylevel': 0.7}

Model with rank: 4
Mean validation score: 0.93017 (std: 0.02015)
Parameters: {'subsample': 0.7, 'scale_pos_weight': 4, 'reg_lambda': 1,
'reg_alpha': 0.01, 'n_estimators': 500, 'min_child_weight': 1, 'max_depth': 8,
'max_delta_step': 1, 'learning_rate': 0.3, 'gamma': 0.4, 'colsample_bytree':
0.5, 'colsample_bylevel': 0.8}

Model with rank: 5
Mean validation score: 0.92990 (std: 0.01727)
Parameters: {'subsample': 0.8, 'scale_pos_weight': 1, 'reg_lambda': 0.1,
'reg_alpha': 1e-05, 'n_estimators': 500, 'min_child_weight': 5, 'max_depth': 7,
'max_delta_step': 1, 'learning_rate': 0.5, 'gamma': 0.4, 'colsample_bytree':
0.7, 'colsample_bylevel': 0.9}
```

[160]:

```
xgb_params = {
    "learning_rate": [0.04,0.05,0.06,0.09,0.1,0.12],
    "gamma": [0.1,0.2,0.3,0.4],
    "max_depth": [4,5,6,7,8,9],
    "min_child_weight": [1,2,3],
    "max_delta_step": [0, 1, 9,10,11],
    "subsample": [0.6,0.7,0.8],
    "colsample_bytree": [0.4,0.5,0.6,0.8,0.9,1.0],
    "colsample_bylevel": [0.7,0.8,0.9,1.0],
    "reg_lambda": [0.000009,0.00001,0.000015,0.009,0.01,0.015],
    "reg_alpha": [0.009,0.01,0.015,0.5,1,1.5],
    "scale_pos_weight": [3, 4, 5, 6, 7, 8],
    "n_estimators": [1450,500,550,650,700,750]
}
```

```
xgb = XGBClassifier(objective='binary:logistic', use_label_encoder=False,␣
↪eval_metric='logloss', random_state=42)
random_search_xgb = RandomizedSearchCV(xgb, param_distributions=xgb_params,␣
↪n_iter=10, cv=7, scoring='f1', verbose=1, n_jobs=-1)
random_search_xgb.fit(x_train, y_train)
```

```
Fitting 7 folds for each of 10 candidates, totalling 70 fits
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning:
[11:06:58] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

[160]:
```
RandomizedSearchCV(cv=7,
                   estimator=XGBClassifier(base_score=None, booster=None,
                                           callbacks=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None, device=None,
                                           early_stopping_rounds=None,
                                           enable_categorical=False,
                                           eval_metric='logloss',
                                           feature_types=None, gamma=None,
                                           grow_policy=None,
                                           importance_type=None,
                                           interaction_constraints=None,
                                           learning…
                                        'gamma': [0.1, 0.2, 0.3, 0.4],
                                        'learning_rate': [0.04, 0.05, 0.06,
                                                          0.09, 0.1, 0.12],
                                        'max_delta_step': [0, 1, 9, 10, 11],
                                        'max_depth': [4, 5, 6, 7, 8, 9],
                                        'min_child_weight': [1, 2, 3],
                                        'n_estimators': [1450, 500, 550, 650,
                                                         700, 750],
                                        'reg_alpha': [0.009, 0.01, 0.015, 0.5,
                                                      1, 1.5],
                                        'reg_lambda': [9e-06, 1e-05, 1.5e-05,
                                                       0.009, 0.01, 0.015],
                                        'scale_pos_weight': [3, 4, 5, 6, 7, 8],
                                        'subsample': [0.6, 0.7, 0.8]},
                   scoring='f1', verbose=1)
```

[161]:
```
report(random_search_xgb.cv_results_, 5)
```

```
Model with rank: 1
Mean validation score: 0.93848 (std: 0.01852)
```

```
Parameters: {'subsample': 0.7, 'scale_pos_weight': 8, 'reg_lambda': 0.009,
'reg_alpha': 0.01, 'n_estimators': 750, 'min_child_weight': 3, 'max_depth': 9,
'max_delta_step': 11, 'learning_rate': 0.05, 'gamma': 0.2, 'colsample_bytree':
0.5, 'colsample_bylevel': 0.9}

Model with rank: 2
Mean validation score: 0.93832 (std: 0.01852)
Parameters: {'subsample': 0.6, 'scale_pos_weight': 4, 'reg_lambda': 0.015,
'reg_alpha': 1, 'n_estimators': 550, 'min_child_weight': 2, 'max_depth': 9,
'max_delta_step': 1, 'learning_rate': 0.09, 'gamma': 0.4, 'colsample_bytree':
1.0, 'colsample_bylevel': 0.8}

Model with rank: 3
Mean validation score: 0.93692 (std: 0.01868)
Parameters: {'subsample': 0.8, 'scale_pos_weight': 8, 'reg_lambda': 0.01,
'reg_alpha': 1, 'n_estimators': 500, 'min_child_weight': 2, 'max_depth': 9,
'max_delta_step': 11, 'learning_rate': 0.12, 'gamma': 0.1, 'colsample_bytree':
0.5, 'colsample_bylevel': 0.7}

Model with rank: 4
Mean validation score: 0.93625 (std: 0.01890)
Parameters: {'subsample': 0.7, 'scale_pos_weight': 6, 'reg_lambda': 0.01,
'reg_alpha': 0.5, 'n_estimators': 750, 'min_child_weight': 2, 'max_depth': 9,
'max_delta_step': 1, 'learning_rate': 0.04, 'gamma': 0.3, 'colsample_bytree':
0.9, 'colsample_bylevel': 0.8}

Model with rank: 5
Mean validation score: 0.93547 (std: 0.01819)
Parameters: {'subsample': 0.7, 'scale_pos_weight': 5, 'reg_lambda': 1e-05,
'reg_alpha': 0.5, 'n_estimators': 750, 'min_child_weight': 1, 'max_depth': 7,
'max_delta_step': 9, 'learning_rate': 0.04, 'gamma': 0.3, 'colsample_bytree':
0.5, 'colsample_bylevel': 0.7}
```

[162]:
```python
xgb_params = {
    "learning_rate": [0.04,0.05,0.06,0.08,0.09,0.1,0.12],
    "gamma": [0.1,0.2,0.3,0.4,0.5,0.6],
    "max_depth": [8,9,10,11],
    "min_child_weight": [1,2,3,4],
    "max_delta_step": [0,1,2,10,11,12],
    "subsample": [0.5,0.6,0.7,0.8],
    "colsample_bytree": [0.4,0.5,0.6,0.8,0.9,1.0],
    "colsample_bylevel": [0.7,0.8,0.9,1.0],
    "reg_lambda": [0.08,0.09,0.1,0.014,0.015,0.016],
    "reg_alpha": [0.009,0.01,0.015,0.5,1,1.5],
    "scale_pos_weight": [3, 4, 5,7, 8,9],
    "n_estimators": [740,750,760,540,550,560]
```

```
}

xgb = XGBClassifier(objective='binary:logistic', use_label_encoder=False,␣
 ↪eval_metric='logloss', random_state=42)
random_search_xgb = RandomizedSearchCV(xgb, param_distributions=xgb_params,␣
 ↪n_iter=10, cv=7, scoring='f1', verbose=1, n_jobs=-1)
random_search_xgb.fit(x_train, y_train)
```

```
Fitting 7 folds for each of 10 candidates, totalling 70 fits
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning:
[11:12:24] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

[162]:
```
RandomizedSearchCV(cv=7,
                   estimator=XGBClassifier(base_score=None, booster=None,
                                           callbacks=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
                                           colsample_bytree=None, device=None,
                                           early_stopping_rounds=None,
                                           enable_categorical=False,
                                           eval_metric='logloss',
                                           feature_types=None, gamma=None,
                                           grow_policy=None,
                                           importance_type=None,
                                           interaction_constraints=None,
                                           learning…
                                        'learning_rate': [0.04, 0.05, 0.06,
                                                          0.08, 0.09, 0.1,
                                                          0.12],
                                        'max_delta_step': [0, 1, 2, 10, 11, 12],
                                        'max_depth': [8, 9, 10, 11],
                                        'min_child_weight': [1, 2, 3, 4],
                                        'n_estimators': [740, 750, 760, 540,
                                                         550, 560],
                                        'reg_alpha': [0.009, 0.01, 0.015, 0.5,
                                                      1, 1.5],
                                        'reg_lambda': [0.08, 0.09, 0.1, 0.014,
                                                       0.015, 0.016],
                                        'scale_pos_weight': [3, 4, 5, 7, 8, 9],
                                        'subsample': [0.5, 0.6, 0.7, 0.8]},
                   scoring='f1', verbose=1)
```

[163]:
```
report(random_search_xgb.cv_results_, 5)
```

```
Model with rank: 1
Mean validation score: 0.93808 (std: 0.01803)
Parameters: {'subsample': 0.6, 'scale_pos_weight': 3, 'reg_lambda': 0.1,
'reg_alpha': 0.009, 'n_estimators': 740, 'min_child_weight': 3, 'max_depth': 9,
'max_delta_step': 12, 'learning_rate': 0.05, 'gamma': 0.2, 'colsample_bytree':
1.0, 'colsample_bylevel': 0.8}

Model with rank: 2
Mean validation score: 0.93768 (std: 0.01574)
Parameters: {'subsample': 0.8, 'scale_pos_weight': 3, 'reg_lambda': 0.09,
'reg_alpha': 0.5, 'n_estimators': 750, 'min_child_weight': 2, 'max_depth': 8,
'max_delta_step': 12, 'learning_rate': 0.06, 'gamma': 0.3, 'colsample_bytree':
0.4, 'colsample_bylevel': 1.0}

Model with rank: 3
Mean validation score: 0.93752 (std: 0.01966)
Parameters: {'subsample': 0.6, 'scale_pos_weight': 8, 'reg_lambda': 0.016,
'reg_alpha': 0.009, 'n_estimators': 740, 'min_child_weight': 4, 'max_depth': 11,
'max_delta_step': 0, 'learning_rate': 0.12, 'gamma': 0.3, 'colsample_bytree':
0.6, 'colsample_bylevel': 1.0}

Model with rank: 4
Mean validation score: 0.93720 (std: 0.01685)
Parameters: {'subsample': 0.6, 'scale_pos_weight': 3, 'reg_lambda': 0.09,
'reg_alpha': 0.5, 'n_estimators': 740, 'min_child_weight': 1, 'max_depth': 11,
'max_delta_step': 11, 'learning_rate': 0.1, 'gamma': 0.5, 'colsample_bytree':
0.6, 'colsample_bylevel': 0.7}

Model with rank: 5
Mean validation score: 0.93692 (std: 0.02220)
Parameters: {'subsample': 0.8, 'scale_pos_weight': 7, 'reg_lambda': 0.1,
'reg_alpha': 1, 'n_estimators': 560, 'min_child_weight': 1, 'max_depth': 10,
'max_delta_step': 2, 'learning_rate': 0.1, 'gamma': 0.3, 'colsample_bytree':
0.4, 'colsample_bylevel': 0.9}
```

[ ]:
```
#After multiple trials, I am selecting the below

# Model with rank: 1
# Mean validation score: 0.93848 (std: 0.01852)
# Parameters: {'subsample': 0.7, 'scale_pos_weight': 8, 'reg_lambda': 0.009,
 ↪'reg_alpha': 0.01, 'n_estimators': 750, 'min_child_weight': 3, 'max_depth':
 ↪9, 'max_delta_step': 11, 'learning_rate': 0.05, 'gamma': 0.2,
 ↪'colsample_bytree': 0.5, 'colsample_bylevel': 0.9}
```

[164]:
```
```

```
xgb_c=XGBClassifier(**{'subsample': 0.7, 'scale_pos_weight': 8, 'reg_lambda': 0.
↪009, 'reg_alpha': 0.01, 'n_estimators': 750, 'min_child_weight': 3,␣
↪'max_depth': 9, 'max_delta_step': 11, 'learning_rate': 0.05, 'gamma': 0.2,␣
↪'colsample_bytree': 0.5, 'colsample_bylevel': 0.9})
```

[165]:
```
xgb_c.fit(x_train,y_train)
```

[165]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=0.9, colsample_bynode=None,
              colsample_bytree=0.5, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=0.2, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.05, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=11,
              max_depth=9, max_leaves=None, min_child_weight=3, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=750,
              n_jobs=None, num_parallel_tree=None, random_state=None, …)
```

[168]:
```
#Feature importance
feat_imp_df1=pd.DataFrame({'features':x_train.columns,
                           'importance':xgb_c.feature_importances_})

feat_imp_df1=feat_imp_df1.sort_values('importance',ascending=False)
feat_imp_df1['normalised_imp']=feat_imp_df1['importance']/np.
↪sum(feat_imp_df1['importance'])
feat_imp_df1['cum_imp']=np.cumsum(feat_imp_df1['normalised_imp'])
```

[169]:
```
feat_imp_df1
```

[169]:

	features	importance	normalised_imp	cum_imp
10	IncomeIncreased	0.110974	0.110974	0.110974
0	Reports_2019	0.092104	0.092104	0.203078
1	Reports_2020	0.082772	0.082772	0.285850
9	QRIncrease	0.047206	0.047206	0.333055
8	Quarterly Rating	0.046880	0.046880	0.379936
15	City_C13	0.031247	0.031247	0.411183
11	tenure	0.025562	0.025562	0.436745
21	City_C19	0.025443	0.025443	0.462189
27	City_C24	0.024845	0.024845	0.487034
28	City_C25	0.024696	0.024696	0.511730
23	City_C20	0.023553	0.023553	0.535283
33	City_C3	0.022216	0.022216	0.557499
39	City_C9	0.022049	0.022049	0.579548
12	City_C10	0.021596	0.021596	0.601143
3	Gender	0.021359	0.021359	0.622503
7	Total Business Value	0.020943	0.020943	0.643446
37	City_C7	0.019944	0.019944	0.663390

30	City_C27	0.019932	0.019932	0.683322
25	City_C22	0.019840	0.019840	0.703162
38	City_C8	0.019725	0.019725	0.722887
20	City_C18	0.017829	0.017829	0.740716
14	City_C12	0.017806	0.017806	0.758522
6	Joining Designation	0.017728	0.017728	0.776250
31	City_C28	0.017489	0.017489	0.793739
22	City_C2	0.017407	0.017407	0.811146
29	City_C26	0.017081	0.017081	0.828228
2	Age	0.016010	0.016010	0.844238
26	City_C23	0.015910	0.015910	0.860148
17	City_C15	0.015474	0.015474	0.875622
18	City_C16	0.015304	0.015304	0.890926
5	Income	0.015211	0.015211	0.906137
4	Education_Level	0.014702	0.014702	0.920839
32	City_C29	0.014541	0.014541	0.935380
34	City_C4	0.011906	0.011906	0.947286
24	City_C21	0.011488	0.011488	0.958774
35	City_C5	0.010834	0.010834	0.969608
13	City_C11	0.010697	0.010697	0.980305
16	City_C14	0.008838	0.008838	0.989143
36	City_C6	0.007656	0.007656	0.996799
19	City_C17	0.003201	0.003201	1.000000

[ ]:
```
# In Boosting we see that IncomeIncreased has the most importance followed by times reported in 2019 and 2020
# But here these have less weightage. Also, all the features seem equally important here.
```

[ ]:
```
#Now I will use this model to predict on the test data
```

[ ]:
```
#Getting the cut-off using the KS method
```

[170]:
```
cutoffs=np.linspace(0.01,0.99,99)

train_score=xgb_c.predict_proba(x_train)[:,1]
real=y_train
```

[171]:
```
KS_all=[]

for cutoff in cutoffs:

    predicted=(train_score>cutoff).astype(int)

    TP=((predicted==1) & (real==1)).sum()
    TN=((predicted==0) & (real==0)).sum()
    FP=((predicted==1) & (real==0)).sum()
```

```
    FN=((predicted==0) & (real==1)).sum()

    P=TP+FN
    N=TN+FP


    KS=(TP/P)-(FP/N)

    KS_all.append(KS)
```

[172]:
```
mycutoff=cutoffs[KS_all==max(KS_all)]
mycutoff
```

[172]:
```
array([0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67,
       0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78,
       0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89,
       0.9 , 0.91, 0.92, 0.93])
```

[173]:
```
#Again we see multiple cutoffs. I am picking the cutoff as 0.57
mycutoff=0.57
```

[ ]:
```
#Now predicting on the test data
```

[174]:
```
test_score=xgb_c.predict_proba(x_test)[:,1]
test_classes=(test_score>mycutoff).astype(int)
```

[175]:
```
y_test.shape,test_classes.shape
```

[175]:
```
((239,), (239,))
```

[176]:
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, test_classes)
print(f"Accuracy of the model: {accuracy:.4f}")
```

```
Accuracy of the model: 0.9205
```

[ ]:
```
#From this we can see that the model has generalized well.
#Earlier with hyperparameter tuning on the train data we had got the accuracy␣
↪of 0.938
#Now on the testing data we are getting 0.9205, which is a good generalization
```

[ ]:
```
#let me now check the other performance metrics
```

[178]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, confusion_matrix
# Calculate metrics
```

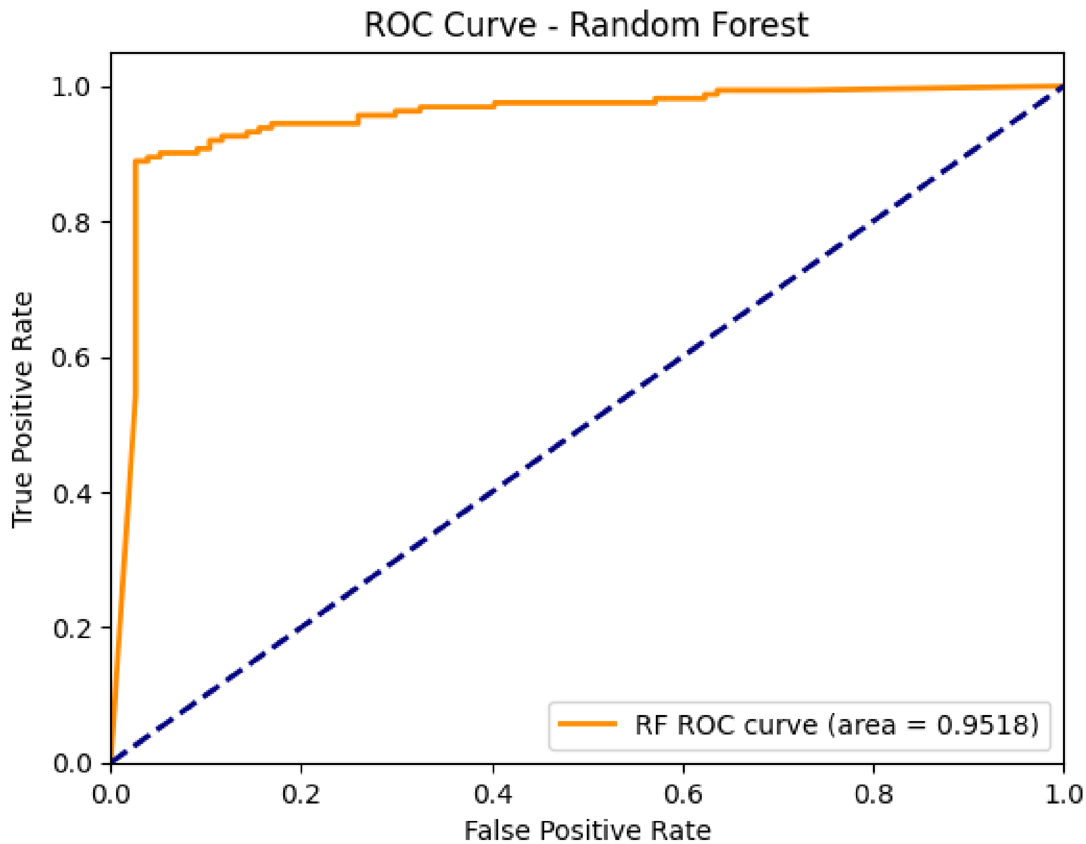
```python
accuracy = accuracy_score(y_test, test_classes)
precision = precision_score(y_test, test_classes, zero_division=0)
recall = recall_score(y_test, test_classes, zero_division=0)
f1 = f1_score(y_test, test_classes, zero_division=0)
conf_matrix = confusion_matrix(y_test, test_classes)

print(f"\nAccuracy: {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("\nConfusion Matrix Interpretation:")
print("True Negatives (TN):", conf_matrix[0, 0])
print("False Positives (FP):", conf_matrix[0, 1])
print("False Negatives (FN):", conf_matrix[1, 0])
print("True Positives (TP):", conf_matrix[1, 1])
```

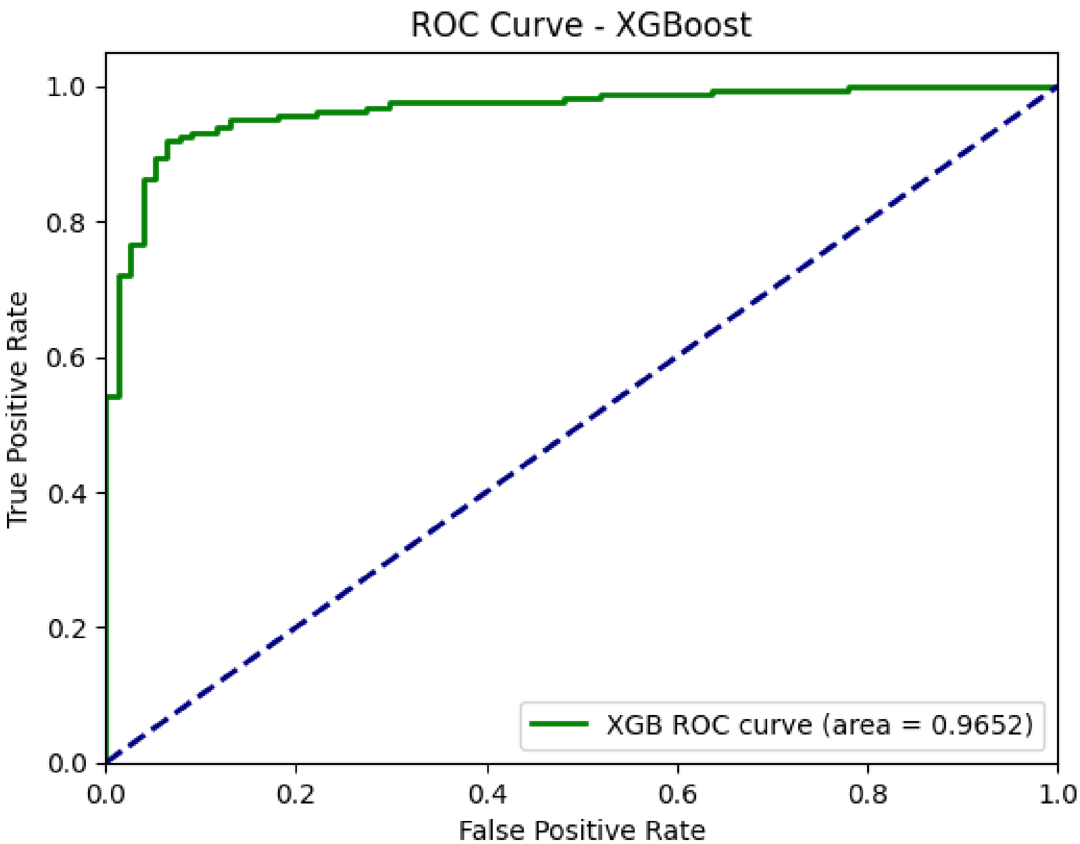
```
Accuracy: 0.9205
Precision: 0.9333
Recall: 0.9506
F1-Score: 0.9419

Confusion Matrix:
[[ 66  11]
 [  8 154]]

Confusion Matrix Interpretation:
True Negatives (TN): 66
False Positives (FP): 11
False Negatives (FN): 8
True Positives (TP): 154
```

[179]:

```python
#ROC-AUC curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

y_proba_rf = rf_best_model.predict_proba(x_test)[:, 1]
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_proba_rf)
roc_auc_rf = auc(fpr_rf, tpr_rf)

plt.figure()
plt.plot(fpr_rf, tpr_rf, color='darkorange', lw=2, label='RF ROC curve (area =␣
↪%0.4f)' % roc_auc_rf)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
```

```
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Random Forest')
plt.legend(loc="lower right")
plt.show()

y_proba_xgb = xgb_c.predict_proba(x_test)[:, 1]
fpr_xgb, tpr_xgb, _ = roc_curve(y_test, y_proba_xgb)
roc_auc_xgb = auc(fpr_xgb, tpr_xgb)

plt.figure()
plt.plot(fpr_xgb, tpr_xgb, color='green', lw=2, label='XGB ROC curve (area = %0.
 ↪4f)' % roc_auc_xgb)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - XGBoost')
plt.legend(loc="lower right")
plt.show()
```


ROC Curve - Random Forest
True Positive Rate
False Positive Rate
RF ROC curve (area = 0.9518)
0.0
0.2
0.4
0.6
0.8
1.0



[ ]:
```
# The ROC AUC score for XGBoost (0.9652) is higher than that of Random Forest␣
↪(0.9518), indicating that XGBoost has slightly better
# capability in distinguishing between the positive and negative classes.

# Both models demonstrate strong classification performance, as their ROC␣
↪curves are close to the top-left corner of the plot,
# showing high sensitivity (TPR) even with low false positive rates (FPR).

# The steep initial rise in TPR for both models suggests that they can␣
↪correctly identify a large proportion of positives with
# minimal false alarms.

# After reaching around 0.9 TPR, the curve begins to plateau, which implies␣
↪that additional gains in recall come at the cost of
# substantially higher FPR, reflecting a trade-off in performance.

# This pattern is typical of high-performing classifiers and suggests both␣
↪models are robust, but XGBoost offers slightly better
# separation between the classes across all thresholds.
```

[180]:
```python
from sklearn.metrics import classification_report, confusion_matrix

# For Random Forest
y_pred_rf = rf_best_model.predict(x_test)
print("Random Forest:\n")
print(confusion_matrix(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))

# For XGBoost
y_pred_xgb = xgb_c.predict(x_test)
print("XGBoost:\n")
print(confusion_matrix(y_test, y_pred_xgb))
print(classification_report(y_test, y_pred_xgb))
```

```
Random Forest:

[[ 69   8]
 [ 13 149]]
              precision    recall  f1-score   support

           0       0.84      0.90      0.87        77
           1       0.95      0.92      0.93       162

    accuracy                           0.91       239
   macro avg       0.90      0.91      0.90       239
weighted avg       0.91      0.91      0.91       239

XGBoost:

[[ 65  12]
 [  8 154]]
              precision    recall  f1-score   support

           0       0.89      0.84      0.87        77
           1       0.93      0.95      0.94       162

    accuracy                           0.92       239
   macro avg       0.91      0.90      0.90       239
weighted avg       0.92      0.92      0.92       239
```

[ ]:
```python
# Random Forest:

# The model achieves high precision (0.95) and recall (0.92) for class 1,␣
↪indicating strong performance in correctly identifying positive cases.
```

```
# Class 0 performance is slightly lower, especially in precision (0.84),
↪showing some false positives.

# The overall accuracy is 91%, with balanced macro and weighted averages around
↪90-91%.

# Suitable when minimizing false negatives is a priority, though slightly prone
↪to false positives on class 0.


# XGBoost:

# Shows improved recall (0.95) and F1-score (0.94) for class 1, outperforming
↪Random Forest in identifying positives.

# Precision for class 0 improves (0.89 vs. 0.84 in RF), indicating better
↪handling of false positives.

# Achieves higher overall accuracy at 92% with better balance across classes.

# Offers a more robust and balanced classification, especially in scenarios
↪where both false positives and false negatives matter.
```

0.0.2 Actionable Insights & Recommendations

1. **High Driver Attrition Rate**
   ~67% of drivers have quit, which is a serious concern. This high attrition rate can severely affect operational continuity and revenue.

2. **Monitor Reporting Frequency**
   `Reports_2020` and `Reports_2019` together contribute over 60% to the model's decision. Drivers who report more frequently are significantly less likely to quit. Encourage regular reporting through incentives or app notifications.

3. **Increase Driver Engagement via Tenure-Linked Benefits**
   `Tenure` is the third most important feature (~13%). Drivers who stay longer are less likely to quit. Offer retention bonuses, loyalty programs, or recognition based on tenure milestones.

4. **Targeted Interventions Based on Income**
   Higher `Income` is associated with lower quit probability. Create income-boosting programs for low-earning drivers (e.g., peak-time bonuses or targeted ride allocations).

5. **Support High Business Value Drivers**
   `Total Business Value` also shows a negative correlation with quitting. Protect and nurture top-performing drivers through premium support and higher visibility on the platform.

6. **Age-Based Programs Aren't Effective Alone**
   `Age` does not consistently correlate with quitting. Avoid generalized age-based policies; instead, segment drivers by behavior or performance.

7. **City-Based Effects Are Minimal**
   One-hot encoded city variables contributed little to model performance. Focus strategy on behavioral and economic features rather than geographic ones.

8. **Improve Driver Support & Experience**
   Since reporting frequency is highly predictive, use it as a proxy for engagement and potential dissatisfaction. Introduce proactive check-ins and regular feedback mechanisms for less active drivers.

9. **Utilize Predictive Model for Early Interventions**
   Deploy the trained model to flag high-risk drivers in real time and trigger automated retention workflows, such as support calls, bonus offers, or feedback surveys.

10. **Conduct Deeper Analysis on Remaining Drivers**
    Analyze why 33% of drivers chose to stay despite similar conditions. Identify traits or patterns that can be amplified in broader driver policies.