```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve
from sklearn.metrics import precision_recall_curve
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.preprocessing import MinMaxScaler
from datetime import datetime
from statsmodels.stats.outliers_influence import variance_inflation_factor
import warnings
warnings.filterwarnings("ignore")
```

```
!gdown "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/002/492/original/ola_driver_scaler.csv"
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/002/492/original/ola_driver_scaler.csv
To: /content/ola_driver_scaler.csv
100% 1.13M/1.13M [00:00<00:00, 9.74MB/s]
```

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

```
ola.head()
```

	Unnamed: 0	MMM-YY	Driver_ID	Age	Gender	City	Education_Level	Income	Dateofjoining	LastWorkingDate	Joining Designation	Grade	Total Business Value	Quarterly Rating
0	0	01/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	2381060	2
1	1	02/01/19	1	28.0	0.0	C23	2	57387	24/12/18	NaN	1	1	-665480	2
2	2	03/01/19	1	28.0	0.0	C23	2	57387	24/12/18	03/11/19	1	1	0	2
3	3	11/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	0	1
4	4	12/01/20	2	31.0	0.0	C7	2	67016	11/06/20	NaN	2	2	0	1

```
#EXPLORATORY DATA ANALYSIS
```

```
print('Rows in the ola dataset: ',ola.shape[0])
print('Columns in the ola dataset: ',ola.shape[1])
```

```
Rows in the ola dataset:  19104
Columns in the ola dataset:  14
```

```
ola.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
```

```
# Reporting Date (MMMM-YY):
# Type: Date-time
# Represents the monthly reporting date.

# Age:
# Type: Numerical
# Denotes the age of the employee.

# Gender:
# Type: Categorical
# Male is represented by 0, Female by 1.

# City:
# Type: Categorical
# City code of the employee.

# Education Level:
# Type: Categorical
# 0 for 10+, 1 for 12+, 2 for graduate.

# Income:
# Type: Numerical
# Monthly average income of the employee.

# Date Of Joining:
# Type: Date-time
# Signifies the joining date for the employee.

# Last Working Date:
# Type: Date-time (to be converted to categorical)
# Target feature; indicates the last date of working for the employee.

# Joining Designation:
# Type: Categorical (Ordinal)
# Designation of the employee at the time of joining.

# Grade:
# Type: Categorical (Ordinal)
# Grade of the employee at the time of reporting.

# Total Business Value:
# Type: Numerical
# Total business value acquired by the employee in a month (negative values indicate cancellations or adjustments).

# Quarterly Rating:
# Type: Categorical (Ordinal)
# Quarterly rating of the employee, with higher values indicating better performance. Ratings range from 1 to 5.
```

```
ola.describe()
```

	Unnamed: 0	Driver_ID	Age	Gender	Education_Level	Income	Joining Designation	Grade	Bu:
count	19104.000000	19104.000000	19043.000000	19052.000000	19104.000000	19104.000000	19104.000000	19104.000000	1.9104
mean	9551.500000	1415.591133	34.668435	0.418749	1.021671	65652.025126	1.690536	2.252670	5.7166
std	5514.994107	810.705321	6.257912	0.493367	0.800167	30914.515344	0.836984	1.026512	1.1283
min	0.000000	1.000000	21.000000	0.000000	0.000000	10747.000000	1.000000	1.000000	-6.0000
25%	4775.750000	710.000000	30.000000	0.000000	0.000000	42383.000000	1.000000	1.000000	0.0000
50%	9551.500000	1417.000000	34.000000	0.000000	1.000000	60087.000000	1.000000	2.000000	2.5000
75%	14327.250000	2137.000000	39.000000	1.000000	2.000000	83969.000000	2.000000	3.000000	6.9970
max	19103.000000	2788.000000	58.000000	1.000000	2.000000	188418.000000	5.000000	5.000000	3.3747

```
ola.describe(include='object')
```

	MMM-YY	City	Dateofjoining	LastWorkingDate
count	19104	19104	19104	1616
unique	24	29	869	493
top	01/01/19	C20	23/07/15	29/07/20
freq	1022	1008	192	70

```
# I am dropping the 'Unnamed' column as it has the highest correlation with the 'driver_id' column, and they contain the same information.
```

```
ola.drop(columns='Unnamed: 0',axis=1,inplace=True)
```

```
ola.nunique()
```

```
MMM-YY                  24
Driver_ID             2381
Age                     36
Gender                   2
City                    29
Education_Level          3
Income                2383
Dateofjoining          869
```

```
LastWorkingDate           493
Joining Designation         5
Grade                       5
Total Business Value    10181
Quarterly Rating            4
dtype: int64
```

```
ola.isna().sum()
```

```
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
```

```
#DATA PROCESSING AND FEATURE ENGINEERING
```

```
ola1 = ola.copy(deep=True)
```

```
# I am currently in the process of creating a target variable by introducing a new column called 'target,' where a value of 1 is
# assigned to drivers who have left the company, based on the presence of their last working day.
```

```
first =  (ola1.groupby('Driver_ID').agg({'LastWorkingDate':'last'})['LastWorkingDate'].isna()).reset_index()
first['LastWorkingDate'].replace({True:1,False:0},inplace=True)
first.rename(columns={'LastWorkingDate':'target'},inplace=True)
first.head()
```

```
# I am creating a new column that indicates whether the quarterly rating has increased for each driver, assigning the value 1 to
# those drivers who have experienced an improvement in their quarterly rating

QR1 =  (ola1.groupby('Driver_ID').agg({'Quarterly Rating':'first'})['Quarterly Rating']).reset_index()
QR2 = (ola1.groupby('Driver_ID').agg({'Quarterly Rating':'last'})['Quarterly Rating']).reset_index()
```

```
QR1.shape,QR2.shape
```

```
((2381, 2), (2381, 2))
```

```
QR1.isna().sum(),QR2.isna().sum()
```

```
(Driver_ID           0
 Quarterly Rating    0
 dtype: int64,
 Driver_ID           0
 Quarterly Rating    0
 dtype: int64)
```

```
first = first.merge(QR1,on='Driver_ID')
first = first.merge(QR2,on='Driver_ID')
```

```
first.head()
```

	Driver_ID	target	Quarterly Rating_x	Quarterly Rating_y
0	1	0	2	2
1	2	1	1	1
2	4	0	1	1
3	5	0	1	1
4	6	1	1	2

```
first['Promotion'] = np.where(first['Quarterly Rating_x'] == first['Quarterly Rating_y'], 0,1)
```

```
# Create a column which tells whether the monthly income has increased for that driver -
# for those whose monthly income has increased we assign the value 1
incm1 =  (ola1.groupby('Driver_ID').agg({'Income':'first'})['Income']).reset_index()
incm2 = (ola1.groupby('Driver_ID').agg({'Income':'last'})['Income']).reset_index()
```

```
incm1.shape,incm2.shape
```

```
((2381, 2), (2381, 2))
```

```
incm1.isna().sum(),incm2.isna().sum()
```

```
(Driver_ID    0
 Income       0
 dtype: int64,
 Driver_ID    0
 Income       0
 dtype: int64)
```

```
first = first.merge(incm1,on='Driver_ID')
first = first.merge(incm2,on='Driver_ID')
```

```
first.head()
```

	Driver_ID	target	Quarterly Rating_x	Quarterly Rating_y	Promotion	Income_x	Income_y
0	1	0	2	2	0	57387	57387
1	2	1	1	1	0	67016	67016
2	4	0	1	1	0	65603	65603
3	5	0	1	1	0	46368	46368
4	6	1	1	2	1	78728	78728

```
first['Raise'] = np.where(first['Income_x'] == first['Income_y'], 0,1)
```

```
first.head()
```

	Driver_ID	target	Quarterly Rating_x	Quarterly Rating_y	Promotion	Income_x	Income_y	Raise
0	1	0	2	2	0	57387	57387	0
1	2	1	1	1	0	67016	67016	0
2	4	0	1	1	0	65603	65603	0
3	5	0	1	1	0	46368	46368	0
4	6	1	1	2	1	78728	78728	0

```
first.tail()
```

	Driver_ID	target	Quarterly Rating_x	Quarterly Rating_y	Promotion	Income_x	Income_y	Raise
2376	2784	1	3	4	1	82815	82815	0
2377	2785	0	1	1	0	12105	12105	0
2378	2786	0	2	1	1	35370	35370	0
2379	2787	0	2	1	1	69498	69498	0
2380	2788	1	1	2	1	70254	70254	0

```
first = first[['Driver_ID','target','Raise','Promotion']]
```

```
first.head()
```

	Driver_ID	target	Raise	Promotion
0	1	0	0	0
1	2	1	0	0
2	4	0	0	0
3	5	0	0	0
4	6	1	0	1

```
functions = {'MMM-YY':'count',
             'Driver_ID':'first',
             'Age':'max',
             'Gender':'last',
            'City':'last',
             'Education_Level':'last',
             'Dateofjoining':'first',
            'LastWorkingDate':'last',
             'Grade':'last',
             'Total Business Value':'sum',
            'Income':'sum',
             'Dateofjoining':'first',
             'LastWorkingDate':'last',
            'Joining Designation':'last',
             'Grade':'last',
             'Quarterly Rating':'first'}
ola1 = ola1.groupby([ola1['Driver_ID']]).aggregate(functions)
ola1['month'] = pd.to_datetime(ola['Dateofjoining']).dt.month
ola1['year'] = pd.DatetimeIndex(ola1['Dateofjoining']).year
ola1.rename(columns={'MMM-YY':'Reportings'},inplace=True)


ola1.reset_index(drop=True, inplace=True)
ola1 = ola1.merge(first,on='Driver_ID')
ola1.head()
```

	Reportings	Driver_ID	Age	Gender	City	Education_Level	Dateofjoining	LastWorkingDate	Grade	Total Business Value	Income	Joining Designation	Quarterly Rating	month	yea
0	3	1	28.0	0.0	C23	2	24/12/18	03/11/19	1	1715580	172161	1	2	12	20
1	2	2	31.0	0.0	C7	2	11/06/20	None	2	0	134032	2	1	12	202
2	5	4	43.0	0.0	C13	2	12/07/19	27/04/20	2	350000	328015	2	1	11	20
3	3	5	29.0	0.0	C9	0	01/09/19	03/07/19	1	120360	139104	1	1	12	20
4	5	6	31.0	1.0	C11	1	31/07/20	None	3	1265000	393640	3	1	12	202

```
import regex
ola1['Age'] = ola1['Age'].astype('int64')
ola1['Cities'] =ola1['City'].astype('str').str.extractall('(\d+)').unstack().fillna('').sum(axis=1).astype(int)


ola1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2381 entries, 0 to 2380
Data columns (total 19 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Reportings           2381 non-null   int64  
 1   Driver_ID            2381 non-null   int64  
 2   Age                  2381 non-null   int64  
 3   Gender               2381 non-null   float64
 4   City                 2381 non-null   object 
 5   Education_Level      2381 non-null   int64  
 6   Dateofjoining        2381 non-null   object 
 7   LastWorkingDate      1616 non-null   object 
 8   Grade                2381 non-null   int64  
 9   Total Business Value 2381 non-null   int64  
 10  Income               2381 non-null   int64  
 11  Joining Designation  2381 non-null   int64  
 12  Quarterly Rating     2381 non-null   int64  
 13  month                2381 non-null   int64  
 14  year                 2381 non-null   int64  
 15  target               2381 non-null   int64  
 16  Raise                2381 non-null   int64  
 17  Promotion            2381 non-null   int64  
 18  Cities               2381 non-null   int64  
dtypes: float64(1), int64(15), object(3)
memory usage: 372.0+ KB
```

```
ola1.drop(columns=['Dateofjoining','LastWorkingDate','City'],axis=1,inplace=True)
ola1['Gender'].replace({'M':0,'F':1},inplace=True)
ola1['Gender'] = ola1['Gender'].astype('int64')
```

```
ola1.head()
```

	Reportings	Driver_ID	Age	Gender	Education_Level	Grade	Total Business Value	Income	Joining Designation	Quarterly Rating	month	year	target	Raise	Promotion	Cities
0	3	1	28	0	2	1	1715580	172161	1	2	12	2018	0	0	0	23
1	2	2	31	0	2	2	0	134032	2	1	12	2020	1	0	0	7
2	5	4	43	0	2	2	350000	328015	2	1	11	2019	0	0	0	13
3	3	5	29	0	0	1	120360	139104	1	1	12	2019	0	0	0	9
4	5	6	31	1	1	3	1265000	393640	3	1	12	2020	1	0	1	11

```
sum(ola1.isna().sum())
```

```
0
```

```
ola1.describe().T
```

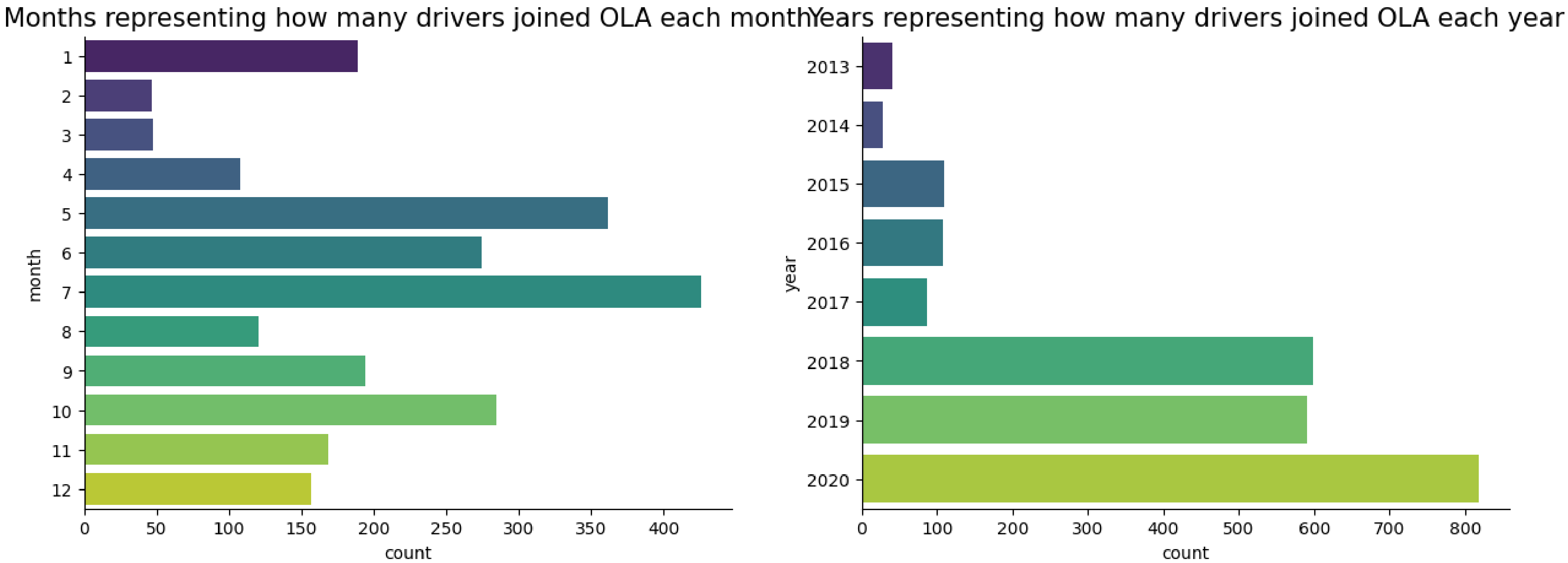
	count	mean	std	min	25%	50%	75%	max
Reportings	2381.0	8.023520e+00	6.783590e+00	1.0	3.0	5.0	10.0	24.0
Driver_ID	2381.0	1.397559e+03	8.061616e+02	1.0	695.0	1400.0	2100.0	2788.0
Age	2381.0	3.366317e+01	5.983375e+00	21.0	29.0	33.0	37.0	58.0
Gender	2381.0	4.103318e-01	4.919972e-01	0.0	0.0	0.0	1.0	1.0
Education_Level	2381.0	1.007560e+00	8.162900e-01	0.0	0.0	1.0	2.0	2.0
Grade	2381.0	2.096598e+00	9.415218e-01	1.0	1.0	2.0	3.0	5.0
Total Business Value	2381.0	4.586742e+06	9.127115e+06	-1385530.0	0.0	817680.0	4173650.0	95331060.0
Income	2381.0	5.267603e+05	6.231633e+05	10883.0	139895.0	292980.0	651456.0	4522032.0
Joining Designation	2381.0	1.820244e+00	8.414334e-01	1.0	1.0	2.0	2.0	5.0
Quarterly Rating	2381.0	1.486350e+00	8.343483e-01	1.0	1.0	1.0	2.0	4.0
month	2381.0	6.975220e+00	3.007801e+00	1.0	5.0	7.0	10.0	12.0
year	2381.0	2.018536e+03	1.609597e+00	2013.0	2018.0	2019.0	2020.0	2020.0
target	2381.0	3.212936e-01	4.670713e-01	0.0	0.0	0.0	1.0	1.0
Raise	2381.0	1.805964e-02	1.331951e-01	0.0	0.0	0.0	0.0	1.0
Promotion	2381.0	3.427131e-01	4.747162e-01	0.0	0.0	0.0	1.0	1.0
Cities	2381.0	1.533557e+01	8.371843e+00	1.0	8.0	15.0	22.0	29.0

```
#DATA VISUALIZATION
```

```
#Univariate
```

```
# figure1
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,1)
sns.countplot(y=ola1.month,palette='viridis')
plt.title('Months representing how many drivers joined OLA each month',fontname='Franklin Gothic Medium', fontsize=15)


ax = fig.add_subplot(1,2,2)
sns.countplot(y=ola1.year,palette='viridis')
plt.title('Years representing how many drivers joined OLA each year',fontname='Franklin Gothic Medium', fontsize=15)
sns.despine()
plt.show()
```

```
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
```



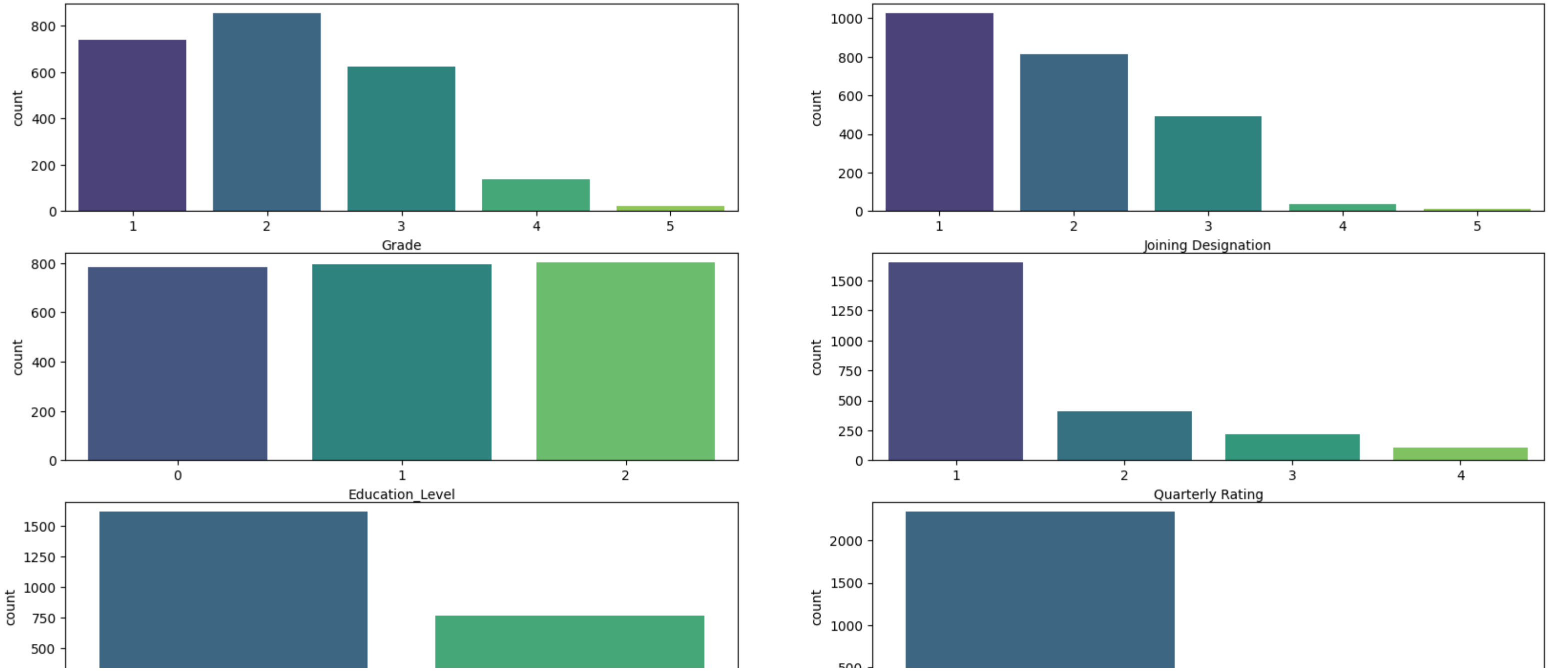
```
# #Observation

# #July Peaks in Driver Joining:

# In the span of 8 years, the month of July experienced the highest influx of new drivers joining OLA, signifying a peak in recruitment during that montl
# February and March Record Lowest Driver Joining:

# February and March stood out as the months with the least number of drivers joining OLA, indicating a period of comparatively lower recruitment
# activity during these months.
# Significant Increase in Driver Joining Post-2017:

# There has been a substantial boost of approximately 500% in the number of drivers joining OLA after the year 2017. This suggests a significant surge
#  in recruitment efforts or increased interest in joining OLA's driver network starting from 2018 onward.


# figure2
fig = plt.figure(figsize=(15,3))
ax = fig.add_subplot(121)
sns.countplot(x=ola1.Age,palette='viridis',width=0.8)
plt.title('Age of Drivers',fontname='Franklin Gothic Medium', fontsize=15)
plt.xticks(rotation=90)

ax = fig.add_subplot(122)
a = pd.cut(ola1.Age,bins=[11,21,31,41,51,61],labels=['11,21','21-31','31-41','41-51','51-61'])
sns.countplot(x=a,palette='viridis')
plt.title('Groupwise Age count of Drivers',fontname='Franklin Gothic Medium', fontsize=15)
sns.despine()
plt.show()
```

```
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
```

```
# figure3
fig = plt.figure(figsize=(22,5))
ax = fig.add_subplot(121)
sns.countplot(x=ola1.Cities,palette='viridis',width=0.6)
plt.title('Cities alloted to Drivers',fontname='Franklin Gothic Medium', fontsize=13)
plt.xticks(rotation=90)

ax = fig.add_subplot(122)
sns.countplot(x=ola1.Reportings,palette='viridis',width=0.6)
plt.title('Number of Reportings of Drivers',fontname='Franklin Gothic Medium', fontsize=13)

# ax = fig.add_subplot(133)
# sns.countplot(x=ola1.Education_Level,palette='viridis')
# plt.title('Educational Level of Drivers',fontname='Franklin Gothic Medium', fontsize=13)
sns.despine()
plt.show()
```

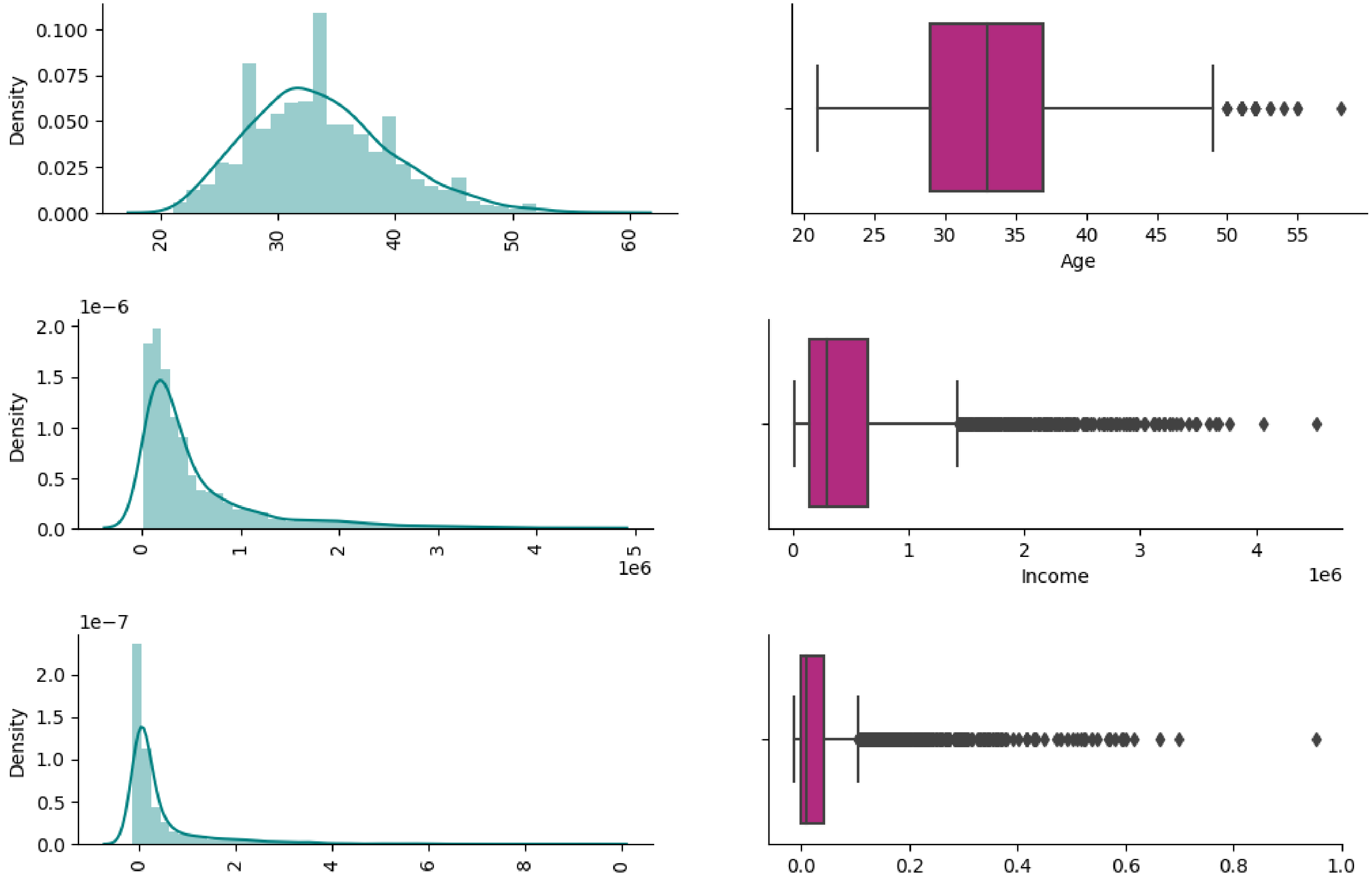
```
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
```

```
# figure4
plt.figure(figsize=(20,13))
plt.subplot(4,2,1)
sns.countplot(x=ola1.Grade,palette='viridis')
plt.subplot(4,2,2)
sns.countplot(x=ola1['Joining Designation'],palette='viridis')
plt.subplot(4,2,3)
sns.countplot(x=ola1.Education_Level,palette='viridis')
plt.subplot(4,2,4)
sns.countplot(x=ola1['Quarterly Rating'],palette='viridis')
plt.subplot(4,2,5)
sns.countplot(x=ola1.target,palette='viridis')
plt.subplot(4,2,6)
sns.countplot(x=ola1.Raise,palette='viridis')
plt.subplot(4,2,7)
sns.countplot(x=ola1.Promotion,palette='viridis')
plt.subplot(4,2,8)
sns.countplot(x=ola1.Gender,palette='viridis')
plt.show()
```



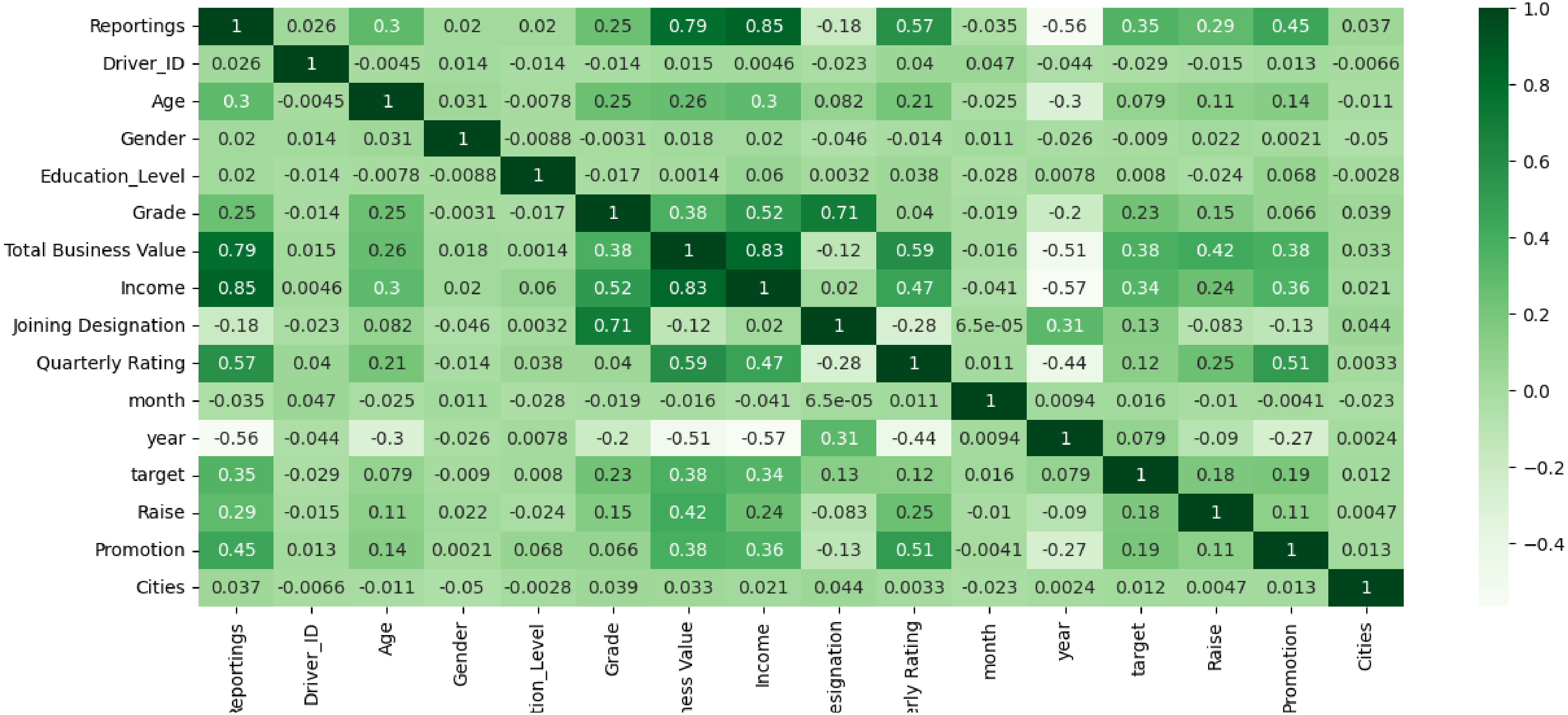

```
# Between 21 years(min age) to 58(max age) years of age, maximum number of drivers are 32 years o
# meanwhile the age group between 31-41 years of age receives the maximum number of drivers.
#  58.9% of the Drivers are male.
#  City C20 has been used by the most of the drivers.
#  There are 3 Education levels and all of them alomst have the equal distribution of Drivers.
#  Grade 2 has been received by most of the Drivers and then the count of grade keeps on falling
```

```
a =ola1[['Age','Income','Total Business Value']]
for i in a:
    plt.figure(figsize=(12,2))
    plt.subplot(121)
    sns.distplot(x=ola1[i],color='teal')
    plt.title('')
    plt.xticks(rotation=90)

    plt.subplot(122)
    sns.boxplot(x=ola1[i],color='mediumvioletred')
    plt.title('')
    sns.despine()
    plt.show()
```




```
#Bivariate and multivariate
```

```
corr = ola1.corr()
plt.figure(figsize=(15,6))
sns.heatmap(corr,annot=True,cmap='Greens')
plt.show()
```


Reporting s 1 0.026 0.3 0.02 0.02 0.25 0.79 0.85 -0.18 0.57 -0.035 -0.56 0.35 0.29 0.45 0.037
Driver_ID 0.026 1 -0.0045 0.014 -0.014 -0.014 0.015 0.0046 -0.023 0.04 0.047 -0.044 -0.029 -0.015 0.013 -0.0066
Age 0.3 -0.0045 1 0.031 -0.0078 0.25 0.26 0.3 0.082 0.21 -0.025 -0.3 0.079 0.11 0.14 -0.011
Gender 0.02 0.014 0.031 1 -0.0088 -0.0031 0.018 0.02 -0.046 -0.014 0.011 -0.026 -0.009 0.022 0.0021 -0.05
Education_Level 0.02 -0.014 -0.0078 -0.0088 1 -0.017 0.0014 0.06 0.0032 0.038 -0.028 0.0078 0.008 -0.024 0.068 -0.0028
Grade 0.25 -0.014 0.25 -0.0031 -0.017 1 0.38 0.52 0.71 0.04 -0.019 -0.2 0.23 0.15 0.066 0.039
Total Business Value 0.79 0.015 0.26 0.018 0.0014 0.38 1 0.83 -0.12 0.59 -0.016 -0.51 0.38 0.42 0.38 0.033
Income 0.85 0.0046 0.3 0.02 0.06 0.52 0.83 1 0.02 0.47 -0.041 -0.57 0.34 0.24 0.36 0.021
Joining Designation -0.18 -0.023 0.082 -0.046 0.0032 0.71 -0.12 0.02 1 -0.28 6.5e-05 0.31 0.13 -0.083 -0.13 0.044
Quarterly Rating 0.57 0.04 0.21 -0.014 0.038 0.04 0.59 0.47 -0.28 1 0.011 -0.44 0.12 0.25 0.51 0.0033
month -0.035 0.047 -0.025 0.011 -0.028 -0.019 -0.016 -0.041 6.5e-05 0.011 1 0.0094 0.016 -0.01 -0.0041 -0.023
year -0.56 -0.044 -0.3 -0.026 0.0078 -0.2 -0.51 -0.57 0.31 -0.44 0.0094 1 0.079 -0.09 -0.27 0.0024
target 0.35 -0.029 0.079 -0.009 0.008 0.23 0.38 0.34 0.13 0.12 0.016 0.079 1 0.18 0.19 0.012
Raise 0.29 -0.015 0.11 0.022 -0.024 0.15 0.42 0.24 -0.083 0.25 -0.01 -0.09 0.18 1 0.11 0.0047
Promotion 0.45 0.013 0.14 0.0021 0.068 0.066 0.38 0.36 -0.13 0.51 -0.0041 -0.27 0.19 0.11 1 0.013
Cities 0.037 -0.0066 -0.011 -0.05 -0.0028 0.039 0.033 0.021 0.044 0.0033 -0.023 0.0024 0.012 0.0047 0.013 1
Driver_ID Age Gender Grade Income month year target Raise Promotion Cities
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

```
fig = plt.figure(figsize=(22,5))
ax = fig.add_subplot(1,3,1)
grouped_months = ola1.groupby(['month'])['Reportings'].count().reset_index()
sns.barplot(data=grouped_months,x='month',y='Reportings',palette='viridis')
plt.title('Reportings of Driver every month',fontname='Franklin Gothic Medium', fontsize=15)

ax = fig.add_subplot(1,3,2)
grouped_years = ola1.groupby(['year'])['Reportings'].count().reset_index()
sns.barplot(x='year', y='Reportings', data=grouped_years,palette='viridis')
plt.title('Reportings of Driver every year',fontname='Franklin Gothic Medium', fontsize=15)

ax = fig.add_subplot(1,3,3)
grouped_gender = ola1.groupby('Gender')['Reportings'].sum().reset_index()
grouped_gender['Reportings'] =(grouped_gender['Reportings']/sum(ola1.Reportings)*100).round(2)
sns.barplot(x=grouped_gender['Gender'],y= grouped_gender['Reportings'],palette='viridis')
plt.title('Reportings of Driver by Gender in percentage',fontname='Franklin Gothic Medium', fontsize=15)
sns.despine()
sns.despine()
plt.show()
```

```
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
```

```
grouped_gender
```

	Gender	Reportings
0	0	58.12
1	1	41.88

```
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
```

```
fig = plt.figure(figsize=(15,4))
ax = fig.add_subplot(1,2,1)
sns.lineplot(x=ola1.Age,y=ola1.Grade,hue=ola1.target,palette='viridis')
plt.title('Age of Drivers in context with Grades and Target variable',fontname='Franklin Gothic Medium', fontsize=15)

ax = fig.add_subplot(1,2,2)
sns.barplot(data=ola1, x="Joining Designation", y="Grade",palette='viridis',hue='target')
plt.title('Joining Designation with Grade',fontname='Franklin Gothic Medium', fontsize=15)
sns.despine()
plt.show()
```

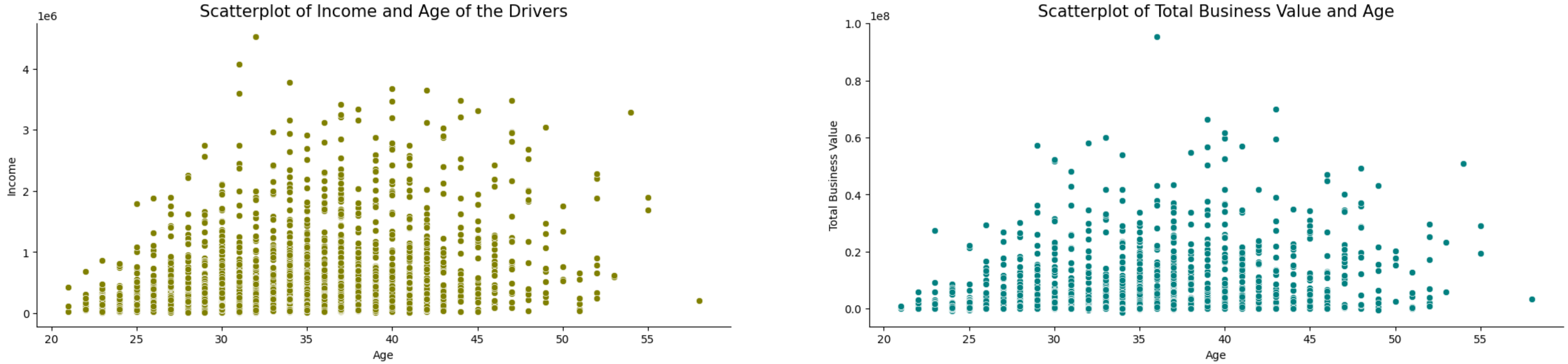
```
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
```

```
# figure7
plt.figure(figsize=(25,7))
plt.subplot(1,4,1)
sns.violinplot(y=ola1.Age,x=ola1['Joining Designation'],palette='viridis')
plt.title('Income of Drivers according to Designation',fontname='Franklin Gothic Medium', fontsize=15)
plt.subplot(1,4,2)
sns.violinplot(x=ola1.Grade,y=ola1.Income,palette='viridis')
plt.title('Income of Drivers according to Grade',fontname='Franklin Gothic Medium', fontsize=15)
plt.xticks(rotation=90)
plt.subplot(1,4,3)
sns.violinplot(x=ola1.Education_Level,y=ola1.Income,palette='viridis')
plt.title('Income of Drivers according to Their Education Level',fontname='Franklin Gothic Medium', fontsize=15)
plt.subplot(1,4,4)
sns.violinplot(x=ola1['Grade'],y=ola1["Quarterly Rating"],palette='viridis')
plt.title('Distribution of Quarterly Rating in context with Grade')
sns.despine()
sns.despine()
plt.show()
```

```
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
```

```
plt.figure(figsize=(25,5))
plt.subplot(1,2,1)
sns.scatterplot(x=ola1.Age,y=ola1.Income,color='olive')
plt.title('Scatterplot of Income and Age of the Drivers',fontname='Franklin Gothic Medium', fontsize=15)
plt.subplot(1,2,2)
sns.scatterplot(x=ola1.Age,y=ola1['Total Business Value'],color='teal')
plt.title('Scatterplot of Total Business Value and Age',fontname='Franklin Gothic Medium', fontsize=15)
sns.despine()
plt.show()
```

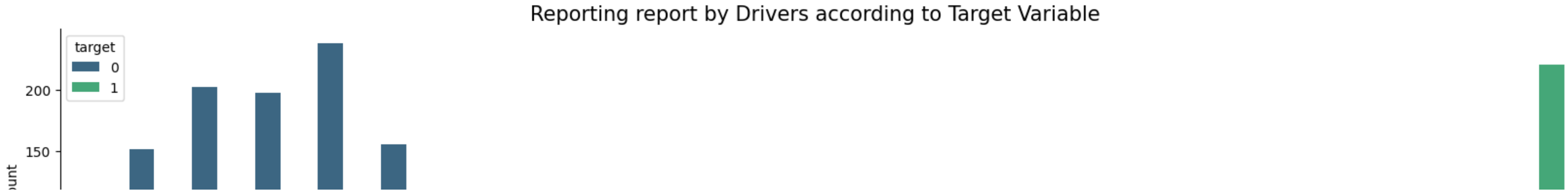
```
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
```



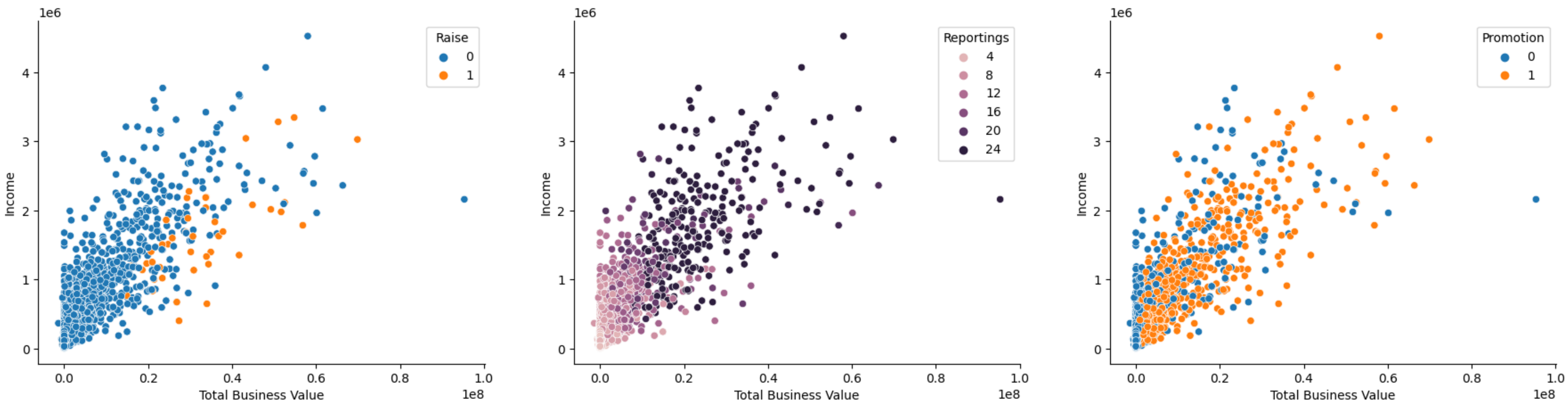
```
grouped_gender = ola1.groupby('Gender')['Income'].sum().reset_index()
grouped_education = ola1.groupby('Education_Level')['Income'].sum().reset_index()
grouped_grade = ola1.groupby('Grade')['Income'].sum().reset_index()
grouped_desig = ola1.groupby('Joining Designation')['Income'].sum().reset_index()
grouped_QR = ola1.groupby('Quarterly Rating')['Income'].sum().reset_index()
grouped_target = ola1.groupby('target')['Income'].sum().reset_index()
grouped_raise = ola1.groupby('Raise')['Income'].sum().reset_index()
grouped_promote = ola1.groupby('Promotion')['Income'].sum().reset_index()
```

```
plt.figure(figsize=(20,9))
plt.subplot(2,1,1)
sns.countplot(x=ola1['Reportings'],hue=ola1.target,palette='viridis')
plt.title('Reporting report by Drivers according to Target Variable',fontname='Franklin Gothic Medium', fontsize=15)
plt.subplot(2,1,2)
grouped_rating = ola1.groupby('Quarterly Rating')['Reportings'].count().reset_index()
sns.barplot(data = grouped_rating,y='Reportings',x='Quarterly Rating',palette='viridis')
plt.title('Reporting report by Drivers according to Quarterly Ratings Given',fontname='Franklin Gothic Medium', fontsize=15)
sns.despine()
plt.show()
```

```
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Franklin Gothic Medium' not found.
```

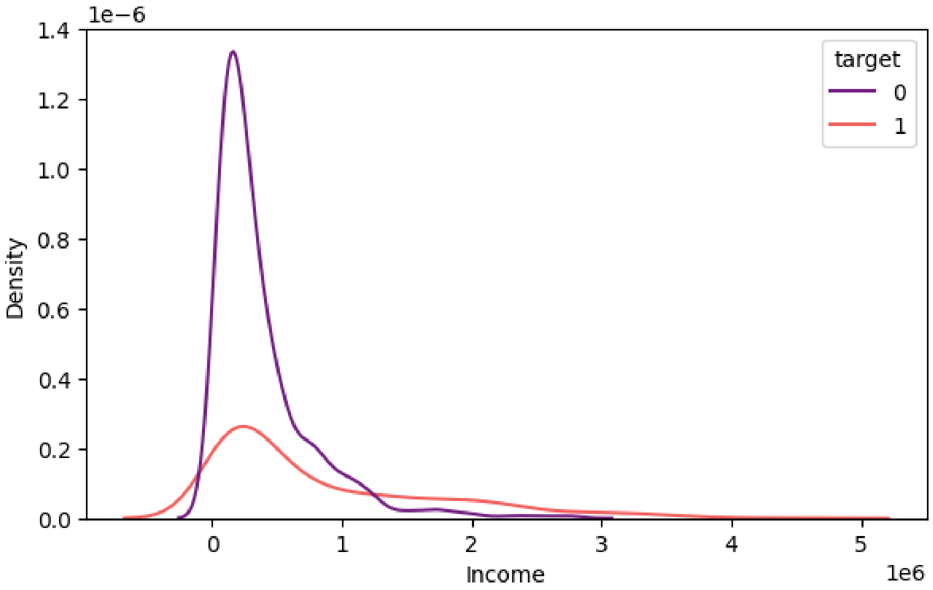


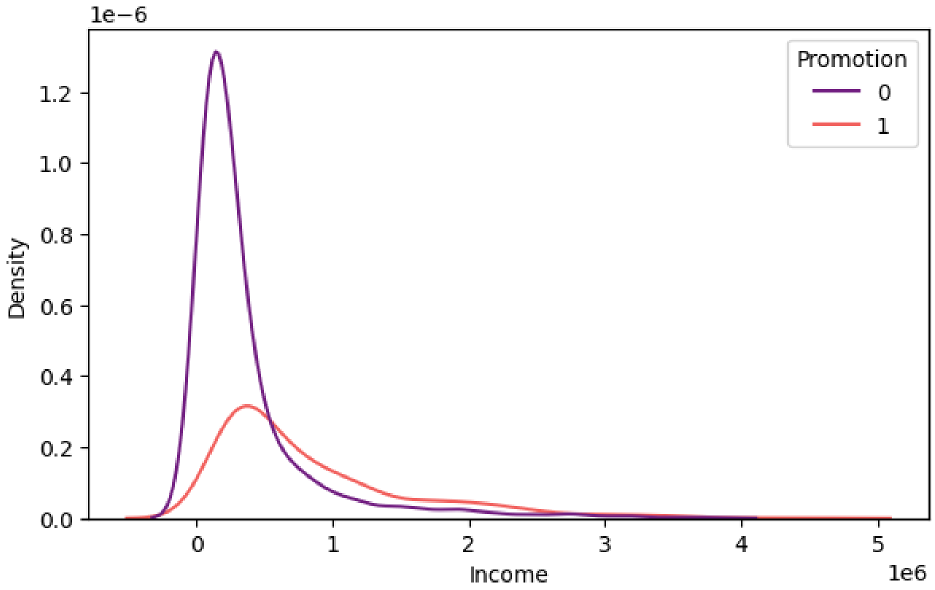

```
plt.figure(figsize=(22,5))
plt.subplot(1,3,1)
sns.scatterplot(x=ola1['Total Business Value'],y=ola1.Income,hue=ola1.Raise)
plt.subplot(1,3,2)
sns.scatterplot(x=ola1['Total Business Value'],y=ola1.Income,hue=ola1.Reportings)
plt.subplot(1,3,3)
sns.scatterplot(x=ola1['Total Business Value'],y=ola1.Income,hue=ola1.Promotion)
sns.despine()
plt.show()
```



```
# Gender Distribution:
# 57% male and 43% female employees.

# Education Level Distribution:
# Similar percentages for education levels 1 & 2.

# Employees without a Raise:
# 97.3% of employees did not receive a raise.

# Joining Designations:
# 43% joined at the lowest designation (1), 34% at level 2, 20% at level 3, and below 2% at higher levels.

# Current Designation Distribution:
# Majority (35%) at designation level 2, followed by level 1 (31%) and level 3 (26%).

# Promotion and Raise Stats:
# 54.6% received a promotion, 45.4% did not, and only 2.6% received a raise.

# Employee Growth:
# Employee count increases with both years and the number of reportings.

# City Association:
# Majority associated with city C20.

# Income and Age Relationship:
# Income generally increases with age, but a subtle decline is noticed after 45-50.

# Total Business Value and Age Relationship:
# Total Business Value increases with age, declining after 45.

# Income and Designation Relationship:
# Income decreases with higher designations, with about 4% of employees holding higher designations.

# Income and Grade Relationship:
# Median income for employees with higher grades is greater.

# Joining Designation and Grade Relationship:
# Joining designation increases with higher grades.

# Max Reporting Days:
# Maximum reporting days is 24.

# Quarterly Rating Distribution:
# About 55% of employee reportings have Quarterly Rating 1.

# Income and Total Business Value Increase:
# Number of reportings increases with higher income and total business value.
```

```
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
sns.kdeplot(x=ola1.Income,hue=ola1['target'],palette='magma')
plt.subplot(1,2,2)
sns.kdeplot(x=ola1.Income,hue=ola1['Promotion'],palette='magma')
plt.show()
```






```
#OUTLIER TREATMENT
```

```
ola1.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Reportings	2381.0	8.023520e+00	6.783590e+00	1.0	3.0	5.0	10.0	24.0
Driver_ID	2381.0	1.397559e+03	8.061616e+02	1.0	695.0	1400.0	2100.0	2788.0
Age	2381.0	3.366317e+01	5.983375e+00	21.0	29.0	33.0	37.0	58.0
Gender	2381.0	4.103318e-01	4.919972e-01	0.0	0.0	0.0	1.0	1.0
Education_Level	2381.0	1.007560e+00	8.162900e-01	0.0	0.0	1.0	2.0	2.0
Grade	2381.0	2.096598e+00	9.415218e-01	1.0	1.0	2.0	3.0	5.0
Total Business Value	2381.0	4.586742e+06	9.127115e+06	-1385530.0	0.0	817680.0	4173650.0	95331060.0
Income	2381.0	5.267603e+05	6.231633e+05	10883.0	139895.0	292980.0	651456.0	4522032.0
Joining Designation	2381.0	1.820244e+00	8.414334e-01	1.0	1.0	2.0	2.0	5.0

```
len(ola1[ola1['Total Business Value'] < 1])
```

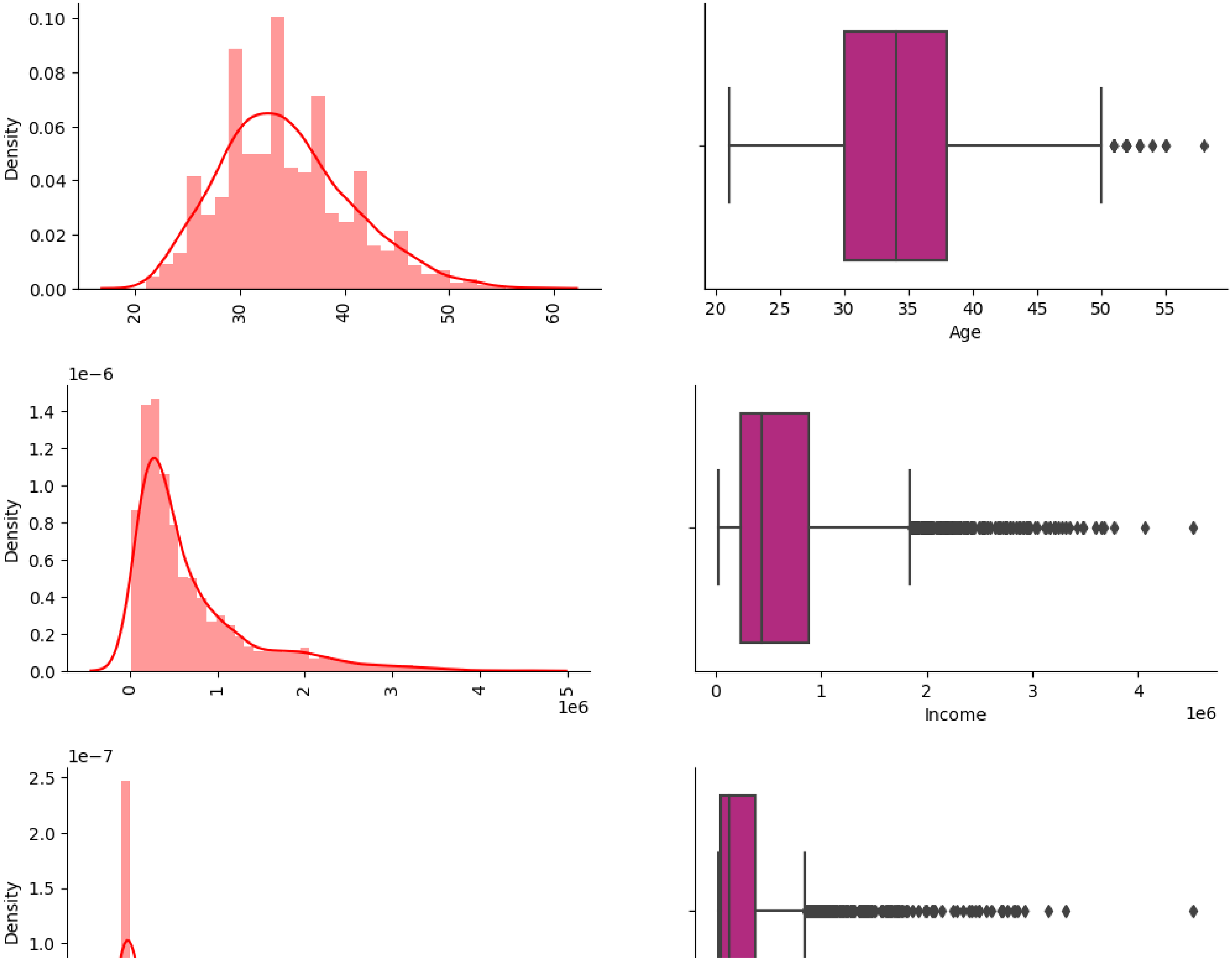
```
729
```

```
# We identify values in the 'Total Business Value' column that are less than 1 as outliers, and upon considering this subset of the
# dataset, we find that precisely 729 drivers have such outlier values.
```

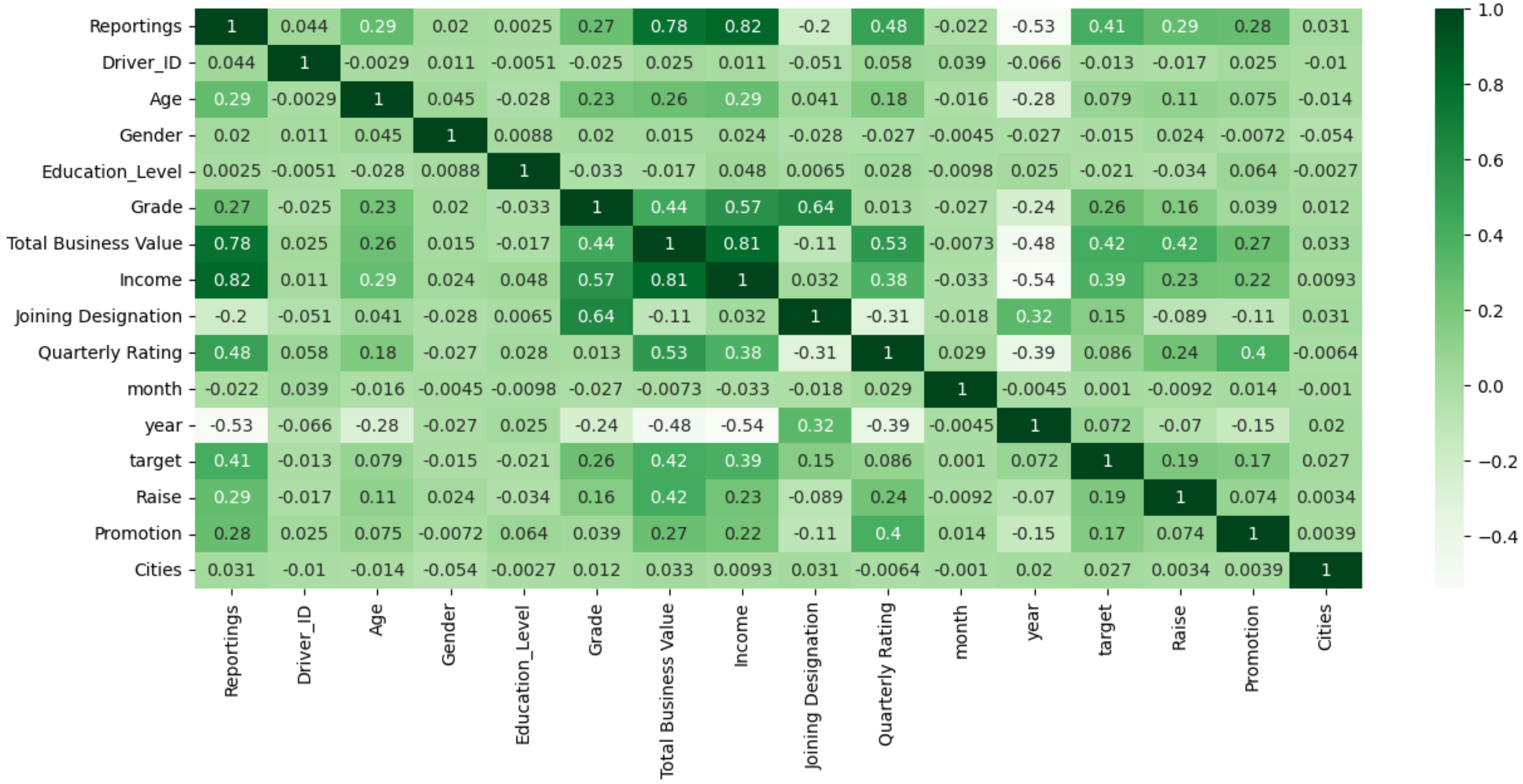
Raise	2381.0	1.805964e-02	1.331951e-01	0.0	0.0	0.0	0.0	1.0

```
ola1= ola1[ola1['Total Business Value'] > 1]
```

```
a =ola1[['Age','Income','Total Business Value']]
for i in a:
    plt.figure(figsize=(12,3))
    plt.subplot(121)
    sns.distplot(x=ola1[i],color='red')
    plt.xticks(rotation=90)
#     plt.figure(figsize=(9,5))
    plt.subplot(122)
    sns.boxplot(x=ola1[i],color='mediumvioletred')
    sns.despine()
    plt.show()
```




```
corr = ola1.corr()
plt.figure(figsize=(15,6))
sns.heatmap(corr,annot=True,cmap='Greens')
plt.show()
```




```
ola1.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Reportings	1652.0	1.026998e+01	6.967589e+00	1.0	5.0	8.0	14.0	24.0
Driver_ID	1652.0	1.390315e+03	8.082919e+02	1.0	679.5	1385.0	2097.0	2788.0
Age	1652.0	3.432385e+01	6.190776e+00	21.0	30.0	34.0	38.0	58.0
Gender	1652.0	4.158596e-01	4.930188e-01	0.0	0.0	0.0	1.0	1.0
Education_Level	1652.0	1.030872e+00	8.093284e-01	0.0	0.0	1.0	2.0	2.0
Grade	1652.0	2.144068e+00	9.719606e-01	1.0	1.0	2.0	3.0	5.0
Total Business Value	1652.0	6.613094e+06	1.032794e+07	19580.0	663022.5	2242080.0	7418392.5	95331060.0
Income	1652.0	6.864932e+05	6.814522e+05	20886.0	236652.5	428960.0	877151.0	4522032.0
Joining Designation	1652.0	1.759685e+00	8.395129e-01	1.0	1.0	2.0	2.0	5.0
Quarterly Rating	1652.0	1.700363e+00	9.237035e-01	1.0	1.0	1.0	2.0	4.0
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Raise	1652.0	2.602906e-02	1.592699e-01	0.0	0.0	0.0	0.0	1.0

```
#ENSEMBLE LEARNING:-
```

```
#Data Prepration:-
```

```
# In model selection, the trade-off between precision and recall is crucial: prioritizing precision minimizes false positives, avoiding unnecessary
# expenses on employees unlikely to leave, while prioritizing recall minimizes false negatives, allowing for better identification of potential
# departures and implementing retention measures for experienced individuals, considering the higher cost of hiring new personnel


from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

```
X = ola1.drop('target',axis=1)
y = ola1['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state= 42)
```

```
from sklearn.model_selection import learning_curve
def plot_learning_curve(estimator, X, Y, title):
    train_sizes, train_scores, test_scores, _, _ = learning_curve(estimator,X,Y,return_times=True)
    fig, axes = plt.subplots(1, 1, figsize = (15, 5))
    axes.set_title(title)
    axes.plot
    axes.set_xlabel("Training examples")
    axes.set_ylabel("Score")
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    # Plot learning curve
#     32
    axes.grid()
    axes.fill_between(
    train_sizes,
    train_scores_mean - train_scores_std,
    train_scores_mean + train_scores_std,
    alpha=0.1,
    color="r",
    )
    axes.fill_between(
    train_sizes,
    test_scores_mean - test_scores_std,
    test_scores_mean + test_scores_std,
    alpha=0.1,
    color="g",
    )
    axes.plot(
    train_sizes, train_scores_mean, "o-", color="r", label="Training score"
    )
    axes.plot(
    train_sizes, test_scores_mean, "o-", color="g", label="Cross-validation score"
    )
    axes.legend(loc="best")
    plt.show()
```

```
X.head()
```

	Reportings	Driver_ID	Age	Gender	Education_Level	Grade	Total Business Value	Income	Joining Designation	Quarterly Rating	month	year	Ra
0	3	1	28	0	2	1	1715580	172161	1	2	12	2018	
2	5	4	43	0	2	2	350000	328015	2	1	11	2019	
3	3	5	29	0	0	1	120360	139104	1	1	12	2019	
4	5	6	31	1	1	3	1265000	393640	3	1	12	2020	

```
ss= StandardScaler()
ss.fit_transform(X_train)
```

```
array([[-0.61446611, -1.09640018,  1.70794584, ..., -0.16737851,
         1.023749  , -0.04979913],
       [ 1.93718866, -1.32951199,  1.54780698, ..., -0.16737851,
        -0.97680193, -0.5247786 ],
       [-0.18919032, -1.0914666 ,  0.26669606, ..., -0.16737851,
         1.023749  ,  1.25639439],
       ...,
       [-0.75622471,  0.03585718, -1.49483144, ..., -0.16737851,
        -0.97680193, -0.88101319],
       [ 0.51960268,  1.32105562, -1.33469258, ..., -0.16737851,
         1.023749  , -1.59348238],
       [-0.33094892,  0.60815284, -0.69413712, ..., -0.16737851,
        -0.97680193, -0.28728886]])
```

```
from sklearn.model_selection import cross_validate
```

```
valid1 = cross_val_score(LogisticRegression(),X,y,cv=5)
print('Logistic Regression:',valid1.round(2))
print('Mean:',valid1.mean())
valid2 = cross_val_score( DecisionTreeClassifier(),X,y,cv=5)
print('Decision Tree:',valid2.round(3))
print('Mean:',valid2.mean())
valid3 = cross_val_score(RandomForestClassifier(),X,y,cv=5)
print('RandomForestClassifier():',valid3.round(2))
print('Mean:',valid3.mean())
valid4 = cross_val_score(GradientBoostingClassifier(),X,y,cv=5)
print('GradientBoostingClassifier:',valid4.round(3))
print('Mean:',valid4.mean())
valid5 =cross_val_score(XGBClassifier(),X,y,cv=5)
print('XGBoostClassifier:',valid1.round(2))
print('Mean:',valid5.mean())
```

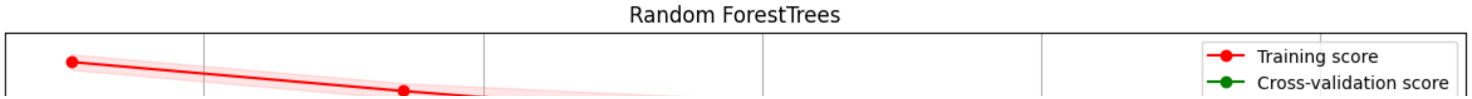
```
Logistic Regression: [0.7  0.75 0.75 0.75 0.76]
Mean: 0.7415453629955141
Decision Tree: [0.843 0.858 0.858 0.852 0.839]
Mean: 0.8498782385791449
RandomForestClassifier(): [0.89 0.89 0.89 0.87 0.91]
Mean: 0.889224572004028
GradientBoostingClassifier: [0.888 0.918 0.885 0.879 0.845]
Mean: 0.8831474869541335
XGBoostClassifier: [0.7  0.75 0.75 0.75 0.76]
Mean: 0.879520278311819
```

```
# MACHINE LEARNING MODEL:-
# WITHOUT THE TREATMENT OF CLASS IMBALANCE
# Random Forest Classifier
```

```
rf_clf1 = RandomForestClassifier(criterion='gini',max_depth=7,max_features='sqrt',n_estimators=10)
rf_clf1.fit(X_train,y_train)
```

```
▾               RandomForestClassifier
RandomForestClassifier(max_depth=7, n_estimators=10)
```

```
plot_learning_curve(rf_clf1, X_train, y_train, "Random ForestTrees")
```




```
y_pred = rf_clf1.predict(X_test)
proba = rf_clf1.predict_proba(X_test)[:,1]
print("Train data accuracy:",rf_clf1.score(X_train, y_train))
print("Test data accuracy:",rf_clf1.score(X_test,y_test))
print('Accuracy of the model:', accuracy_score(y_test, y_pred))
print("ROC-AUC score test dataset: ", roc_auc_score(y_test, proba))
print('-'*70)
print(classification_report(y_test, y_pred))
print('-'*70)
cm1 = (confusion_matrix(y_test, y_pred))
print('Confusion Metrix')
print(confusion_matrix(y_test, y_pred))
```

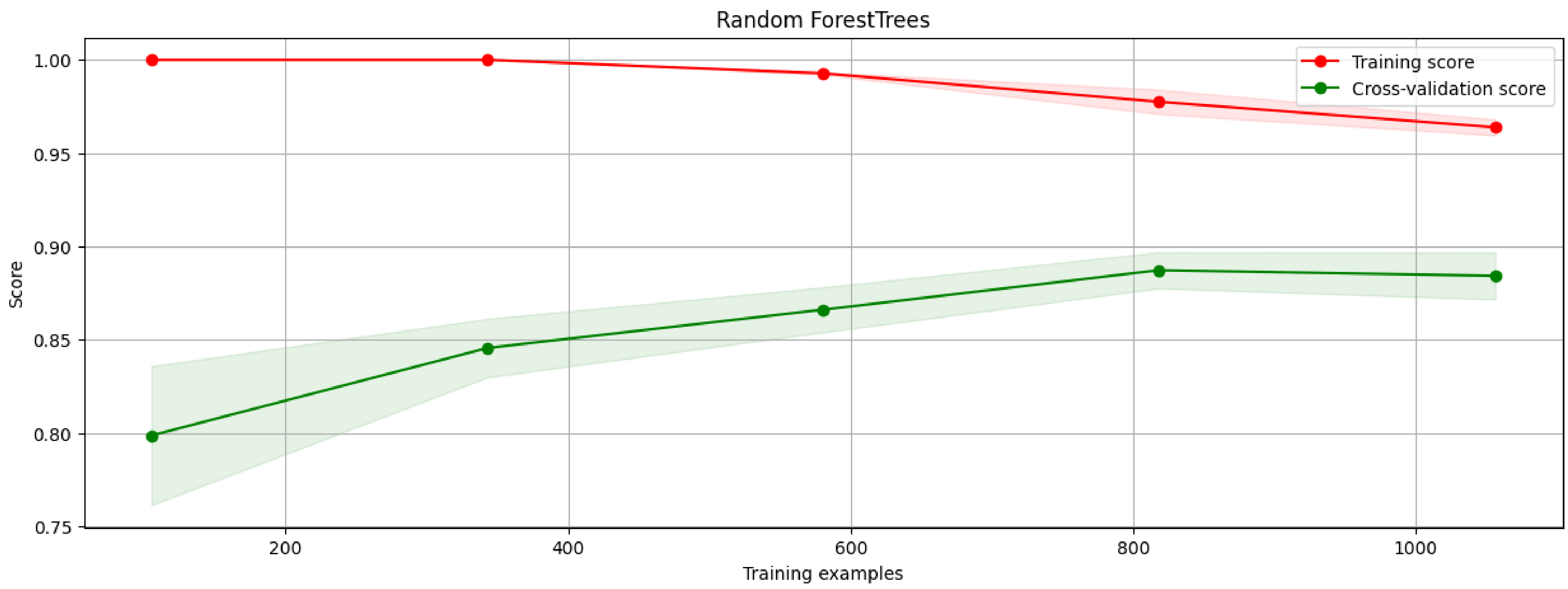
```
Train data accuracy: 0.9280847842543528
Test data accuracy: 0.8429003021148036
Accuracy of the model: 0.8429003021148036
ROC-AUC score test dataset:  0.9268349696119682
----------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.84      0.93      0.88       207
           1       0.86      0.69      0.77       124

    accuracy                           0.84       331
   macro avg       0.85      0.81      0.82       331
weighted avg       0.84      0.84      0.84       331

----------------------------------------------------------------------
Confusion Metrix
[[193  14]
 [ 38  86]]
```

```
rf_clf_imp1 = rf_clf1.feature_importances_
```

```
# XG Boosting Classifier
```

```
gbc1 = GradientBoostingClassifier()
gbc1.fit(X_train, y_train)
y_pred = gbc1.predict(X_test)
proba =gbc1.predict_proba(X_test)[:, 1]
```

```
plot_learning_curve(gbc1, X_train, y_train, "Random ForestTrees")
```




```
gbc_clf_imp1 = gbc1.feature_importances_
```

```
print('Train Score : ', gbc1.score(X_train, y_train))
print('Test Score : ', gbc1.score(X_test, y_test))
print('Accuracy Score : ', accuracy_score(y_test, y_pred))
print("ROC-AUC score test dataset: ", roc_auc_score(y_test, proba))
print('-'*60)
print(classification_report(y_test, y_pred))
print('-'*60)
print('Confusion Matrix')
cm2 = (confusion_matrix(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print('-'*60)
```

```
Train Score :  0.9553368660105981
Test Score :  0.9003021148036254
Accuracy Score :  0.9003021148036254
ROC-AUC score test dataset:  0.9492753623188407
------------------------------------------------------------
```

```
              precision    recall  f1-score   support

           0       0.91      0.94      0.92       207
           1       0.89      0.84      0.86       124

    accuracy                           0.90       331
   macro avg       0.90      0.89      0.89       331
weighted avg       0.90      0.90      0.90       331

------------------------------------------------------------
Confusion Matrix
[[194  13]
 [ 20 104]]
------------------------------------------------------------
```

```
# CLASS IMBALANCE TREATMENT
```

```
plt.figure(figsize=(15,4))
sns.countplot(x=y_train,palette='Set2')
plt.title('Class Imbalance in the Data')
plt.show()
```


```
(y_train.value_counts()*100)/len(y_train)
```

```
0    64.118092
1    35.881908
Name: target, dtype: float64
```

```
from imblearn.over_sampling import SMOTE
```

```
smot = SMOTE(random_state=42)
X_train_smot,y_train_smot = smot.fit_resample(X_train,y_train.ravel())
```

```
X_train_smot.shape,y_train_smot.shape
```

```
((1694, 15), (1694,))
```

```
X_test.shape,y_test.shape
```

```
((331, 15), (331,))
```

```
from collections import Counter
c = Counter(y_train_smot)
print(c)
```

```
Counter({0: 847, 1: 847})
```

```
# Randome Forest Classifier
```

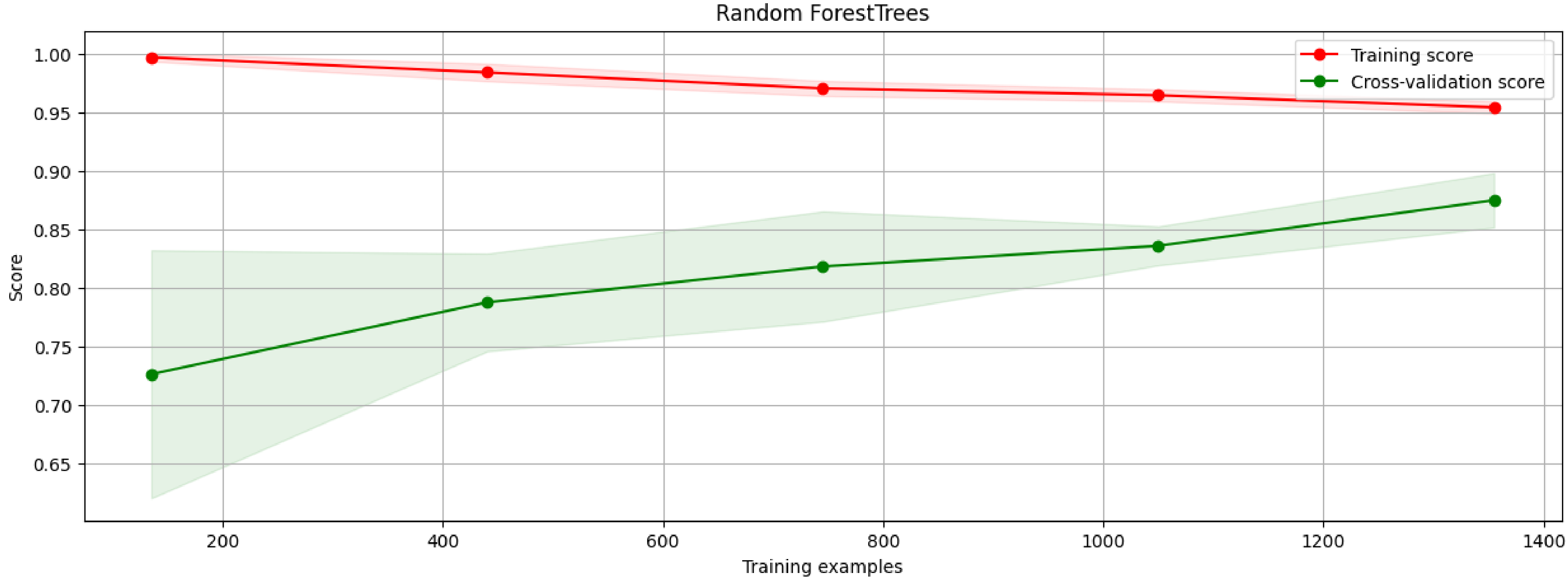
```
clf = RandomForestClassifier()
clf.fit(X_train_smot,y_train_smot)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
clf = RandomForestClassifier(criterion='gini',max_depth=8,
                             max_features='sqrt',n_estimators= 19)
clf.fit(X_train_smot,y_train_smot)
```

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=8, n_estimators=19)
```

```
plot_learning_curve(clf, X_train_smot, y_train_smot, "Random ForestTrees")
```




```
y_pred = clf.predict(X_test)
print('-'*70)
print(classification_report(y_test, y_pred))
print('-'*70)
print('Confusion Metrix')
cm3 = confusion_matrix(y_test, y_pred)
print(confusion_matrix(y_test, y_pred))
```

```
----------------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.93      0.87      0.90       207
           1       0.81      0.90      0.85       124

    accuracy                           0.88       331
   macro avg       0.87      0.88      0.88       331
weighted avg       0.89      0.88      0.88       331

----------------------------------------------------------------------
Confusion Metrix
[[181  26]
 [ 13 111]]
```

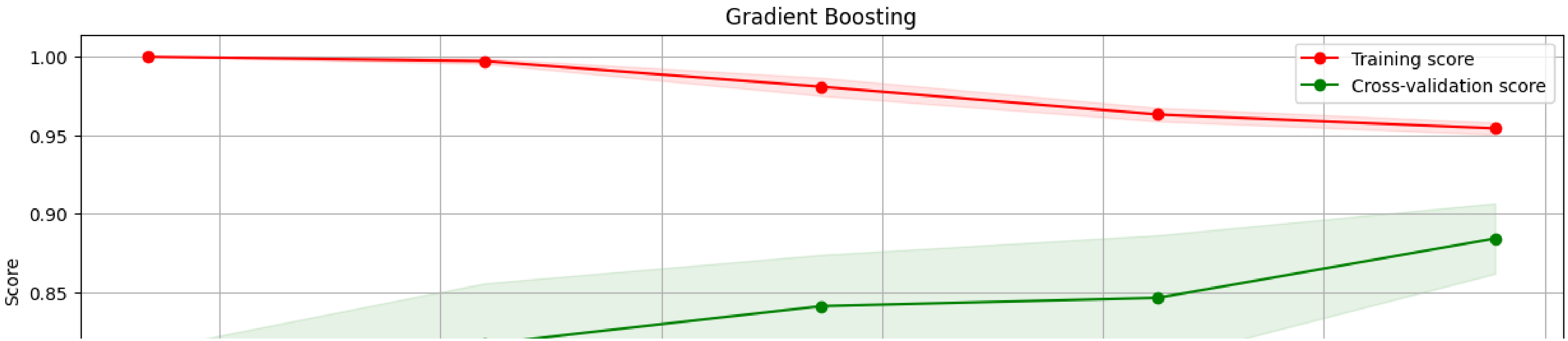
```
rf_clf_imp2= clf.feature_importances_
```

```
# Gradient Boosting
```

```
gbc2 = GradientBoostingClassifier()
gbc2.fit(X_train_smot, y_train_smot)
y_pred1 = gbc2.predict(X_test)
gbc_clf_imp2 = gbc2.feature_importances_
print('-'*60)
print(classification_report(y_test, y_pred1))
print('-'*60)
cm4 = confusion_matrix(y_test, y_pred1)
print('Confusion Matrix')
print(cm4)
print('-'*60)
```

```
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.93      0.89      0.91       207
           1       0.83      0.90      0.86       124

    accuracy                           0.89       331
   macro avg       0.88      0.89      0.89       331
weighted avg       0.90      0.89      0.89       331

------------------------------------------------------------
Confusion Matrix
[[185  22]
 [ 13 111]]
------------------------------------------------------------
```

```
plot_learning_curve(gbc2, X_train_smot, y_train_smot, "Gradient Boosting")
```




```
data1 = pd.DataFrame({'Column_Name':X.columns,
                      'RandomForestClassifier':rf_clf_imp1,
                      'XGBClassifier':gbc_clf_imp1})
```

```
data2 = pd.DataFrame({'Column_Name':X.columns,
                      'RandomForestClassifier':rf_clf_imp2,
                      'XGBClassifier':gbc_clf_imp2})
```

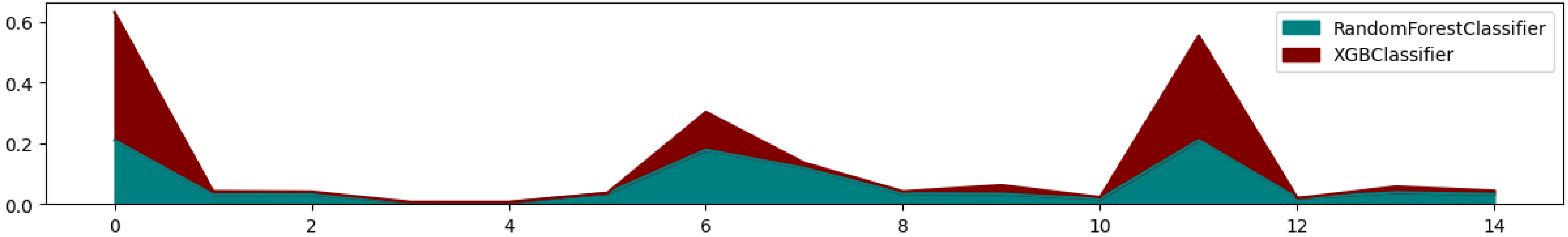
```
data1
```

	Column_Name	RandomForestClassifier	XGBClassifier
0	Reportings	0.209871	0.420978
1	Driver_ID	0.030580	0.011521
2	Age	0.031879	0.008586

```
data2
```

	Column_Name	RandomForestClassifier	XGBClassifier
0	Reportings	0.244572	0.300373
1	Driver_ID	0.035055	0.009928
2	Age	0.042561	0.009589
3	Gender	0.015603	0.009140
4	Education_Level	0.017783	0.004790
5	Grade	0.030698	0.004796
6	Total Business Value	0.156366	0.205282
7	Income	0.105710	0.024849
8	Joining Designation	0.029745	0.003067
9	Quarterly Rating	0.043415	0.028597
10	month	0.024704	0.003328
11	year	0.184614	0.363063
12	Raise	0.009132	0.000000
13	Promotion	0.033418	0.025453
14	Cities	0.026624	0.007745

```
data1.plot(kind="area", figsize = (15,2),color=['teal','maroon'])

data2.plot(kind="area", figsize = (15,2),color=['teal','black'])

plt.show()
```

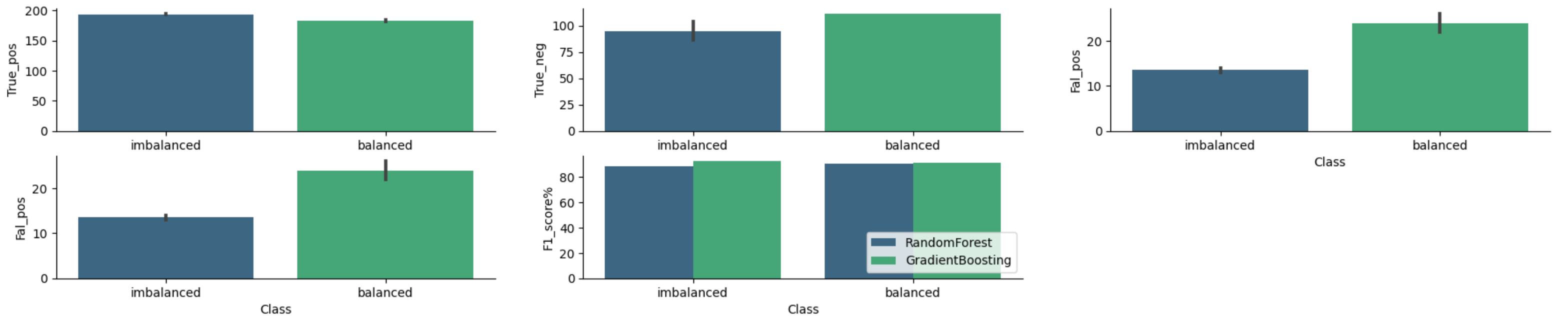



```
# calculating precision, reall and f1_score for every
tp1,fp1,fn1,tn1 =cm1[0][0],cm1[0][1],cm1[1][0],cm1[1][1]
tp2,fp2,fn2,tn2 =cm2[0][0],cm2[0][1],cm2[1][0],cm2[1][1]
tp3,fp3,fn3,tn3 =cm3[0][0],cm3[0][1],cm3[1][0],cm3[1][1]
tp4,fp4,fn4,tn4 =cm4[0][0],cm4[0][1],cm4[1][0],cm4[1][1]
precision1 = tp1/(tp1+fp1)
recall1 = tp1/(tp1+fn1)
precision2 = tp2/(tp2+fp2)
recall2 = tp2/(tp2+fn2)
precision3 = tp3/(tp3+fp3)
recall3 = tp3/(tp3+fn3)
precision4 = tp4/(tp4+fp4)
recall4 = tp4/(tp4+fn4)
f1_1 = (2*precision1*recall1)/(precision1+recall1)
f1_2 = (2*precision2*recall2)/(precision2+recall2)
f1_3 = (2*precision3*recall3)/(precision3+recall3)
f1_4 =(2*precision4*recall4)/(precision4+recall4)


df = pd.DataFrame({'Model':['RandomForest','GradientBoosting','RandomForest','GradientBoosting'],
                  'Class':['imbalanced','imbalanced','balanced','balanced'],
                  'True_pos':[tp1,tp2,tp3,tp4],
                  'Fal_pos':[fp1,fp2,fp3,fp4],
                  'Fal_neg':[fn1,fn2,fn3,fn4],
                  'True_neg':[tn1,tn2,tn3,tn4],
                  'F1_score%':[f1_1*100,f1_2*100,f1_3*100,f1_4*100],
                  'Precision%':[precision1*100,precision2*100,precision3*100,precision4*100],
                  'Recall%':[recall1*100,recall2*100,recall3*100,recall4*100]})
```

```
df
```

	Model	Class	True_pos	Fal_pos	Fal_neg	True_neg	F1_score%	Precision%	Recall%
0	RandomForest	imbalanced	193	14	38	86	88.127854	93.236715	83.549784
1	GradientBoosting	imbalanced	194	13	20	104	92.161520	93.719807	90.654206
2	RandomForest	balanced	181	26	13	111	90.274314	87.439614	93.298969
3	GradientBoosting	balanced	185	22	13	111	91.358025	89.371981	93.434343

```
plt.figure(figsize=(22,4))
plt.subplot(2,3,1)
sns.barplot(x=df.Class,y=df.True_pos,palette='viridis')
# plt.show()
plt.subplot(2,3,2)
sns.barplot(x=df.Class,y=df.True_neg,palette='viridis')
# plt.show()
plt.subplot(2,3,3)
sns.barplot(x=df.Class,y=df.Fal_pos,palette='viridis')
# plt.show()
plt.subplot(2,3,4)
sns.barplot(x=df.Class,y=df.Fal_pos,palette='viridis')
plt.subplot(2,3,5)
sns.barplot(x=df.Class,y=df['F1_score%'],palette='viridis',hue=df.Model)
plt.legend(loc='lower right')
sns.despine()
plt.show()
```



```
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
sns.barplot(x=df.Model,y=df['Precision%'],hue=df.Class,palette='viridis')
plt.title('Precision according to classes and Models')
plt.legend(loc='lower right')
plt.subplot(1,2,2)
sns.barplot(x=df.Model,y=df['Recall%'],hue=df.Class,palette='viridis')
plt.title('Recall according to classes and Models')
plt.legend(loc='lower right')
sns.despine()
plt.show()
```