1.0 Project Overview

This project is about the Syrian Telecommunication company that was assessing the bevaiour of the customers to leave their services and move to another telecommunication company(competitor).This will mean customers will soon stop accessing their services such as calling,sms,etc and switch to another service provider.In this project therefore we will explore the available data to classify the customer into two classifiable predictions as: will soon stop using the telcos services and will retain the services of the the telcos.In the longrun we shall determine which features will contribute to the customer discontinuing(soon) services of Syriatel in favour of another telcomunication company.

1.1 Objetcives of the Project

1. Determine how long a customer will stay on the Syriatel services
2. Determinie the retention ration of customers by Syriatel
3. Determine possible strategies to retain customers on Syriatel

2.0 Business and Data Understanding

2.1 Business Understanding

This project is about assessing why the Syriatel Telecom company is going to loose customers,very soon to anothr service provider within the industry.We shall therefore seek insight on why customers will leave Syriatel or for this matter any company within the industry to cross-over another network.We shall establish the customer trends across various services provided within the network and see what factors will lead the customer abandon the service of one company for the other.

In particular we shall seek to answer the following questions:

1. How long does it take the customers to stay with the Syriatel?

2. What is the rention ratio of customers by the Telcos?

3. What are the likely causes of customers to leave the Syriatel to another telcos?

4. What are the likely strategies to be deployed by the teclos to avoid soon losing customers?

5. What is the behaviour of the customer before soon leaving the Syriatel to another service customer?

2.2 Data Understanding

In this section we explore the data provided for this project applying Exploration Data Analysisis Techniques to determine how we shall utilise the data provided.

2.2.1 Importing Libraries

In [279…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_sco
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import accuracy_score,f1_score,recall_score,pr
from sklearn.preprocessing import MinMaxScaler # to scale the numer
from scipy import stats
```

2.2.2 Import the provided data for the project

We shall load our csv file and see the characteristics of the data provided and identify the features required for this project.

In [280…

```
data= pd.read_csv('bigml_59c28831336c6604c800002a.csv')
data
```

Out[280…

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	tot d minut
0	KS	128	415	382-4657	no	yes	25	265
1	OH	107	415	371-7191	no	yes	26	161
2	NJ	137	415	358-1921	no	no	0	243
3	OH	84	408	375-9999	yes	no	0	299
4	OK	75	415	330-6626	yes	no	0	166
...	...	...	...	...	...	...	...	
3328	AZ	192	415	414-4276	no	yes	36	156
3329	WV	68	415	370-3271	no	no	0	231
3330	RI	28	510	328-8230	no	no	0	180
3331	CT	184	510	364-6381	yes	no	0	213
3332	TN	74	415	400-4344	no	yes	25	234

3333 rows × 21 columns

2.3 Data preparations

In this section we shall undertake data preparation to enable us conduct Exploratory Data Analysis and Modelling by;

(a). Determine any missing values in the data set (b). Identify any duplicated rows and columns (c). Identify any irreleant columns that may not be needed to conduct any analysis and therefore they are of no value to us in conducting this modelling for Machine Learning.This will be achieved by dropping such columns.

In [281…

```
# checking for missing values
data.isnull().sum()
```

Out[281…

```
state                     0
account length            0
area code                 0
phone number              0
international plan        0
voice mail plan           0
number vmail messages     0
total day minutes         0
total day calls           0
total day charge          0
total eve minutes         0
total eve calls           0
total eve charge          0
total night minutes       0
total night calls         0
total night charge        0
total intl minutes        0
total intl calls          0
total intl charge         0
customer service calls    0
churn                     0
dtype: int64
```

No missing value in the data set

In [282…

```
# Check for duplicates in the dataset
duplicates = data.duplicated().sum()
duplicates
```

Out[282…

```
0
```

No duplicates in the dataset

In [283…

```
# Drop any irrelevant columns that will not be required or used in 
data.drop(columns=['phone number'], inplace=True)
data.head(5)
```

Out[283…

	state	account length	area code	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	t cha
0	KS	128	415	no	yes	25	265.1	110	4
1	OH	107	415	no	yes	26	161.6	123	2
2	NJ	137	415	no	no	0	243.4	114	4
3	OH	84	408	yes	no	0	299.4	71	5
4	OK	75	415	yes	no	0	166.7	113	2

2.4 Conducting Exploratory Data

Analysis

In this section we shall explore the data to see the type of data we are dealing with,establish some relationships and visualise the data.

In [284…
```
# checking the data types
data.dtypes
```

Out[284…
```
state                      object
account length              int64
area code                   int64
international plan         object
voice mail plan            object
number vmail messages       int64
total day minutes         float64
total day calls             int64
total day charge          float64
total eve minutes         float64
total eve calls             int64
total eve charge          float64
total night minutes       float64
total night calls           int64
total night charge        float64
total intl minutes        float64
total intl calls            int64
total intl charge         float64
customer service calls      int64
churn                        bool
dtype: object
```

The data contains both numeric data and categorical data:

1. Categorical data include; state,international plan and voice plan

2. Numeric data include;

- number vmail message
- total day minutes
- total day calls
- total day charge
- total eve minutes
- total eve calls
- total eve charge
- total night minutes
- total night calls
- total night charge
- total intl minutes
- total intl calls
- total intl charge

- customer service calls

3. We have data known as **churn** which is boolen in nature meaning that it either true or False.This may mean it will determine if the customer left the telcom company soon(Syriatel) or not.True will denote that the customer left the company and False will denote that the customer did not leave the service.

We can further analyise the churn data and create a list for numeric and categorical data as follows to enable us use the data better in our analysis going forward:

(a) Creating numerical and categorical features or lists

In [285…

```
# Creating a list of numeric features
numeric_features = data.select_dtypes(include=[np.number]).columns.
numeric_features
```

Out[285…

```
['account length',
 'area code',
 'number vmail messages',
 'total day minutes',
 'total day calls',
 'total day charge',
 'total eve minutes',
 'total eve calls',
 'total eve charge',
 'total night minutes',
 'total night calls',
 'total night charge',
 'total intl minutes',
 'total intl calls',
 'total intl charge',
 'customer service calls']
```

In [286…

```
# Creating a list of categorical  features
categorical_features = data.select_dtypes(exclude=[np.number]).colu
categorical_features
```

Out[286…

```
['state', 'international plan', 'voice mail plan', 'churn']
```

We can further count the boolen entries in the churn feature as follows;

In [287…

```
# counting the boolean values in the churn data as 1 and 0
data['churn'].value_counts() # 1 means churn or the customer left t
```

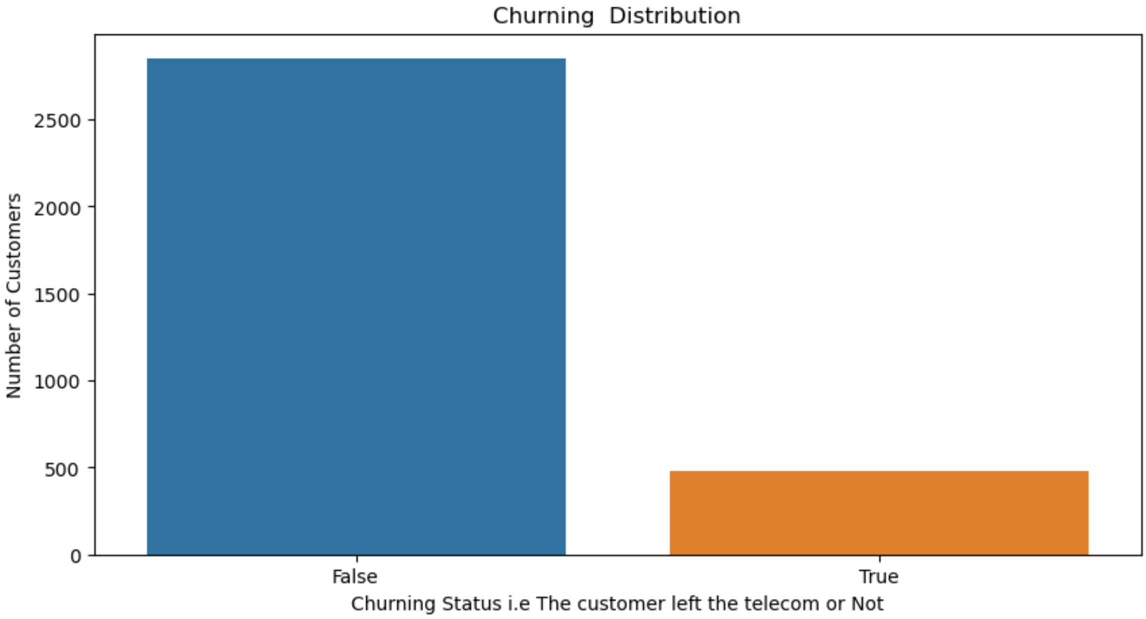
Out[287…

```
churn
False    2850
True      483
Name: count, dtype: int64
```

In [288…

```
# Visualize the churned data on a histograph
```

```
plt.figure(figsize=(10, 5))
sns.countplot(x='churn', data=data);
plt.title('Churning  Distribution')
plt.xlabel('Churning Status i.e The customer left the telecom or No
plt.ylabel('Number of Customers')
plt.show()
```



From the above graph we can see that the data is imbalanced as the number of customers who churned is less than the number of customers who did not churn.i.e. the number of customers who left the telecom is less than the number of customers who are still with the telecom.

Those customers who left were 483 and those who remained were 2,850

In [289…
```
# Identfying the unique values in the categorical features
for feature in categorical_features:
    print(f"{feature}: {data[feature].unique()}")
```

```
state: ['KS' 'OH' 'NJ' 'OK' 'AL' 'MA' 'MO' 'LA' 'WV' 'IN' 'RI' 'IA'
'MT' 'NY'
 'ID' 'VT' 'VA' 'TX' 'FL' 'CO' 'AZ' 'SC' 'NE' 'WY' 'HI' 'IL' 'NH' 'G
A'
 'AK' 'MD' 'AR' 'WI' 'OR' 'MI' 'DE' 'UT' 'CA' 'MN' 'SD' 'NC' 'WA' 'N
M'
 'NV' 'DC' 'KY' 'ME' 'MS' 'TN' 'PA' 'CT' 'ND']
international plan: ['no' 'yes']
voice mail plan: ['yes' 'no']
churn: [False  True]
```

In [290…
```
# identifying the unique values in the numeric features
for feature in numeric_features:
    print(f"{feature}: {data[feature].unique()}")
```

```
account length: [128 107 137  84  75 118 121 147 117 141  65  74 168
95  62 161  85  93
  76  73  77 130 111 132 174  57  54  20  49 142 172  12  72  36  78
136
```

```
 149  98 135  34 160  64  59 119  97  52  60  10  96  87  81  68 125
116
  38  40  43 113 126 150 138 162  90  50  82 144  46  70  55 106  94
155
  80 104  99 120 108 122 157 103  63 112  41 193  61  92 131 163  91
127
 110 140  83 145  56 151 139   6 115 146 185 148  32  25 179  67  19
170
 164  51 208  53 105  66  86  35  88 123  45 100 215  22  33 114  24
101
 143  48  71 167  89 199 166 158 196 209  16  39 173 129  44  79  31
124
  37 159 194 154  21 133 224  58  11 109 102 165  18  30 176  47 190
152
  26  69 186 171  28 153 169  13  27   3  42 189 156 134 243  23   1
205
 200   5   9 178 181 182 217 177 210  29 180   2  17   7 212 232 192
195
 197 225 184 191 201  15 183 202   8 175   4 188 204 221]
area code: [415 408 510]
number vmail messages: [25 26  0 24 37 27 33 39 30 41 28 34 46 29 35
21 32 42 36 22 23 43 31 38
 40 48 18 17 45 16 20 14 19 51 15 11 12 47  8 44 49  4 10 13 50  9]
total day minutes: [265.1 161.6 243.4 ... 321.1 231.1 180.8]
total day calls: [110 123 114  71 113  98  88  79  97  84 137 127  9
6  70  67 139  66  90
 117  89 112 103  86  76 115  73 109  95 105 121 118  94  80 128  64
106
 102  85  82  77 120 133 135 108  57  83 129  91  92  74  93 101 146
72
  99 104 125  61 100  87 131  65 124 119  52  68 107  47 116 151 126
122
 111 145  78 136 140 148  81  55  69 158 134 130  63  53  75 141 163
59
 132 138  54  58  62 144 143 147  36  40 150  56  51 165  30  48  60
42
   0  45 160 149 152 142 156  35  49 157  44]
total day charge: [45.07 27.47 41.38 ... 54.59 39.29 30.74]
total eve minutes: [197.4 195.5 121.2 ... 153.4 288.8 265.9]
total eve calls: [ 99 103 110  88 122 101 108  94  80 111  83 148  7
1  75  76  97  90  65
  93 121 102  72 112 100  84 109  63 107 115 119 116  92  85  98 118
74
 117  58  96  66  67  62  77 164 126 142  64 104  79  95  86 105  81
113
 106  59  48  82  87 123 114 140 128  60  78 125  91  46 138 129  89
133
 136  57 135 139  51  70 151 137 134  73 152 168  68 120  69 127 132
143
  61 124  42  54 131  52 149  56  37 130  49 146 147  55  12  50 157
155
  45 144  36 156  53 141  44 153 154 150  43   0 145 159 170]
total eve charge: [16.78 16.62 10.3  ... 13.04 24.55 22.6 ]
total night minutes: [244.7 254.4 162.6 ... 280.9 120.1 279.1]
total night calls: [ 91 103 104  89 121 118  96  90  97 111  94 128
115  99  75 108  74 133
```

```
  64  78 105  68 102 148  98 116  71 109 107 135  92  86 127  79  87
129
  57  77  95  54 106  53  67 139  60 100  61  73 113  76 119  88  84
62
 137  72 142 114 126 122  81 123 117  82  80 120 130 134  59 112 132
110
 101 150  69 131  83  93 124 136 125  66 143  58  55  85  56  70  46
42
 152  44 145  50 153  49 175  63 138 154 140 141 146  65  51 151 158
155
 157 147 144 149 166  52  33 156  38  36  48 164]
total night charge: [11.01 11.45  7.32  8.86  8.41  9.18  9.57  9.53
9.71 14.69  9.4   8.82
  6.35  8.65  9.14  7.23  4.02  5.83  7.46  8.68  9.43  8.18  8.53 1
0.67
 11.28  8.22  4.59  8.17  8.04 11.27 11.08 13.2  12.61  9.61  6.88
5.82
 10.25  4.58  8.47  8.45  5.5  14.02  8.03 11.94  7.34  6.06 10.9
6.44
  3.18 10.66 11.21 12.73 10.28 12.16  6.34  8.15  5.84  8.52  7.5
7.48
  6.21 11.95  7.15  9.63  7.1   6.91  6.69 13.29 11.46  7.76  6.86
8.16
 12.15  7.79  7.99 10.29 10.08 12.53  7.91 10.02  8.61 14.54  8.21
9.09
  4.93 11.39 11.88  5.75  7.83  8.59  7.52 12.38  7.21  5.81  8.1  1
1.04
 11.19  8.55  8.42  9.76  9.87 10.86  5.36 10.03 11.15  9.51  6.22
2.59
  7.65  6.45  9.    6.4   9.94  5.08 10.23 11.36  6.97 10.16  7.88 1
1.91
  6.61 11.55 11.76  9.27  9.29 11.12 10.69  8.8  11.85  7.14  8.71 1
1.42
  4.94  9.02 11.22  4.97  9.15  5.45  7.27 12.91  7.75 13.46  6.32 1
2.13
 11.97  6.93 11.66  7.42  6.19 11.41 10.33 10.65 11.92  4.77  4.38
7.41
 12.1   7.69  8.78  9.36  9.05 12.7   6.16  6.05 10.85  8.93  3.48 1
0.4
  5.05 10.71  9.37  6.75  8.12 11.77 11.49 11.06 11.25 11.03 10.82
8.91
  8.57  8.09 10.05 11.7  10.17  8.74  5.51 11.11  3.29 10.13  6.8
8.49
  9.55 11.02  9.91  7.84 10.62  9.97  3.44  7.35  9.79  8.89  8.14
6.94
 10.49 10.57 10.2   6.29  8.79 10.04 12.41 15.97  9.1  11.78 12.75 1
1.07
 12.56  8.63  8.02 10.42  8.7   9.98  7.62  8.33  6.59 13.12 10.46
6.63
  8.32  9.04  9.28 10.76  9.64 11.44  6.48 10.81 12.66 11.34  8.75 1
3.05
 11.48 14.04 13.47  5.63  6.6   9.72 11.68  6.41  9.32 12.95 13.37
9.62
  6.03  8.25  8.26 11.96  9.9   9.23  5.58  7.22  6.64 12.29 12.93 1
1.32
  6.85  8.88  7.03  8.48  3.59  5.86  6.23  7.61  7.66 13.63  7.9  1
```

```
1.82
  7.47  6.08  8.4   5.74 10.94 10.35 10.68  4.34  8.73  5.14  8.24
9.99
 13.93  8.64 11.43  5.79  9.2  10.14 12.11  7.53 12.46  8.46  8.95
9.84
 10.8  11.23 10.15  9.21 14.46  6.67 12.83  9.66  9.59 10.48  8.36
4.84
 10.54  8.39  7.43  9.06  8.94 11.13  8.87  8.5   7.6  10.73  9.56 1
0.77
  7.73  3.47 11.86  8.11  9.78  9.42  9.65  7.    7.39  9.88  6.56
5.92
  6.95 15.71  8.06  4.86  7.8   8.58 10.06  5.21  6.92  6.15 13.49
9.38
 12.62 12.26  8.19 11.65 11.62 10.83  7.92  7.33 13.01 13.26 12.22 1
1.58
  5.97 10.99  8.38  9.17  8.08  5.71  3.41 12.63 11.79 12.96  7.64
6.58
 10.84 10.22  6.52  5.55  7.63  5.11  5.89 10.78  3.05 11.89  8.97 1
0.44
 10.5   9.35  5.66 11.09  9.83  5.44 10.11  6.39 11.93  8.62 12.06
6.02
  8.85  5.25  8.66  6.73 10.21 11.59 13.87  7.77 10.39  5.54  6.62 1
3.33
  6.24 12.59  6.3   6.79  8.28  9.03  8.07  5.52 12.14 10.59  7.54
7.67
  5.47  8.81  8.51 13.45  8.77  6.43 12.01 12.08  7.07  6.51  6.84
9.48
 13.78 11.54 11.67  8.13 10.79  7.13  4.72  4.64  8.96 13.03  6.07
3.51
  6.83  6.12  9.31  9.58  4.68  5.32  9.26 11.52  9.11 10.55 11.47
9.3
 13.82  8.44  5.77 10.96 11.74  8.9  10.47  7.85 10.92  4.74  9.74 1
0.43
  9.96 10.18  9.54  7.89 12.36  8.54 10.07  9.46  7.3  11.16  9.16 1
0.19
  5.99 10.88  5.8   7.19  4.55  8.31  8.01 14.43  8.3  14.3   6.53
8.2
 11.31 13.    6.42  4.24  7.44  7.51 13.1   9.49  6.14  8.76  6.65 1
0.56
  6.72  8.29 12.09  5.39  2.96  7.59  7.24  4.28  9.7   8.83 13.3  1
1.37
  9.33  5.01  3.26 11.71  8.43  9.68 15.56  9.8   3.61  6.96 11.61 1
2.81
 10.87 13.84  5.03  5.17  2.03 10.34  9.34  7.95 10.09  9.95  7.11
9.22
  6.13 11.05  9.89  9.39 14.06 10.26 13.31 15.43 16.39  6.27 10.64 1
1.5
 12.48  8.27 13.53 10.36 12.24  8.69 10.52  9.07 11.51  9.25  8.72
6.78
  8.6  11.84  5.78  5.85 12.3   5.76 12.07  9.6   8.84 12.39 10.1
9.73
  2.85  6.66  2.45  5.28 11.73 10.75  7.74  6.76  6.    7.58 13.69
7.93
  7.68  9.75  4.96  5.49 11.83  7.18  9.19  7.7   7.25 10.74  4.27 1
3.8
  9.12  4.75  7.78 11.63  7.55  2.25  9.45  9.86  7.71  4.95  7.4  1
```

```
1.17
 11.33  6.82 13.7   1.97 10.89 12.77 10.31  5.23  5.27  9.41  6.09 1
0.61
  7.29  4.23  7.57  3.67 12.69 14.5   5.95  7.87  5.96  5.94 12.23
4.9
 12.33  6.89  9.67 12.68 12.87  3.7   6.04 13.13 15.74 11.87  4.7
4.67
  7.05  5.42  4.09  5.73  9.47  8.05  6.87  3.71 15.86  7.49 11.69
6.46
 10.45 12.9   5.41 11.26  1.04  6.49  6.37 12.21  6.77 12.65  7.86
9.44
  4.3   7.38  5.02 10.63  2.86 17.19  8.67  8.37  6.9  10.93 10.38
7.36
 10.27 10.95  6.11  4.45 11.9  15.01 12.84  7.45  6.98 11.72  7.56 1
1.38
 10.    4.42  9.81  5.56  6.01 10.12 12.4  16.99  5.68 11.64  3.78
7.82
  9.85 13.74 12.71 10.98 10.01  9.52  7.31  8.35 11.35  9.5  14.03
3.2
  7.72 13.22 10.7   8.99 10.6  13.02  9.77 12.58 12.35 12.2  11.4  1
3.91
  3.57 14.65 12.28  5.13 10.72 12.86 14.    7.12 12.17  4.71  6.28
8.
  7.01  5.91  5.2  12.   12.02 12.88  7.28  5.4  12.04  5.24 10.3  1
0.41
 13.41 12.72  9.08  7.08 13.5   5.35 12.45  5.3  10.32  5.15 12.67
5.22
  5.57  3.94  4.41 13.27 10.24  4.25 12.89  5.72 12.5  11.29  3.25 1
1.53
  9.82  7.26  4.1  10.37  4.98  6.74 12.52 14.56  8.34  3.82  3.86 1
3.97
 11.57  6.5  13.58 14.32 13.75 11.14 14.18  9.13  4.46  4.83  9.69 1
4.13
  7.16  7.98 13.66 14.78 11.2   9.93 11.    5.29  9.92  4.29 11.1  1
0.51
 12.49  4.04 12.94  7.09  6.71  7.94  5.31  5.98  7.2  14.82 13.21 1
2.32
 10.58  4.92  6.2   4.47 11.98  6.18  7.81  4.54  5.37  7.17  5.33 1
4.1
  5.7  12.18  8.98  5.1  14.67 13.95 16.55 11.18  4.44  4.73  2.55
6.31
  2.43  9.24  7.37 13.42 12.42 11.8  14.45  2.89 13.23 12.6  13.18 1
2.19
 14.81  6.55 11.3  12.27 13.98  8.23 15.49  6.47 13.48 13.59 13.25 1
7.77
 13.9   3.97 11.56 14.08 13.6   6.26  4.61 12.76 15.76  6.38  3.6  1
2.8
  5.9   7.97  5.   10.97  5.88 12.34 12.03 14.97 15.06 12.85  6.54 1
1.24
 12.64  7.06  5.38 13.14  3.99  3.32  4.51  4.12  3.93  2.4  11.75
4.03
 15.85  6.81 14.25 14.09 16.42  6.7  12.74  2.76 12.12  6.99  6.68 1
1.81
  7.96  5.06 13.16  2.13 13.17  5.12  5.65 12.37 10.53]
total intl minutes: [10.  13.7 12.2  6.6 10.1  6.3  7.5  7.1  8.7 1
1.2 12.7  9.1 12.3 13.1
```

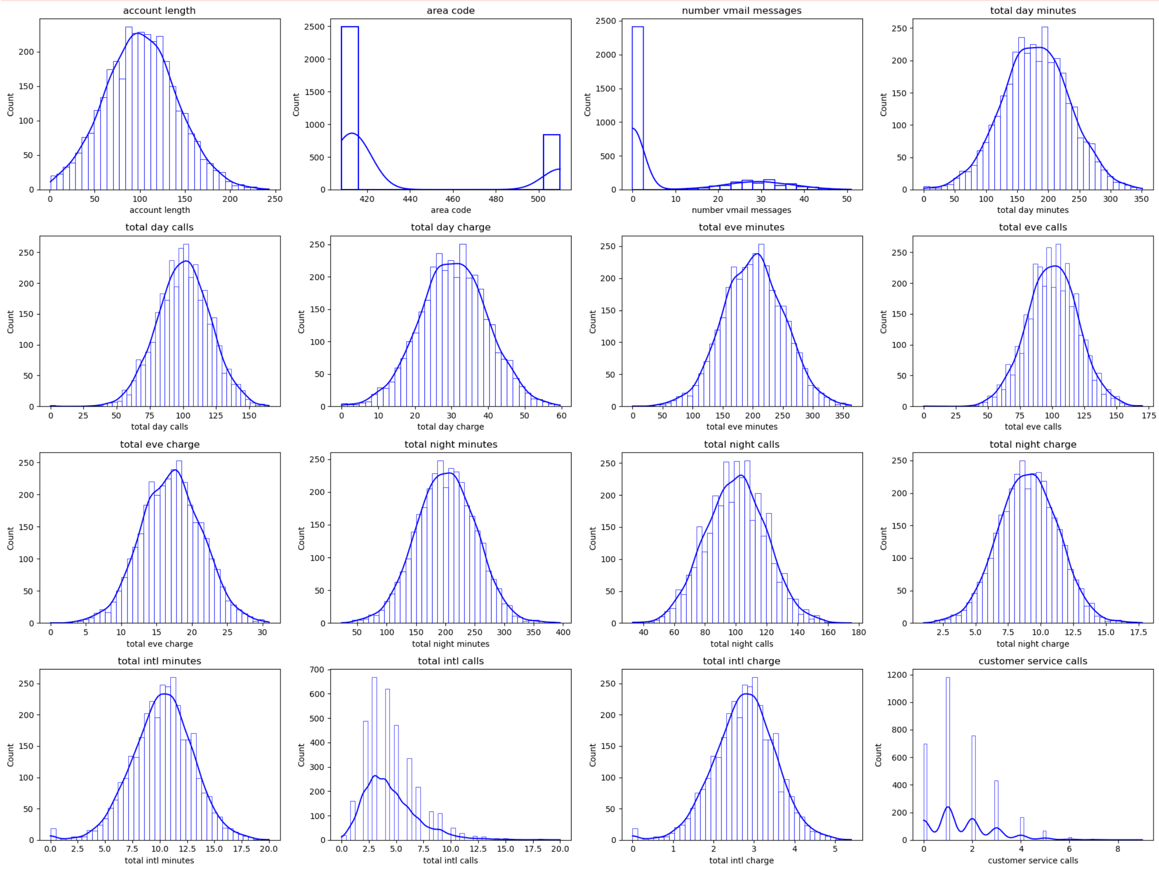
```
  5.4 13.8  8.1 13.  10.6  5.7  9.5  7.7 10.3 15.5 14.7 11.1 14.2 1
2.6
 11.8  8.3 14.5 10.5  9.4 14.6  9.2  3.5  8.5 13.2  7.4  8.8 11.
7.8
  6.8 11.4  9.3  9.7 10.2  8.   5.8 12.1 12.  11.6  8.2  6.2  7.3
6.1
 11.7 15.   9.8 12.4  8.6 10.9 13.9  8.9  7.9  5.3  4.4 12.5 11.3
9.
  9.6 13.3 20.   7.2  6.4 14.1 14.3  6.9 11.5 15.8 12.8 16.2  0.  1
1.9
  9.9  8.4 10.8 13.4 10.7 17.6  4.7  2.7 13.5 12.9 14.4 10.4  6.7 1
5.4
  4.5  6.5 15.6  5.9 18.9  7.6  5.   7.  14.  18.  16.  14.8  3.7
2.
  4.8 15.3  6.  13.6 17.2 17.5  5.6 18.2  3.6 16.5  4.6  5.1  4.1 1
6.3
 14.9 16.4 16.7  1.3 15.2 15.1 15.9  5.5 16.1  4.  16.9  5.2  4.2 1
5.7
 17.   3.9  3.8  2.2 17.1  4.9 17.9 17.3 18.4 17.8  4.3  2.9  3.1
3.3
  2.6  3.4  1.1 18.3 16.6  2.1  2.4  2.5]
total intl calls: [ 3  5  7  6  4  2  9 19  1 10 15  8 11  0 12 13 1
8 14 16 20 17]
total intl charge: [2.7  3.7  3.29 1.78 2.73 1.7  2.03 1.92 2.35 3.0
2 3.43 2.46 3.32 3.54
 1.46 3.73 2.19 3.51 2.86 1.54 2.57 2.08 2.78 4.19 3.97 3.   3.83 3.
4
 3.19 2.24 3.92 2.84 2.54 3.94 2.48 0.95 2.3  3.56 2.   2.38 2.97 2.
11
 1.84 3.08 2.51 2.62 2.75 2.16 1.57 3.27 3.24 3.13 2.21 1.67 1.97 1.
65
 3.16 4.05 2.65 3.35 2.32 2.94 3.75 2.4  2.13 1.43 1.19 3.38 3.05 2.
43
 2.59 3.59 5.4  1.94 1.73 3.81 3.86 1.86 3.11 4.27 3.46 4.37 0.   3.
21
 2.67 2.27 2.92 3.62 2.89 4.75 1.27 0.73 3.65 3.48 3.89 2.81 1.81 4.
16
 1.22 1.76 4.21 1.59 5.1  2.05 1.35 1.89 3.78 4.86 4.32 4.   1.   0.
54
 1.3  4.13 1.62 3.67 4.64 4.73 1.51 4.91 0.97 4.46 1.24 1.38 1.11 4.
4
 4.02 4.43 4.51 0.35 4.1  4.08 4.29 1.49 4.35 1.08 4.56 1.4  1.13 4.
24
 4.59 1.05 1.03 0.59 4.62 1.32 4.83 4.67 4.97 4.81 1.16 0.78 0.84 0.
89
 0.7  0.92 0.3  4.94 4.48 0.57 0.65 0.68]
customer service calls: [1 0 2 3 4 5 7 9 6 8]
```

In [291…

```
# checking the distribution of the numeric features
plt.figure(figsize=(20, 15))
for i, feature in enumerate(numeric_features):
    plt.subplot(4, 4, i + 1)  # Adjusted to a 4x4 grid to fit all 1…
    sns.histplot(data[feature], kde=True, fill=False, color='blue')
    plt.title(feature)
plt.tight_layout()
plt.show()
```

```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
```

```
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



The area code,number of voice mail messages and cusomers service calls are not normally distributed.This means

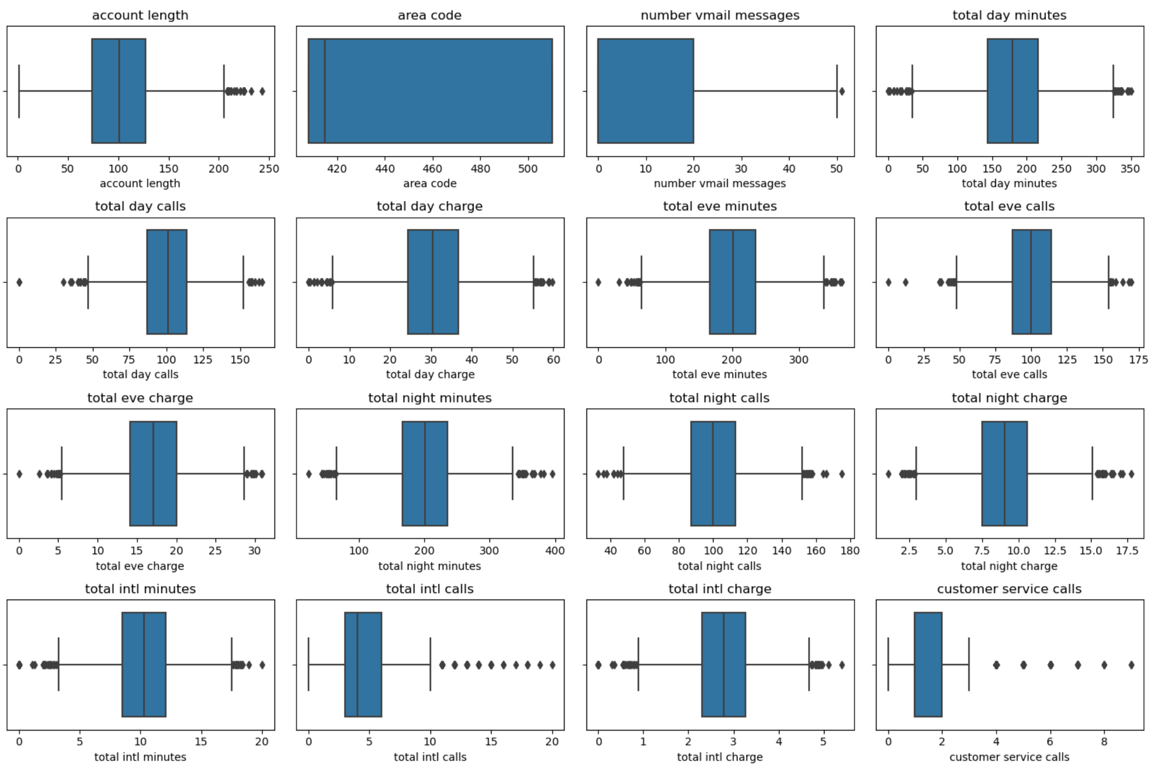
that we will need to scale the numeric features before we can use them in our model

Identify Outliers in the Numeric Features of the Data

- Identifying the outliers will help us to undertstand the data better.
- It will explain any data points that are far away from the rest of the data points

In [292…

```
# Find out any outliers in the numeric features
plt.figure(figsize=(15, 10))
for i, feature in enumerate(numeric_features):
    plt.subplot(4, 4, i + 1)  # Adjusted to a 4x4 grid to fit all 1
    sns.boxplot(x=data[feature])
    plt.title(feature)
plt.tight_layout()
plt.show()
```



- Most of the numeric features have outliers except area code
- This means that the data is not normally distributed and therefore we may drop and reduce the data points that are an outlier or replace their data

with the mean or median

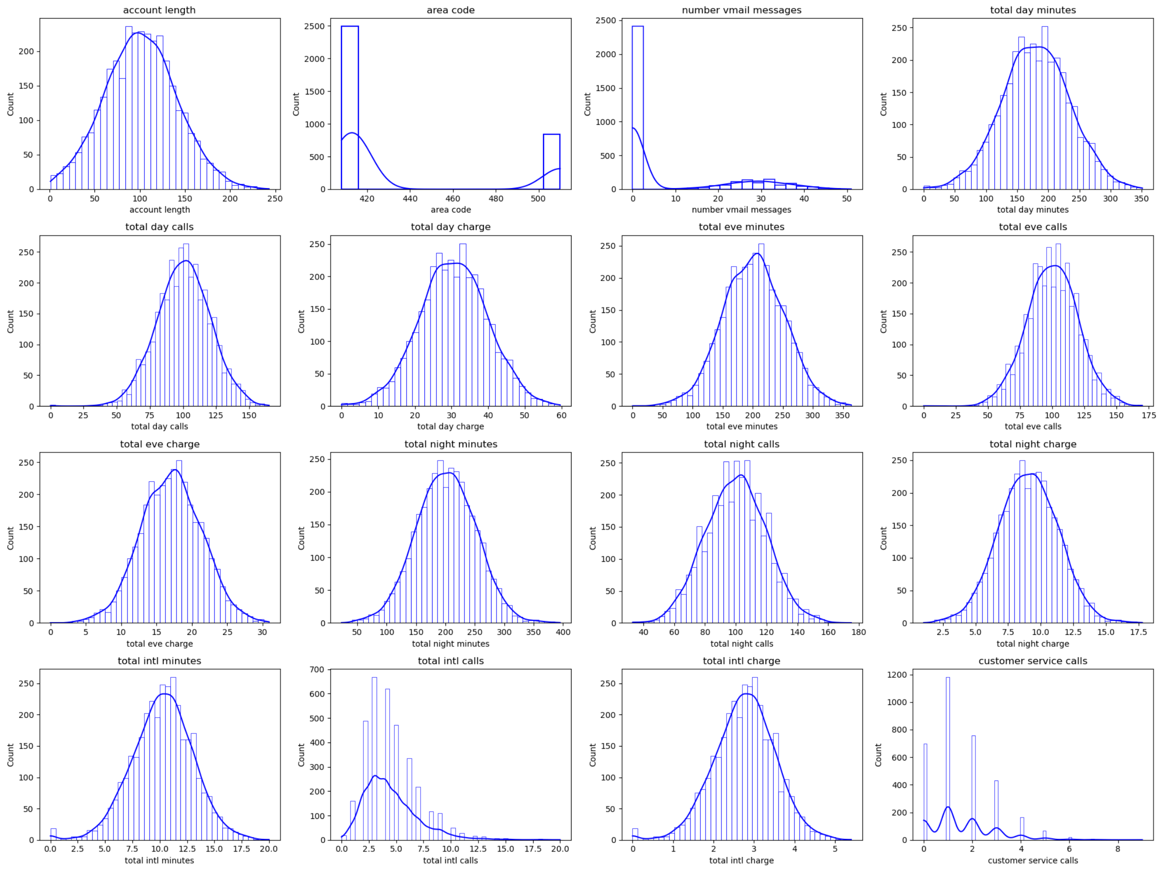
We can further anlysse the distribution nature of the numerical data as follows;

In [293…

```
# distribution of numeric features
plt.figure(figsize=(20, 15))
for i, feature in enumerate(numeric_features):
    plt.subplot(4, 4, i + 1)  # Adjusted to a 4x4 grid to fit all 1
    sns.histplot(data[feature], kde=True, fill=False, color='blue')
    plt.title(feature)
plt.tight_layout()
plt.show()
```

```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
```

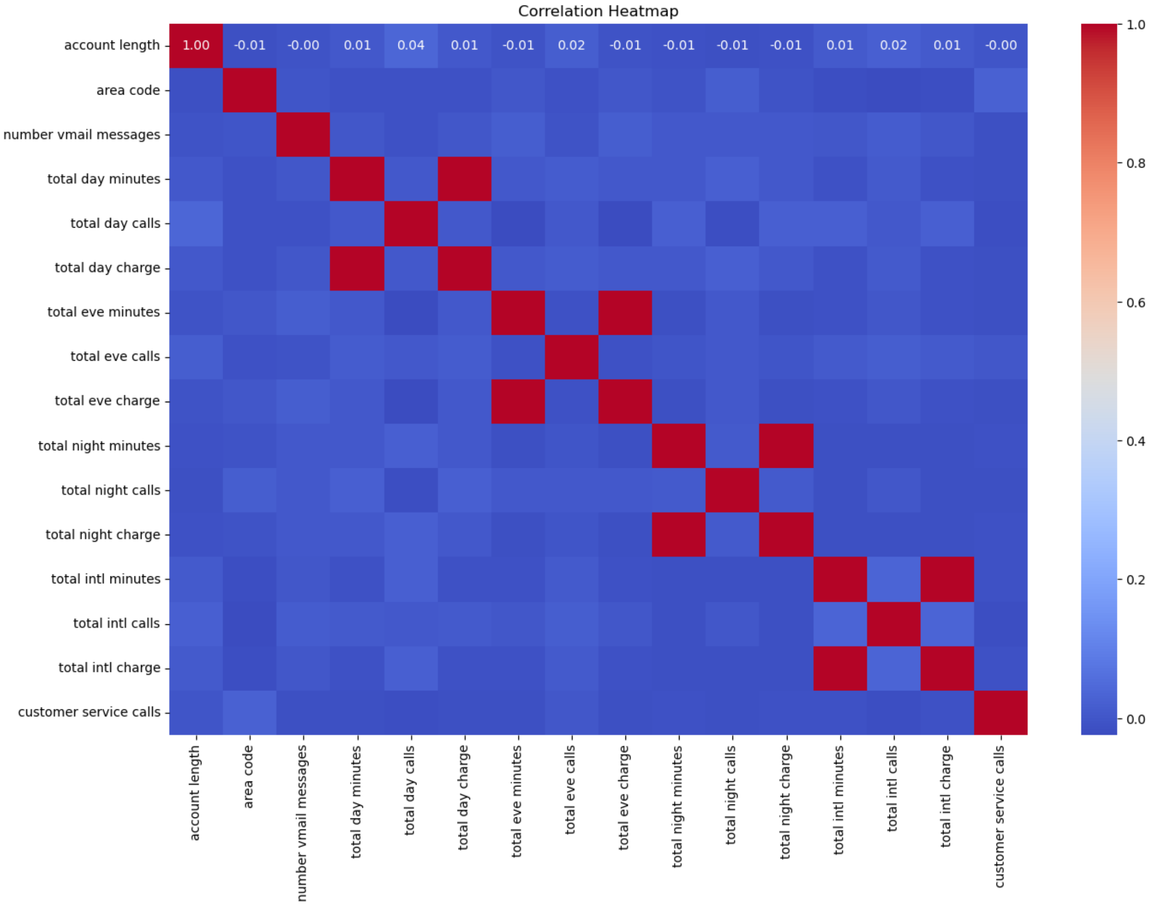
```
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will be rem
oved in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



- All numeric features except customer service calls, have a normal distribution.
- Total international calls seems to be skewed to the right side however it is still normally distributed.
- Customer service calls has a few peaks, which indicates there are a few modes in the population. This is so because customer service calls has to be a integer and not a float number.

In [294…

```
# Creating a heatmap to show the correlation between the numeric fe
plt.figure(figsize=(15, 10))
sns.heatmap(data[numeric_features].corr(), annot=True, cmap='coolwa
plt.title('Correlation Heatmap')
plt.show()
```



In [295…

```
# We can check for the skewness of the numeric features
skewness = data[numeric_features].skew()
skewness = skewness[abs(skewness) > 0.5]  # Filter for skewed featu
print("Skewed Features:")
print(skewness)
```

```
Skewed Features:
area code                 1.126823
number vmail messages     1.264824
total intl calls          1.321478
customer service calls    1.091359
dtype: float64
```

Dealing with Outliers in the numerical features

In [302…

```
# We can deal with outliers of the numeric features by using the z-
z_scores = np.abs(stats.zscore(data[numeric_features]))
threshold = 3
outliers = np.where(z_scores > threshold)
print("Outliers detected at indices:")
print(outliers)
```

```
Outliers detected at indices:
(array([  22,   32,   32,   41,   58,  115,  115,  179,  179,  182,
185,
         219,  244,  244,  272,  301,  314,  314,  329,  332,  343,
343,
```

```
         365,  365,  377,  416,  468,  474,  483,  488,  488,  493,
504,
         514,  522,  533,  533,  542,  595,  595,  636,  642,  646,
674,
         692,  694,  712,  712,  721,  740,  756,  762,  762,  778,
817,
         821,  821,  837,  845,  854,  863,  878,  878,  883,  883,
883,
         889,  889,  902,  908,  921,  922,  922,  957,  960,  974,
982,
         985,  985, 1021, 1028, 1028, 1052, 1052, 1080, 1080, 1092, 1
113,
        1113, 1121, 1142, 1144, 1179, 1233, 1233, 1260, 1260, 1273, 1
317,
        1317, 1325, 1333, 1345, 1345, 1345, 1355, 1392, 1397, 1397, 1
397,
        1400, 1400, 1407, 1408, 1419, 1445, 1445, 1502, 1551, 1564, 1
564,
        1567, 1615, 1638, 1694, 1751, 1831, 1865, 1886, 1889, 1912, 1
919,
        1986, 1986, 1989, 2001, 2212, 2223, 2269, 2288, 2321, 2321, 2
327,
        2331, 2331, 2345, 2345, 2362, 2362, 2380, 2387, 2428, 2513, 2
513,
        2551, 2551, 2553, 2594, 2594, 2621, 2659, 2663, 2663, 2669, 2
669,
        2703, 2716, 2732, 2732, 2733, 2733, 2736, 2736, 2753, 2753, 2
775,
        2786, 2835, 2887, 2903, 2906, 2906, 2918, 2918, 2930, 2932, 2
932,
        2932, 2947, 2953, 2956, 2958, 2961, 2970, 2979, 2988, 3025, 3
026,
        3071, 3081, 3107, 3107, 3109, 3112, 3187, 3190, 3206, 3211, 3
216,
        3219, 3230, 3247, 3247, 3275, 3275, 3290, 3290, 3291, 3310]),
array([13,  6,  8, 13,  7, 12, 14, 12, 14, 13, 13, 13,  9, 11, 13,
7, 12,
        14, 13, 15, 12, 14,  3,  5, 13,  0,  4, 13, 13, 12, 14, 10, 1
3, 13,
        15,  6,  8, 15, 12, 14, 13, 13,  7, 13,  4, 15, 12, 14, 15,
4, 13,
        12, 14, 15,  0,  6,  8, 13,  2, 13, 13, 12, 14,  9, 11, 13,
6,  8,
        15, 15, 13,  9, 11, 13,  7, 15, 13,  3,  5, 13, 12, 14,  3,
5, 12,
        14, 13,  9, 11,  4, 15,  4, 13,  6,  8,  9, 11, 15,  9, 11, 1
5, 13,
         3,  4,  5, 13, 13,  3,  4,  5, 12, 14, 15,  0, 13,  9, 11, 1
5,  0,
        12, 14, 13,  7, 15, 15,  0, 15, 15,  0, 13, 15, 15,  3,  5,
4, 13,
        13, 15, 13, 10,  9, 11, 15,  6,  8, 12, 14, 12, 14, 15, 15, 1
5, 12,
        14,  6,  8, 15,  3,  5, 13, 10,  9, 11, 12, 14, 13,  2,  6,
8, 12,
        14,  3,  5,  3,  5, 13, 15, 13,  2, 10, 12, 14, 12, 14, 13,
```

```
6,  7,
        8, 13, 15, 13, 15, 15, 13, 15, 10, 13, 15, 13, 15,  9, 11, 1
3, 15,
        4, 15, 13, 10,  0,  7, 13,  9, 11, 12, 14, 12, 14, 13, 13]))
```

In [297…

```
# We can drop the outliers from the dataset
data_cleaned = data[(z_scores < threshold).all(axis=1)]
print("Shape of cleaned data:", data_cleaned.shape)
print("Shape of original data:", data.shape)
```

```
Shape of cleaned data: (3169, 20)
Shape of original data: (3333, 20)
```

In [311…

```
print("The original dataframe has {} columns.".format(data.shape[1]
# Calculate the correlation matrix and take the absolute value
corr_matrix =(data[numeric_features]).corr().abs()

# Create a True/False mask and apply it
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
tri_df = corr_matrix.mask(mask)

# List column names of highly correlated features (r > 0.90)
to_drop = [c for c in tri_df.columns if any(tri_df[c] >  0.90)]

reduced_data = data.drop(to_drop, axis=1) # Drop the features
print("The reduced dataframe has {} columns.".format(reduced_data.sl
```

```
The original dataframe has 20 columns.
The reduced dataframe has 16 columns.
```

In [312…

```
reduced_data['churn'].value_counts()
```

Out[312…

```
churn
False    2850
True      483
Name: count, dtype: int64
```

Transfrom the 'churn' from False and True to 0s and 1's

In [313…

```
reduced_data['churn'] = reduced_data['churn'].map({True: 1, False: (
reduced_data.head()
```

Out[313…

	state	account length	area code	international plan	voice mail plan	number vmail messages	total day calls	total day charge	tota eve calls
0	KS	128	415	no	yes	25	110	45.07	99
1	OH	107	415	no	yes	26	123	27.47	103
2	NJ	137	415	no	no	0	114	41.38	110
3	OH	84	408	yes	no	0	71	50.90	88
4	OK	75	415	yes	no	0	113	28.34	122

In [ ]:

```
```

Categorical Features Analysis

We review and analyse the categorical features

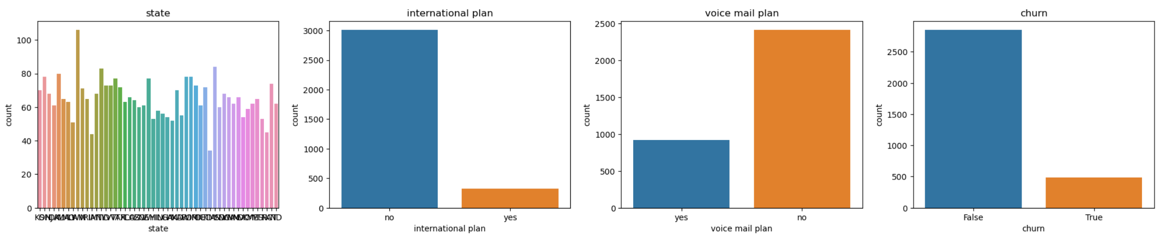
In [304…

```
# List of categorical features
categorical_features = ['state', 'area code', 'international plan',
categorical_features
```

Out[304… ['state', 'area code', 'international plan', 'voice mail plan']

In [298…

```
# checking the distribution of the categorical features
plt.figure(figsize=(20, 15))
for i, feature in enumerate(categorical_features):
    plt.subplot(4, 4, i + 1)  # Adjusted to a 4x4 grid to fit all 1
    sns.countplot(x=feature, data=data,)
    plt.title(feature)
plt.tight_layout()
plt.show()
```



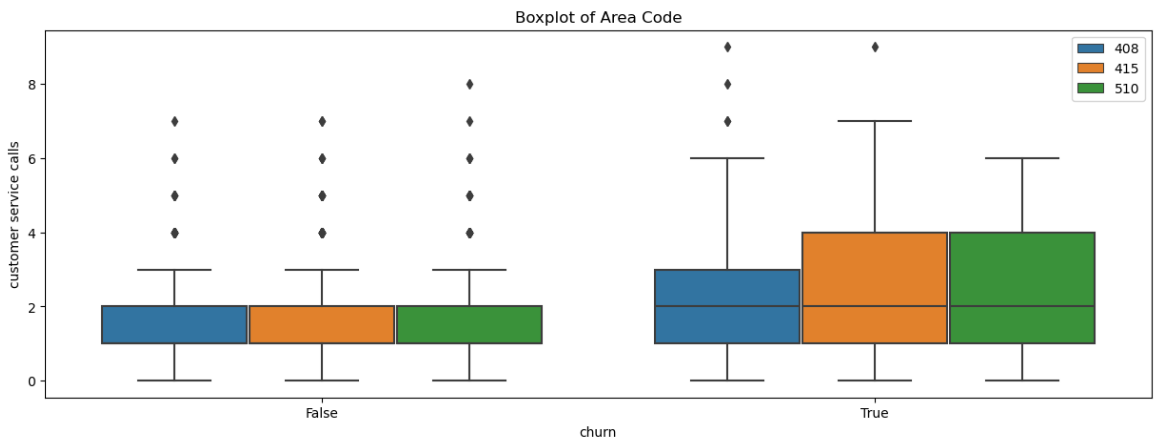
From the above we can see that the categorical features have imbalanced data.This means that the data is not evenly distributed across the different categories. This can lead to biased results in machine learning models, as

the model may be more likely to predict the majority class.This will be corrected later in the analysis by using the SMOTE technique to oversample the minority class and undersample the majority class.

In [299…
```
# the Analysis of Area code
# checking the distribution of the area code
Area_code = data['area code'].value_counts()
Area_code
```

Out[299…
```
area code
415    1655
510     840
408     838
Name: count, dtype: int64
```

In [300…
```
# Assess the boxplot to identify the outliers in the area code
plt.figure(figsize=(15, 5))
sns.boxplot(x=data['churn'], y=data['customer service calls'], hue=
plt.title('Boxplot of Area Code')
plt.xlabel('churn')
plt.ylabel('customer service calls')
plt.legend(loc='upper right');
plt.show()
```



- There are outliers, in all area codes, amongst the customers who have not terminated their accounts ie not left Syriatell.
- The customers who have terminated their account with Syriatel are in area code 415 and 510.,

OneHot Encoding for categorical data

We shall transform the categorical features into dummy variables as 0 and 1 to be able to use them in classification models.

In [314…
```
# Onehot Encoding for categorical features
dummy_data = pd.get_dummies(reduced_data, columns=categorical_featu
dummy_data.head()
```

Out[314…

	account length	number vmail messages	total day calls	total day charge	total eve calls	total eve charge	total night calls	total night charge	total intl calls	ch
0	128	25	110	45.07	99	16.78	91	11.01	3	
1	107	26	123	27.47	103	16.62	103	11.45	3	
2	137	0	114	41.38	110	10.30	104	7.32	5	
3	84	0	71	50.90	88	5.26	89	8.86	7	
4	75	0	113	28.34	122	12.61	121	8.41	3	

5 rows × 66 columns

Scaling the Numerical Features

In [316…
```
# scaling numerical features
Transformer = MinMaxScaler()

# Filter numeric_features to include only columns present in dummy_
numeric_features = [feature for feature in numeric_features if featu

# Fit the transformer to the data
Transformer.fit(dummy_data[numeric_features])
# Transform the data
scaled_data = Transformer.transform(dummy_data[numeric_features])
# Convert the transformed data back to a DataFrame
scaled_data = pd.DataFrame(scaled_data, columns=numeric_features)
# Concatenate the scaled data with the remaining columns in dummy_d
dummy_data = pd.concat([scaled_data, dummy_data.drop(columns=numeri
dummy_data.head()
```

Out[316…

	account length	number vmail messages	total day calls	total day charge	total eve calls	total eve charge	total night calls
0	0.524793	0.490196	0.666667	0.755701	0.582353	0.542866	0.408451
1	0.438017	0.509804	0.745455	0.460597	0.605882	0.537690	0.492958
2	0.561983	0.000000	0.690909	0.693830	0.647059	0.333225	0.500000
3	0.342975	0.000000	0.430303	0.853454	0.517647	0.170171	0.394366
4	0.305785	0.000000	0.684848	0.475184	0.717647	0.407959	0.619718

5 rows × 66 columns

In [ ]:

3.0 Modelling

Since this is a classifcation of binary data,we shall use two claffiers ie the logistic model and decision tree classification

In [343…

```
#  basemodeling using logistic regression for numeric features

# Logistic Regression model
X = dummy_data.drop('churn', axis=1)
y = dummy_data['churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
y_pred = log_reg.predict(X_test)
# Model evaluation
print("Logistic Regression Model Evaluation:")
```

```
Logistic Regression Model Evaluation:
```

In [ ]:

4.0 Model Evaluation

```
# 4.1 Logistic Regression Model Evaluation
```

In [324…

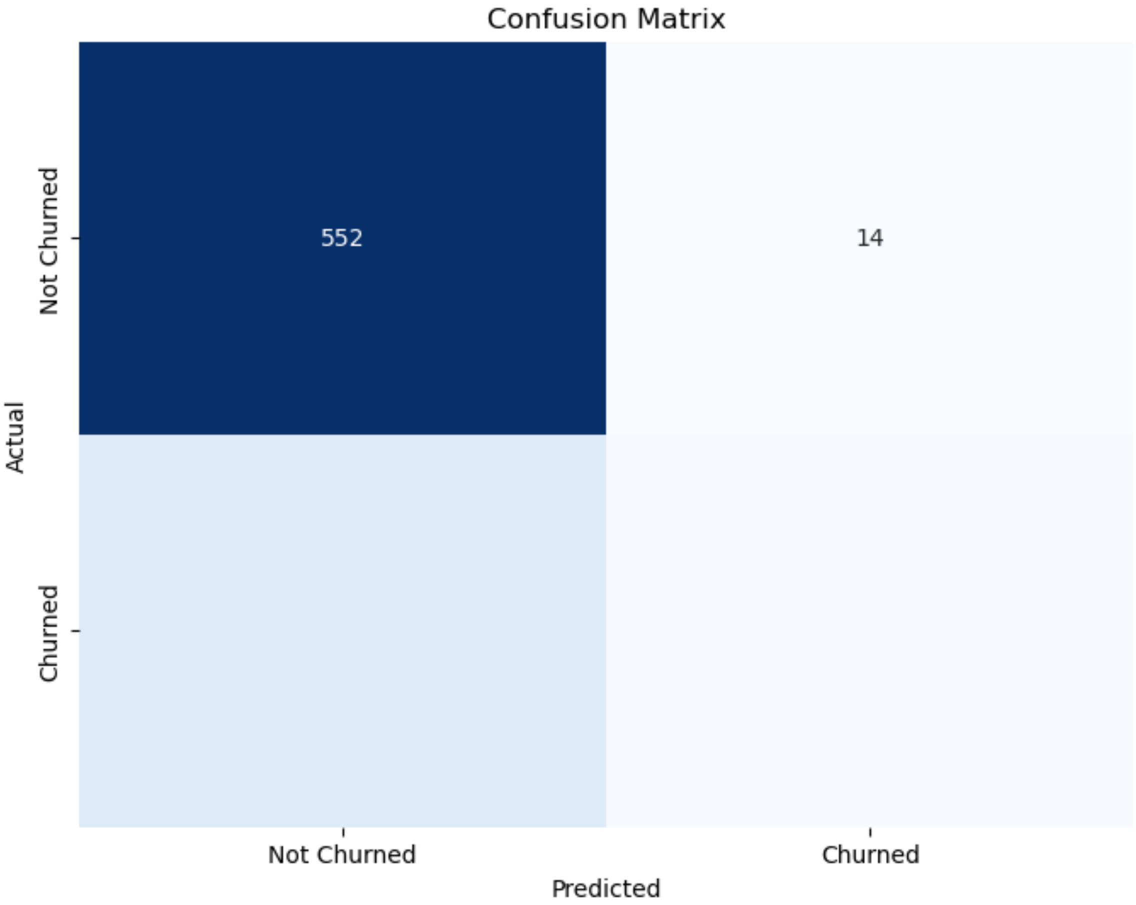
```
# Predicting the test set results
y_pred = log_reg.predict(X_test)
# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
# F1 Score
f1 = f1_score(y_test, y_pred)
```

```
print("F1 Score:", f1)
# Recall
recall = recall_score(y_test, y_pred)
print("Recall:", recall)
# Precision
precision = precision_score(y_test, y_pred)
print("Precision:", precision)
```

```
Accuracy: 0.8530734632683659
F1 Score: 0.25757575757575757
Recall: 0.16831683168316833
Precision: 0.5483870967741935
```

In [327…

```
# confusion_matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False, xtic
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



In [328…

```
# classification_report
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=['Not Chur
```

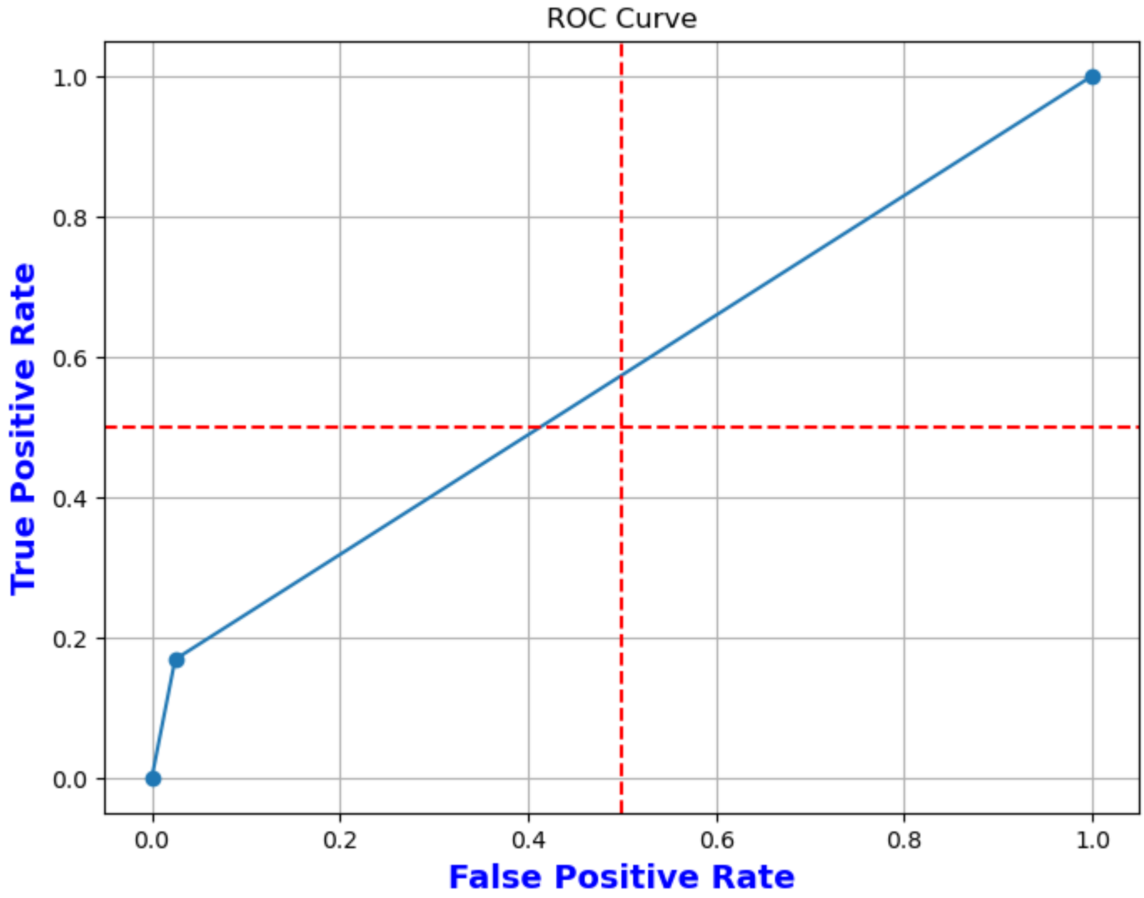
```
Classification Report:
              precision    recall  f1-score   support

 Not Churned       0.87      0.98      0.92       566
     Churned       0.55      0.17      0.26       101

    accuracy                           0.85       667
   macro avg       0.71      0.57      0.59       667
weighted avg       0.82      0.85      0.82       667
```

From the above classification report we can see that the model is not performing well:

1. The accuracy is 0.85, which is not very high.This means that the model is not able to predict the churned customers accurately.
2. The F1 score is 0.26, which is also not very high. This means that the model is not able to balance precision and recall well.
3. The recall is 0.17, which means that the model is able to identify 17% of the churned customers correctly.
4. The precision is 0.55, which means that the model is able to identify 55% of the non-churned customers correctly.
5. The confusion matrix shows that the model is making a lot of false positives and false negatives.
6. The classification report shows that the model is not able to predict the churned customers accurately.
7. The model is not performing well, and we need to improve it.

In [329…

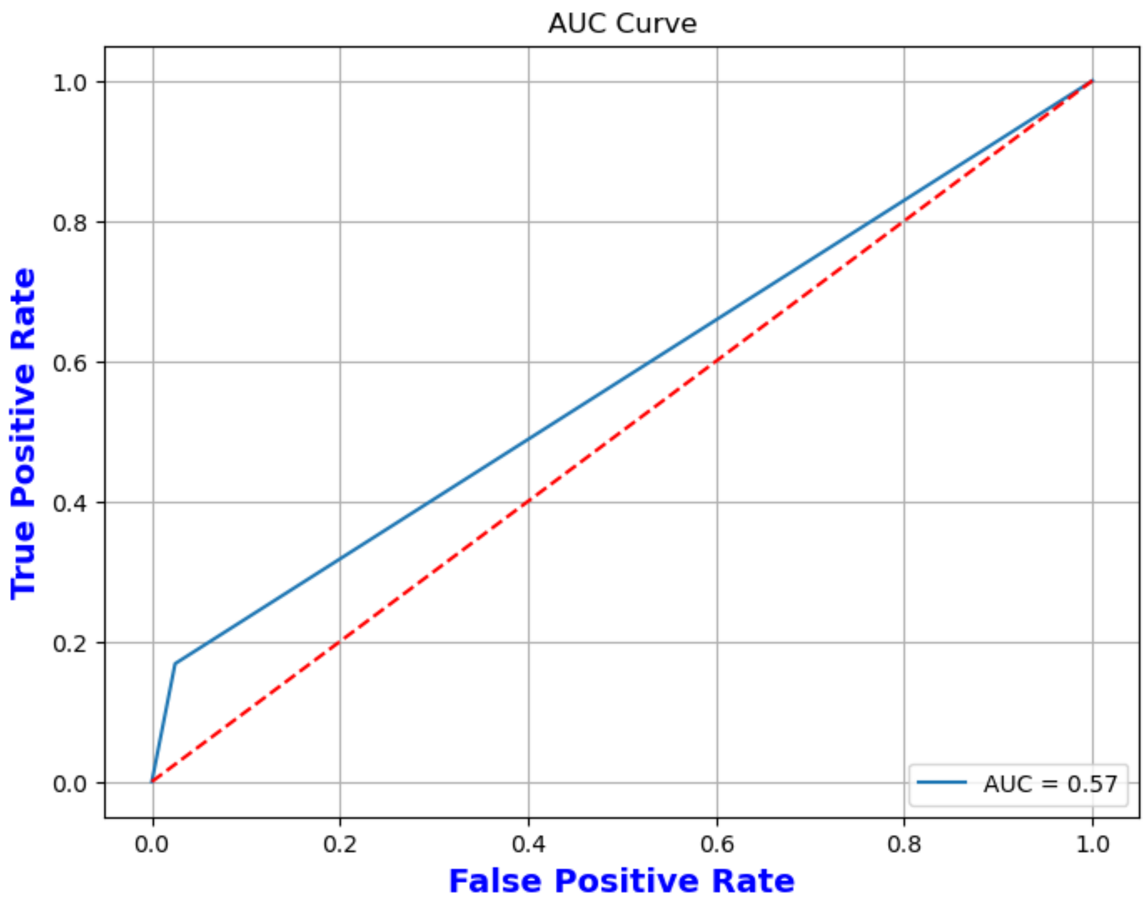
```
# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, marker='o')
plt.title('ROC Curve')
plt.xlabel('False Positive Rate', fontsize=14, fontweight='bold', c
plt.ylabel('True Positive Rate', fontsize=14, fontweight='bold', co
plt.axhline(y=0.5, color='r', linestyle='--')
plt.axvline(x=0.5, color='r', linestyle='--')
plt.grid()
plt.show()
```



In [330…
```
# AUC
auc = roc_auc_score(y_test, y_pred)
print("AUC:", auc)
```

```
AUC: 0.5717909246755064
```

In [331…
```
# AUC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label='AUC = %.2f' % auc)
plt.plot([0, 1], [0, 1], 'r--')
plt.title('AUC Curve')
plt.xlabel('False Positive Rate', fontsize=14, fontweight='bold', co
plt.ylabel('True Positive Rate', fontsize=14, fontweight='bold', co
plt.legend(loc='lower right')
plt.grid()
plt.show()
```



4.2 Decision Tree

In [336…

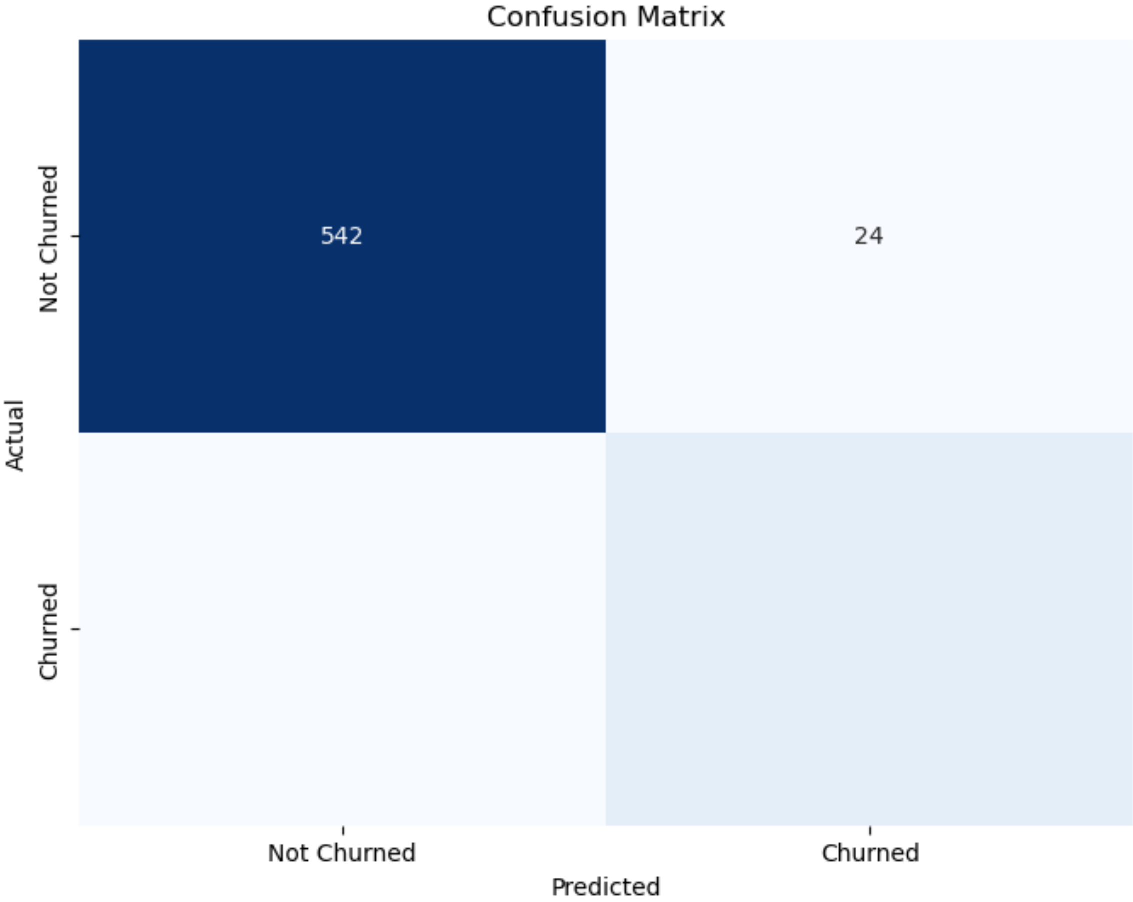
```
# Decision Tree Classifier model
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train, y_train)
y_pred_dt = dt_classifier.predict(X_test)

# Model evaluation
print("Decision Tree Classifier Model Evaluation:")

# Predicting the test set results
y_pred_dt = dt_classifier.predict(X_test)
# Accuracy
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print("Accuracy:", accuracy_dt)
print("F1 Score:", f1_score(y_test, y_pred_dt))
print("Recall:", recall_score(y_test, y_pred_dt))
print("Precision:", precision_score(y_test, y_pred_dt))
```

```
Decision Tree Classifier Model Evaluation:
Accuracy: 0.9250374812593704
F1 Score: 0.75
Recall: 0.7425742574257426
Precision: 0.7575757575757576
```

In [337…

```
# confusion_matrix
cm_dt = confusion_matrix(y_test, y_pred_dt)
plt.figure(figsize=(8, 6))
sns.heatmap(cm_dt, annot=True, fmt='d', cmap='Blues', cbar=False, x
```

```
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



In [338…

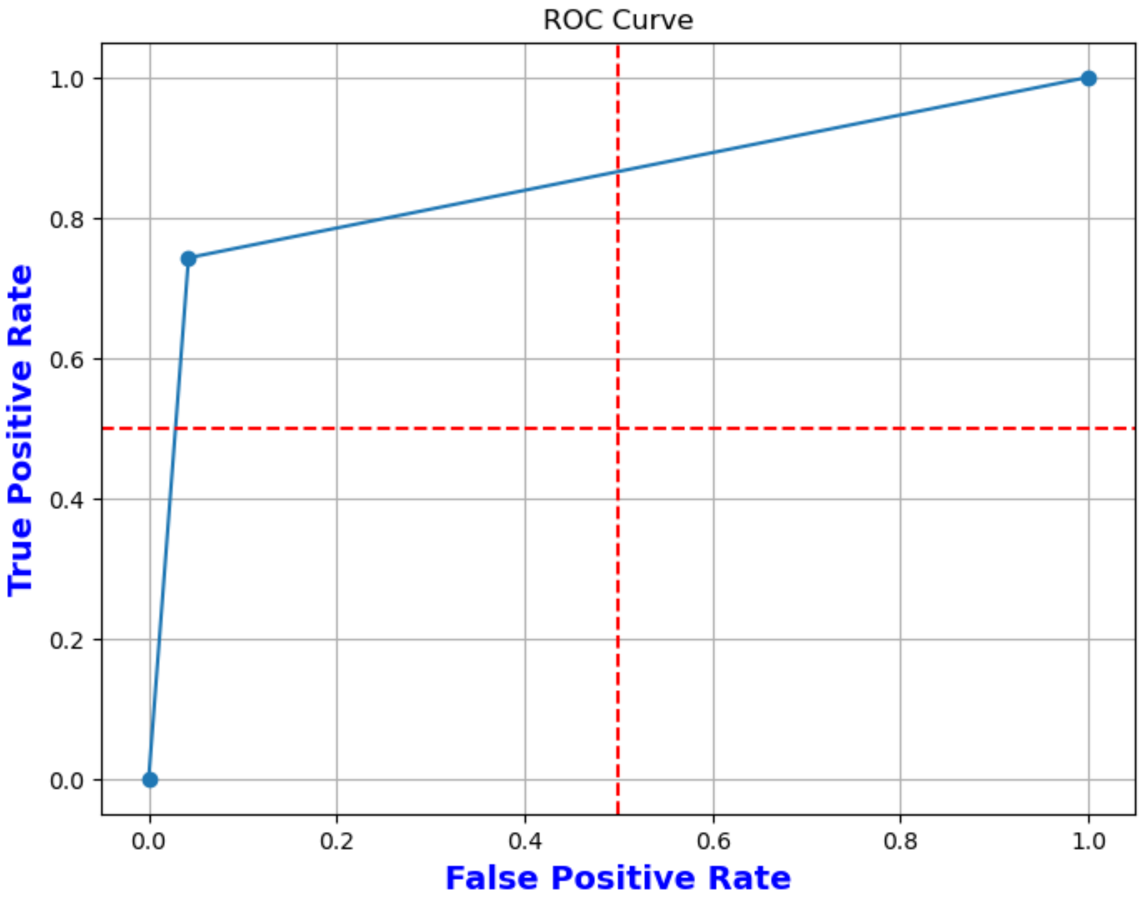
```
# classification_report
print("Classification Report:")
print(classification_report(y_test, y_pred_dt, target_names=['Not Ch
```

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

 Not Churned       0.95      0.96      0.96       566
     Churned       0.76      0.74      0.75       101

    accuracy                           0.93       667
   macro avg       0.86      0.85      0.85       667
weighted avg       0.92      0.93      0.92       667
```

In [339…

```
#ROC Curve
fpr_dt, tpr_dt, thresholds_dt = roc_curve(y_test, y_pred_dt)
plt.figure(figsize=(8, 6))
plt.plot(fpr_dt, tpr_dt, marker='o')
plt.title('ROC Curve')
plt.xlabel('False Positive Rate', fontsize=14, fontweight='bold', co
plt.ylabel('True Positive Rate', fontsize=14, fontweight='bold', co
plt.axhline(y=0.5, color='r', linestyle='--')
plt.axvline(x=0.5, color='r', linestyle='--')
plt.grid()
```

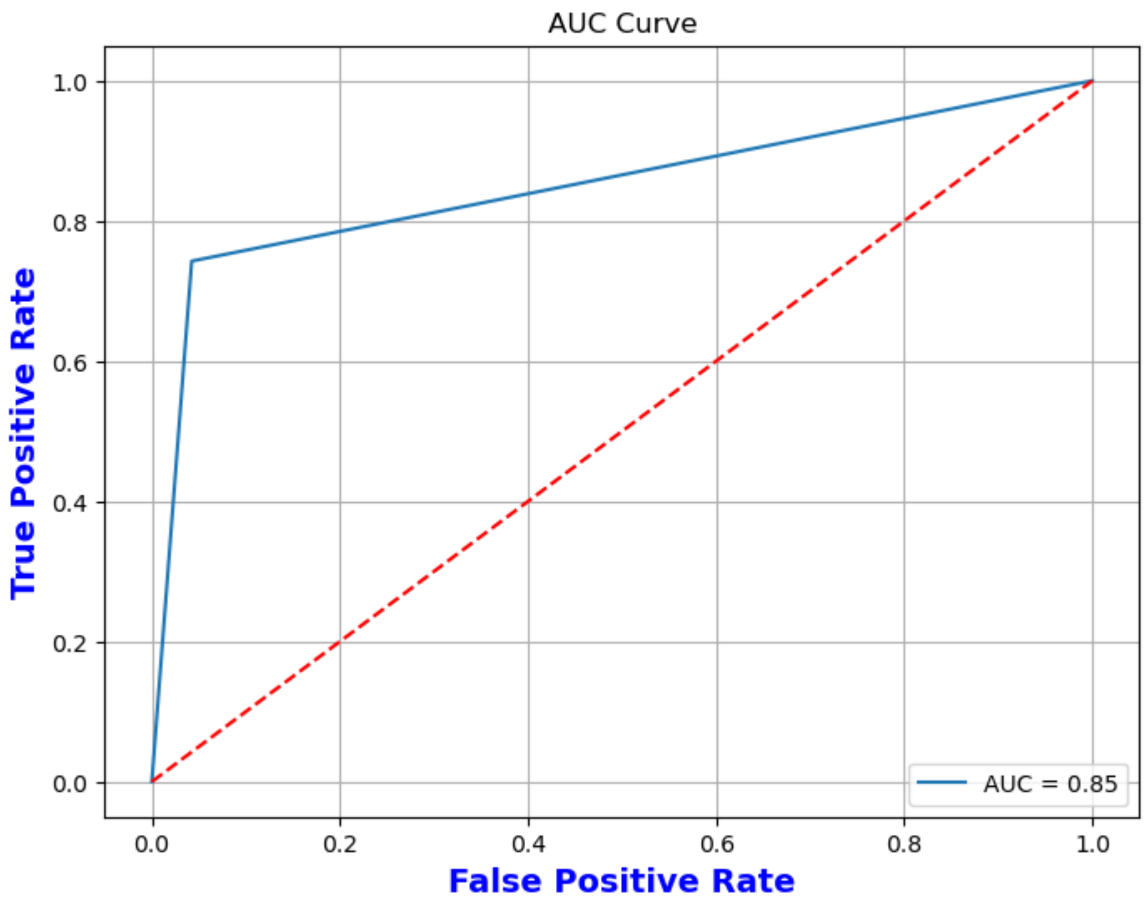
```
plt.show()
```



In [340…

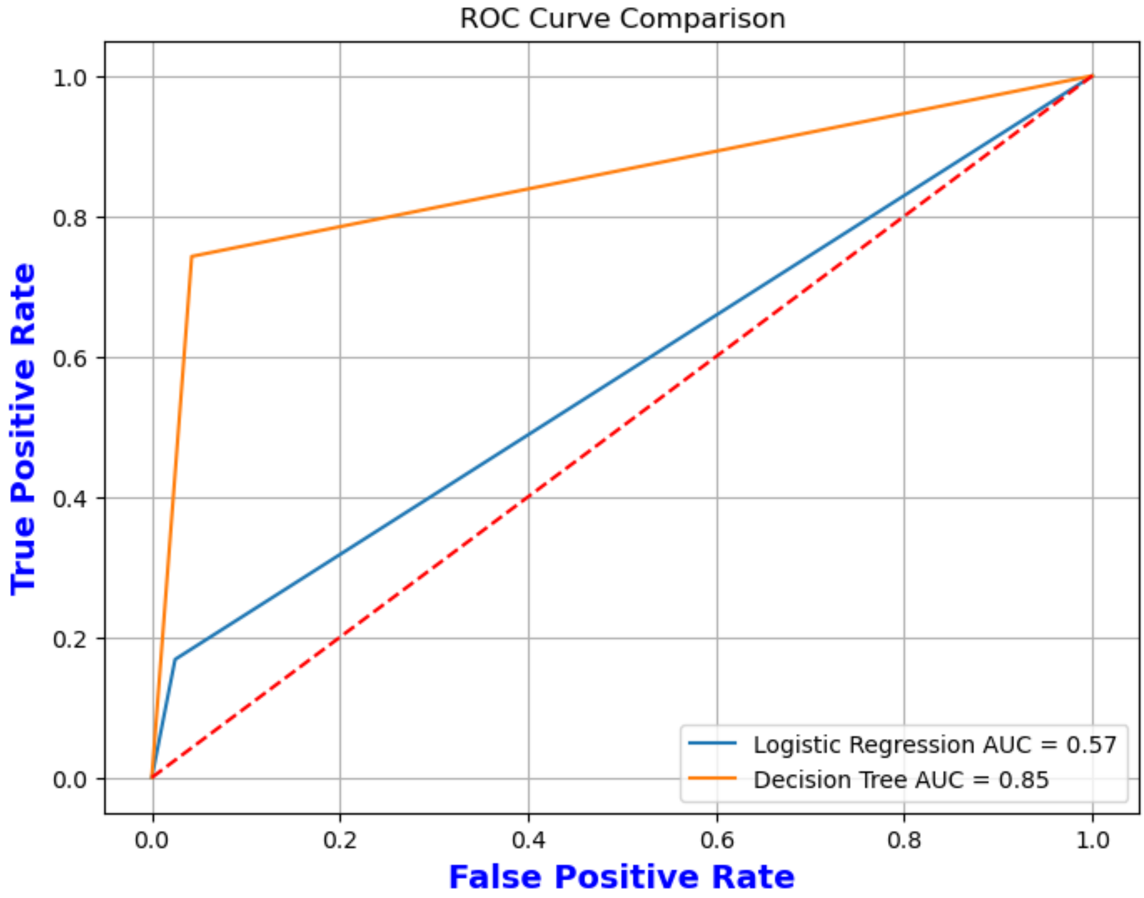
```
# AUC curve
auc_dt = roc_auc_score(y_test, y_pred_dt)
print("AUC:", auc_dt)
plt.figure(figsize=(8, 6))
plt.plot(fpr_dt, tpr_dt, label='AUC = %.2f' % auc_dt)
plt.plot([0, 1], [0, 1], 'r--')
plt.title('AUC Curve')
plt.xlabel('False Positive Rate', fontsize=14, fontweight='bold', c
plt.ylabel('True Positive Rate', fontsize=14, fontweight='bold', co
plt.legend(loc='lower right')
plt.grid()
plt.show()
```

```
AUC: 0.8500857152853095
```



In [344…

```
# combining ROC curve and AUC for Logistic regresssion and Decision
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label='Logistic Regression AUC = %.2f' % auc)
plt.plot(fpr_dt, tpr_dt, label='Decision Tree AUC = %.2f' % auc_dt)
plt.plot([0, 1], [0, 1], 'r--')
plt.title('ROC Curve Comparison')
plt.xlabel('False Positive Rate', fontsize=14, fontweight='bold', co
plt.ylabel('True Positive Rate', fontsize=14, fontweight='bold', co
plt.legend(loc='lower right')
plt.grid()
plt.show()
```



5.0 Conclusion

In [ ]:
```
# in conclusion the for churned vs not churned customers
```