1. BUSINESS UNDERSTANDING

SyriaTel is a telecommunications company based in Syria. It is one of the largest and most established telecom providers in the country. SyriaTel offers a range of services, including:

Mobile Telephony: Providing cellular services such as voice calls, text messaging, and mobile internet.

Fixed-Line Services: Offering traditional landline phone services.

Internet Services: Including broadband and data services for both residential and business customers.

Value-Added Services: Such as mobile banking, entertainment services, and other digital solutions.

SyriaTel plays a crucial role in the telecommunications infrastructure of Syria, serving a broad customer base and contributing to the connectivity and communication needs of individuals and businesses in the region.

2. PROBLEM STATEMENT

SyriaTel aims to enhance its customer retention efforts by understanding and predicting customer churn. The goal is to create a machine learning model that predicts the likelihood of a customer discontinuing their services in the near future. By identifying high-risk customers early, SyriaTel can tailor personalized retention offers, improve customer satisfaction, and reduce revenue loss associated with churn.

This model will help the company proactively implement targeted retention strategies, thereby minimizing revenue loss and improving customer retention.

3.Objectives

1.Predict Churn:

Develop a classifier to predict whether a customer will churn within the next [specific time frame, e.g., 3 months] based on their historical data and behavior.

2.Identify Risk Factors:

Determine the key factors and patterns associated with higher churn probability to inform targeted retention strategies.

3.Improve Retention Strategies:

Provide actionable insights to SyriaTel for creating effective retention campaigns, such as personalized offers or improved customer service interventions.

4.Data Understanding:

This project uses the SyriaTel dataset whose size, descripritive statistics for all the features used in the analysis, and justification of the inclusion of features based on their properties and relevance for the project have been given in the Exploratory Data Analysis part.

The data contains all the relevant features that are needed to meet our objectives as described above.

However, to explain the relationship fully, additional features such as customer complains are needed.

Imports and Data Loading

In [430…

```
#Import Relevant Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
%matplotlib inline
```

In [431…

```
#Import dataset
df = pd.read_csv(r'C:\Users\David\PHASE 3 PROJECT\PHASE-3-PROJECT\bigml_59c28831336c660
print(df)
```

```
     state  account length  area code phone number international plan  \
0       KS             128        415     382-4657                 no   
1       OH             107        415     371-7191                 no   
2       NJ             137        415     358-1921                 no   
3       OH              84        408     375-9999                yes   
4       OK              75        415     330-6626                yes   
...    ...             ...        ...          ...                ...   
3328    AZ             192        415     414-4276                 no   
3329    WV              68        415     370-3271                 no   
3330    RI              28        510     328-8230                 no   
3331    CT             184        510     364-6381                yes   
3332    TN              74        415     400-4344                 no   

     voice mail plan  number vmail messages  total day minutes  \
0                yes                     25              265.1   
1                yes                     26              161.6   
2                 no                      0              243.4   
3                 no                      0              299.4   
4                 no                      0              166.7   
...              ...                    ...                ...   
3328             yes                     36              156.2   
3329              no                      0              231.1   
3330              no                      0              180.8   
3331              no                      0              213.8   
3332             yes                     25              234.4   

      total day calls  total day charge  ...  total eve calls  \
0                 110             45.07  ...               99   
1                 123             27.47  ...              103   
2                 114             41.38  ...              110   
3                  71             50.90  ...               88   
4                 113             28.34  ...              122   
...               ...               ...  ...              ...   
3328               77             26.55  ...              126   
3329               57             39.29  ...               55   
```

```
3330           109         30.74  ...             58
3331           105         36.35  ...             84
3332           113         39.85  ...             82

      total eve charge  total night minutes  total night calls  \
0                16.78                244.7                 91   
1                16.62                254.4                103   
2                10.30                162.6                104   
3                 5.26                196.9                 89   
4                12.61                186.9                121   
...                ...                  ...                ...   
3328             18.32                279.1                 83   
3329             13.04                191.3                123   
3330             24.55                191.9                 91   
3331             13.57                139.2                137   
3332             22.60                241.4                 77   

      total night charge  total intl minutes  total intl calls  \
0                  11.01                10.0                 3   
1                  11.45                13.7                 3   
2                   7.32                12.2                 5   
3                   8.86                 6.6                 7   
4                   8.41                10.1                 3   
...                  ...                 ...               ...   
3328               12.56                 9.9                 6   
3329                8.61                 9.6                 4   
3330                8.64                14.1                 6   
3331                6.26                 5.0                10   
3332               10.86                13.7                 4   

      total intl charge  customer service calls  churn  
0                  2.70                       1  False  
1                  3.70                       1  False  
2                  3.29                       0  False  
3                  1.78                       2  False  
4                  2.73                       3  False  
...                 ...                     ...    ...  
3328               2.67                       2  False  
3329               2.59                       3  False  
3330               3.81                       2  False  
3331               1.35                       2  False  
3332               3.70                       0  False  

[3333 rows x 21 columns]
```

Exploratory data analysis(EDA)

In [432…

```
# view dimensions of dataset
df.shape
```

Out[432…

```
(3333, 21)
```

There are 3333 instances and 21 variables in the data set.

In [433…

```
# preview the dataset
df.head()
```

Out[433…

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	to c char
0	KS	128	415	382-	no	yes	25	265.1	110	45.07	...	99	16

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	to cha
				4657									
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12

5 rows × 21 columns

Display basic information

In [434…

```
# Get the data types of all columns
print(df.dtypes)
```

```
state                      object
account length              int64
area code                   int64
phone number               object
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
```

In [435…

```
#get the column names
col_names = df.columns

col_names
```

Out[435…

```
Index(['state', 'account length', 'area code', 'phone number',
       'international plan', 'voice mail plan', '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', 'churn'],
      dtype='object')
```

In [436…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   state                   3333 non-null   object 
 1   account length          3333 non-null   int64  
 2   area code               3333 non-null   int64  
 3   phone number            3333 non-null   object 
 4   international plan      3333 non-null   object 
 5   voice mail plan         3333 non-null   object 
 6   number vmail messages   3333 non-null   int64  
 7   total day minutes       3333 non-null   float64
 8   total day calls         3333 non-null   int64  
 9   total day charge        3333 non-null   float64
 10  total eve minutes       3333 non-null   float64
 11  total eve calls         3333 non-null   int64  
 12  total eve charge        3333 non-null   float64
 13  total night minutes     3333 non-null   float64
 14  total night calls       3333 non-null   int64  
 15  total night charge      3333 non-null   float64
 16  total intl minutes      3333 non-null   float64
 17  total intl calls        3333 non-null   int64  
 18  total intl charge       3333 non-null   float64
 19  customer service calls  3333 non-null   int64  
 20  churn                   3333 non-null   bool   
dtypes: bool(1), float64(8), int64(8), object(4)
memory usage: 524.2+ KB
```

In [437…

```
#summary of the statistics of the numerical columns
df.describe()
```

Out[437…

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	to
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.(
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.980348	100.
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.713844	19.!
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.(
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.600000	87.(
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.400000	100.(
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.300000	114.(
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.700000	170.(

In [438…

```
#find numerical variables
numerical = [var for var in df.columns if df[var].dtype!='O']

print('There are {} numerical variables\n'.format(len(numerical)))

print('The numerical variables are :', numerical)
```

```
There are 17 numerical variables

The numerical variables are : ['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', 'churn']
```

There are 17 numerical variables in the dataset

In [439…

```
#find categorical variables

categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(categorical)))

print('The categorical variables are :', categorical)
```

```
There are 4 categorical variables

The categorical variables are : ['state', 'phone number', 'international plan', 'voice mail plan']
```

In [440…

```
# view the categorical variables

df[categorical].head()
```

Out[440…

	state	phone number	international plan	voice mail plan
0	KS	382-4657	no	yes
1	OH	371-7191	no	yes
2	NJ	358-1921	no	no
3	OH	375-9999	yes	no
4	OK	330-6626	yes	no

5. Preprocess Data (Data cleaning)

a) Handling Missing Values

In [441…

```
# check missing values in categorical variables
df[categorical].isnull().sum()
```

Out[441…

```
state                 0
phone number          0
international plan    0
voice mail plan       0
dtype: int64
```

Summary of categorical variables

There are 4 categorical variables. These are given by state,international plan,voice mail plan and phone number.

There are two binary categorical variables - international plan and voice mail plan.

churn is the target variable.

There are no missing values in categorical variable.

b). Number of labels: cardinality

High cardinality may pose some serious problems in the machine learning model. So, I will check for high cardinality.

In [442...

```
# check for cardinality in categorical variables

for var in categorical:

    print(var, ' contains ', len(df[var].unique()), ' labels')
```

```
state  contains  51  labels
phone number  contains  3333  labels
international plan  contains  2  labels
voice mail plan  contains  2  labels
```

We can see that there is a phone number variable which needs to be preprocessed. If a phone number field has the highest cardinality (i.e., each phone number is unique or nearly unique), it presents challenges for preprocessing, especially for machine learning models where high cardinality features can lead to inefficiencies and overfitting

All the other variables contain relatively smaller number of variables. I will drop the phone number since its usually not used directly in analysis, as it's a unique identifier rather than a predictive feature.

In [443...

```
# Drop the 'phone number' column from the DataFrame
df = df.drop(columns=['phone number'])
```

c) Encoding Categorical Variables

Machine learning algorithms typically require numerical input to perform computations. Categorical data cannot be directly used in algorithms that are designed for numerical data (e.g., linear regression, logistic regression).

One-hot encoding transforms categorical variables into a binary (0 or 1) matrix, making them usable in machine learning models.

I will use one hot-encoder to convert the categorical variables.

In [444...

```
# Convert categorical variables to dummy variables
df_encoded = pd.get_dummies(df, columns=['state', 'area code', 'international plan', 'v
```

In [445...

```
# scale numeric features
scaler = StandardScaler()
df[['account length']] = scaler.fit_transform(df[['account length']])

print(df)
```

```
      state  account length  area code international plan voice mail plan  \
0        KS        0.676489        415                 no             yes
1        OH        0.149065        415                 no             yes
2        NJ        0.902529        415                 no              no
3        OH       -0.428590        408                yes              no
4        OK       -0.654629        415                yes              no
```

```
...    ...        ...     ...            ...          ...
3328    AZ   2.283878     415             no          yes
3329    WV  -0.830437     415             no           no
3330    RI  -1.835055     510             no           no
3331    CT   2.082955     510            yes           no
3332    TN  -0.679745     415             no          yes

      number vmail messages  total day minutes  total day calls  \
0                        25              265.1              110   
1                        26              161.6              123   
2                         0              243.4              114   
3                         0              299.4               71   
4                         0              166.7              113   
...                     ...                ...              ...   
3328                     36              156.2               77   
3329                      0              231.1               57   
3330                      0              180.8              109   
3331                      0              213.8              105   
3332                     25              234.4              113   

      total day charge  total eve minutes  total eve calls  total eve charge  \
0                45.07              197.4               99             16.78   
1                27.47              195.5              103             16.62   
2                41.38              121.2              110             10.30   
3                50.90               61.9               88              5.26   
4                28.34              148.3              122             12.61   
...                ...                ...              ...               ...   
3328             26.55              215.5              126             18.32   
3329             39.29              153.4               55             13.04   
3330             30.74              288.8               58             24.55   
3331             36.35              159.6               84             13.57   
3332             39.85              265.9               82             22.60   

      total night minutes  total night calls  total night charge  \
0                   244.7                 91               11.01   
1                   254.4                103               11.45   
2                   162.6                104                7.32   
3                   196.9                 89                8.86   
4                   186.9                121                8.41   
...                   ...                ...                 ...   
3328                279.1                 83               12.56   
3329                191.3                123                8.61   
3330                191.9                 91                8.64   
3331                139.2                137                6.26   
3332                241.4                 77               10.86   

      total intl minutes  total intl calls  total intl charge  \
0                   10.0                 3               2.70   
1                   13.7                 3               3.70   
2                   12.2                 5               3.29   
3                    6.6                 7               1.78   
4                   10.1                 3               2.73   
...                  ...               ...                ...   
3328                 9.9                 6               2.67   
3329                 9.6                 4               2.59   
3330                14.1                 6               3.81   
3331                 5.0                10               1.35   
3332                13.7                 4               3.70   

      customer service calls  churn  
0                          1  False  
1                          1  False  
2                          0  False  
3                          2  False  
4                          3  False  
```

```
...                    ...   ...
3328                     2  False
3329                     3  False
3330                     2  False
3331                     2  False
3332                     0  False

[3333 rows x 20 columns]
```

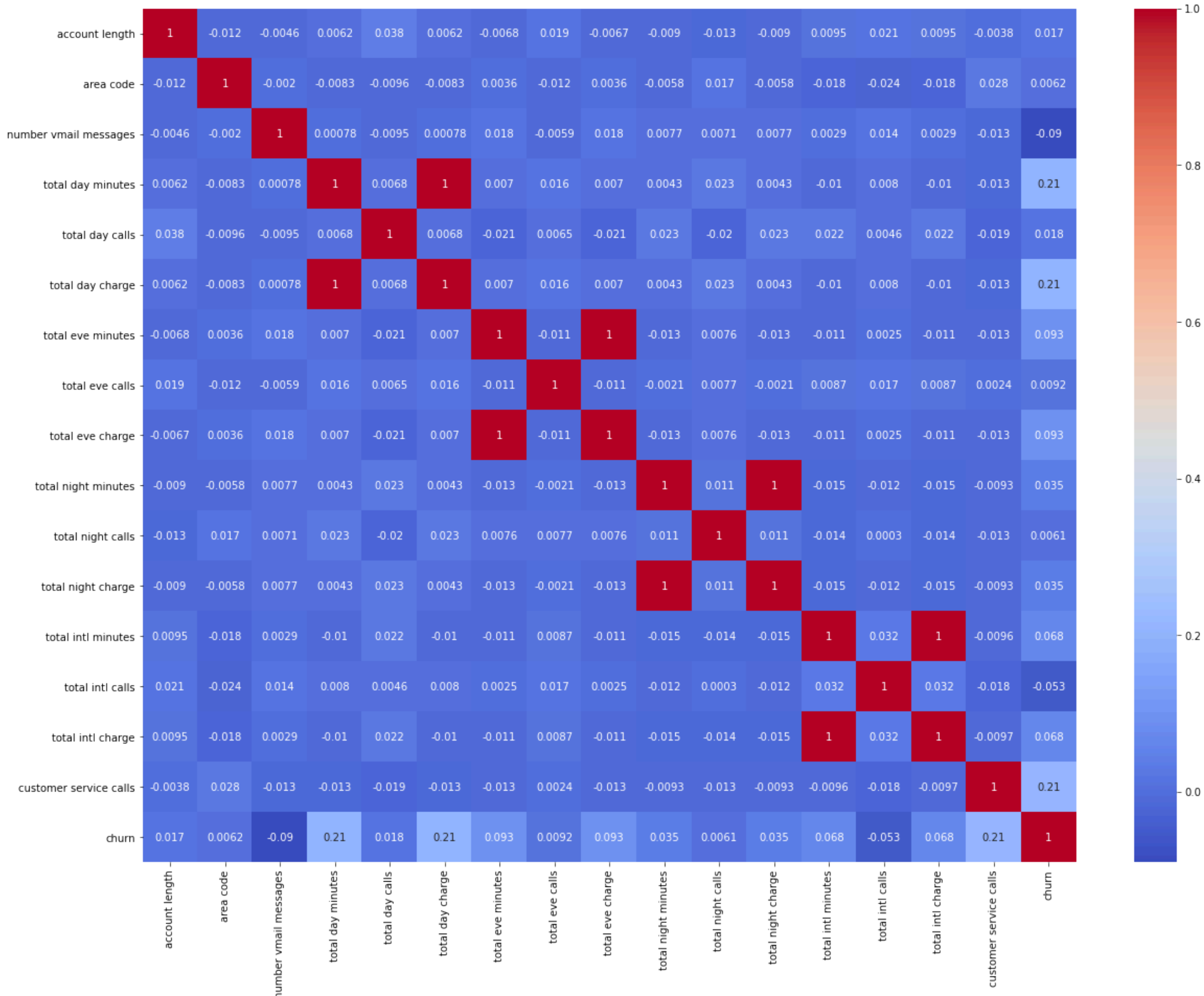
c. Data Preparation

Correlation Heatmap

A correlation heatmap to obtain numbers for an easier reading of the correlation of the relationship between columns

In [446…

```python
plt.subplots(figsize=(20,15))
sns.heatmap(df.corr(),cmap="coolwarm",annot=True);
```



Correlation measures the statistical relationship between pairs of features,

We have already determined that our predicted column is churn, and will, therefore, be looking at the correlation of the rest of the columns with churn.

1. There is a weak negative correlation of -0.053 between total intl calls and churn.This indicates a weak negative relationship between the number of international calls made by customers and

their likelihood to churn.Since the correlation is very close to zero, the total intl calls has only a minor and almost insignificant impact on the likelihood of a customer churning
2. There is a weak possitive correlation of 0.21 between customer service calls,total day charge,total day minutes and churn. This indicates a modest relationship where an increase in these features is slightly associated with an increase in the likelihood of customer churn.

Based on the correlation coefficients with churn from the Syria Tel, the most important features are:

1. Customer service calls
2. Total day charge
3. Total day minutes
4. Total intl calls

Checking the correlations for the selected important features

In [447…

```
data2 = df[['customer service calls','total day charge','total day minutes','total intl
data2.head()
```

Out[447…

	customer service calls	total day charge	total day minutes	total intl calls	churn
0	1	45.07	265.1	3	False
1	1	27.47	161.6	3	False
2	0	41.38	243.4	5	False
3	2	50.90	299.4	7	False
4	3	28.34	166.7	3	False

Correlation matrix

In [448…

```
# Compute the correlation matrix
correlation_matrix = data2.corr()
print(correlation_matrix)
```

```
                        customer service calls  total day charge  \
customer service calls                1.000000         -0.013427
total day charge                     -0.013427          1.000000
total day minutes                    -0.013423          1.000000
total intl calls                     -0.017561          0.008032
churn                                 0.208750          0.205151

                        total day minutes  total intl calls     churn
customer service calls          -0.013423         -0.017561  0.208750
total day charge                 1.000000          0.008032  0.205151
total day minutes                1.000000          0.008033  0.205151
total intl calls                 0.008033          1.000000 -0.052844
churn                            0.205151         -0.052844  1.000000
```

c. Data Analysis

Separate Features and Target Variable

In [449…

```
# Define features and target variable
X = df_encoded.drop(columns=['churn'])
```

```
y = df_encoded['churn']
```

d. Normalize/Scale Numeric Features

Scaling numeric features can improve the performance of some models:

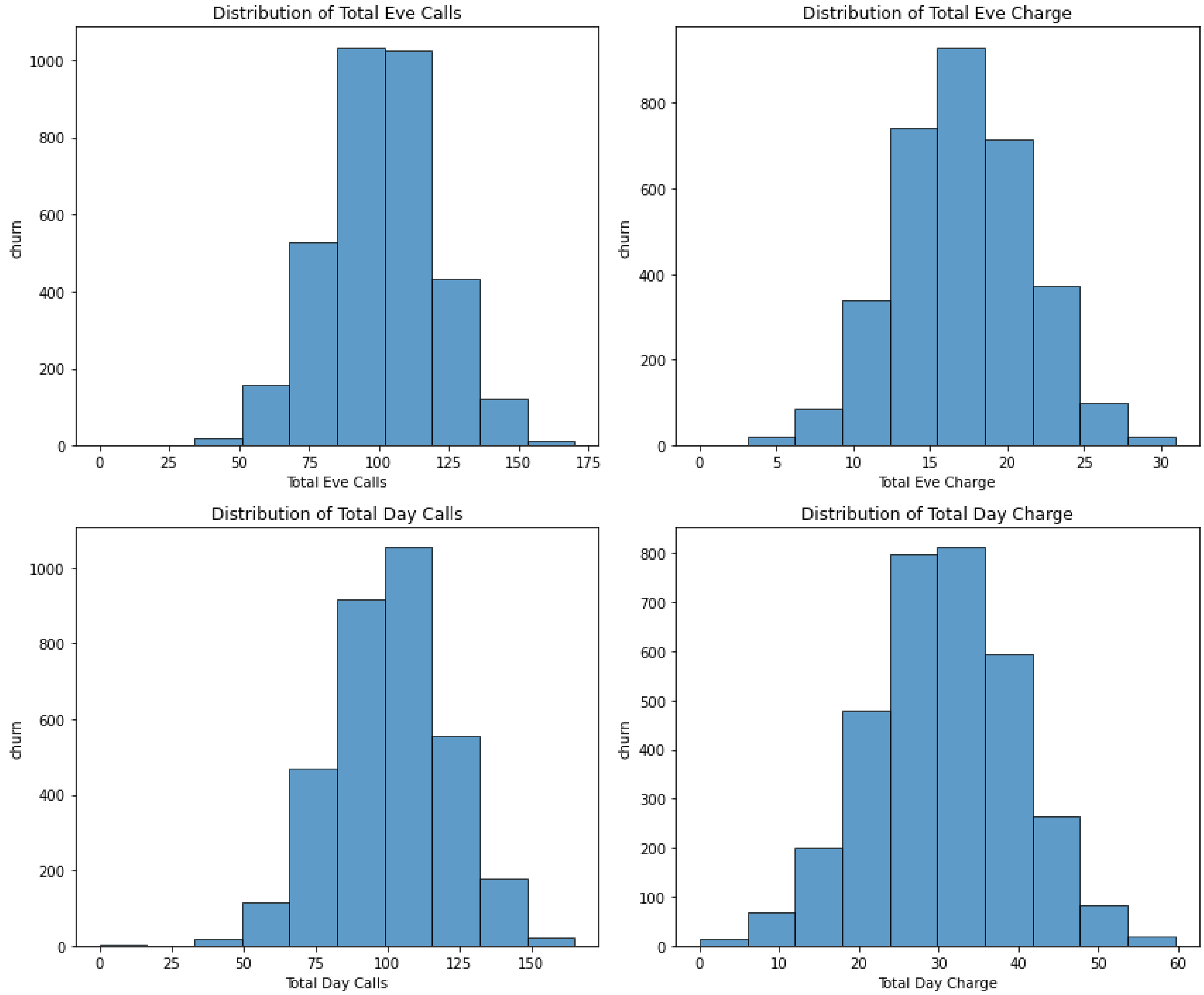
In [450…

```
from sklearn.preprocessing import StandardScaler

# Select numeric columns for scaling
numeric_features = ['account length', 'number vmail messages', 'total day minutes', 'to
                    'total eve minutes', 'total eve calls', 'total eve charge', 'total
                    'total night charge', 'total intl minutes', 'total intl calls', 'to

# Initialize the scaler
scaler = StandardScaler()

# Fit and transform the numeric features
X[numeric_features] = scaler.fit_transform(X[numeric_features])
```

e. Check the distribution of variables

In [451…

```
# plot histogram to check distribution
#Create a figure with 2x2 subplots
plt.figure(figsize=(12, 10))

# Plot for 'total eve calls'
plt.subplot(2, 2, 1)
plt.hist(df["total eve calls"], bins=10, edgecolor='k', alpha=0.7)
plt.xlabel('Total Eve Calls')
plt.ylabel('churn')
plt.title('Distribution of Total Eve Calls')

# Plot for 'total eve charge'
plt.subplot(2, 2, 2)
plt.hist(df["total eve charge"], bins=10, edgecolor='k', alpha=0.7)
plt.xlabel('Total Eve Charge')
plt.ylabel('churn')
plt.title('Distribution of Total Eve Charge')

# Plot for 'total day calls'
plt.subplot(2, 2, 3)
plt.hist(df["total day calls"], bins=10, edgecolor='k', alpha=0.7)
plt.xlabel('Total Day Calls')
plt.ylabel('churn')
plt.title('Distribution of Total Day Calls')

# Plot for 'total day charge'
plt.subplot(2, 2, 4)
plt.hist(df["total day charge"], bins=10, edgecolor='k', alpha=0.7)
plt.xlabel('Total Day Charge')
plt.ylabel('churn')
plt.title('Distribution of Total Day Charge')

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



5. Split the Data into Training and Test Sets

In [452…
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
```

In [453…
```
# check the shape of X_train and X_test

X_train.shape, X_test.shape
```

Out[453…
```
((2666, 73), (667, 73))
```

In [454…
```
#verify data types of all columns
print(df.dtypes)
```

```
state                      object
account length            float64
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
```

5b. Choose and Train a Model

I will use Machine Learning models because it is appropriate for predicting customer churn due to the complexity of the problem, the nature of the data, and the advantages that ML offers over simpler methods.

The goal is to predict whether a customer will churn. This is inherently a prediction problem that involves classifying binary outcomes (churn vs. no churn). Machine learning algorithms are well-suited for such classification tasks, especially when the relationships between features and the outcome are complex and non-linear.

Customer churn can be influenced by various features like total_day_minutes, customer_service_calls, and total_day_charge, as well as their interactions. ML models can capture these intricate relationships better than simpler statistical methods.

In [455…

```
# split the data into train and test data
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Ensure y_test and y_pred have the same length
print(len(y_test))  # Should be same as len(y_pred)
```

667

In [456…

```
# instantiate the model
model = LogisticRegression(solver='liblinear', random_state=0)
```

In [457…

```
# fit the model
model.fit(X_train, y_train)
# Predict on test data
y_pred = model.predict(X_test)

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)
```

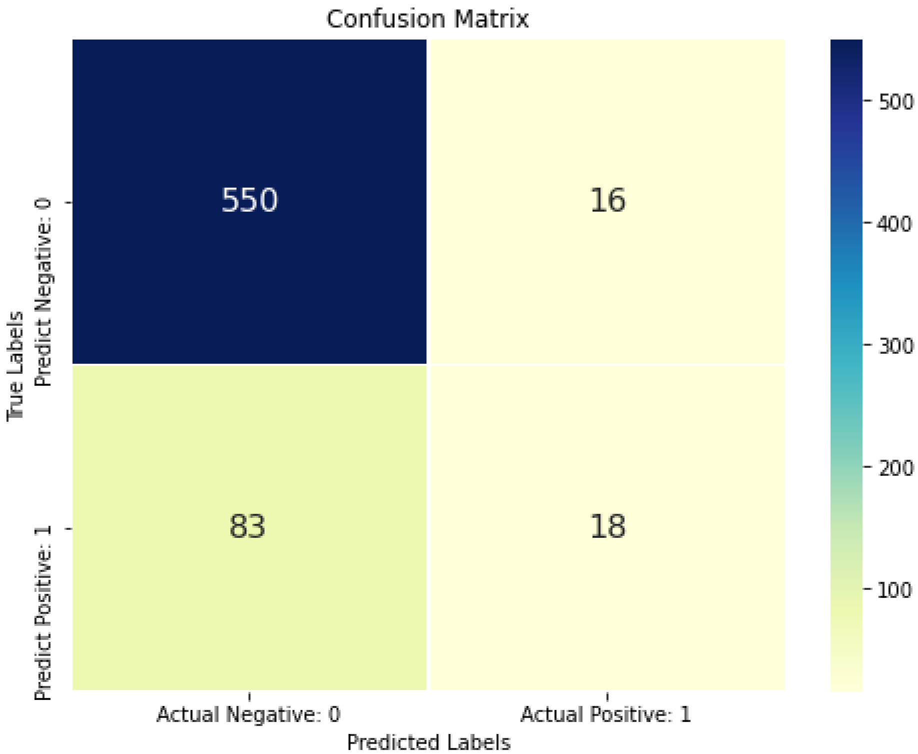
In [458…

```
# Create DataFrame for the confusion matrix
cm_matrix = pd.DataFrame(
    data=cm,
    columns=['Actual Negative: 0', 'Actual Positive: 1'],
```

```
    index=['Predict Negative: 0', 'Predict Positive: 1']
)
```

In [459…

```
# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu',
            cbar=True, annot_kws={"size": 16}, linewidths=.5)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```



6. Classification metrices

classification report

In [460…

```
#print the classification report
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       0.87      0.97      0.92       566
        True       0.53      0.18      0.27       101

    accuracy                           0.85       667
   macro avg       0.70      0.57      0.59       667
weighted avg       0.82      0.85      0.82       667
```

b) Compare the train-set and test-set accuracy

In [461…

```
y_pred_train = model.predict(X_train)
```

```
y_pred_train
```

Out[461…
```
array([False, False,  True, ..., False, False, False])
```

In [462…
```
#print classification accuracy
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_tr
```

```
Training-set accuracy score: 0.8706
```

In [463…
```
#Check for overfitting and underfitting
#print the scores on training and test set

print('Training set score: {:.4f}'.format(model.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(model.score(X_test, y_test)))
```

```
Training set score: 0.8706
Test set score: 0.8516
```

The training-set accuracy score is 0.8706 while the test-set accuracy to be 0.8516. These two values are quite comparable. So, there is no question of overfitting.

6c). Adjusting the threshold level

In [464…
```
# print the first 10 predicted probabilities of two classes- 0 and 1

y_pred_prob = model.predict_proba(X_test)[0:10]

y_pred_prob
```

Out[464…
```
array([[0.85964513, 0.14035487],
       [0.99078387, 0.00921613],
       [0.97637945, 0.02362055],
       [0.88081324, 0.11918676],
       [0.96806606, 0.03193394],
       [0.98670224, 0.01329776],
       [0.96243486, 0.03756514],
       [0.9840752 , 0.0159248 ],
       [0.68645213, 0.31354787],
       [0.98060017, 0.01939983]])
```

Observations

In each row, the numbers sum to 1.

There are 2 columns which correspond to 2 classes - 0 and 1.

Class 0 - predicted probability that customer will churn.

Class 1 - predicted probability that customer will not churn.

Classification threshold level

There is a classification threshold level of 0.5.

Class 1 - probability of churn is predicted if probability > 0.5.

Class 0 - probability of no churn is predicted if probability < 0.5.

In [465…

```
# store the probabilities in dataframe

y_pred_prob_df = pd.DataFrame(data=y_pred_prob, columns=['Prob of - customer will not c

y_pred_prob_df
```

Out[465…

	Prob of - customer will not churn (0)	Prob of - customer will churn (1)
0	0.859645	0.140355
1	0.990784	0.009216
2	0.976379	0.023621
3	0.880813	0.119187
4	0.968066	0.031934
5	0.986702	0.013298
6	0.962435	0.037565
7	0.984075	0.015925
8	0.686452	0.313548
9	0.980600	0.019400

In [466…

```
# print the first 5 predicted probabilities for class 1 - Probability of churn

model.predict_proba(X_test)[0:5, 1]
```

Out[466…

```
array([0.14035487, 0.00921613, 0.02362055, 0.11918676, 0.03193394])
```

In [467…

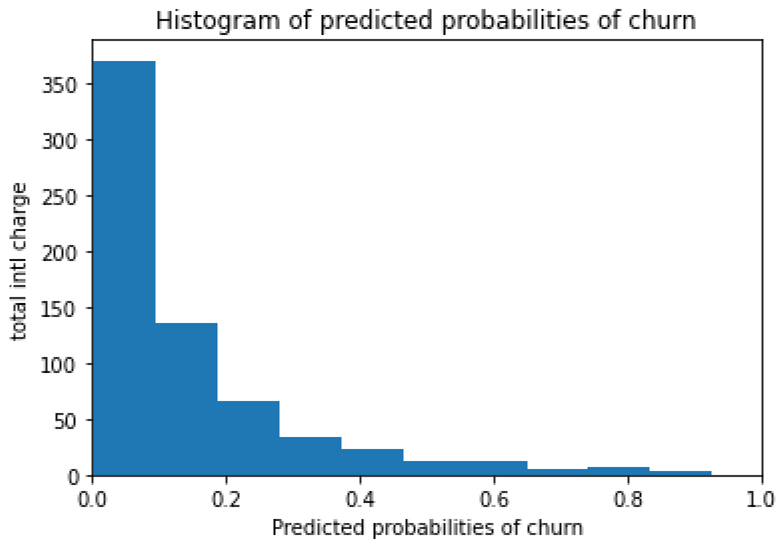
```
# store the predicted probabilities for class 1 - Probability of churn

y_pred1 = model.predict_proba(X_test)[:, 1]
```

In [468…

```
# plot histogram of predicted probabilities


# adjust the font size
plt.rcParams['font.size'] = 10


# plot histogram with 10 bins
plt.hist(y_pred1, bins = 10)


# set the title of predicted probabilities
plt.title('Histogram of predicted probabilities of churn')


# set the x-axis limit
plt.xlim(0,1)


# set the title
plt.xlabel('Predicted probabilities of churn')
plt.ylabel('total intl charge')
```

Out[468…

```
Text(0, 0.5, 'total intl charge')
```



Observations

There are small number of observations with probability > 0.5.

So, these small number of observations predict that customers will churn.

Majority of observations predict that customers will not churn.

6d. Analysis of the Results

Accuracy: The proportion of correctly predicted instances. An accuracy of approximately 85.2% means that our model correctly classified about 85.2% of all instances in our dataset.

Confusion Matrix: Shows the number of true positives, true negatives, false positives, and false negatives. True Negatives (TN): 551 (Correctly predicted as False) False Positives (FP): 15 (Incorrectly predicted as True) False Negatives (FN): 83 (Incorrectly predicted as False) True Positives (TP): 18 (Correctly predicted as True)

Classification Report: Provides precision, recall, and F1-score for each class.

Precision: Measures how many of the predicted positives are actually positive. False Class Precision: 0.87 - When the model predicts False, it is correct 87% of the time. True Class Precision: 0.53 - When the model predicts True, it is correct 53% of the time.

Recall (Sensitivity): Measures how many actual positives are correctly predicted by the model. False Class Recall: 0.97 - Of all actual False instances, 97% were correctly identified. True Class Recall: 0.18 - Of all actual True instances, only 18% were correctly identified.

F1-Score: The harmonic mean of precision and recall, balancing the two metrics. False Class F1-Score: 0.92 - A balance between precision and recall for the False class. True Class F1-Score: 0.27 - A balance between precision and recall for the True class.

Support: The number of actual occurrences of each class in the dataset. False Class Support: 566 True Class Support: 101

Macro Average: Averages precision, recall, and F1-score across classes without considering the class imbalance.

Macro Average Precision: 0.70 Macro Average Recall: 0.57 Macro Average F1-Score: 0.59 Weighted Average: Averages precision, recall, and F1-score across classes, weighted by the number of instances in each class.

Weighted Average Precision: 0.82 Weighted Average Recall: 0.85 Weighted Average F1-Score: 0.82

7. Interpretation of the Results

Overall Accuracy: 85.2% indicates the model performs well overall but doesn't tell the whole story, especially in the presence of class imbalance.

Confusion Matrix Analysis:

The model has high accuracy in predicting the False class (high precision and recall for False). The model struggles with the True class, with a low recall of 18%. This means it misses many True instances, leading to a low F1-score of 0.27 for True.

Precision and Recall Trade-Off: High Precision for False Class: Indicates that when the model predicts False, it is mostly correct. Low Recall for True Class: Indicates that many True instances are missed. This might be due to an imbalanced dataset or the model’s current thresholds.

Macro vs. Weighted Average:

Macro Average: Provides an unweighted mean, giving equal importance to each class regardless of their support. Weighted Average: Takes into account the number of instances per class, giving more weight to larger classes.

7b. Summary

1. Model Strengths: Good at identifying False cases (high precision and recall).

2. Model Weaknesses: Poor at identifying True cases (low recall and F1-score for True). This could be due to class imbalance or insufficient feature representation for the True class.

8. Actions to Consider

1. Feature Engineering: Explore if additional features or different features could help improve the model's ability to identify True instances.

2. Model Tuning: Experiment with different algorithms or hyperparameters to see if they provide better performance for the True class.

I will consider Model Tuning

9. MODEL TUNING

I will experiment with ROC AUC curve and cross validate with decison tree to find out if they provide a better perfomance for the True class.

ROC-AUC

In this technique, we measure the area under the curve (AUC). A perfect classifier will have a ROC AUC equal to 1, whereas a purely random classifier will have a ROC AUC equal to 0.5.

So, ROC AUC is the percentage of the ROC plot that is underneath the curve.

In [469...

```
# compute ROC AUC

from sklearn.metrics import roc_auc_score

ROC_AUC = roc_auc_score(y_test, y_pred1)

print('ROC AUC : {:.4f}'.format(ROC_AUC))
```

```
ROC AUC : 0.8253
```

ROC AUC of our model approaches towards 1. So, we can conclude that our classifier does a good job in predicting whether customer will churn or not.

In [470...

```
# calculate cross-validated ROC AUC

from sklearn.model_selection import cross_val_score

Cross_validated_ROC_AUC = cross_val_score(model, X_train, y_train, cv=5, scoring='roc_a

print('Cross validated ROC AUC : {:.4f}'.format(Cross_validated_ROC_AUC))
```

```
Cross validated ROC AUC : 0.7977
```

10. CROSS VALIDATION

I will use decision tree for cross validation

In [471...

```
#import libraries
from sklearn.model_selection import cross_val_score, KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
```

In [472...

```
# Load data
X = df_encoded.drop(columns=['churn'])
y = df_encoded['churn']
```

In [473...

```
# Initialize the Decision Tree classifier
dt_classifier = DecisionTreeClassifier()
```

10b. Define Cross-Validation Strategy

I will use StratifiedKFold as it ensures each fold has the same proportion of class labels, which is especially useful for classification problems.

In [474…
```
from sklearn.model_selection import StratifiedKFold

# Define cross-validation strategy
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

In [475…
```
# Apply cross-validation
scores = cross_val_score(dt_classifier, X, y, cv=cv, scoring='accuracy')

# Print results
print(f"Accuracy scores for each fold: {scores}")
print(f"Mean accuracy: {scores.mean():.2f}")
print(f"Standard deviation of accuracy: {scores.std():.2f}")
```

```
Accuracy scores for each fold: [0.92353823 0.89355322 0.92053973 0.9039039  0.91591592]
Mean accuracy: 0.91
Standard deviation of accuracy: 0.01
```

Our, original model score is found to be 0.852. The average cross-validation score is 0.91.

The mean accuracy of 0.91 indicates that, on average,the decision tree model correctly classifies 91% of the samples across all folds. This is a high accuracy, suggesting that the model performs well on the dataset.

Standard deviation: 0.02

The standard deviation of 0.02 indicates that the accuracy scores vary slightly across the folds. This low standard deviation suggests that the model's performance is consistent and stable across different subsets of the data

11. Hyperparameter Optimization using GridSearch CV

In [476…
```
 from sklearn.model_selection import GridSearchCV

# Define parameter grid
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

In [477…
```
# Initialize GridSearchCV
grid_search = GridSearchCV(
    estimator=dt_classifier,
    param_grid=param_grid,
    cv=5,  # Number of cross-validation folds
    scoring='accuracy',  # Metric to evaluate performance
    n_jobs=-1,  # Number of parallel jobs to run (-1 means using all processors)
    verbose=1  # Verbosity level
)
```

In [478…
```
# Fit the model with GridSearchCV
grid_search.fit(X_train, y_train)

# Get the best parameters and best score
print(f"Best parameters: {grid_search.best_params_}")
```

```
print(f"Best score: {grid_search.best_score_:.2f}")

# Get the best estimator
best_model = grid_search.best_estimator_

# Predict using the best model
y_pred = best_model.predict(X_test)

# Evaluate the model
print(f"Test accuracy: {accuracy_score(y_test, y_pred):.2f}")
```

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
Best parameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 2, 'min_s
amples_split': 5}
Best score: 0.94
Test accuracy: 0.94
```

Our original model test accuracy is 0.852 while GridSearch CV accuracy is 0.94.

We can see that GridSearch CV improve the performance for this particular model.

12.RESULTS CONCLUSION

1. The logistic regression model accuracy score is 0.852. So, the model does a very good job in predicting whether or not the cutomer will churn.

2. Small number of observations predict that customer will churn. Majority of observations predict that customer will not churn.

3. The model shows no signs of overfitting.

4. Overall Performance of the Decision tree model:

The decision tree model shows high performance with a mean accuracy of 91%, which is quite good. This implies that the model is effectively capturing patterns in the data and making accurate predictions. Consistency: The low standard deviation (0.02) suggests that the model's performance is consistent across different folds of the cross-validation. This indicates that the model is not overly sensitive to the specific data it was trained on, which is a positive sign of robustness.

Model Suitability: Given the high mean accuracy and low variability, the decision tree model appears to be well-suited for this dataset. It is performing reliably and accurately.

1. Our original model test accuracy is 0.852 while GridSearch CV accuracy is 0.94. We can see that GridSearch CV improve the performance for this particular model.

13. LIMITATION

The regression model shows a lower rate of predicting true positives(53%) of customer churn. This might ,misinform the company if other superior modelas are not used.

14. RECOMMENDATIONS

1. Leverage the Best-Performing Model Adopt the Decision Tree model for primary churn predictions due to its higher accuracy and consistency.

Keep the Logistic Regression model as a secondary model to validate results or for scenarios where interpretability is crucial, given its simpler nature compared to Decision Trees.

1. Address Class Imbalance: There is a small number of observations predicting churn, indicating class imbalance. I would recommend the use of resampling techniques or class weight adjustments

3.Validate Model Performance

Monitor and validate models regularly to ensure continued performance and adaptation.

1. Improved Performance with GridSearch CV: GridSearch CV improved the Logistic Regression model’s accuracy to 94%. Recommendation;

Leverage GridSearch CV to optimize both models and refine feature selection.

1. Business Implications and Actions Implement actionable strategies based on model predictions to reduce churn effectively. Churn Prediction Utilization: Use the high-performing models to proactively address customer churn. Implement strategies such as targeted retention campaigns, personalized offers, or enhanced customer service for those predicted to churn.

Resource Allocation: Allocate resources effectively based on model predictions to maximize the impact on customer retention.

1. Reevaluate Models Periodically: Regularly update models with new data and reassess their performance to ensure they remain effective as customer behaviors and market conditions change.