Predicting Customer Churn for SyriaTel

Business Understanding

Here, I will define the stakeholder, and the business objective:

#Stakeholder: SyriaTel Customer Retention Team

#Objective: Predict customer churn using classification.

Then I will import and define the dataset.

In [10]:
```
import pandas as pd
df = pd.read_csv('bigml_59c28831336c6604c800002a.csv')
df.head()
```

Out[10]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages
0	KS	128	415	382-4657	no	yes	25
1	OH	107	415	371-7191	no	yes	26
2	NJ	137	415	358-1921	no	no	0
3	OH	84	408	375-9999	yes	no	0
4	OK	75	415	330-6626	yes	no	0

5 rows × 21 columns

Data Preparation

Here, I convert the required data into integers., and handle categorical variables.

Next up I split into x and y, then use train_test_split with stratify=y

In [11]:
```
df['churn'] = df['churn'].astype(int)
X = pd.get_dummies(df.drop('churn', axis=1), drop_first=True)
y = df['churn']
```

In [12]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
stratify=y, random_state=42)
```

Modeling

Here, I model using Logistic Regression for evaluation via classification, and Random Forest method for tuned hyperparameters.

In [15]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

log_reg = LogisticRegression(max_iter=10000, class_weight='balanced')
log_reg.fit(X_train, y_train)
print(classification_report(y_test, log_reg.predict(X_test)))
```

```
                   precision    recall  f1-score   support

                0       0.93      0.82      0.87       570
                1       0.37      0.63      0.47        97

         accuracy                           0.79       667
        macro avg       0.65      0.72      0.67       667
     weighted avg       0.85      0.79      0.81       667
```

In [14]:
```
rf = RandomForestClassifier(n_estimators=300, max_depth=10,
class_weight='balanced', random_state=42)
rf.fit(X_train, y_train)
print(classification_report(y_test, rf.predict(X_test)))
```

```
                   precision    recall  f1-score   support

                0       0.93      0.92      0.92       570
                1       0.56      0.57      0.56        97

         accuracy                           0.87       667
        macro avg       0.74      0.74      0.74       667
     weighted avg       0.87      0.87      0.87       667
```

#Logistic Regression
Accuracy - 0.79
Recall: Churn =1. - 0.63

#Random Forest
Accuracy - 0.87
Recall: Churn = 1. - 0.57

Logistic Regression has better Recall and catches more churners
Random Forest has a higher accuracy overall, improving overall predictive strength.

My baseline model is Logistic Regression because of its simplicity and interpretability. Stakeholders can easily understand what features influence churn thanks to clear coefficients.

My second model was Random Forest, which helped me capture non-linear relationshipsand interactions between features.
After tuning hyperparameters,(n_estimators = 300, max_depth = 10, class_weight = 'balanced') accuracy improved to 87%.
Recall for churners decreased slightly in the process, indicating a tradeoff between accuracy and identifying churners.

Given my primary objective is to identify churners for retention campaigns, I would advise on using Logistic Regression at the cost of lower accuracy.