1 Customer Retention for SyriaTel

Here at SyriaTel, a telecommunications company, we're committed to making our consumers happy and keeping them part of the family. We will look at our data with different models to predict which customers are more likely to cancel their service with our company in order to better prevent patron loss.

We will start with loading the necessary libraries and looking into our dataset.

In [120]:

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from sklearn.model_selection import train_test_split, cross_val_sc
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegre
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.tree import plot_tree
from sklearn import tree
from six import StringIO
from IPython.display import Image
%matplotlib inline
```

In [121]:

In [122]:

Out[122]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...

5 rows × 21 columns

In [123]:
```
#dropping columns that won't be used
data.drop(columns=['state', 'account length', 'area code', 'phone
```

In [124]:
```
#renaming columns
data = data.rename(columns = {'international plan': 'International
```

In [125]:
```
#change international plan, voicemail plan and churn to numerical
#0 = no/false, 1 = yes/true
#also making sure these values are numbers and not strings

data['International Plan'] = data['International Plan'].replace('n
data['International Plan'] = data['International Plan'].replace('y
data['Voicemail Plan'] = data['International Plan'].replace('no',
data['Voicemail Plan'] = data['International Plan'].replace('yes',
data['Churn'] = data['International Plan'].replace('False', '0')
data['Churn'] = data['International Plan'].replace('True', '1')

data['International Plan'] = data['International Plan'].astype(int
data['Voicemail Plan'] = data['International Plan'].astype(int)
data['Churn'] = data['International Plan'].astype(int)
```

In [126]:
```
#defining x and y
y = data[['Churn']].values.ravel()
#x = data[['International Plan', 'Voicemail Plan', 'Number of Voic
x = data.drop('Churn', axis=1)
```

In [127]:
```
#defining x train, y train, x test and y test
```

In [128]:
```
#using a defined function for modeling to streamline the process
def run_model(model, x_train, y_train, x_test, y_test):

    #fitting
    model.fit(x_train, y_train)

    #predictions
    y_hat_train = model.predict(x_train)
    y_hat_test = model.predict(x_test)

    print('Classification Report: Train Set \n')
    print(classification_report(y_train, y_hat_train))
    print('Classification Report: Test Set \n')
    print(classification_report(y_test, y_hat_test))

    fig, (ax0, ax1) = plt.subplots(1, 2, figsize=(18,6))

    plot_confusion_matrix(model, x_train, y_train, ax=ax0)
    plot_confusion_matrix(model, x_test, y_test, ax=ax1)

    ax0.title.set_text('Train Confusion Matrix')
    ax1.title.set_text('Test Confusion Matrix')

    return model
```

In [129]:

```
#looking and checking for null or missing data
```

Out[129]:

	International Plan	Voicemail Plan	Number of Voicemail Messages	Total Day Minutes	Total Day Calls	Total Evening Minutes	Eve
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.00
mean	0.096910	0.096910	8.099010	179.775098	100.435644	200.980348	100.1
std	0.295879	0.295879	13.688365	54.467389	20.069084	50.713844	19.92
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.000000	0.000000	0.000000	143.700000	87.000000	166.600000	87.00
50%	0.000000	0.000000	0.000000	179.400000	101.000000	201.400000	100.00
75%	0.000000	0.000000	20.000000	216.400000	114.000000	235.300000	114.00
max	1.000000	1.000000	51.000000	350.800000	165.000000	363.700000	170.00

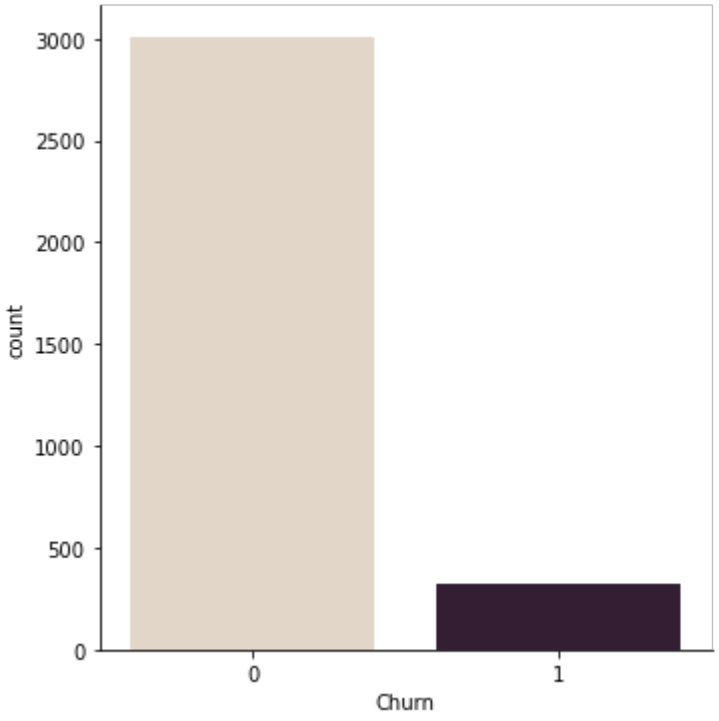
In [130]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 13 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   International Plan            3333 non-null   int32  
 1   Voicemail Plan                3333 non-null   int32  
 2   Number of Voicemail Messages  3333 non-null   int64  
 3   Total Day Minutes             3333 non-null   float64
 4   Total Day Calls               3333 non-null   int64  
 5   Total Evening Minutes         3333 non-null   float64
 6   Total Evening Calls           3333 non-null   int64  
 7   Total Night Minutes           3333 non-null   float64
 8   Total Night Calls             3333 non-null   int64  
 9   Total International Minutes   3333 non-null   float64
 10  Total International Calls     3333 non-null   int64  
 11  Customer Service Calls        3333 non-null   int64  
 12  Churn                         3333 non-null   int32  
dtypes: float64(4), int32(3), int64(6)
memory usage: 299.6 KB
```

In [131]:
```
#looking at retention/churn values
print(data["Churn"].value_counts())
```

```
0    3010
1     323
Name: Churn, dtype: int64
```

Out[131]:
```
<seaborn.axisgrid.FacetGrid at 0x271185671c0>
```



Looking here at the graph above we can see that with customer churn, or otherwise known as customer retention, 90.3% (or 3,010 people) of customers continued to do business with SyriaTel. While that is a relatively high rate, there was still the 9.7% (or 323 people) that did not. Below we will look at some models to better understand the features in our data and how they may play a pivotal role in client loyalty.

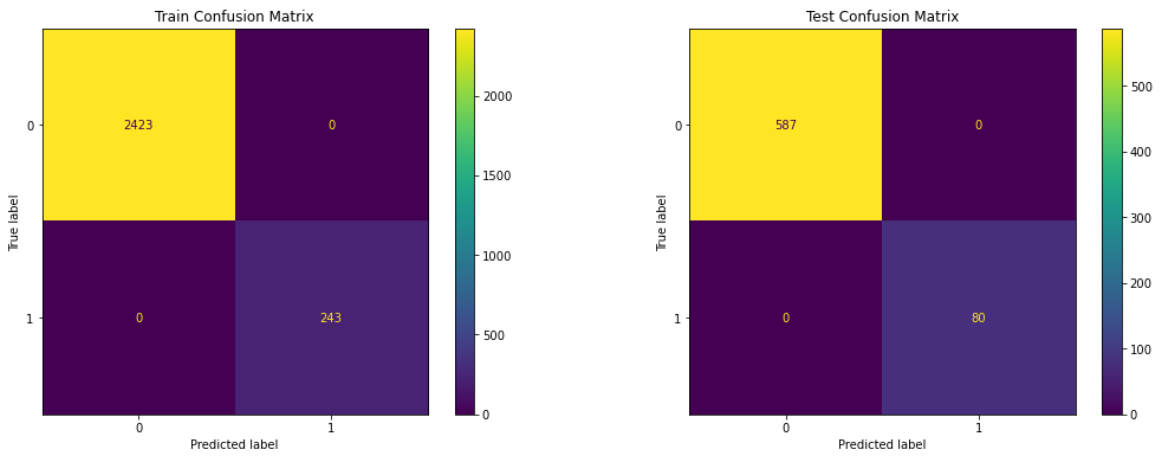
2 Decision Tree Modeling

In [132]:
```
#using Decision Trees as first model
```

In [133]:

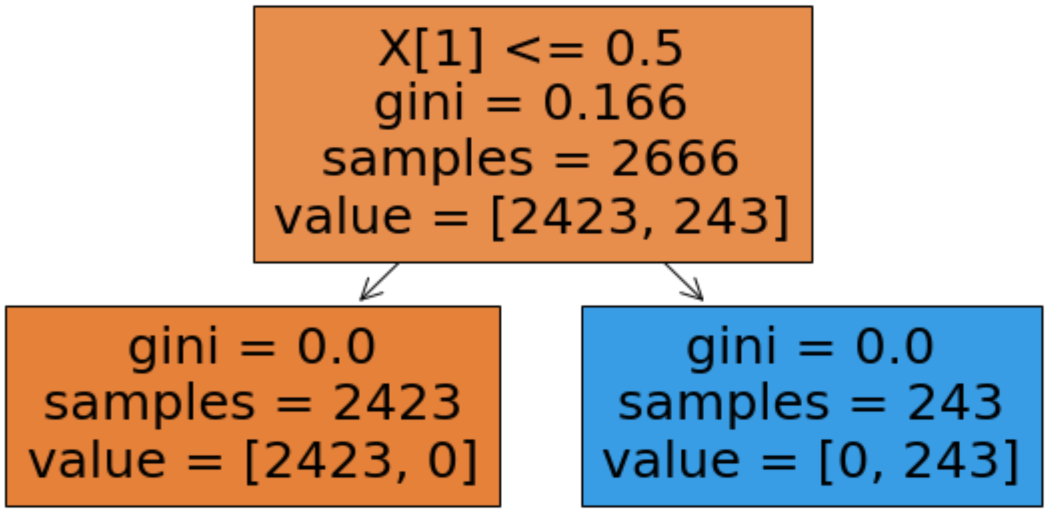
```
Classification Report: Train Set

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2423
           1       1.00      1.00      1.00       243

    accuracy                           1.00      2666
   macro avg       1.00      1.00      1.00      2666
weighted avg       1.00      1.00      1.00      2666

Classification Report: Test Set
```

Out[133]:
```
DecisionTreeClassifier()
```



In [134]:
```
#plot the decision tree
plt.figure(figsize=(10, 5))
```

Out[134]:
```
[Text(279.0, 203.85000000000002, 'X[1] <= 0.5\ngini = 0.166\nsamples
= 2666\nvalue = [2423, 243]'),
 Text(139.5, 67.94999999999999, 'gini = 0.0\nsamples = 2423\nvalue =
[2423, 0]'),
 Text(418.5, 67.94999999999999, 'gini = 0.0\nsamples = 243\nvalue =
[0, 243]')]
```



In [135]:
```
#feature importances
dtc.feature_importances_
df = pd.DataFrame({'Feature Names':x.columns, 'Importance':dtc.fea
```

Out[135]:

	Feature Names	Importance
1	Voicemail Plan	1.0
0	International Plan	0.0
2	Number of Voicemail Messages	0.0
3	Total Day Minutes	0.0
4	Total Day Calls	0.0
5	Total Evening Minutes	0.0
6	Total Evening Calls	0.0
7	Total Night Minutes	0.0
8	Total Night Calls	0.0
9	Total International Minutes	0.0
10	Total International Calls	0.0
11	Customer Service Calls	0.0

For the Decision Tree Model, we can see that there is a nice attribute of it where we can check on feature importances. With the upwards graph, we can see that customers who had the Voicemail Plan had a high impact on customer Churn compared to the other features, so therefore we will keep this in mind when we discuss our recommendations.

Over all the Decision Tree model had very good numbers across accuracy, precision, recall and F1 scores. This held true even with testing variations to the default model, but due to Decision Trees having the tendency to overgeneralize the data and potentially leading to over fitting, this will most likely not be the final model choice.

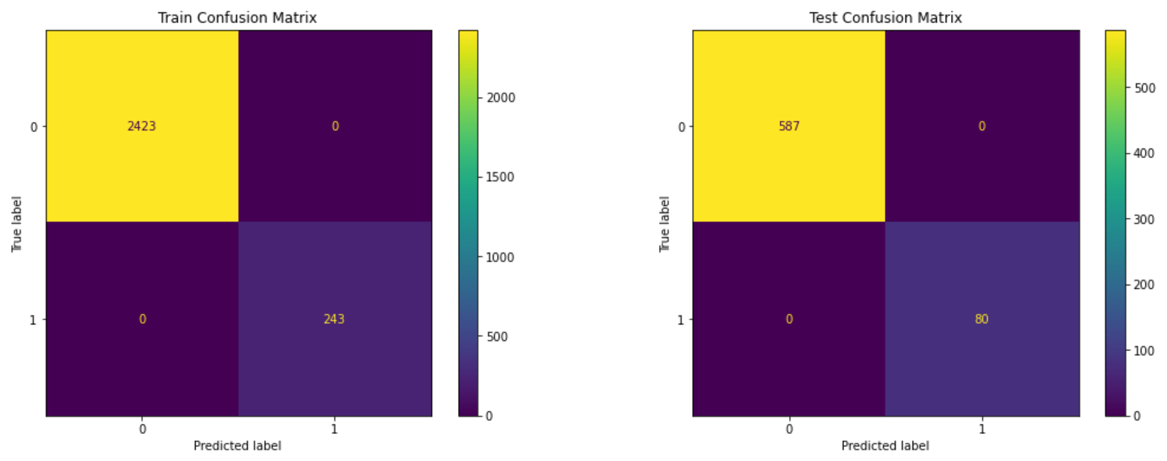
3 Random Forest Modeling

In [136]:
```
#using Random Forest as second model
```

In [137]:

```
Classification Report: Train Set

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2423
           1       1.00      1.00      1.00       243

    accuracy                           1.00      2666
   macro avg       1.00      1.00      1.00      2666
weighted avg       1.00      1.00      1.00      2666

Classification Report: Test Set
```

Out[137]: `RandomForestClassifier()`



In [138]:

```python
#feature importances
ranfor.feature_importances_
df = pd.DataFrame({'Feature Names':x.columns, 'Importance':ranfor.
```

Out[138]:

	Feature Names	Importance
0	International Plan	0.513414
1	Voicemail Plan	0.465918
5	Total Evening Minutes	0.004716
7	Total Night Minutes	0.003284
9	Total International Minutes	0.003015
3	Total Day Minutes	0.002950
6	Total Evening Calls	0.001700
4	Total Day Calls	0.001481
8	Total Night Calls	0.001236
10	Total International Calls	0.001071
11	Customer Service Calls	0.000931
2	Number of Voicemail Messages	0.000283

As with the Decision Tree Model, the Random Forest Model also deals with feature importances and shows that once again that the VoiceMail Plan is noted having great influence on customer Churn. The difference between the two models showing that the International Plan was slightly more relevant than VoiceMail Plans, but the latter still highly rated with importance.

In addition, just like the Decision Tree Model, the Random Forest Model had high accuracy, precision, recall and F1 scores. These scores remained up to par, even after changing the default settings. Regardless of the number of estimators, the criterion and even max depth, the Random Forest Model performed very well.

With all of this in mind, there is the fact that with Random Forest Models, the downside is that if they have a large number of trees, this can slow the prediction process. This may not be an issue just now, but we will keep this in mind when choosing what the final model for this dataset will be.

4 Logistic Regression Modeling [...]

5 Gaussian Naive Bayes Modeling [...]

6 Nearest Neighbors Modeling [...]

7 Final Model Outcomes and Recommendations

In [145]:

```
#random forest with gridsearch
ranfor_grid = {'n_estimators': [150], 'max_depth':[15], 'criterion

ranfor_search = GridSearchCV(RandomForestClassifier(), ranfor_grid

ranfor_search.fit(x_train, y_train)
```

Out[145]:

```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'class_weight': ['balanced', 'balanced_subsa
mple'],
                         'criterion': ['gini', 'entropy'], 'max_depth
': [15],
                         'n_estimators': [150]},
             scoring='f1')
```

In [146]:

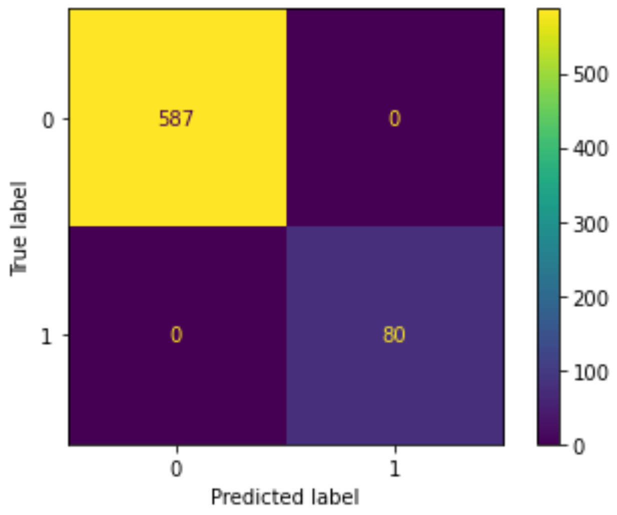
Out[146]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_class_weight	param_c
0	0.268570	0.029482	0.017490	0.000524	balanced	

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_class_weight	param_c
1	0.262155	0.009203	0.016260	0.000494	balanced	
2	0.353106	0.022463	0.017501	0.001146	balanced_subsample	
3	0.361153	0.003663	0.016869	0.000398	balanced_subsample	

In [147]:
```
best_params = ranfor_search.best_params_
```
Out[147]:
```
{'class_weight': 'balanced',
 'criterion': 'gini',
 'max_depth': 15,
 'n_estimators': 150}
```

In [148]:
```
test_preds = ranfor_search.predict(x_test)
```
Out[148]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2
711ba50b80>
```



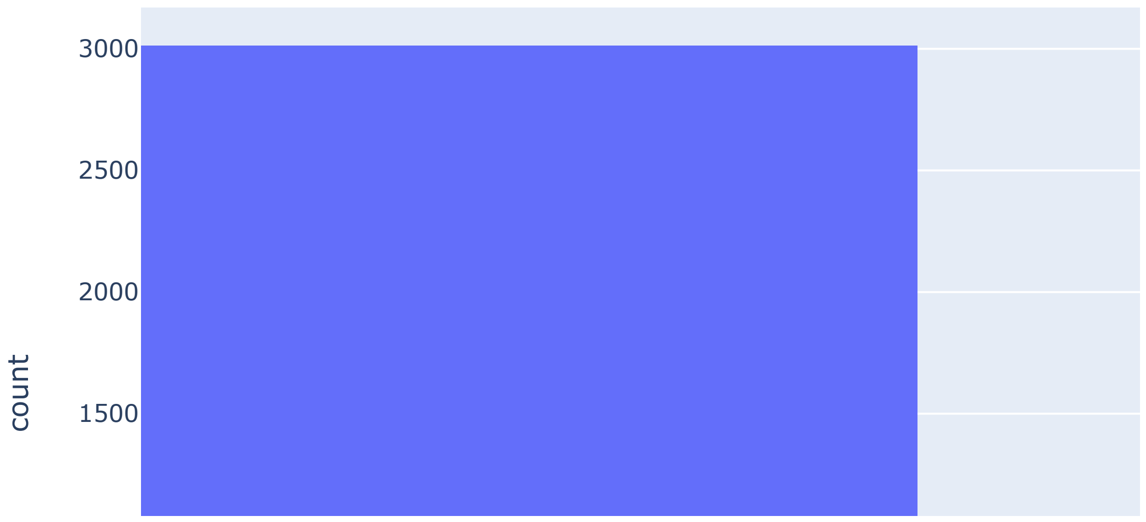
In [149]:
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       587
           1       1.00      1.00      1.00        80

    accuracy                           1.00       667
   macro avg       1.00      1.00      1.00       667
weighted avg       1.00      1.00      1.00       667
```

In [160]:
```
data['Churn'] = data['Churn'].replace({0: 'stayed', 1: 'left'})
```

In [161]:
```
px.histogram(data, x="International Plan", color="Churn")
```

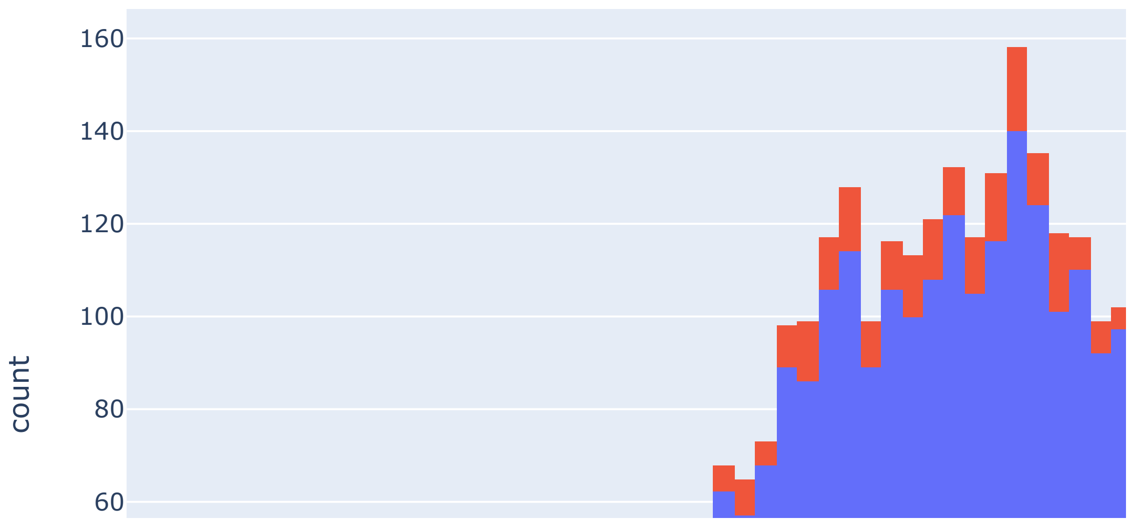
In [158]:

```
[illegible]
```



Here with the graphs above, we can see after our model has run with the best parameters that within the International Plan and Voicemail Plan, the same amount of people left SyriaTel. This indicates that both features are equally important as to why consumers are leaving and both should be the focus of improvement rather than just one of them.

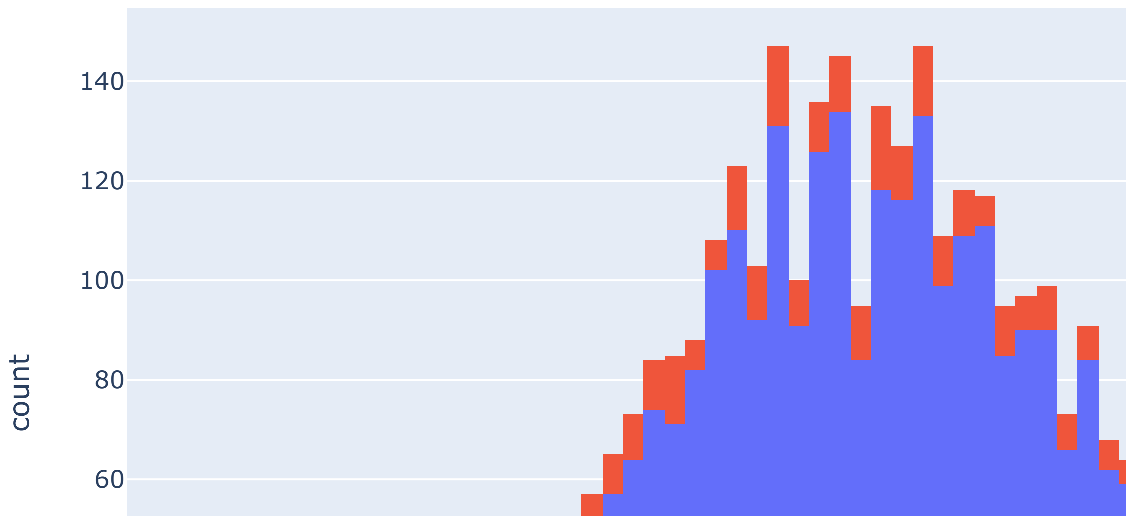
```
In [157]: [illegible]
```



With the graphs on either side of this paragraph, we can tell that for the following important features listed for Random Forest Total Evening Minutes and Total Night Minutes, both had similar results in that when the minutes began to enter the 150 to 280 range, more people began to leave than stay. This could indicate that perhaps the plans for evenings and nights need to be revised to better suit the needs of people talking quite a bit in these hours.

In [159]:

```
[illegible]
```



The final model ended up being the Random Forest Classifier for a few reasons. Even though most of the models run above had good results, Random Forest not only had a high f1 score, but the results were reliable and quick. These models are known to run efficiently on large datasets, resistant with outliers and have a lower risk of over fitting to the data.

Focusing on the f1 scores allowed us to have a harmonious balance between precision and recall. And since having high accuracy does not necessarily mean you'll have class equity results, f1 seemed to be the most reliable score to use.

Our final model accurately predicted that 608 people who would stay with SyriaTel would stay and those 59 people who would churn, did so. Random Forest also has feature importances, which gave us insight as to which service should be the focus of improving at SyriaTel. With almost half the results for the feature importance being the International Plan as well as the Voicemail Plan, these should be the main focuses of enhancement for the company to prevent customer Churn.

The only downside to Random Forest, as mentioned earlier, is that as the data set gets bigger it can over fit. However, with the best parameters feature, we were able to find that the most efficient class weight is balance, the criterion (a way to measure the quality of a split) was gini, with a max depth of trees being 15 and 150 estimators, our model should run smoothly.

With any model, there can always be improvement. Here, it would be recommended to have other models tested with more datasets (both past and future company data) as the multiple iterations on different aspects of customer data would provide a clearer picture as to why consumers are ending their services with SyriaTel.

In [ ]: