



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 [1]: 1 #importing libraries
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import pickle
7 from sklearn.model_selection import train_test_split, cross_val_score
8 from sklearn.linear_model import LogisticRegression
9 from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
10 from sklearn.ensemble import RandomForestClassifier
11 from sklearn.naive_bayes import GaussianNB
12 from sklearn.neighbors import KNeighborsClassifier
13 from sklearn.metrics import plot_confusion_matrix
14 from sklearn.metrics import classification_report
15 from sklearn.metrics import mean_squared_error
16 from sklearn.metrics import accuracy_score
17 from sklearn.metrics import roc_curve, auc, roc_auc_score
18 from sklearn.tree import plot_tree
19 from sklearn import tree
20 from six import StringIO
21 from IPython.display import Image
```

```
In [2]:
```

```
In [3]:
```

Out[3]:

	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 [4]: 1 #dropping columns that won't be used
        2 data.drop(columns=['state', 'account length', 'area code', 'phone

In [5]: 1 #renaming columns
        2 data = data.rename(columns = {'international plan': 'International

In [6]: 1 #change international plan, voicemail plan and churn to numerical
        2 #0 = no/false, 1 = yes/true
        3 #also making sure these values are numbers and not strings
        4
        5 data['International Plan'] = data['International Plan'].replace('n
        6 data['International Plan'] = data['International Plan'].replace('y
        7 data['Voicemail Plan'] = data['International Plan'].replace('no',
        8 data['Voicemail Plan'] = data['International Plan'].replace('yes',
        9 data['Churn'] = data['International Plan'].replace('False', '0')
       10 data['Churn'] = data['International Plan'].replace('True', '1')
       11
       12 data['International Plan'] = data['International Plan'].astype(int
       13 data['Voicemail Plan'] = data['International Plan'].astype(int)
       14 data['Churn'] = data['International Plan'].astype(int)
       15

In [7]: 1 #defining x and y
        2 y = data[['Churn']].values.ravel()
        3 #x = data[['International Plan', 'Voicemail Plan', 'Number of Voic
        4 x = data.drop('Churn', axis=1)

In [8]: 1 #defining x train, y train, x test and y test
        2

In [9]: 1 #using a defined function for modeling to streamline the process
        2 def run_model(model, x_train, y_train, x_test, y_test):
        3
        4     #fitting
        5     model.fit(x_train, y_train)
        6
        7     #predictions
        8     y_hat_train = model.predict(x_train)
        9     y_hat_test = model.predict(x_test)
       10
       11     print('Classification Report: Train Set \n')
       12     print(classification_report(y_train, y_hat_train))
       13     print('Classification Report: Test Set \n')
       14     print(classification_report(y_test, y_hat_test))
       15
       16     fig, (ax0, ax1) = plt.subplots(1, 2, figsize=(18,6))
       17
       18     plot_confusion_matrix(model, x_train, y_train, ax=ax0)
       19     plot_confusion_matrix(model, x_test, y_test, ax=ax1)
       20
       21     ax0.title.set_text('Train Confusion Matrix')
       22     ax1.title.set_text('Test Confusion Matrix')
       23
       24     return model
```

```
In [10]: 1 #looking and checking for null or missing data
```

```
Out[10]:
```

	International Plan	Voicemail Plan	Number of Voicemail Messages	Total Day Minutes	Total Day Calls	Total Evening Minutes	Evening Calls
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000
mean	0.096910	0.096910	8.099010	179.775098	100.435644	200.980348	100.111111
std	0.295879	0.295879	13.688365	54.467389	20.069084	50.713844	19.922222
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	143.700000	87.000000	166.600000	87.000000
50%	0.000000	0.000000	0.000000	179.400000	101.000000	201.400000	100.000000
75%	0.000000	0.000000	20.000000	216.400000	114.000000	235.300000	114.000000
max	1.000000	1.000000	51.000000	350.800000	165.000000	363.700000	170.000000

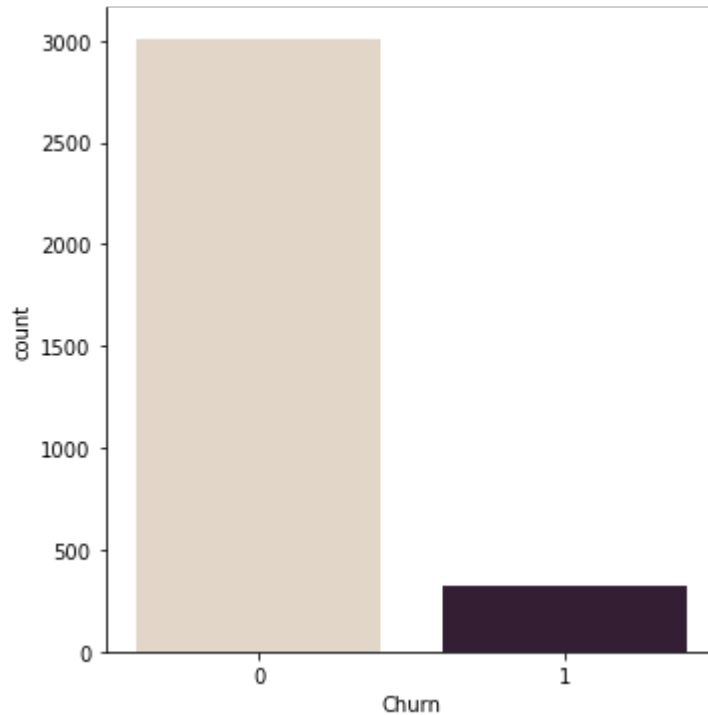
```
In [11]: <class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 13 columns):
```

#	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 [12]: 1 #looking at retention/churn values
          2 print(data["Churn"].value_counts())
          3
          0    3010
          1     323
          Name: Churn, dtype: int64
```

Out[12]: <seaborn.axisgrid.FacetGrid at 0x15f77c25400>



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 [13]: 1 #using Decision Trees as first model
```

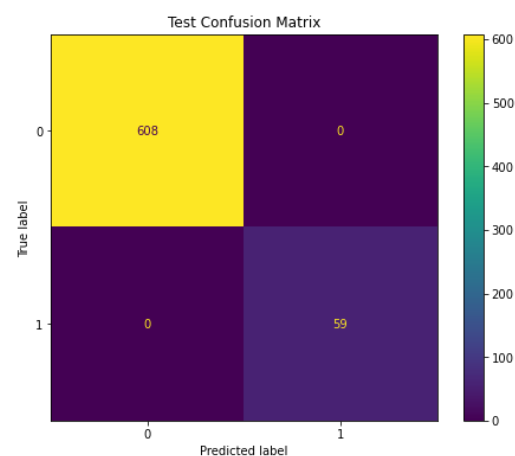
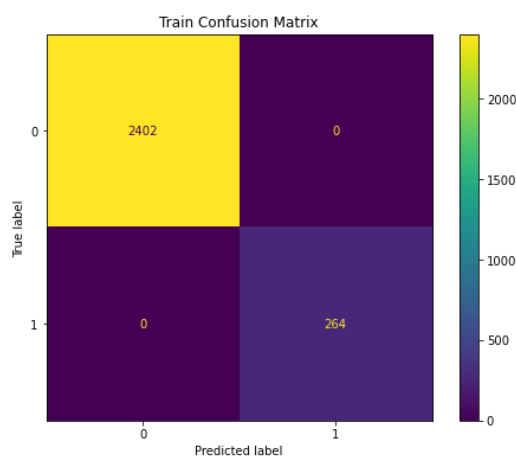
```
In [14]:
```

Classification Report: Train Set

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2402
1	1.00	1.00	1.00	264
accuracy			1.00	2666
macro avg	1.00	1.00	1.00	2666
weighted avg	1.00	1.00	1.00	2666

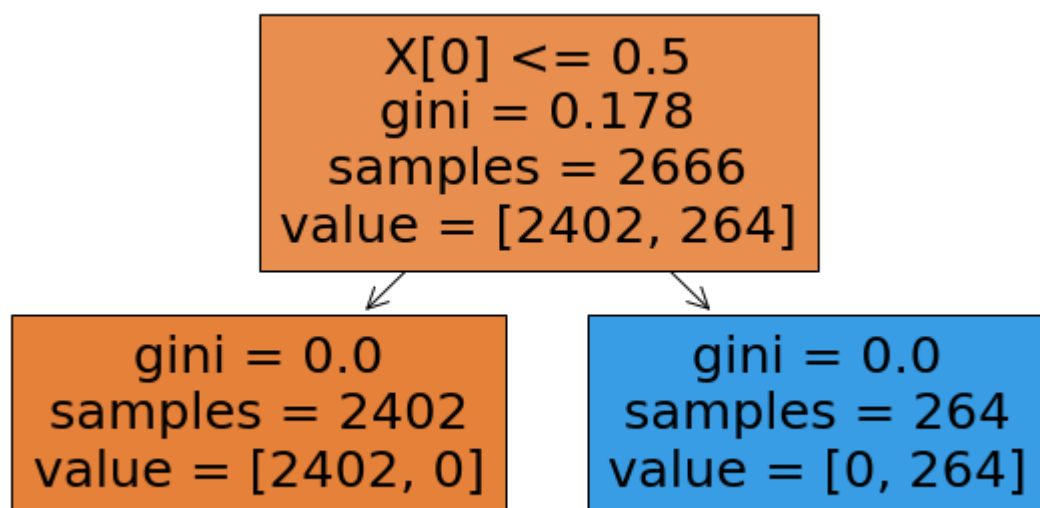
Classification Report: Test Set

Out[14]: DecisionTreeClassifier()



In [15]: `#plot the decision tree`
`plt.figure(figsize=(10, 5))`

Out[15]: [Text(279.0, 203.85000000000002, 'X[0] <= 0.5\ngini = 0.178\nsamples = 2666\nvalue = [2402, 264]'),
 Text(139.5, 67.94999999999999, 'gini = 0.0\nsamples = 2402\nvalue = [2402, 0]'),
 Text(418.5, 67.94999999999999, 'gini = 0.0\nsamples = 264\nvalue = [0, 264]')]



```
In [16]: 1 #feature importances
2 dtc.feature_importances_
3 df = pd.DataFrame({'Feature Names':x.columns, 'Importance':dtc.fea
```

Out[16]:

	Feature Names	Importance
0	International Plan	1.0
1	Voicemail 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 [17]: 1 #using Random Forest as second model
```

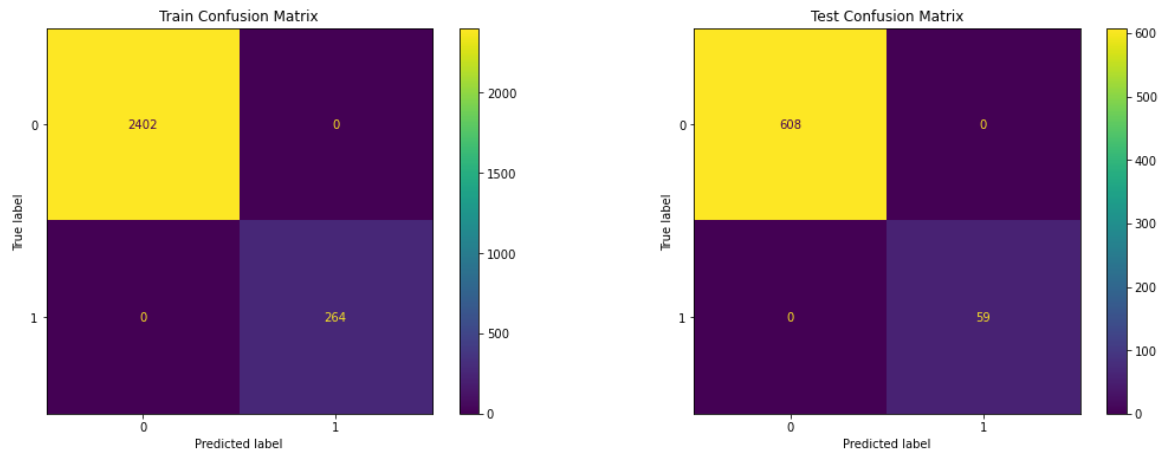
```
In [18]: 1 #using Random Forest as second model
```

Classification Report: Train Set

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2402
1	1.00	1.00	1.00	264
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[18]: RandomForestClassifier()



```
In [19]: 1 #feature importances
2 ranfor.feature_importances_
3 df = pd.DataFrame({'Feature Names':x.columns, 'Importance':ranfor.
```

Out[19]:

	Feature Names	Importance
0	International Plan	0.503334
1	Voicemail Plan	0.470974
3	Total Day Minutes	0.004971
9	Total International Minutes	0.003749
7	Total Night Minutes	0.003599
5	Total Evening Minutes	0.003076
4	Total Day Calls	0.002476
10	Total International Calls	0.001708
8	Total Night Calls	0.001667
6	Total Evening Calls	0.001585
11	Customer Service Calls	0.001483
2	Number of Voicemail Messages	0.001378

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

```
In [20]: 1 #using Logistic Regression as third model
```


In [21]:

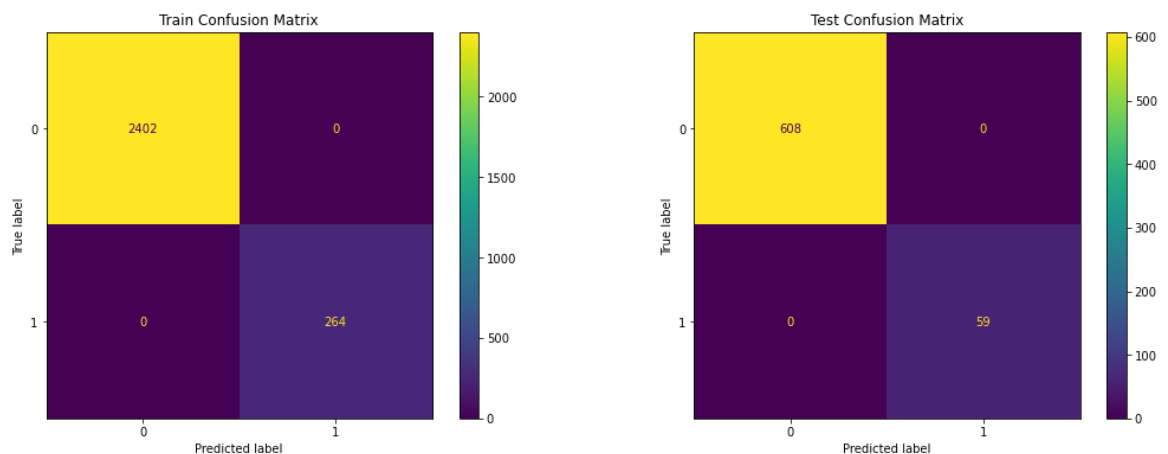
Classification Report: Train Set

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2402
1	1.00	1.00	1.00	264
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

	precision	recall	f1-score	support
0	1.00	1.00	1.00	608
1	1.00	1.00	1.00	59
accuracy			1.00	667
macro avg	1.00	1.00	1.00	667
weighted avg	1.00	1.00	1.00	667

Out[21]: LogisticRegression(max_iter=500)



The results of our Logistic Regression model are the same as the previous ones in terms of having good scores across the confusion matrices, which is great for our dataset but makes it that much tougher choosing one for our final model. Logistic Regression models are great since they are easy to implement and edit information to reflect new data, less prone to over fitting and when dealing with simplistic datasets, tend to be more accurate than others.

There are a few downsides to Logistic Regression models though, including needing a significant amount of data since otherwise it will lead to over fitting, sometimes having difficulty dealing with more complex relationships among data. Also keeping in mind that not all problems are linear, so it may not be worth the time or effort transforming the data to work with the model. These are all valid points for and against Logistic Regression when deciding which model will be the best fit for us.

5 Gaussian Naive Bayes Modeling

In [22]: `1 #using Naive Bayes GaussianNB as fourth model`

In [23]: `1`

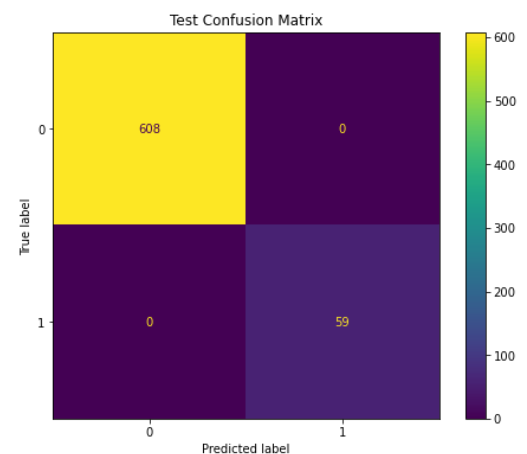
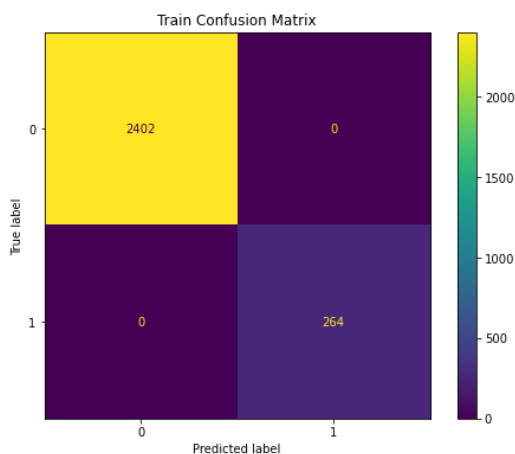
Classification Report: Train Set

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2402
1	1.00	1.00	1.00	264
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

	precision	recall	f1-score	support
0	1.00	1.00	1.00	608
1	1.00	1.00	1.00	59
accuracy			1.00	667
macro avg	1.00	1.00	1.00	667
weighted avg	1.00	1.00	1.00	667

Out[23]: `GaussianNB()`



Once again looking at the scores across the matrix board, we see great results amongst all categories. Seeing these indicate that the Naive Bayes Gaussian Model might be a good fit for our customer retention problem. This model does better with less training data (if the assumption of independent feature holds true), is quick to use and can easily predict multiple class prediction problem.

However, it should be noted that there are a few catches when working with this type of model.

This model assumes that all features are independent to each other, which is tad unrealistic when dealing with data collected from the real world. Also, this model is notoriously known for being an inadequate estimator so using the "predict_proba" feature can't be relied on.

▼ 6 Nearest Neighbors Modeling

In [24]:  1 *#using KNN for fifth model*

In [25]:  1 *#using KNN for fifth model*

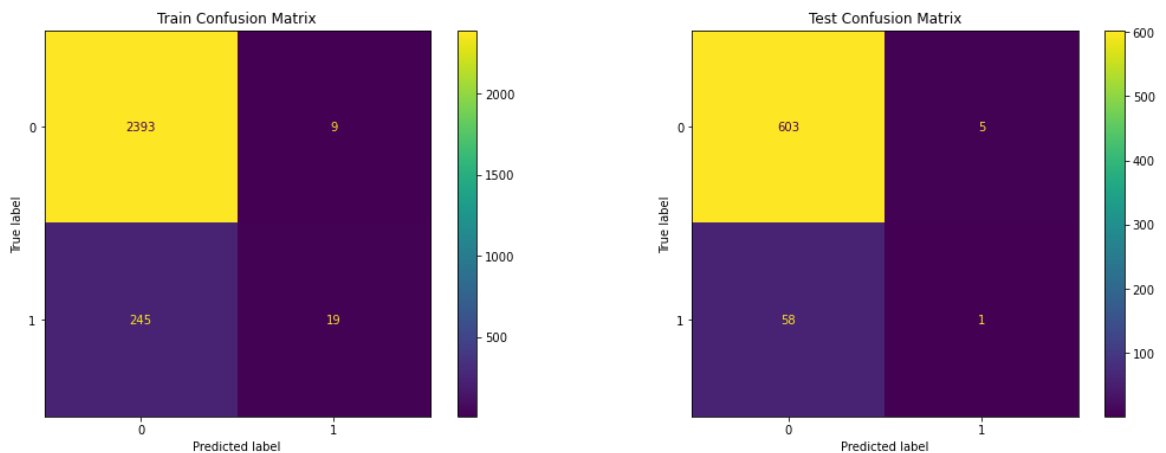
Classification Report: Train Set

	precision	recall	f1-score	support
0	0.91	1.00	0.95	2402
1	0.68	0.07	0.13	264
accuracy			0.90	2666
macro avg	0.79	0.53	0.54	2666
weighted avg	0.88	0.90	0.87	2666

Classification Report: Test Set

	precision	recall	f1-score	support
0	0.91	0.99	0.95	608
1	0.17	0.02	0.03	59
accuracy			0.91	667
macro avg	0.54	0.50	0.49	667
weighted avg	0.85	0.91	0.87	667

Out[25]: KNeighborsClassifier()



It seems the Nearest Neighbors Model (or the KNN for short) has finally broken the record for results. Here we can see in the confusion matrices that our model did not perform as well as the others did.

With our train set data we can see that 2,393 of the customers that did not churn were correctly classified by the model to have stayed with SyriaTel but 9 were not. Also, for the consumers who did leave/churn, we see that 19 customers were correctly classified as ending their services with SyriaTel while that 245 customers were predicted to leave, didn't.

Meanwhile, within our test set data we see that 603 of the customers who stayed with SyriaTel were correctly classified by the model while 5 weren't. In addition, for 1 customer they were correctly classified as canceling while 58 people who were expected to do so, remained instead.

What this means is that this model predicts more people deciding to break away from our company than they actually do. This can be a good thing in a way, as that our results for who will end up leaving will better than what we can expect. However, on the other side of the coin, some of those customers that we expect to stay, will instead leave.

With the KNN model, the upsides of using it is that the model is straightforward, it doesn't make any assumptions with the data and with the more data it deals with, the more it learns and evolves on it's own. It also very simple to use for multi-class problems and can be used for both classification and regression problems.

Just like the others, there are some disadvantages to using the Nearest Neighbors Model. For instance, it is true that the model will learn more as the data set grows, but with it are the potential for outlier sensitivity that may skew results as well as the difficulty of finding out which is the true optimal number of neighbors to use. Furthermore, as the data set gets bigger, not only does KNN not have the capability to deal with missing value points, but the speed and efficiency of the model declines. With this in mind, while Nearest Neighbors might be the right model to use from some datasets, it seems unlikely that it will be our final model.

▼ 7 Final Model Outcomes and Recommendations

```
In [26]: 1 #random forest with gridsearch
          2 ranfor_grid = {'n_estimators': [150], 'max_depth':[15], 'criterion': 'gini'}
          3
          4 ranfor_search = GridSearchCV(RandomForestClassifier(), ranfor_grid)
          5
          6 ranfor_search.fit(x_train, y_train)
```

```
Out[26]: GridSearchCV(estimator=RandomForestClassifier(),
                      param_grid={'class_weight': ['balanced', 'balanced_subsample'],
                                   'criterion': ['gini', 'entropy'], 'max_depth': [15],
                                   'n_estimators': [150]},
                      scoring='f1')
```

```
In [27]: 1
          2
          3
          4
          5
          6
          7
          8
          9
          10
          11
          12
          13
          14
          15
          16
          17
          18
          19
          20
          21
          22
          23
          24
          25
          26
          27
          28
          29
          30
          31
          32
          33
          34
          35
          36
          37
          38
          39
          40
          41
          42
          43
          44
          45
          46
          47
          48
          49
          50
          51
          52
          53
          54
          55
          56
          57
          58
          59
          60
          61
          62
          63
          64
          65
          66
          67
          68
          69
          70
          71
          72
          73
          74
          75
          76
          77
          78
          79
          80
          81
          82
          83
          84
          85
          86
          87
          88
          89
          90
          91
          92
          93
          94
          95
          96
          97
          98
          99
          100
          101
          102
          103
          104
          105
          106
          107
          108
          109
          110
          111
          112
          113
          114
          115
          116
          117
          118
          119
          120
          121
          122
          123
          124
          125
          126
          127
          128
          129
          130
          131
          132
          133
          134
          135
          136
          137
          138
          139
          140
          141
          142
          143
          144
          145
          146
          147
          148
          149
          150
          151
          152
          153
          154
          155
          156
          157
          158
          159
          160
          161
          162
          163
          164
          165
          166
          167
          168
          169
          170
          171
          172
          173
          174
          175
          176
          177
          178
          179
          180
          181
          182
          183
          184
          185
          186
          187
          188
          189
          190
          191
          192
          193
          194
          195
          196
          197
          198
          199
          200
          201
          202
          203
          204
          205
          206
          207
          208
          209
          210
          211
          212
          213
          214
          215
          216
          217
          218
          219
          220
          221
          222
          223
          224
          225
          226
          227
          228
          229
          230
          231
          232
          233
          234
          235
          236
          237
          238
          239
          240
          241
          242
          243
          244
          245
          246
          247
          248
          249
          250
          251
          252
          253
          254
          255
          256
          257
          258
          259
          260
          261
          262
          263
          264
          265
          266
          267
          268
          269
          270
          271
          272
          273
          274
          275
          276
          277
          278
          279
          280
          281
          282
          283
          284
          285
          286
          287
          288
          289
          290
          291
          292
          293
          294
          295
          296
          297
          298
          299
          300
          301
          302
          303
          304
          305
          306
          307
          308
          309
          310
          311
          312
          313
          314
          315
          316
          317
          318
          319
          320
          321
          322
          323
          324
          325
          326
          327
          328
          329
          330
          331
          332
          333
          334
          335
          336
          337
          338
          339
          340
          341
          342
          343
          344
          345
          346
          347
          348
          349
          350
          351
          352
          353
          354
          355
          356
          357
          358
          359
          360
          361
          362
          363
          364
          365
          366
          367
          368
          369
          370
          371
          372
          373
          374
          375
          376
          377
          378
          379
          380
          381
          382
          383
          384
          385
          386
          387
          388
          389
          390
          391
          392
          393
          394
          395
          396
          397
          398
          399
          400
          401
          402
          403
          404
          405
          406
          407
          408
          409
          410
          411
          412
          413
          414
          415
          416
          417
          418
          419
          420
          421
          422
          423
          424
          425
          426
          427
          428
          429
          430
          431
          432
          433
          434
          435
          436
          437
          438
          439
          440
          441
          442
          443
          444
          445
          446
          447
          448
          449
          450
          451
          452
          453
          454
          455
          456
          457
          458
          459
          460
          461
          462
          463
          464
          465
          466
          467
          468
          469
          470
          471
          472
          473
          474
          475
          476
          477
          478
          479
          480
          481
          482
          483
          484
          485
          486
          487
          488
          489
          490
          491
          492
          493
          494
          495
          496
          497
          498
          499
          500
          501
          502
          503
          504
          505
          506
          507
          508
          509
          510
          511
          512
          513
          514
          515
          516
          517
          518
          519
          520
          521
          522
          523
          524
          525
          526
          527
          528
          529
          530
          531
          532
          533
          534
          535
          536
          537
          538
          539
          540
          541
          542
          543
          544
          545
          546
          547
          548
          549
          550
          551
          552
          553
          554
          555
          556
          557
          558
          559
          560
          561
          562
          563
          564
          565
          566
          567
          568
          569
          570
          571
          572
          573
          574
          575
          576
          577
          578
          579
          580
          581
          582
          583
          584
          585
          586
          587
          588
          589
          590
          591
          592
          593
          594
          595
          596
          597
          598
          599
          600
          601
          602
          603
          604
          605
          606
          607
          608
          609
          610
          611
          612
          613
          614
          615
          616
          617
          618
          619
          620
          621
          622
          623
          624
          625
          626
          627
          628
          629
          630
          631
          632
          633
          634
          635
          636
          637
          638
          639
          640
          641
          642
          643
          644
          645
          646
          647
          648
          649
          650
          651
          652
          653
          654
          655
          656
          657
          658
          659
          660
          661
          662
          663
          664
          665
          666
          667
          668
          669
          670
          671
          672
          673
          674
          675
          676
          677
          678
          679
          680
          681
          682
          683
          684
          685
          686
          687
          688
          689
          690
          691
          692
          693
          694
          695
          696
          697
          698
          699
          700
          701
          702
          703
          704
          705
          706
          707
          708
          709
          710
          711
          712
          713
          714
          715
          716
          717
          718
          719
          720
          721
          722
          723
          724
          725
          726
          727
          728
          729
          730
          731
          732
          733
          734
          735
          736
          737
          738
          739
          740
          741
          742
          743
          744
          745
          746
          747
          748
          749
          750
          751
          752
          753
          754
          755
          756
          757
          758
          759
          760
          761
          762
          763
          764
          765
          766
          767
          768
          769
          770
          771
          772
          773
          774
          775
          776
          777
          778
          779
          780
          781
          782
          783
          784
          785
          786
          787
          788
          789
          790
          791
          792
          793
          794
          795
          796
          797
          798
          799
          800
          801
          802
          803
          804
          805
          806
          807
          808
          809
          810
          811
          812
          813
          814
          815
          816
          817
          818
          819
          820
          821
          822
          823
          824
          825
          826
          827
          828
          829
          830
          831
          832
          833
          834
          835
          836
          837
          838
          839
          840
          841
          842
          843
          844
          845
          846
          847
          848
          849
          850
          851
          852
          853
          854
          855
          856
          857
          858
          859
          860
          861
          862
          863
          864
          865
          866
          867
          868
          869
          870
          871
          872
          873
          874
          875
          876
          877
          878
          879
          880
          881
          882
          883
          884
          885
          886
          887
          888
          889
          890
          891
          892
          893
          894
          895
          896
          897
          898
          899
          900
          901
          902
          903
          904
          905
          906
          907
          908
          909
          910
          911
          912
          913
          914
          915
          916
          917
          918
          919
          920
          921
          922
          923
          924
          925
          926
          927
          928
          929
          930
          931
          932
          933
          934
          935
          936
          937
          938
          939
          940
          941
          942
          943
          944
          945
          946
          947
          948
          949
          950
          951
          952
          953
          954
          955
          956
          957
          958
          959
          960
          961
          962
          963
          964
          965
          966
          967
          968
          969
          970
          971
          972
          973
          974
          975
          976
          977
          978
          979
          980
          981
          982
          983
          984
          985
          986
          987
          988
          989
          990
          991
          992
          993
          994
          995
          996
          997
          998
          999
          1000
          1001
          1002
          1003
          1004
          1005
          1006
          1007
          1008
          1009
          1010
          1011
          1012
          1013
          1014
          1015
          1016
          1017
          1018
          1019
          1020
          1021
          1022
          1023
          1024
          1025
          1026
          1027
          1028
          1029
          1030
          1031
          1032
          1033
          1034
          1035
          1036
          1037
          1038
          1039
          1040
          1041
          1042
          1043
          1044
          1045
          1046
          1047
          1048
          1049
          1050
          1051
          1052
          1053
          1054
          1055
          1056
          1057
          1058
          1059
          1060
          1061
          1062
          1063
          1064
          1065
          1066
          1067
          1068
          1069
          1070
          1071
          1072
          1073
          1074
          1075
          1076
          1077
          1078
          1079
          1080
          1081
          1082
          1083
          1084
          1085
          1086
          1087
          1088
          1089
          1090
          1091
          1092
          1093
          1094
          1095
          1096
          1097
          1098
          1099
          1100
          1101
          1102
          1103
          1104
          1105
          1106
          1107
          1108
          1109
          1110
          1111
          1112
          1113
          1114
          1115
          1116
          1117
          1118
          1119
          1120
          1121
          1122
          1123
          1124
          1125
          1126
          1127
          1128
          1129
          1130
          1131
          1132
          1133
          1134
          1135
          1136
          1137
          1138
          1139
          1140
          1141
          1142
          1143
          1144
          1145
          1146
          1147
          1148
          1149
          1150
          1151
          1152
          1153
          1154
          1155
          1156
          1157
          1158
          1159
          1160
          1161
          1162
          1163
          1164
          1165
          1166
          1167
          1168
          1169
          1170
          1171
          1172
          1173
          1174
          1175
          1176
          1177
          1178
          1179
          1180
          1181
          1182
          1183
          1184
          1185
          1186
          1187
          1188
          1189
          1190
          1191
          1192
          1193
          1194
          1195
          1196
          1197
          1198
          1199
          1200
          1201
          1202
          1203
          1204
          1205
          1206
          1207
          1208
          1209
          1210
          1211
          1212
          1213
          1214
          1215
          1216
          1217
          1218
          1219
          1220
          1221
          1222
          1223
          1224
          1225
          1226
          1227
          1228
          1229
          1230
          1231
          1232
          1233
          1234
          1235
          1236
          1237
          1238
          1239
          1240
          1241
          1242
          1243
          1244
          1245
          1246
          1247
          1248
          1249
          1250
          1251
          1252
          1253
          1254
          1255
          1256
          1257
          1258
          1259
          1260
          1261
          1262
          1263
          1264
          1265
          1266
          1267
          1268
          1269
          1270
          1271
          1272
          1273
          1274
          1275
          1276
          1277
          1278
          1279
          1280
          1281
          1282
          1283
          1284
          1285
          1286
          1287
          1288
          1289
          1290
          1291
          1292
          1293
          1294
          1295
          1296
          1297
          1298
          1299
          1300
          1301
          1302
          1303
          1304
          1305
          1306
          1307
          1308
          1309
          1310
          1311
          1312
          1313
          1314
          1315
          1316
          1317
          1318
          1319
          1320
          1321
          1322
          1323
          1324
          1325
          1326
          1327
          1328
          1329
          1330
          1331
          1332
          1333
          1334
          1335
          1336
          1337
          1338
          1339
          1340
          1341
          1342
          1343
          1344
          1345
          1346
          1347
          1348
          1349
          1350
          1351
          1352
          1353
          1354
          1355
          1356
          1357
          1358
          1359
          1360
          1361
          1362
          1363
          1364
          1365
          1366
          1367
          1368
          1369
          1370
          1371
          1372
          1373
          1374
          1375
          1376
          1377
          1378
          1379
          1380
          1381
          1382
          1383
          1384
          1385
          1386
          1387
          1388
          1389
          1390
          1391
          1392
          1393
          1394
          1395
          1396
          1397
          1398
          1399
          1400
          1401
          1402
          1403
          1404
          1405
          1406
          1407
          1408
          1409
          1410
          1411
          1412
          1413
          1414
          1415
          1416
          1417
          1418
          1419
          1420
          1421
          1422
          1423
          1424
          1425
          1426
          1427
          1428
          1429
          1430
          1431
          1432
          1433
          1434
          1435
          1436
          1437
          1438
          1439
          1440
          1441
          1442
          1443
          1444
          1445
          1446
          1447
          1448
          1449
          1450
          1451
          1452
          1453
          1454
          1455
          1456
          1457
          1458
          1459
          1460
          1461
          1462
          1463
          1464
          1465
          1466
          1467
          1468
          1469
          1470
          1471
          1472
          1473
          1474
          1475
          1476
          1477
          1478
          1479
          1480
          1481
          1482
          1483
          1484
          1485
          1486
          1487
          1488
          1489
          1490
          1491
          1492
          1493
          1494
          1495
          1496
          1497
          1498
          1499
          1500
          1501
          1502
          1503
          1504
          1505
          1506
          1507
          1508
          1509
          1510
          1511
          1512
          1513
          1514
          1515
          1516
          1517
          1518
          1519
          1520
          1521
          1522
          1523
          1524
          1525
          1526
          1527
          1528
          1529
          1530
          1531
          1532
          1533
          1534
          1535
          1536
          1537
          1538
          1539
          1540
          1541
          1542
          1543
          1544
          1545
          1546
          1547
          1548
          1549
          1550
          1551
          1552
          1553
          1554
          1555
          1556
          1557
          1558
          1559
          1560
          1561
          1562
          1563
          1564
          1565
          1566
          1567
          1568
          1569
          1570
          1571
          1572
          1573
          1574
          1575
          1576
          1577
          1578
          1579
          1580
          1581
          1582
          1583
          1584
          1585
          1586
          1587
          1588
          1589
          1590
          1591
          1592
          1593
          1594
          1595
          1596
          1597
          1598
          1599
          1600
          1601
          1602
          1603
          1604
          1605
          1606
          1607
          1608
          1609
          1610
          1611
          1612
          1613
          1614
          1615
          1616
          1617
          1618
          1619
          1620
          1621
          1622
          1623
          1624
          1625
          1626
          1627
          1628
          1629
          1630
          1631
          1632
          1633
          1634
          1635
          1636
          1637
          1638
          1639
          1640
          1641
          1642
          1643
          1644
          1645
          1646
          1647
          1648
          1649
          1650
          1651
          1652
          1653
          1654
          1655
          1656
          1657
          1658
          1659
          1660
          1661
          1662
          1663
          1664
          1665
          1666
          1667
          1668
          1669
          1670
          1671
          1672
          1673
          1674
          1675
          1676
          1677
          1678
          1679
          1680
          1681
          1682
          1683
          1684
          1685
          1686
          1687
          1688
          1689
          1690
          1691
          1692
          1693
          1694
          1695
          1696
          1697
          1698
          1699
          1700
          1701
          1702
          1703
          1704
          1705
          1706
          1707
          1708
          1709
          1710
          1711
          1712
          1713
          1714
          1715
          1716
          1717
          1718
          1719
          1720
          1721
          1722
          1723
          1724
          1725
          1726
          1727
          1728
          1729
          1730
          1731
          1732
          1733
          1734
          1735
          1736
          1737
          1738
          1739
          1740
          1741
          1742
          1743
          1744
          1745
          1746
          1747
          1748
          1749
          1750
          1751
          1752
          1753
          1754
          1755
          1756
          1757
          1758
          1759
          1760
          1761
          1762
          1763
          1764
          1765
          1766
          1767
          1768
          1769
          1770
          1771
          1772
          1773
          1774
          1775
          1776
          1777
          1778
          1779
          1780
          1781
          1782
          1783
          1784
          1785
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_class_weight	param_c
0	0.285950	0.018427	0.018445	0.001088	balanced	
1	0.322477	0.015225	0.018911	0.000746	balanced	
2	0.401056	0.009298	0.018805	0.000762	balanced_subsample	

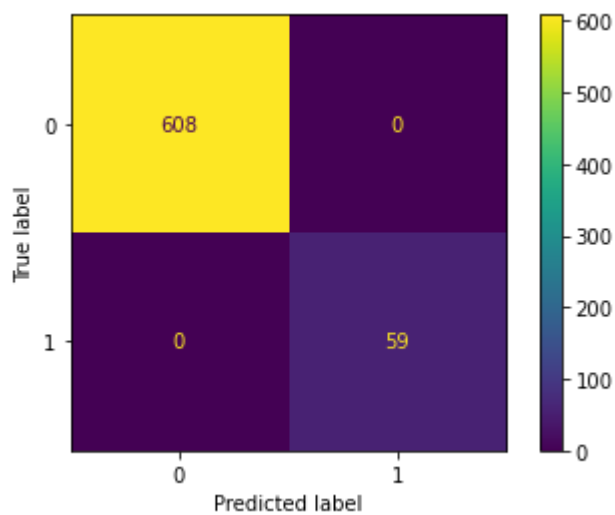
In [28]:

```
Out[28]: {'class_weight': 'balanced',
          'criterion': 'gini',
          'max_depth': 15,
          'n_estimators': 150}
```

In [29]:

```
1 test_preds = ranfor_search.predict(x_test)
```

```
Out[29]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x15f7d839130>
```



In [30]:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	608
1	1.00	1.00	1.00	59
accuracy			1.00	667
macro avg	1.00	1.00	1.00	667
weighted avg	1.00	1.00	1.00	667

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 followed by 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 SvriaTel.