Phase 3 Final Project Submission ¶

- Student name: Charlie Jin
- Student pace: Full time
- Scheduled project review date/time: Apr 21
- Instructor name: Daniel
- Blog post URL:

SyriaTel

Overview

This project analyzes the telephone/internet company SyriaTel - and their 3300 users listed in this study among their mobile telephone userbase. The primary focus will be the churn amount, or the amount of users who have reported to have left the service. This presents the business problem at hand - at which we'll attempt to solve.

Business Problem

A large amount of users have left the service. We need to retain further users to prevent them from leaving.

Data

https://www.kaggle.com/datasets/becksddf/churn-in-telecoms-dataset (https://www.kaggle.com/datasets/becksddf/churn-in-telecoms-dataset)

Methodology

1. Initial Trials - We will use multiple algorithms and models to predict user churn rate. The model will predict who it believes is most likely to leave the service. The goal will be to target those users to prevent them from leaving.

2. States and Area Codes - Through data manipulating, we can find out what states and area codes produces the highest churn amount. The users in those specific states and area codes or state/area code combinations will additionally be our target.

3. Decision Tree Method - The Decision Tree Method we believe will be the best method of approach and yield the most accurate predictions. However, we will first attempt to try out other methods to make sure.

4. Final Model - We will settle on a final model. Likely the model with the highest recall or precision score. Believing that to be the most accurate prediction model. The False Positive

prediction becomes the item of main importance after the model produces its final predictions, and therefore the False Positive amount. False Positives are users who the model predicts a high probability of leaving, but has not yet left. Those are the users specifically we need to prevent from leaving.

Part 1: Initial Trials

Trial 1: Logistic Regression Model

Dropping columns state and phone number, but using dummy variables for international plan and voice mail plan.

In [3]:
```
import numpy as np
import pandas as pd

from matplotlib import pyplot as plt

from sklearn.utils import resample
from sklearn.datasets import load_breast_cancer, load_iris, make_classificatio
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import confusion_matrix, plot_confusion_matrix,\
    precision_score, recall_score, accuracy_score, f1_score, log_loss,\
    roc_curve, roc_auc_score, classification_report
```

In [4]:
```
df = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Untitled F
```

In [207]:
```
df.head()
```

Out[207]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	tot ev cal
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	9
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	10
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	11
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	8
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	12

5 rows × 21 columns

In [208]:
```
df.describe()
```

Out[208]:

	account length	area code	number vmail messages	total day minutes	total day calls	total day charge	total ev minute
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.0000(
mean	101.064806	437.182418	8.099010	179.775098	100.435644	30.562307	200.9803
std	39.822106	42.371290	13.688365	54.467389	20.069084	9.259435	50.7138
min	1.000000	408.000000	0.000000	0.000000	0.000000	0.000000	0.0000(
25%	74.000000	408.000000	0.000000	143.700000	87.000000	24.430000	166.6000(
50%	101.000000	415.000000	0.000000	179.400000	101.000000	30.500000	201.4000(
75%	127.000000	510.000000	20.000000	216.400000	114.000000	36.790000	235.3000(
max	243.000000	510.000000	51.000000	350.800000	165.000000	59.640000	363.7000(

In [209]:
```
pd.get_dummies(df['international plan'])
```

Out[209]:

	no	yes
0	1	0
1	1	0
2	1	0
3	0	1
4	0	1
...	...	...
3328	1	0
3329	1	0
3330	1	0
3331	0	1
3332	1	0

3333 rows × 2 columns

In [210]:
```
pd.get_dummies(df['voice mail plan'])
```

Out[210]:

	no	yes
0	0	1
1	0	1
2	1	0
3	1	0
4	1	0
...	...	...
3328	0	1
3329	1	0
3330	1	0
3331	1	0
3332	0	1

3333 rows × 2 columns

In [211]:
```
internationaldummy = pd.get_dummies(df['international plan'])
```

In [212]:
```
voicemaildummy = pd.get_dummies(df['voice mail plan'])
```

In [213]:
```
df1 = pd.concat([voicemaildummy, df], axis=1)
df1.head()
```

Out[213]:

	no	yes	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	...	total eve calls	ch
0	0	1	KS	128	415	382-4657	no	yes	25	265.1	...	99	
1	0	1	OH	107	415	371-7191	no	yes	26	161.6	...	103	
2	1	0	NJ	137	415	358-1921	no	no	0	243.4	...	110	
3	1	0	OH	84	408	375-9999	yes	no	0	299.4	...	88	
4	1	0	OK	75	415	330-6626	yes	no	0	166.7	...	122	

5 rows × 23 columns

In [214]:
```
df1.rename(columns = {'no':'voicemailplan-no', 'yes':'voicemailplan-yes'}, inp
df1.head()
```

Out[214]:

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

5 rows × 23 columns

In [215]:
```
df2 = pd.concat([internationaldummy, df1], axis=1)
df2.head()
```

Out[215]:

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

5 rows × 25 columns

In [216]:
```
df2.rename(columns = {'no':'internationalplan-no', 'yes':'internationalplan-yes
df2.head()
```

Out[216]:

	internationalplan-no	internationalplan-yes	voicemailplan-no	voicemailplan-yes	state	account length	area code	ph num
0	1	0	0	1	KS	128	415	3 4
1	1	0	0	1	OH	107	415	3 7
2	1	0	1	0	NJ	137	415	3 1
3	0	1	1	0	OH	84	408	3 9
4	0	1	1	0	OK	75	415	3 6

5 rows × 25 columns

In [217]:
```
df2.drop(['state', 'phone number', 'international plan', 'voice mail plan'], a
```

In [218]:
```
df2.head()
```

Out[218]:

	internationalplan-no	internationalplan-yes	voicemailplan-no	voicemailplan-yes	account length	area code	number vmail messages
0	1	0	0	1	128	415	25
1	1	0	0	1	107	415	26
2	1	0	1	0	137	415	0
3	0	1	1	0	84	408	0
4	0	1	1	0	75	415	0

5 rows × 21 columns

In [219]:
```
X = df2.drop('churn', axis = 1)
y = df2['churn']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.25,
                                                    random_state=1)
# Scale the data for modeling
cred_scaler = StandardScaler()
cred_scaler.fit(X_train)
X_train_sc = cred_scaler.transform(X_train)
X_test_sc = cred_scaler.transform(X_test)

# Train a logistic regresssion model with the train data
cred_model = LogisticRegression(random_state=42)
cred_model.fit(X_train_sc, y_train)
```

Out[219]:
```
LogisticRegression(random_state=42)
```

In [220]:
```
cred_model.score(X_test_sc, y_test)
```

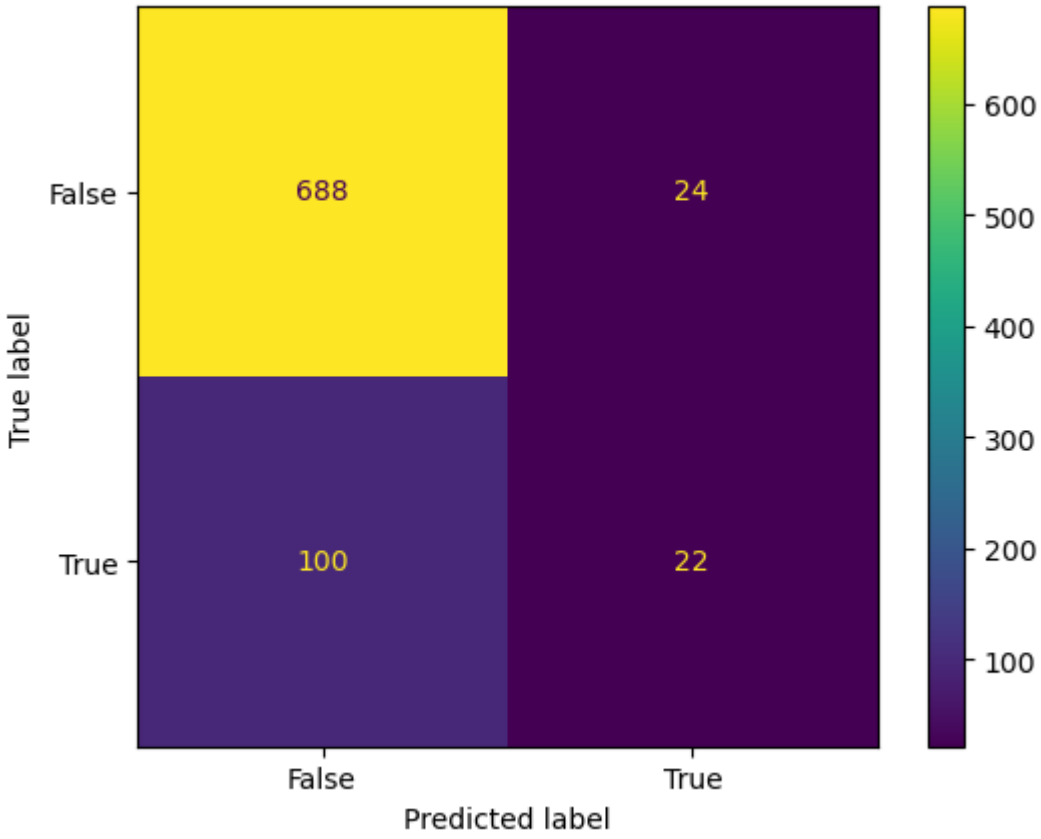
Out[220]:
```
0.8513189448441247
```

In [222]:
```
y_pred = cred_model.predict(X_test_sc)
cm_1 = confusion_matrix(y_test, y_pred)
cm_1
```

Out[222]:
```
array([[688,  24],
       [100,  22]], dtype=int64)
```

In [223]:
```
plot_confusion_matrix(cred_model, X_test_sc, y_test);
```

```
C:\Users\somep\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: F
utureWarning: Function plot_confusion_matrix is deprecated; Function `plot_co
nfusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of t
he class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixD
isplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



In [224]:
```
cm_1
```

Out[224]:
```
array([[688,  24],
       [100,  22]], dtype=int64)
```

In [225]:
```
tn = cm_1[0, 0]
fp = cm_1[0, 1]
fn = cm_1[1, 0]
tp = cm_1[1, 1]
```

In [226]:
```
acc = (tp + tn) / (tp + tn + fp + fn)
print(acc)
```

```
0.8513189448441247
```

In [227]:
```
rec = tp / (tp + fn)
print(rec)
```

```
0.18032786885245902
```

In [228]:
```
prec = tp / (tp + fp)
print(prec)
```

```
0.4782608695652174
```

In [229]:
```
f1_score = 2*prec*rec / (prec + rec)
print(f1_score)
```

```
0.2619047619047619
```

In [230]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       0.87      0.97      0.92       712
        True       0.48      0.18      0.26       122

    accuracy                           0.85       834
   macro avg       0.68      0.57      0.59       834
weighted avg       0.82      0.85      0.82       834
```

Trial 1 produces a recall score of .180327 - further trials are needed.

In [ ]:

In [ ]:

Trial 2: Logistic Regression Model (Removing the Dummy Variables)

Dropping columns state, area code, phone number, international plan, and voice mail plan.

In [4]:
```
dfday2 = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Untitl
```

In [5]:
```
dfday2.head()
```

Out[5]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	tot ev cal
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	9
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	10
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	11
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	8
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	12

5 rows × 21 columns

In [6]:
```
dfday2.drop(['state', 'area code', 'phone number', 'international plan', 'voic
```

In [8]:
```
X = dfday2.drop('churn', axis = 1)
y = dfday2['churn']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.25,
                                                    random_state=1)
# Scale the data for modeling
cred_scaler = StandardScaler()
cred_scaler.fit(X_train)
X_train_sc = cred_scaler.transform(X_train)
X_test_sc = cred_scaler.transform(X_test)

# Train a logistic regresssion model with the train data
cred_model = LogisticRegression(random_state=42)
cred_model.fit(X_train_sc, y_train)
```

Out[8]:
```
LogisticRegression(random_state=42)
```

In [9]:
```
cred_model.score(X_test_sc, y_test)
```

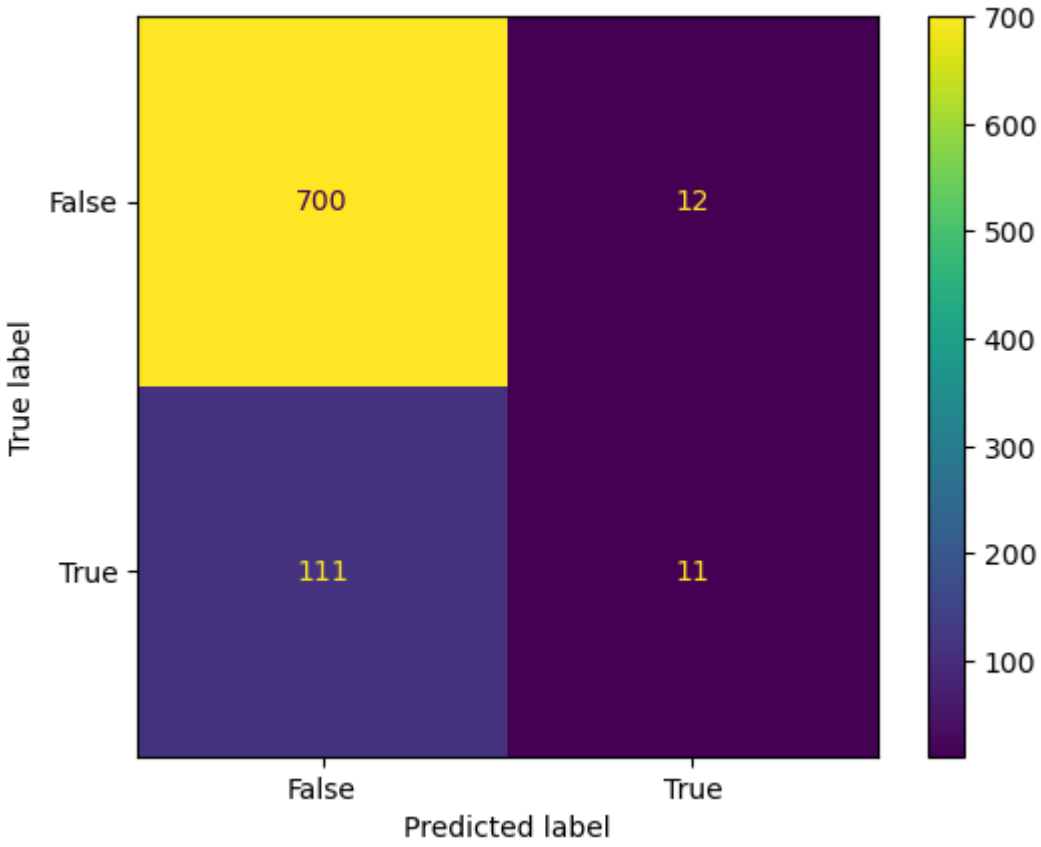
Out[9]:
```
0.8525179856115108
```

In [10]:
```
y_pred = cred_model.predict(X_test_sc)
cm_1 = confusion_matrix(y_test, y_pred)
cm_1
```

Out[10]:
```
array([[700,  12],
       [111,  11]], dtype=int64)
```

In [11]:
```
plot_confusion_matrix(cred_model, X_test_sc, y_test);
```

```
C:\Users\somep\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: F
utureWarning: Function plot_confusion_matrix is deprecated; Function `plot_co
nfusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of t
he class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixD
isplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



In [12]:
```
cm_1
```

Out[12]:
```
array([[700,  12],
       [111,  11]], dtype=int64)
```

In [13]:
```
tn = cm_1[0, 0]
fp = cm_1[0, 1]
fn = cm_1[1, 0]
tp = cm_1[1, 1]
```

In [14]:
```
rec = tp / (tp + fn)
print(rec)
```

```
0.09016393442622951
```

Trial 2 produces a recall score of .0901639 - removing the dummy variables of voice mail plan and

international plan seems to not be the right choice.

In [ ]:

In [ ]:

Trial 3: Cross Validation Model

Dropping columns state, area code, phone number, international plan, and voice mail plan.

In [4]:
```
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold
```

In [63]:
```
dfday2_2 = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Unti
```

In [64]:
```
dfday2_2.drop(['state', 'area code', 'phone number', 'international plan', 'vo
```

In [65]:
```
X = dfday2_2.drop('churn', axis = 1)
y = dfday2_2['churn']
```

In [66]:
```
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

In [67]:
```
clf = LogisticRegression(random_state=0)
cv = StratifiedKFold(n_splits=5)
```

In [68]:
```
scoring = ['precision_macro', 'recall_macro']
scores = cross_validate(clf, X, y, scoring=scoring, cv=cv, return_train_score=
```

In [70]:
```
print("Precision scores: ", scores['test_precision_macro'])
print("Recall scores: ", scores['test_recall_macro'])
print("Average precision score: ", np.mean(scores['test_precision_macro']))
print("Average recall score: ", np.mean(scores['test_recall_macro']))
```

```
Precision scores:  [0.73964949 0.71992481 0.67998478 0.65432099 0.74519231]
Recall scores:  [0.53685115 0.55396093 0.52138723 0.53289474 0.5745614 ]
Average precision score:  0.7078144761826403
Average recall score:  0.5439310906131307
```

Trial 3 produces a recall score of .543931 - an improvement over the previous 2 methods.

In [ ]:

In [ ]:

Trial 4: K Nearest Neighbors Model

Dropping columns state, area code, phone number, international plan, and voice mail plan.

In [75]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [76]:
```
dfday2_3 = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Unti
```

In [77]:
```
dfday2_3.drop(['state', 'area code', 'phone number', 'international plan', 'vo
```

In [78]:
```
X = dfday2_3.drop('churn', axis = 1)
y = dfday2_3['churn']
```

In [79]:
```
neighborsclassifier = KNeighborsClassifier(n_neighbors=3)
```

In [80]:
```
neighborsclassifier.fit(X_train, y_train)
```

Out[80]:
```
KNeighborsClassifier(n_neighbors=3)
```

In [81]:
```
y_pred = neighborsclassifier.predict(X_test)
```

```
C:\Users\somep\anaconda3\lib\site-packages\sklearn\neighbors\_classification.
py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtos
is`), the default behavior of `mode` typically preserves the axis it acts alo
ng. In SciPy 1.11.0, this behavior will change: the default value of `keepdim
s` will become False, the `axis` over which the statistic is taken will be el
iminated, and the value None will no longer be accepted. Set `keepdims` to Tr
ue or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [82]:
```
y_pred
```

Out[82]:
```
array([False,  True, False, ..., False, False,  True])
```

In [83]:
```
neighborsclassifier.score(X_test, y_test)
```

```
C:\Users\somep\anaconda3\lib\site-packages\sklearn\neighbors\_classification.
py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtos
is`), the default behavior of `mode` typically preserves the axis it acts alo
ng. In SciPy 1.11.0, this behavior will change: the default value of `keepdim
s` will become False, the `axis` over which the statistic is taken will be el
iminated, and the value None will no longer be accepted. Set `keepdims` to Tr
ue or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[83]:
```
0.8554545454545455
```

In [85]:
```
cm2 = confusion_matrix(y_test, y_pred)
cm2
```

Out[85]:
```
array([[898,  42],
       [117,  43]], dtype=int64)
```

In [87]:
```
tn = cm2[0, 0]
fp = cm2[0, 1]
fn = cm2[1, 0]
tp = cm2[1, 1]
```

In [88]:
```
rec = tp / (tp + fn)
print(rec)
```

```
0.26875
```

Trial 4 produces a recall score of .26875 - will continue to try more methods until the best one is achieved.

In [ ]:

In [ ]:

Trial 5: Dummy Classifier Model

In [89]:
```
from sklearn.dummy import DummyClassifier
```

In [91]:
```
dfday2_4 = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Unti
```

In [93]:
```
X = dfday2_4.drop('churn', axis = 1)
y = dfday2_4['churn']
```

In [95]:
```
frequentclf = DummyClassifier(strategy='most_frequent').fit(X_train, y_train)
```

In [96]:
```
yfreqpredict = frequentclf.predict(X_test)
```

In [97]:
```
uniformclf = DummyClassifier(strategy='uniform').fit(X_train, y_train)
```

In [98]:
```
yuniformpredict = uniformclf.predict(X_test)
```

In [99]:
```
frequentclf.score(X_test, y_test)
```

Out[99]:
```
0.8545454545454545
```

In [100]:
```
uniformclf.score(X_test, y_test)
```

Out[100]:
```
0.4781818181818182
```

In [101]:
```
from sklearn.metrics import confusion_matrix
```

In [102]:
```
confusion_matrix(y_test, y_pred)
```

Out[102]:
```
array([[898,  42],
       [117,  43]], dtype=int64)
```

In [105]:
```
cm3 = confusion_matrix(y_test, yfreqpredict)
cm3
```

Out[105]:
```
array([[940,   0],
       [160,   0]], dtype=int64)
```

In [106]:
```
tn = cm3[0, 0]
fp = cm3[0, 1]
fn = cm3[1, 0]
tp = cm3[1, 1]
```

In [107]:
```
rec = tp / (tp + fn)
print(rec)
```

```
0.0
```

In [108]:
```
cm4 = confusion_matrix(y_test, yuniformpredict)
cm4
```

Out[108]:
```
array([[482, 458],
       [ 80,  80]], dtype=int64)
```

In [109]:
```
tn = cm4[0, 0]
fp = cm4[0, 1]
fn = cm4[1, 0]
tp = cm4[1, 1]
```

In [110]:
```
rec = tp / (tp + fn)
print(rec)
```

```
0.5
```

Trial 5 produces a recall score of .00 for the most frequent predict classifier and a recall score of .50 for the uniform predictor classifier. Does not appear like this method changed much.

In [ ]:

In [ ]:

Trial 6: Random Forest Model

In [113]:
```
dfday2_5 = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Unti
```

In [114]:
```
X = dfday2_5.drop('churn', axis = 1)
y = dfday2_5['churn']
```

In [117]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

Out[117]:
```
RandomForestClassifier()
```

In [118]:
```
y_predforest = rf.predict(X_test)
```

In [119]:
```
cm5 = confusion_matrix(y_test, y_predforest)
cm5
```

Out[119]:
```
array([[930,  10],
       [ 85,  75]], dtype=int64)
```

In [120]:
```
tn = cm5[0, 0]
fp = cm5[0, 1]
fn = cm5[1, 0]
tp = cm5[1, 1]
```

In [121]:
```
rec = tp / (tp + fn)
print(rec)
```

```
0.46875
```

Trial 6 produces a recall score of .46875 - Will continue to try more methods.

In [ ]:

In [ ]:

Part 2: What States and Area Codes Have the Highest Churn Counts?

Taking a break from the previously using of algorithm models. Will need to find the States and Area Codes with the most users leaving. This valuable information is needed before continuing.

In [172]:
```
dfday3_1 = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Unti
```

In [173]:
```
pd.set_option('display.max_rows', None)
```

In [174]:
```
dfday3_1[['state', 'churn']].value_counts()
```

Out[174]:

state	churn	
WV	False	96
AL	False	72
VA	False	72
WI	False	71
MN	False	69
NY	False	68
WY	False	68
OH	False	68
OR	False	67
VT	False	65
ID	False	64
UT	False	62
IN	False	62
CT	False	62
AZ	False	60
RI	False	59
NC	False	57
CO	False	57
KS	False	57
MI	False	57
MO	False	56
NM	False	56
ND	False	56
NE	False	56
FL	False	55
TX	False	54
MT	False	54
MA	False	54
IL	False	53
MD	False	53
SD	False	52
NV	False	52
OK	False	52
DE	False	52
WA	False	52
MS	False	51
KY	False	51
NJ	False	50
HI	False	50
AK	False	49
ME	False	49
DC	False	49
TN	False	48
LA	False	47
NH	False	47
SC	False	46
GA	False	46
AR	False	44
IA	False	41
PA	False	37
CA	False	25
NJ	True	18
TX	True	18
MD	True	17
MI	True	16
NY	True	15

```
MN     True     15
WA     True     14
SC     True     14
NV     True     14
MS     True     14
MT     True     14
ME     True     13
KS     True     13
CT     True     12
NC     True     11
AR     True     11
MA     True     11
OR     True     11
WV     True     10
OH     True     10
UT     True     10
NH     True      9
IN     True      9
CA     True      9
CO     True      9
DE     True      9
ID     True      9
WY     True      9
OK     True      9
GA     True      8
AL     True      8
VT     True      8
KY     True      8
FL     True      8
PA     True      8
SD     True      8
MO     True      7
WI     True      7
RI     True      6
NM     True      6
ND     True      6
TN     True      5
VA     True      5
NE     True      5
IL     True      5
DC     True      5
LA     True      4
AZ     True      4
HI     True      3
IA     True      3
AK     True      3
dtype: int64
```

In [176]:
```
dfday3_1[['area code', 'churn', 'state']].value_counts()
```

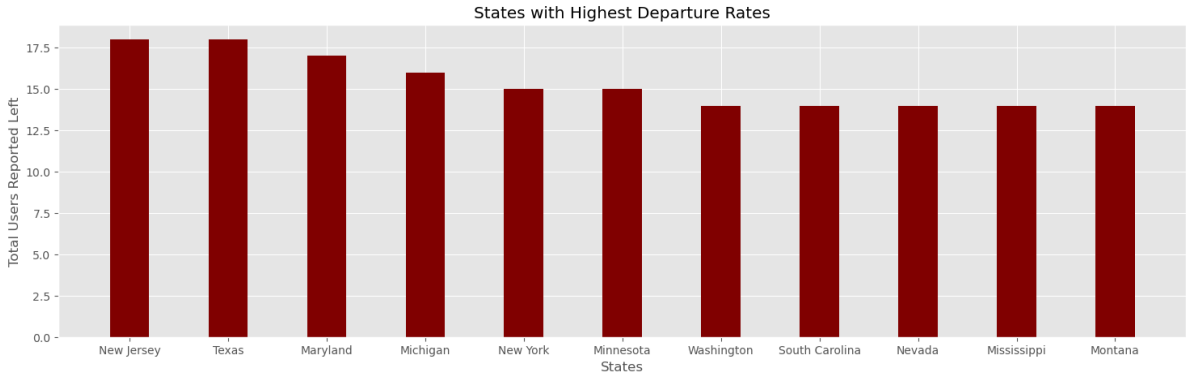
Out[176]:
```
area code  churn  state
415        False  WV       49
                  AL       37
                  NY       37
                  WY       36
                  OH       36
                  OR       36
                  RI       35
                  AZ       35
                  MO       34
                  MN       33
                  MD       33
                  ID       33
                  VA       33
                  WI       32
                  MI       31
                  VT       31
                  KS       31
                  NM       30
```

In [185]:
```
# creating the dataset
data = {'New Jersey':18, 'Texas':18, 'Maryland':17,
        'Michigan':16, 'New York':15, 'Minnesota':15, 'Washington':14, 'South (
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (18, 5))

# creating the bar plot
plt.bar(courses, values, color ='maroon',
        width = 0.4)

plt.xlabel("States")
plt.ylabel("Total Users Reported Left")
plt.title("States with Highest Departure Rates")
plt.show()
```

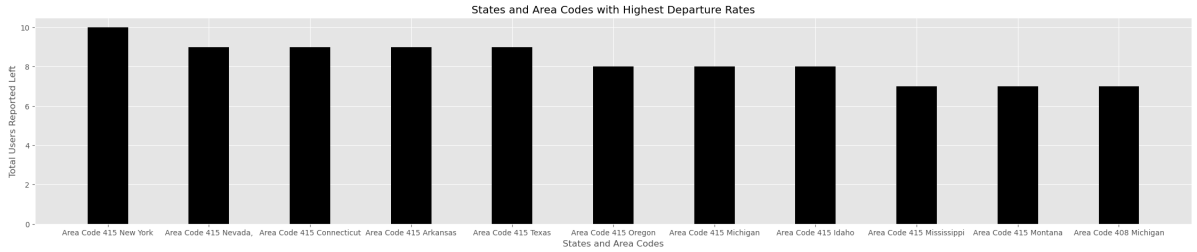


In [194]:
```
# creating the dataset
data = {'Area Code 415 New York':10, 'Area Code 415 Nevada,':9, 'Area Code 415
        'Area Code 415 Oregon':8, 'Area Code 415 Michigan':8, 'Area Code 415 I
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (28, 5))

# creating the bar plot
plt.bar(courses, values, color ='black',
        width = 0.4)

plt.xlabel("States and Area Codes")
plt.ylabel("Total Users Reported Left")
plt.title("States and Area Codes with Highest Departure Rates")
plt.show()
```



We find that states of Texas, New Jersey, Maryland, Michigan, and New York have the highest churn counts. The highest amount of users reported to have left the service. This is when you use states (no area code) to obtain the churn counts.

We also find that Area Code 415 in the states of New York, Connecticut, Arkansas, Texas, and Nevada experience the highest churn counts. This is found when you apply the combination of area code and state to obtain the churn counts - not just solely states.

In [ ]:

In [ ]:

Part 3: Decision Tree Method

Trial 1: Decision Tree Using Dummy Variables International Plan and Voice Mail Plan

believing Decision Tree is the best method for obtaining the optimum recall value, I will attempt multiple trials manipulating as many variables and parameters as I can.

I will compare using Decision Tree with Dummy Variables and without it to see which one produces the better recall score.

In [38]:
```
dfday3_5 = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Unti
```

In [39]:
```
dfday3_5.head()
```

Out[39]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	tot ev cal
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	9
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	10
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	11
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	8
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	12

5 rows × 21 columns

In [40]:
```
dfday3_5['international plan'].replace({'no':0, 'yes':1}, inplace=True)
```

In [41]:
```
dfday3_5['voice mail plan'].replace({'no':0, 'yes':1}, inplace=True)
```

In [42]:
```
dfday3_5.head()
```

Out[42]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	tot ev cal
0	KS	128	415	382-4657	0	1	25	265.1	110	45.07	...	9
1	OH	107	415	371-7191	0	1	26	161.6	123	27.47	...	10
2	NJ	137	415	358-1921	0	0	0	243.4	114	41.38	...	11
3	OH	84	408	375-9999	1	0	0	299.4	71	50.90	...	8
4	OK	75	415	330-6626	1	0	0	166.7	113	28.34	...	12

5 rows × 21 columns

In [43]:
```
dfday3_5.drop(['state', 'area code', 'phone number'], axis = 1, inplace=True)
```

In [44]:
```
X = dfday3_5.drop('churn', axis = 1)
y = dfday3_5['churn']
```

In [45]:
```
X_train,X_test,y_train,y_test, = train_test_split(X,y,test_size=0.33,random_st

X_train.shape, X_test.shape
```

Out[45]:
```
((2233, 17), (1100, 17))
```

In [46]:
```
fit2 = DecisionTreeClassifier(random_state=42)
fit2.fit(X_train,y_train)
```

Out[46]:
```
DecisionTreeClassifier(random_state=42)
```

In [47]:
```
ypred2 = fit2.predict(X_test)
ypred2
```

Out[47]:
```
array([False, False,  True, ...,  True,  True, False])
```

In [48]:
```
from sklearn.metrics import recall_score
recall_score(y_test, ypred2)
```

Out[48]:
```
0.75625
```

In [ ]:
```

```

Now Use Cross Validation on a new X value

In [34]:
```
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

In [35]:
```
clf = LogisticRegression(random_state=0)
cv = StratifiedKFold(n_splits=5)
```

In [36]:
```
scoring = ['precision_macro', 'recall_macro']
scores = cross_validate(clf, X, y, scoring=scoring, cv=cv, return_train_score=
```

In [37]:
```
print("Precision scores: ", scores['test_precision_macro'])
print("Recall scores: ", scores['test_recall_macro'])
print("Average precision score: ", np.mean(scores['test_precision_macro']))
print("Average recall score: ", np.mean(scores['test_recall_macro']))
```

```
Precision scores:  [0.73777349 0.70814699 0.679566   0.75613208 0.72304012]
Recall scores:  [0.60024417 0.60101284 0.57183939 0.58930921 0.59967105]
Average precision score:  0.7209317346618054
Average recall score:  0.5924153327907398
```

The recall score using the Decision Tree method with dummy variables international plan and voice mail plan is .75625

And the cross validation average recall score is .592415

This appears like some form of the Decision Tree method is the best method of choice for my project.

In [ ]:

In [ ]:

Trial 2: Decision Tree With No Dummy Variables

Just to compare if it's better using the dummy variables or not.

In [7]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, plot_roc_curve, plot_confusion_mat
from sklearn.datasets import load_iris

%matplotlib inline
```

In [8]:
```
dfday2_1 = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Unti
```

In [9]:
```
dfday2_1.drop(['state', 'area code', 'phone number', 'international plan', 'vo
```

In [11]:
```
X = dfday2_1.drop('churn', axis = 1)
y = dfday2_1['churn']
```

In [12]:
```
X_train,X_test,y_train,y_test, = train_test_split(X,y,test_size=0.33,random_st

X_train.shape, X_test.shape
```

Out[12]:
```
((2233, 15), (1100, 15))
```

In [13]:
```
dffit = DecisionTreeClassifier(random_state=42)
dffit.fit(X_train,y_train)
```

Out[13]:
```
DecisionTreeClassifier(random_state=42)
```

In [14]:
```
X_test
```

Out[14]:

	account length	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	tota nigh charg
438	113	0	155.0	93	26.35	330.6	106	28.10	189.4	123	8.5
2674	67	0	109.1	117	18.55	217.4	124	18.48	188.4	141	8.4
1345	98	0	0.0	0	0.00	159.6	130	13.57	167.1	88	7.5
1957	147	0	212.8	79	36.18	204.1	91	17.35	156.2	113	7.0
2148	96	0	144.0	102	24.48	224.7	73	19.10	227.7	91	10.2
...	...	...	...	...	...	...	...	...	...	...	..
2678	25	0	242.6	69	41.24	209.0	117	17.77	219.7	82	9.8
1506	136	0	252.4	74	42.91	167.9	81	14.27	248.3	110	11.1
2787	78	0	87.7	74	14.91	214.8	58	18.26	201.3	147	9.0
1133	64	0	148.1	73	25.18	164.9	101	14.02	216.0	125	9.7
3017	141	0	242.8	90	41.28	234.1	80	19.90	211.5	104	9.5

1100 rows × 15 columns

In [16]:
```
y_predicted = dffit.predict(X_test)
y_predicted
```

Out[16]:
```
array([False, False,  True, ..., False,  True, False])
```

In [17]:
```
from sklearn.metrics import recall_score
recall_score(y_test, y_predicted)
```

Out[17]:
```
0.60625
```

Decision Tree Method using no Dummy Variables produces a recall score of .60625 vs when using Dummy Variables the score is .75625

It is definitely a more accurate measure to use the dummy variables of international plan and voice mail plan in our dataset.

In [ ]:

In [ ]:

Trial 3: Decision Tree Tuning Hyperparameters with GridSearch

Dummy Variables international plan and voice mail plan included

In [18]:
```
import numpy as np
import pandas as pd

from sklearn.preprocessing import LabelBinarizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedK
from sklearn.metrics import roc_curve, precision_recall_curve, auc, make_scorer

import matplotlib.pyplot as plt
plt.style.use("ggplot")
```

In [19]:
```
dfday3_6 = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Untit
```

In [116]:
```
internationaldummy = pd.get_dummies(dfday3_6['international plan'])
```

In [117]:
```
voicemaildummy = pd.get_dummies(dfday3_6['voice mail plan'])
```

In [118]:
```
areacodedummy = pd.get_dummies(dfday3_6['area code'])
```

In [119]:
```
dfday3_7 = pd.concat([voicemaildummy, dfday3_6], axis=1)
```

In [120]:
```
dfday3_7.rename(columns = {'no':'voicemailplan-no', 'yes':'voicemailplan-yes'}
dfday3_7.head()
```

Out[120]:

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

5 rows × 23 columns

In [121]:
```
dfday3_8 = pd.concat([internationaldummy, dfday3_7], axis=1)
```

In [122]:
```
dfday3_8.rename(columns = {'no':'internationalplan-no', 'yes':'internationalpl
dfday3_8.head()
```

Out[122]:

	internationalplan-no	internationalplan-yes	voicemailplan-no	voicemailplan-yes	state	account length	area code	ph num
0	1	0	0	1	KS	128	415	3 4
1	1	0	0	1	OH	107	415	3 7
2	1	0	1	0	NJ	137	415	3 1
3	0	1	1	0	OH	84	408	3 9
4	0	1	1	0	OK	75	415	3 6

5 rows × 25 columns

In [123]:
```
dfday3_9 = pd.concat([areacodedummy, dfday3_8], axis=1)
```

In [124]:
```
dfday3_9.rename(columns = {'415':'areacode415', '408':'areacode408', '510':'ar
dfday3_9.head(7)
```

Out[124]:

	408	415	510	internationalplan-no	internationalplan-yes	voicemailplan-no	voicemailplan-yes	state	acco len
0	0	1	0	1	0	0	1	KS	
1	0	1	0	1	0	0	1	OH	
2	0	1	0	1	0	1	0	NJ	
3	1	0	0	0	1	1	0	OH	
4	0	1	0	0	1	1	0	OK	
5	0	0	1	0	1	1	0	AL	
6	0	0	1	1	0	0	1	MA	

7 rows × 28 columns

In [125]:
```
dfday3_9.drop(['state', 'area code', 'phone number', 'international plan', 'vo
```

In [126]:
```
X = dfday3_9.drop('churn', axis = 1)
y = dfday3_9['churn']
```

In [127]:
```
X_train,X_test,y_train,y_test, = train_test_split(X,y,test_size=0.33,random_st

X_train.shape, X_test.shape
```

Out[127]:
```
((2233, 22), (1100, 22))
```

In [128]:
```
fit3 = DecisionTreeClassifier(random_state=42)
fit3.fit(X_train,y_train)
```

```
C:\Users\somep\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got fea
ture names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[128]:
```
DecisionTreeClassifier(random_state=42)
```

In [129]:
```
ypred3 = fit3.predict(X_test)
ypred3
```

```
C:\Users\somep\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got fea
ture names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[129]:
```
array([False, False,  True, ...,  True,  True, False])
```

In [130]:
```
from sklearn.metrics import recall_score
recall_score(y_test, ypred3)
```

Out[130]:
```
0.7625
```

In [131]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [132]:
```
X=scaler.fit_transform(X)
X
```

```
C:\Users\somep\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got fea
ture names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
C:\Users\somep\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got fea
ture names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[132]:
```
array([[-0.57954443,  1.00692466, -0.5804683 , ..., -0.60119509,
        -0.0856905 , -0.42793202],
       [-0.57954443,  1.00692466, -0.5804683 , ..., -0.60119509,
         1.2411686 , -0.42793202],
       [-0.57954443,  1.00692466, -0.5804683 , ...,  0.21153386,
         0.69715637, -1.1882185 ],
       ...,
       [-0.57954443, -0.99312296,  1.72274698, ...,  0.61789834,
         1.3871231 ,  0.33235445],
       [-0.57954443, -0.99312296,  1.72274698, ...,  2.24335625,
        -1.87695028,  0.33235445],
       [-0.57954443,  1.00692466, -0.5804683 , ..., -0.19483061,
         1.2411686 , -1.1882185 ]])
```

In [133]:
```
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, random_st
```

In [134]:
```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()
```

In [135]:
```
clf.fit(X_train,y_train)
```

Out[135]:
```
DecisionTreeClassifier()
```

In [145]:
```
grid_values = {'max_features': ['sqrt', 0.25, 0.5, 0.75, 1.0],
               'max_depth' : [4,5,6,7,8],
              }
```

In [146]:
```
from sklearn.model_selection import GridSearchCV
grid=GridSearchCV(clf, param_grid=grid_values, cv=10)
```

In [147]:
```
grid.fit(X_train, y_train)
```

Out[147]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'max_depth': [4, 5, 6, 7, 8],
                         'max_features': ['sqrt', 0.25, 0.5, 0.75, 1.0]})
```

In [148]:
```
grid.best_params_
```

Out[148]:
```
{'max_depth': 7, 'max_features': 0.75}
```

In [151]:
```
grid_clf_acc = GridSearchCV(clf, param_grid = grid_values,scoring = 'recall')
grid_clf_acc.fit(X_train, y_train)

#Predict values based on new parameters
y_pred_acc = grid_clf_acc.predict(X_test)

# New Model Evaluation metrics
print('Accuracy Score : ' + str(accuracy_score(y_test,y_pred_acc)))
print('Precision Score : ' + str(precision_score(y_test,y_pred_acc)))
print('Recall Score : ' + str(recall_score(y_test,y_pred_acc)))
print('F1 Score : ' + str(f1_score(y_test,y_pred_acc)))

#Logistic Regression (Grid Search) Confusion matrix
confusion_matrix(y_test,y_pred_acc)
```

```
Accuracy Score : 0.9280359820089955
Precision Score : 0.7976190476190477
Recall Score : 0.6836734693877551
F1 Score : 0.7362637362637362
```

Out[151]:
```
array([[552,  17],
       [ 31,  67]], dtype=int64)
```

In [153]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_acc))
```

```
              precision    recall  f1-score   support

       False       0.95      0.97      0.96       569
        True       0.80      0.68      0.74        98

    accuracy                           0.93       667
   macro avg       0.87      0.83      0.85       667
weighted avg       0.92      0.93      0.93       667
```

Decision Tree Method using GridSearch produces a recall score of .683673, a fine score - but I will use the common Decision Tree method previous to that. The one which uses dummy variables International Plan and Voice Mail Plan.

That produces the best recall score of .7625

In [ ]:
```

```

In [ ]:

Part 4: Final Model

This Decision Tree Model using Dummy Variables international plan and voice mail plan will be my final model. Believing it to be the most effective model based on the optimum recall score .7625 and the highest counts of data entries accessed by this prediction model.

In [21]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, plot_roc_curve, plot_confusion_matr
from sklearn.datasets import load_iris

%matplotlib inline
```

In [22]:
```
dfday3_2 = pd.read_csv(r"C:\Users\somep\Documents\Flatiron\DS-Live-022023\Untit
```

In [24]:
```
dfday3_2.head()
```

Out[24]:

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	tot ev cal
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	9
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	10
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	11
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	8
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	12

5 rows × 21 columns

In [25]:
```
internationaldummy = pd.get_dummies(dfday3_2['international plan'])
```

In [26]:
```
voicemaildummy = pd.get_dummies(dfday3_2['voice mail plan'])
```

In [27]:
```
dfday3_3 = pd.concat([voicemaildummy, dfday3_2], axis=1)
dfday3_3.head()
```

Out[27]:

	no	yes	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	...	total eve calls	cl
0	0	1	KS	128	415	382-4657	no	yes	25	265.1	...	99	
1	0	1	OH	107	415	371-7191	no	yes	26	161.6	...	103	
2	1	0	NJ	137	415	358-1921	no	no	0	243.4	...	110	
3	1	0	OH	84	408	375-9999	yes	no	0	299.4	...	88	
4	1	0	OK	75	415	330-6626	yes	no	0	166.7	...	122	

5 rows × 23 columns

In [28]:
```
dfday3_3.rename(columns = {'no':'voicemailplan-no', 'yes':'voicemailplan-yes'}
dfday3_3.head()
```

Out[28]:

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

5 rows × 23 columns

In [29]:
```
dfday3_4 = pd.concat([internationaldummy, dfday3_3], axis=1)
dfday3_4.head()
```

Out[29]:

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

5 rows × 25 columns

In [30]:
```
dfday3_4.rename(columns = {'no':'internationalplan-no', 'yes':'internationalpl
dfday3_4.head()
```

Out[30]:

	internationalplan-no	internationalplan-yes	voicemailplan-no	voicemailplan-yes	state	account length	area code	ph num
0	1	0	0	1	KS	128	415	3 4
1	1	0	0	1	OH	107	415	3 7
2	1	0	1	0	NJ	137	415	3 1
3	0	1	1	0	OH	84	408	3 9
4	0	1	1	0	OK	75	415	3 6

5 rows × 25 columns

In [31]:
```
dfday3_4.drop(['state', 'area code', 'phone number', 'international plan', 'vo
```

In [32]:
```
dfday3_4.head()
```

Out[32]:

	internationalplan-no	internationalplan-yes	voicemailplan-no	voicemailplan-yes	account length	number vmail messages	tot da minut
0	1	0	0	1	128	25	265
1	1	0	0	1	107	26	161
2	1	0	1	0	137	0	243
3	0	1	1	0	84	0	299
4	0	1	1	0	75	0	166

In [33]:
```
X = dfday3_4.drop('churn', axis = 1)
y = dfday3_4['churn']
```

In [34]:
```
X_train,X_test,y_train,y_test, = train_test_split(X,y,test_size=0.33,random_st

X_train.shape, X_test.shape
```

Out[34]:
```
((2233, 19), (1100, 19))
```

In [35]:
```
dtfit = DecisionTreeClassifier(random_state=42)
dtfit.fit(X_train,y_train)
```

Out[35]:
```
DecisionTreeClassifier(random_state=42)
```

In [36]:
```
y_predict = dtfit.predict(X_test)
y_predict
```

Out[36]:
```
array([False, False,  True, ...,  True,  True, False])
```

In [37]:
```
from sklearn.metrics import recall_score
recall_score(y_test, y_predict)
```

Out[37]:
```
0.7625
```

In [38]:
```
confusion_matrix(y_test,y_predict)
```

Out[38]:
```
array([[882,  58],
       [ 38, 122]], dtype=int64)
```

Our final recall score is .7625 and the final False Positive (the main item of importance) count is 58.

In [ ]:
```

```

In [ ]:

Results

1. Final recall score is .7625
2. Final False Positive prediction count is 58
3. Users in states of New York, Texas, New Jersey, Maryland, and Michigan are reported to have left the service at a higher rate.
4. Users in Area Code 415 in the States of New York, Connecticut, Nevada, Arkansas, and Texas are reported to have left the service the most when you adjust to include area code.

Conclusion and Final Goals

We are successful in finding the most important results to move further with our goals. Our final goal is to prevent further users from leaving our service. We have narrowed down our search to target more specifically what state and area code they might be in. We have also an approximate prediction of just exactly how many users might leave the service. It is now much easier to move forward with our plans to retain these users. The discount, bundles, and coupons offered in high probability of leaving states in those area codes will be a more generous offer than normally.

In [ ]: