In [110…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.calibration import CalibratedClassifierCV
```

In [75]:
```
df1 = pd.read_csv('churn-bigml-80.csv')
df2 = pd.read_csv('churn-bigml-20.csv')
```

In [76]:
```
df_1=df1
df_2=df2
```

In [77]:
```
df_1
```

Out[77]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	To minu
0	KS	128	415	No	Yes	25	265.1	110	45.07	19
1	OH	107	415	No	Yes	26	161.6	123	27.47	19
2	NJ	137	415	No	No	0	243.4	114	41.38	12
3	OH	84	408	Yes	No	0	299.4	71	50.90	6
4	OK	75	415	Yes	No	0	166.7	113	28.34	14
…	…	…	…	…	…	…	…	…	…	
2661	SC	79	415	No	No	0	134.7	98	22.90	18
2662	AZ	192	415	No	Yes	36	156.2	77	26.55	21
2663	WV	68	415	No	No	0	231.1	57	39.29	15
2664	RI	28	510	No	No	0	180.8	109	30.74	28
2665	TN	74	415	No	Yes	25	234.4	113	39.85	26

2666 rows × 20 columns

In [78]:
```
df_1
```

Out[78]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	To minu
0	KS	128	415	No	Yes	25	265.1	110	45.07	19
1	OH	107	415	No	Yes	26	161.6	123	27.47	19
2	NJ	137	415	No	No	0	243.4	114	41.38	12
3	OH	84	408	Yes	No	0	299.4	71	50.90	6
4	OK	75	415	Yes	No	0	166.7	113	28.34	14
...	...	...	...	...	...	...	...	...	...	
2661	SC	79	415	No	No	0	134.7	98	22.90	18
2662	AZ	192	415	No	Yes	36	156.2	77	26.55	21
2663	WV	68	415	No	No	0	231.1	57	39.29	15
2664	RI	28	510	No	No	0	180.8	109	30.74	28
2665	TN	74	415	No	Yes	25	234.4	113	39.85	26

2666 rows × 20 columns

In [79]:
```
df_1.shape
```

Out[79]: (2666, 20)

In [80]:
```
df_2.shape
```

Out[80]: (667, 20)

In [81]:
```
df_1.head()
```

Out[81]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3

In [82]:
```
df_2.head()
```

Out[82]:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes
0	LA	117	408	No	No	0	184.5	97	31.37	351.6
1	IN	65	415	No	No	0	129.1	137	21.95	228.5
2	NY	161	415	No	No	0	332.9	67	56.59	317.8
3	SC	111	415	No	No	0	110.4	103	18.77	137.3
4	HI	49	510	No	No	0	119.3	117	20.28	215.1

In [83]:
```
df_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2666 entries, 0 to 2665
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   State                   2666 non-null   object 
 1   Account length          2666 non-null   int64  
 2   Area code               2666 non-null   int64  
 3   International plan      2666 non-null   object 
 4   Voice mail plan         2666 non-null   object 
 5   Number vmail messages   2666 non-null   int64  
 6   Total day minutes       2666 non-null   float64
 7   Total day calls         2666 non-null   int64  
 8   Total day charge        2666 non-null   float64
 9   Total eve minutes       2666 non-null   float64
 10  Total eve calls         2666 non-null   int64  
 11  Total eve charge        2666 non-null   float64
 12  Total night minutes     2666 non-null   float64
 13  Total night calls       2666 non-null   int64  
 14  Total night charge      2666 non-null   float64
 15  Total intl minutes      2666 non-null   float64
 16  Total intl calls        2666 non-null   int64  
 17  Total intl charge       2666 non-null   float64
 18  Customer service calls  2666 non-null   int64  
 19  Churn                   2666 non-null   bool   
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 398.5+ KB
```

In [84]:
```
df_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 667 entries, 0 to 666
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   667 non-null    object
 1   Account length          667 non-null    int64
 2   Area code               667 non-null    int64
 3   International plan      667 non-null    object
 4   Voice mail plan         667 non-null    object
 5   Number vmail messages   667 non-null    int64
 6   Total day minutes       667 non-null    float64
 7   Total day calls         667 non-null    int64
 8   Total day charge        667 non-null    float64
 9   Total eve minutes       667 non-null    float64
 10  Total eve calls         667 non-null    int64
 11  Total eve charge        667 non-null    float64
 12  Total night minutes     667 non-null    float64
 13  Total night calls       667 non-null    int64
 14  Total night charge      667 non-null    float64
 15  Total intl minutes      667 non-null    float64
 16  Total intl calls        667 non-null    int64
 17  Total intl charge       667 non-null    float64
 18  Customer service calls  667 non-null    int64
 19  Churn                   667 non-null    bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 99.8+ KB
```

In [85]:
```
df_1.isna().sum()
```

Out[85]:
```
State                     0
Account length            0
Area code                 0
International plan        0
Voice mail plan           0
Number vmail messages     0
Total day minutes         0
Total day calls           0
Total day charge          0
Total eve minutes         0
Total eve calls           0
Total eve charge          0
Total night minutes       0
Total night calls         0
Total night charge        0
Total intl minutes        0
Total intl calls          0
Total intl charge         0
Customer service calls    0
Churn                     0
dtype: int64
```

In [86]:
```
df_2.isna().sum()
```

Out[86]:
```
State                     0
Account length            0
Area code                 0
International plan        0
Voice mail plan           0
Number vmail messages     0
Total day minutes         0
Total day calls           0
Total day charge          0
Total eve minutes         0
Total eve calls           0
Total eve charge          0
Total night minutes       0
Total night calls         0
Total night charge        0
Total intl minutes        0
Total intl calls          0
Total intl charge         0
Customer service calls    0
Churn                     0
dtype: int64
```

In [87]:
```
df_1.duplicated().sum()
```

Out[87]:
```
0
```

In [88]:
```
df_2.duplicated().sum()
```

Out[88]:
```
0
```

In [89]:
```
df_1.describe()
```

Out[89]:

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Tot m
count	2666.000000	2666.000000	2666.000000	2666.00000	2666.000000	2666.000000	2666.0
mean	100.620405	437.438860	8.021755	179.48162	100.310203	30.512404	200.3
std	39.563974	42.521018	13.612277	54.21035	19.988162	9.215733	50.9
min	1.000000	408.000000	0.000000	0.00000	0.000000	0.000000	0.0
25%	73.000000	408.000000	0.000000	143.40000	87.000000	24.380000	165.3
50%	100.000000	415.000000	0.000000	179.95000	101.000000	30.590000	200.9
75%	127.000000	510.000000	19.000000	215.90000	114.000000	36.700000	235.1
max	243.000000	510.000000	50.000000	350.80000	160.000000	59.640000	363.7

In [90]:
```
df_2.describe()
```

Out[90]:

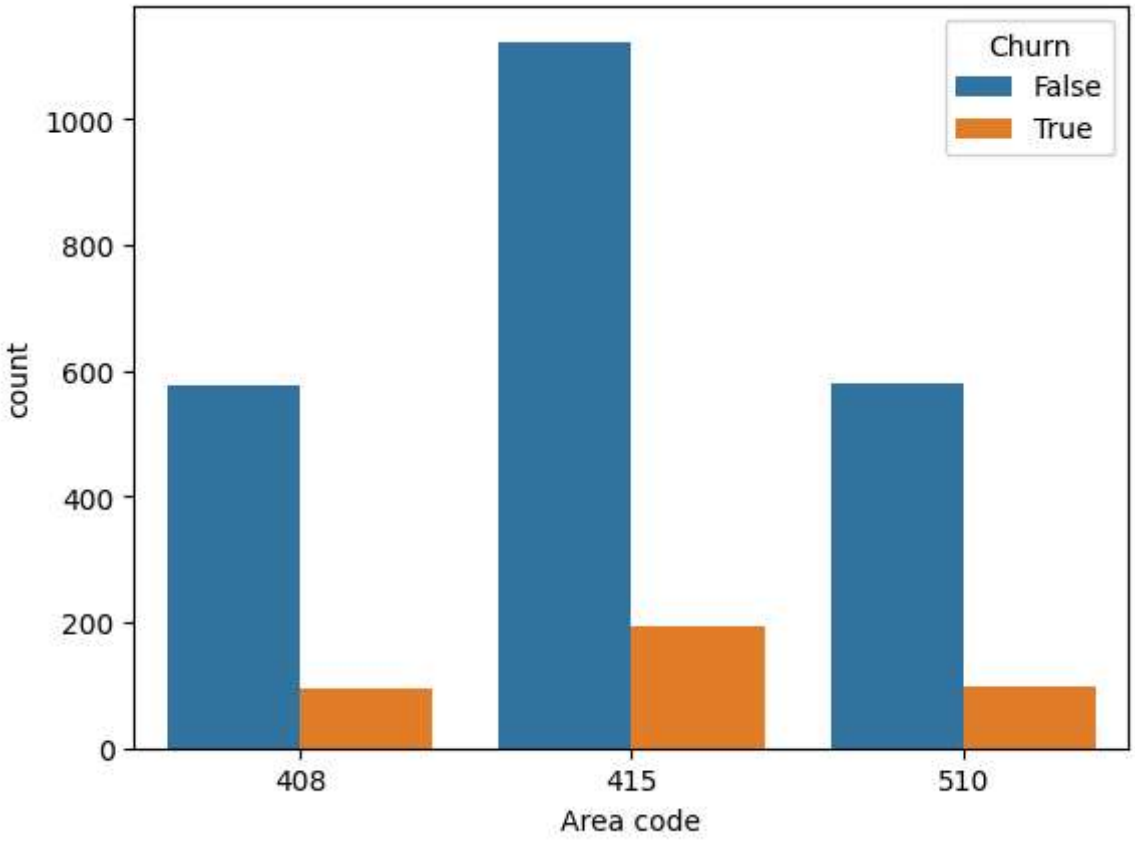
	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes
count	667.000000	667.000000	667.000000	667.000000	667.000000	667.000000	667.000000
mean	102.841079	436.157421	8.407796	180.948126	100.937031	30.761769	203.355322
std	40.819480	41.783305	13.994480	55.508628	20.396790	9.436463	49.719268
min	1.000000	408.000000	0.000000	25.900000	30.000000	4.400000	48.100000
25%	76.000000	408.000000	0.000000	146.250000	87.500000	24.860000	171.050000
50%	102.000000	415.000000	0.000000	178.300000	101.000000	30.310000	203.700000
75%	128.000000	415.000000	20.000000	220.700000	115.000000	37.520000	236.450000
max	232.000000	510.000000	51.000000	334.300000	165.000000	56.830000	361.800000

In [91]:

```
def crosstab_function(df, var):
    tab = pd.DataFrame(pd.crosstab(df[var], df["Churn"], margins=True)).reset_index
    tab['Percentage'] = tab[1] / tab['All'] * 100
    tab.columns = [var, 'Churn_NO', 'Churn_YES', 'Total', 'Churn Percentage']
    return tab

df_1['Area code'] = df_1['Area code'].astype('category')

sns.countplot(data=df_1, x=df_1['Area code'], hue='Churn');

crosstab_function(df_1,'Area code')
```

Out[91]:

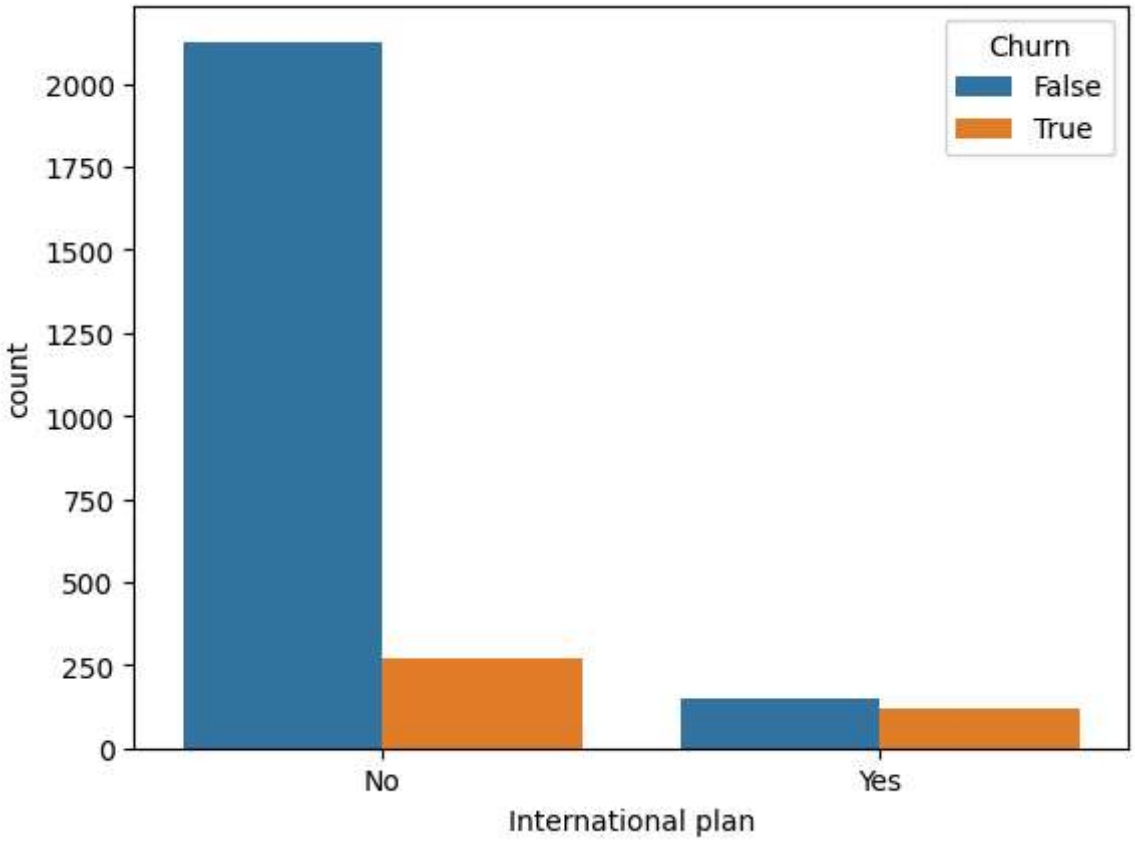
	Area code	Churn_NO	Churn_YES	Total	Churn Percentage
0	408	575	94	669	14.050822
1	415	1123	195	1318	14.795144
2	510	580	99	679	14.580265
3	All	2278	388	2666	14.553638



In [92]:
```
sns.countplot(data=df_1, x=df_1['International plan'], hue='Churn');

crosstab_function(df_1,'International plan')
```

Out[92]:

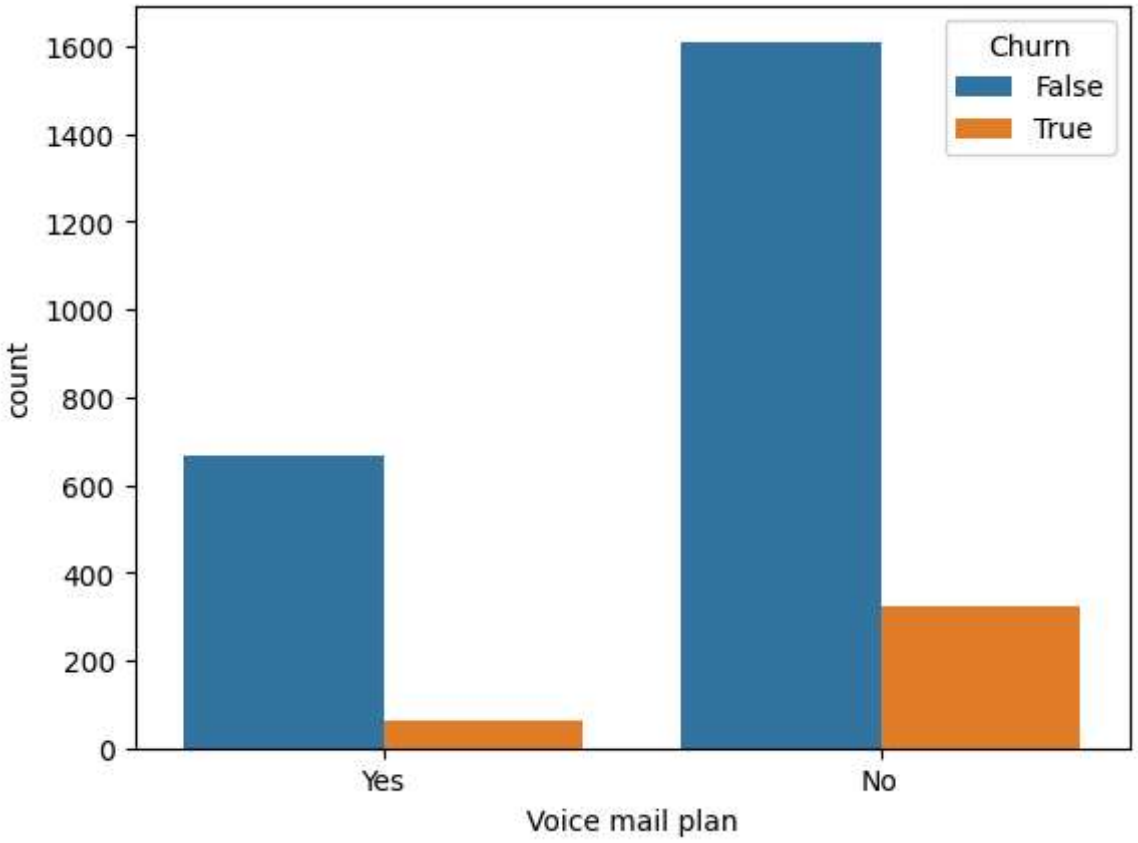
	International plan	Churn_NO	Churn_YES	Total	Churn Percentage
0	No	2126	270	2396	11.268781
1	Yes	152	118	270	43.703704
2	All	2278	388	2666	14.553638



In [93]:
```
sns.countplot(data=df_1, x=df_1['Voice mail plan'], hue='Churn');

crosstab_function(df_1,'Voice mail plan')
```

Out[93]:

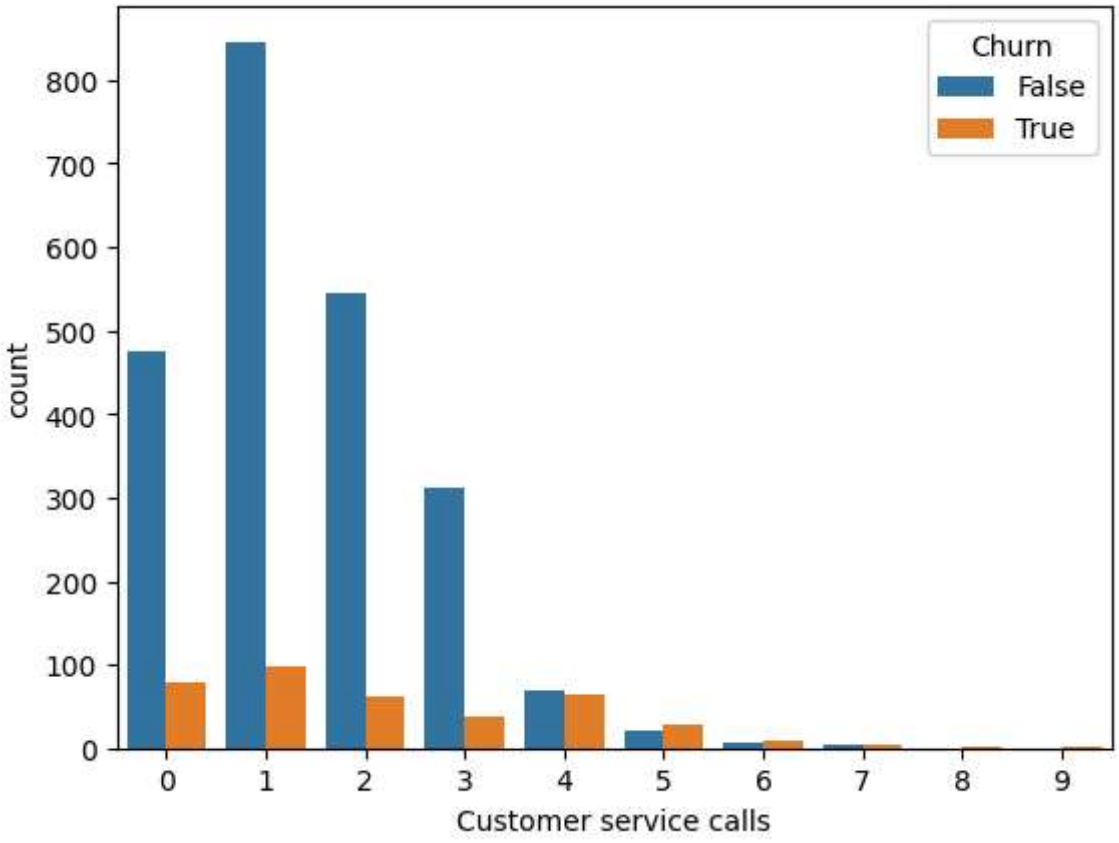
	Voice mail plan	Churn_NO	Churn_YES	Total	Churn Percentage
0	No	1610	323	1933	16.709778
1	Yes	668	65	733	8.867667
2	All	2278	388	2666	14.553638



In [94]:
```
sns.countplot(data=df_1, x=df_1['Customer service calls'], hue=df_1['Churn']);

crosstab_function(df_1,'Customer service calls')
```

Out[94]:

	Customer service calls	Churn_NO	Churn_YES	Total	Churn Percentage
0	0	476	79	555	14.234234
1	1	846	99	945	10.476190
2	2	546	62	608	10.197368
3	3	311	37	348	10.632184
4	4	69	64	133	48.120301
5	5	20	29	49	59.183673
6	6	7	10	17	58.823529
7	7	3	5	8	62.500000
8	8	0	1	1	100.000000
9	9	0	2	2	100.000000
10	All	2278	388	2666	14.553638



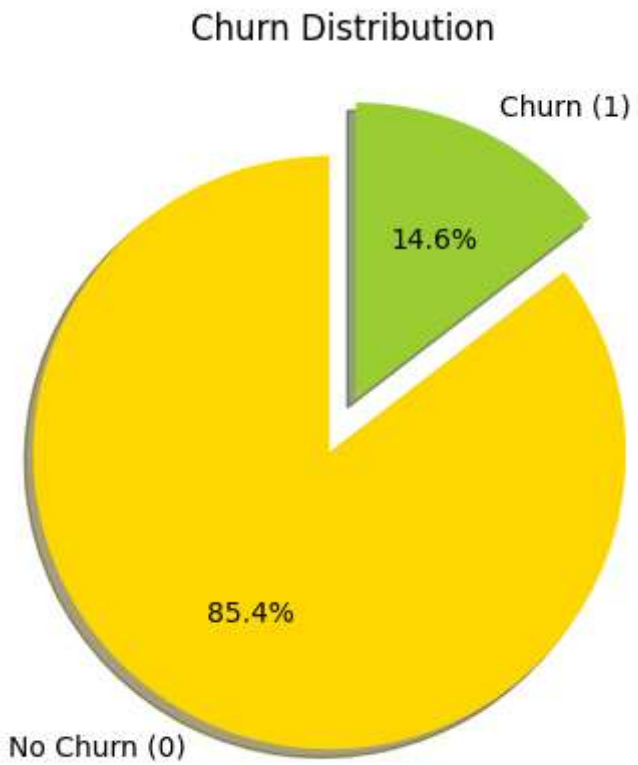
In [95]:
```
df_1[(df_1['Voice mail plan']=='No') & (df_1['International plan']=='Yes')].Churn.v
```

Out[95]:
```
False    0.54359
True     0.45641
Name: Churn, dtype: float64
```

In [96]:
```
df_1[(df_1['Customer service calls']>=4) & (df_1['Total day minutes']>=35)].Churn.v
```

Out[96]:
```
True     0.526316
False    0.473684
Name: Churn, dtype: float64
```

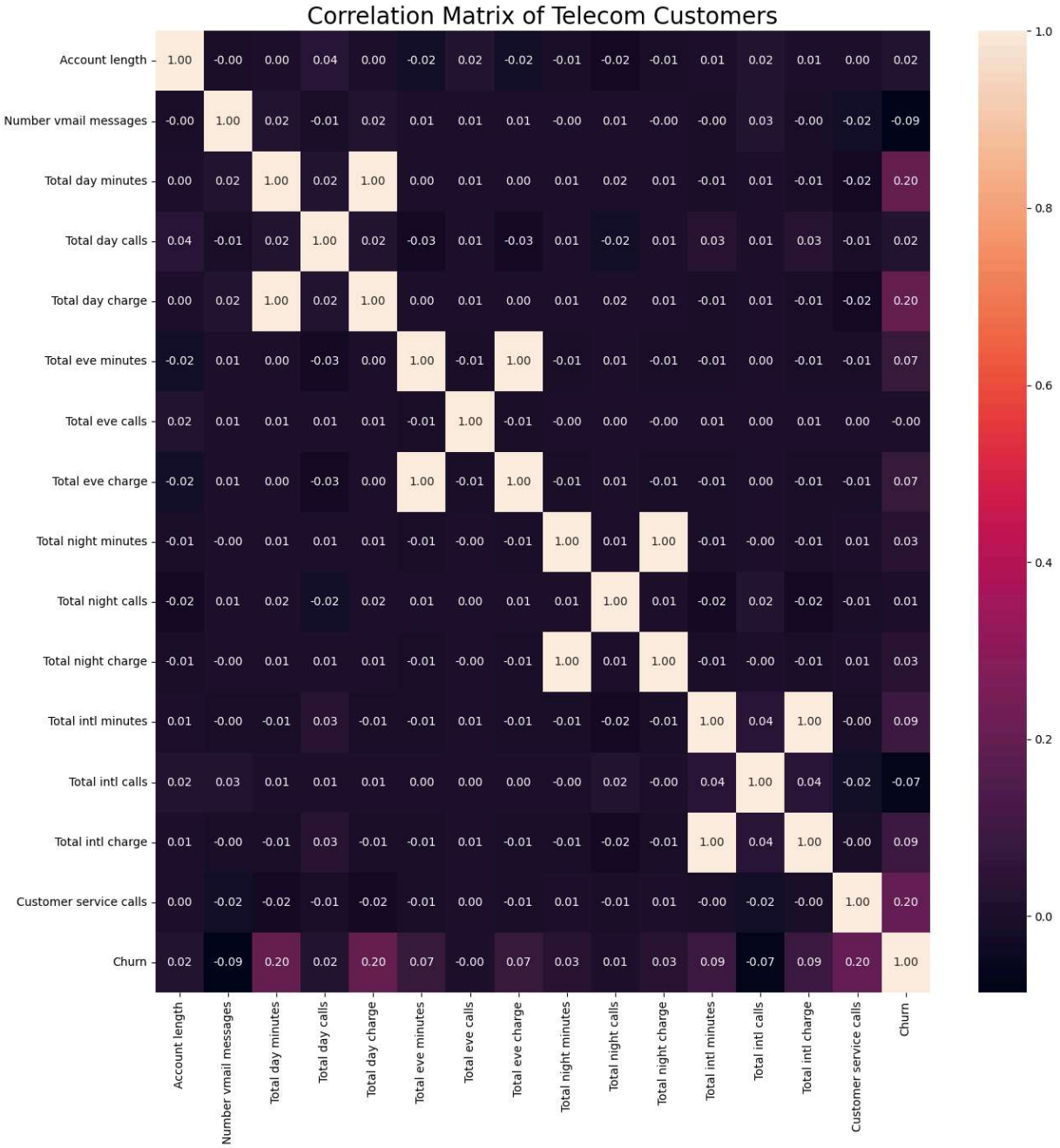
In [97]:
```
labels=['No Churn (0)', 'Churn (1)']
sizes = [df_1['Churn'].value_counts()[0], df_1['Churn'].value_counts()[1]]
colors = ['gold', 'yellowgreen']
explode = [0.1, 0.1]

plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', sh
plt.title('Churn Distribution')
plt.show()
```



In [101…

```
corr_matrix = df_1.corr()
plt.figure(figsize = (15, 15))
sns.heatmap(corr_matrix, annot = True, fmt = '0.2f')
plt.title("Correlation Matrix of Telecom Customers", fontsize = 20)
plt.show()
```

```
<ipython-input-101-25dce9ee55bb>:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future versi
on, it will default to False. Select only valid columns or specify the value of nume
ric_only to silence this warning.
```



In [102…
```
df_1.drop(['Area code', 'State'], axis=1, inplace=True)
```

In [103…
```
df_1.head()
```

Out[103...

	Account length	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Tota ev charg
0	128	No	Yes	25	265.1	110	45.07	197.4	99	16.7
1	107	No	Yes	26	161.6	123	27.47	195.5	103	16.6
2	137	No	No	0	243.4	114	41.38	121.2	110	10.3
3	84	Yes	No	0	299.4	71	50.90	61.9	88	5.2
4	75	Yes	No	0	166.7	113	28.34	148.3	122	12.6

In [104...
```
df_2.drop(['Area code', 'State'], axis=1, inplace=True)
```

In [105...
```
df_2.head()
```

Out[105...

	Account length	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Tota ev charg
0	117	No	No	0	184.5	97	31.37	351.6	80	29.8
1	65	No	No	0	129.1	137	21.95	228.5	83	19.4
2	161	No	No	0	332.9	67	56.59	317.8	97	27.0
3	111	No	No	0	110.4	103	18.77	137.3	102	11.6
4	49	No	No	0	119.3	117	20.28	215.1	109	18.2

In [106...
```
train = pd.get_dummies(df_1)
```

In [107...
```
train.head()
```

Out[107...

	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	ch
0	128	25	265.1	110	45.07	197.4	99	16.78	244.7	91	
1	107	26	161.6	123	27.47	195.5	103	16.62	254.4	103	
2	137	0	243.4	114	41.38	121.2	110	10.30	162.6	104	
3	84	0	299.4	71	50.90	61.9	88	5.26	196.9	89	
4	75	0	166.7	113	28.34	148.3	122	12.61	186.9	121	

In [108…
```
test = pd.get_dummies(df_2)
```

In [109…
```
test.head()
```

Out[109…

	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	ch
0	117	0	184.5	97	31.37	351.6	80	29.89	215.8	90	
1	65	0	129.1	137	21.95	228.5	83	19.42	208.8	111	
2	161	0	332.9	67	56.59	317.8	97	27.01	160.6	128	
3	111	0	110.4	103	18.77	137.3	102	11.67	189.6	105	
4	49	0	119.3	117	20.28	215.1	109	18.28	178.7	90	

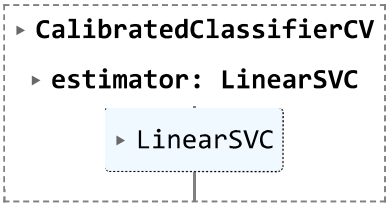
In [114…
```
X_train = train.drop('Churn', axis=1)
y_train = train['Churn']
```

In [115…
```
X_test = test.drop('Churn', axis=1)
y_test = test['Churn']
```

In [116…
```
svm = LinearSVC(max_iter = 20000)
svm = CalibratedClassifierCV(svm)
svm.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarnin
g:

Liblinear failed to converge, increase the number of iterations.

/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarnin
g:

Liblinear failed to converge, increase the number of iterations.

/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarnin
g:

Liblinear failed to converge, increase the number of iterations.

/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarnin
g:

Liblinear failed to converge, increase the number of iterations.

/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244: ConvergenceWarnin
g:

Liblinear failed to converge, increase the number of iterations.
```

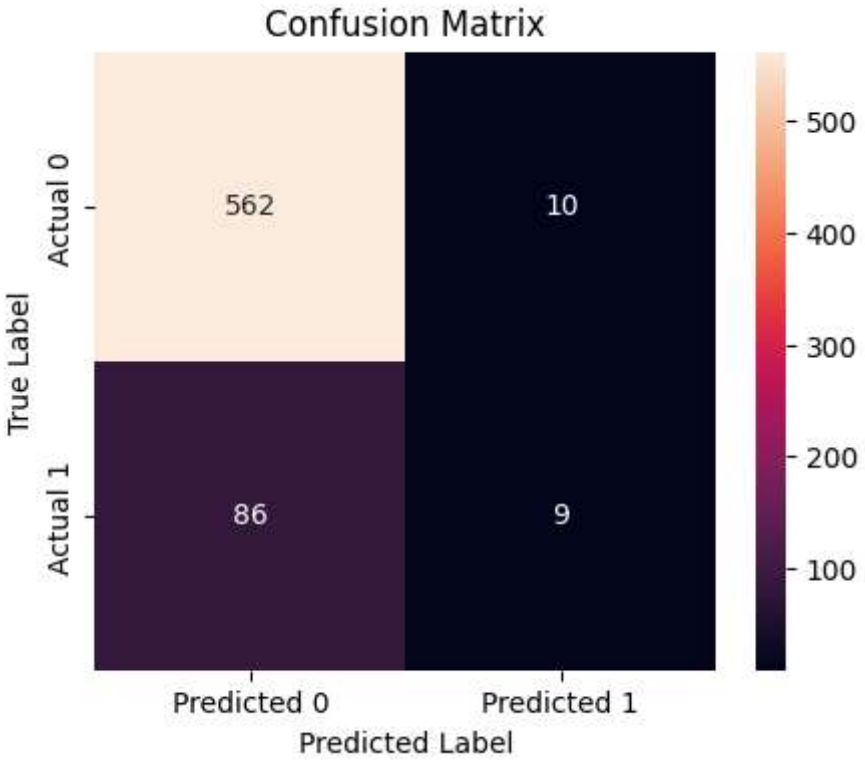
Out[116…



In [117…
```
y_predict = svm.predict(X_test)
```

In [118…
```
y_predict = svm.predict(X_test)
```

In [119…
```
cm = confusion_matrix(y_test, y_predict)
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d',
            xticklabels=['Predicted 0', 'Predicted 1'],
            yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
print("Confusion Matrix:")
print(cm)
```




```
Confusion Matrix:
[[562  10]
 [ 86   9]]
```

In [120…
```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

Out[120…

RandomForestClassifier

RandomForestClassifier()

In [121…

```
y_predict = rf.predict(X_test)
```

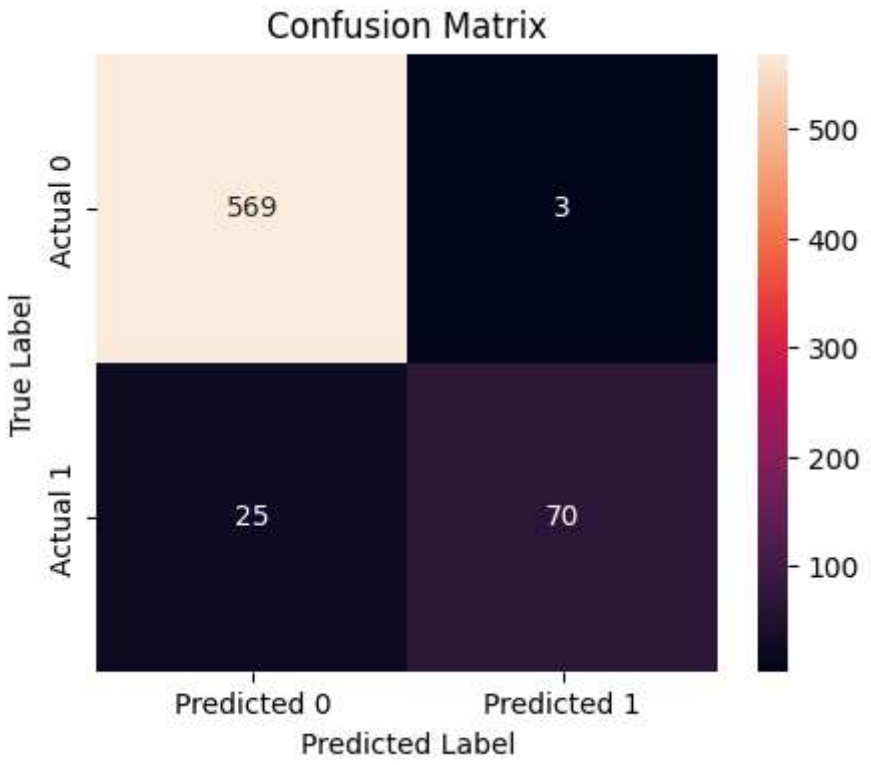
In [122…

```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

       False       0.96      0.99      0.98       572
        True       0.96      0.74      0.83        95

    accuracy                           0.96       667
   macro avg       0.96      0.87      0.90       667
weighted avg       0.96      0.96      0.96       667
```

In [123…

```
cm = confusion_matrix(y_test, y_predict)
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d',
            xticklabels=['Predicted 0', 'Predicted 1'],
            yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
print("Confusion Matrix:")
print(cm)
```



```
Confusion Matrix:
[[569   3]
 [ 25  70]]
```

In [124…
```
nb = GaussianNB()
nb.fit(X_train, y_train)
```

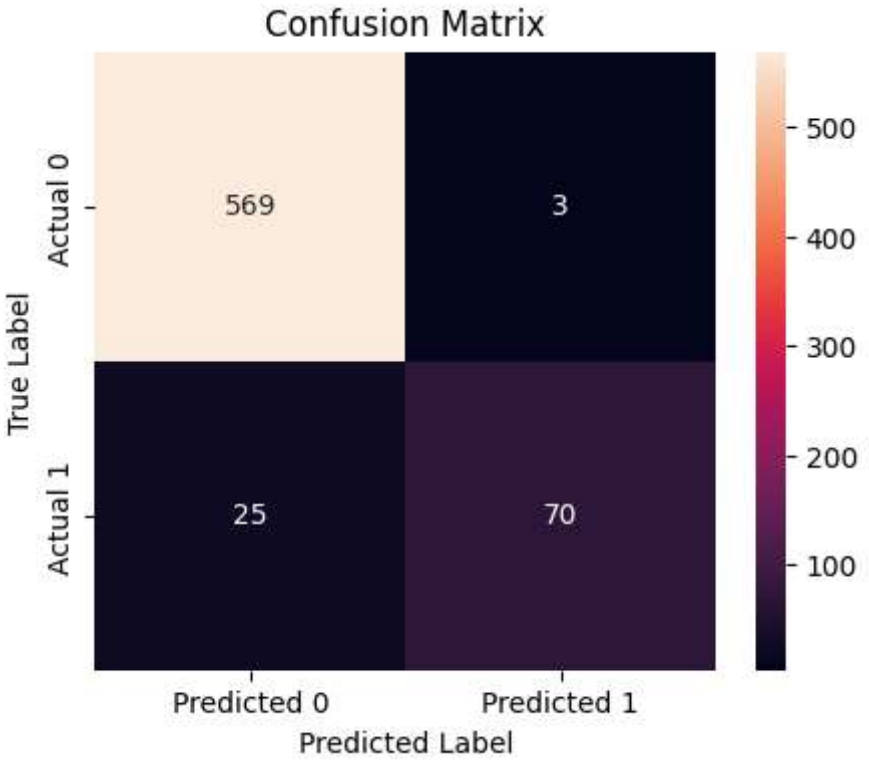
Out[124…
```
▾ GaussianNB

GaussianNB()
```

In [125…
```
y_predcit = nb.predict(X_test)
```

In [126…
```
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

       False       0.96      0.99      0.98       572
        True       0.96      0.74      0.83        95

    accuracy                           0.96       667
   macro avg       0.96      0.87      0.90       667
weighted avg       0.96      0.96      0.96       667
```

In [127…
```
cm = confusion_matrix(y_test, y_predict)
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d',
            xticklabels=['Predicted 0', 'Predicted 1'],
            yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
print("Confusion Matrix:")
print(cm)
```



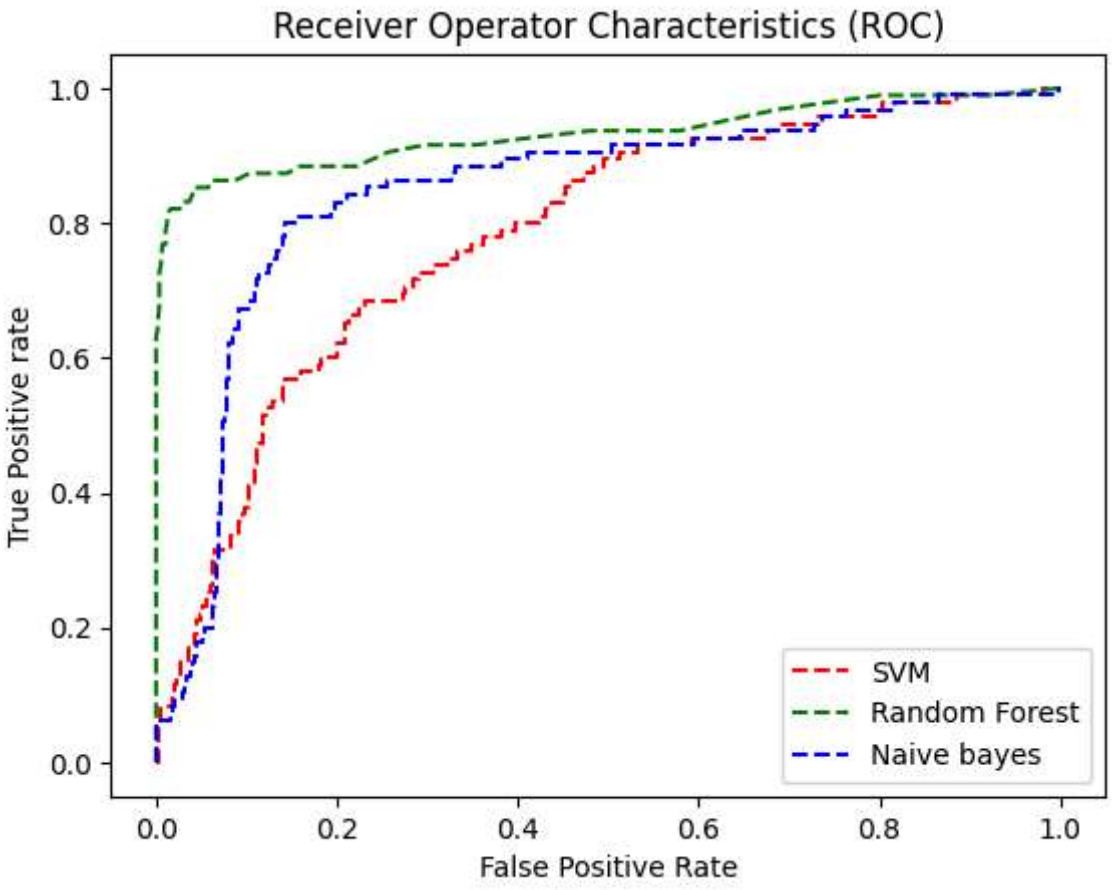

```
Confusion Matrix:
[[569   3]
 [ 25  70]]
```

In [128…
```
auc_score1 = roc_auc_score(y_test, svm.predict_proba(X_test)[:, 1])
auc_score2 = roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1])
auc_score4 = roc_auc_score(y_test, nb.predict_proba(X_test)[:, 1])

print("Support Vector Machine: ", auc_score1)
print("Random Forest: ", auc_score2)
print("Naive Bayes: ", auc_score4)
```

```
Support Vector Machine:  0.7815053367684947
Random Forest:  0.9320758189179241
Naive Bayes:  0.8459514170040486
```

In [129…
```
fpr1, tpr1, thresh1 = roc_curve(y_test, svm.predict_proba(X_test)[:, 1], pos_label
fpr2, tpr2, thresh2 = roc_curve(y_test, rf.predict_proba(X_test)[:, 1], pos_label =
fpr4, tpr4, thresh4 = roc_curve(y_test, nb.predict_proba(X_test)[:, 1], pos_label =
```

In [130…
```
plt.plot(fpr1, tpr1, linestyle = "--", color = "red", label = "SVM")
plt.plot(fpr2, tpr2, linestyle = "--", color = "green", label = "Random Forest")
plt.plot(fpr4, tpr4, linestyle = "--", color = "blue", label = "Naive bayes")

plt.title('Receiver Operator Characteristics (ROC)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')

plt.legend(loc = 'best')
plt.savefig('ROC', dpi = 300)
plt.show()
```



Random Forest Model obtain the highest AUC and ROC score.

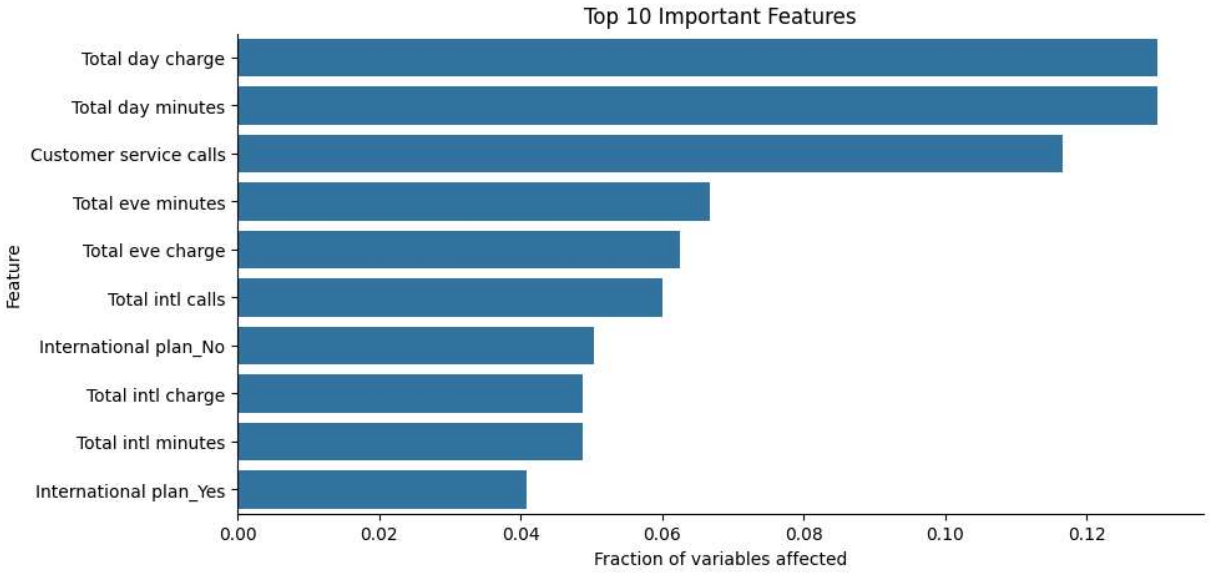
In [131…
```
y_predict = rf.predict(X_test)
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

       False       0.96      0.99      0.98       572
        True       0.96      0.74      0.83        95

    accuracy                           0.96       667
   macro avg       0.96      0.87      0.90       667
weighted avg       0.96      0.96      0.96       667
```

Plot important features

In [133…
```
X = train.drop('Churn', axis=1)

feat_scores = pd.DataFrame({"Fraction of variables affected": rf.feature_importance
feat_scores = feat_scores.sort_values(by="Fraction of variables affected", ascendin

plt.figure(figsize=(10, 5))
sns.barplot(x=feat_scores["Fraction of variables affected"], y=feat_scores.index)
plt.xlabel("Fraction of variables affected")
plt.ylabel("Feature")
plt.title("Top 10 Important Features")
```

```
sns.despine()
plt.show()
```



In [ ]: