In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
churn = pd.read_csv('Churn.csv')
```

In [3]:

```
churn.shape
```

Out[3]:

```
(3333, 21)
```

In [4]:

```
churn.head()
```

Out[4]:

	Account Length	VMail Message	Day Mins	Eve Mins	Night Mins	Intl Mins	CustServ Calls	Churn	Intl Plan	VMail Plan	...	Day Charge	E Ca
0	128	25	265.1	197.4	244.7	10.0	1	0	0	1	...	45.07	
1	107	26	161.6	195.5	254.4	13.7	1	0	0	1	...	27.47	
2	137	0	243.4	121.2	162.6	12.2	0	0	0	0	...	41.38	
3	84	0	299.4	61.9	196.9	6.6	2	0	1	0	...	50.90	
4	75	0	166.7	148.3	186.9	10.1	3	0	1	0	...	28.34	

5 rows × 21 columns

In [5]:

```
churn['Churn'].value_counts(normalize=True)
```

Out[5]:

```
0    0.855086
1    0.144914
Name: Churn, dtype: float64
```

In [6]:

```
churn.columns
```

Out[6]:

```
Index(['Account Length', 'VMail Message', 'Day Mins', 'Eve Mins', 'Nig
ht Mins',
       'Intl Mins', 'CustServ Calls', 'Churn', 'Intl Plan', 'VMail Pla
n',
       'Day Calls', 'Day Charge', 'Eve Calls', 'Eve Charge', 'Night Ca
lls',
       'Night Charge', 'Intl Calls', 'Intl Charge', 'State', 'Area Cod
e',
       'Phone'],
      dtype='object')
```

In [7]:

```
## drop some columns
churn = churn.drop(columns = ['State', 'Area Code','Phone'], axis=1)
```

In [8]:

```
churn.shape
```

Out[8]:

```
(3333, 18)
```

In [9]:

```
target = churn['Churn']
```

In [10]:

```
churn = churn.drop(columns=['Churn'], axis=1)
```

In [14]:

```
churn.std(axis=0)
```

Out[14]:

```
Account Length    39.822106
VMail Message     13.688365
Day Mins          54.467389
Eve Mins          50.713844
Night Mins        50.573847
Intl Mins          2.791840
CustServ Calls     1.315491
Intl Plan          0.295879
VMail Plan         0.447398
Day Calls         20.069084
Day Charge         9.259435
Eve Calls         19.922625
Eve Charge         4.310668
Night Calls       19.568609
Night Charge       2.275873
Intl Calls         2.461214
Intl Charge        0.753773
dtype: float64
```

In [15]:

```
mu = churn.mean(axis=0)
std = churn.std(axis=0)
```

In [16]:

```
churn = (churn-mu)/std
```

In [ ]:

```

```

train test split

In [18]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [19]:

```
X_train, X_test, y_train, y_test = train_test_split(churn, target, test_size=0.2, ra
```

In [20]:

```
logit = LogisticRegression()
```

In [21]:

```
logit.fit(X_train, y_train)
```

Out[21]:

```
LogisticRegression()
```

In [22]:

```
pred = logit.predict(X_test)
pred[:5]
```

Out[22]:

```
array([1, 0, 0, 0, 0])
```

Evaluation Metrics

In [23]:

```
from sklearn.metrics import accuracy_score
```

In [24]:

```
print("Accuracy:",accuracy_score(y_test, pred))
```

```
Accuracy: 0.8695652173913043
```

Introduce Confusion Matrix

In [25]:

```
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_scor
```

In [26]:

```
confusion_matrix(y_test, pred)
```

Out[26]:

```
array([[557,  15],
       [ 72,  23]])
```

In [27]:

```
print("Accuracy :", accuracy_score(y_test, pred))
print("Precision :", precision_score(y_test, pred))
print("Recall :", recall_score(y_test, pred))
print("F1-Score :", f1_score(y_test, pred))
```

```
Accuracy : 0.8695652173913043
Precision : 0.6052631578947368
Recall : 0.24210526315789474
F1-Score : 0.3458646616541354
```

In [28]:

```
23/(23+15)
```

Out[28]:

```
0.6052631578947368
```

In [30]:

```
23/(23+72)
```

Out[30]:

```
0.24210526315789474
```

In [ ]:

```

```

getting the raw probabilities from the model

In [31]:

```
y_pred = logit.predict_proba(X_test)
y_pred[:5]
```

Out[31]:

```
array([[0.38525546, 0.61474454],
       [0.86413771, 0.13586229],
       [0.9764602 , 0.0235398 ],
       [0.91352978, 0.08647022],
       [0.84980336, 0.15019664]])
```

In [32]:

```
# putting a threshold of 0.3
predc = np.where(y_pred[:,1] >=0.3, 1, 0)
predc[:5]
```

Out[32]:

```
array([1, 0, 0, 0, 0])
```

In [33]:

```
confusion_matrix(y_test, predc)
```

Out[33]:

```
array([[533,  39],
       [ 39,  56]])
```

In [34]:

```
print("Accuracy :", accuracy_score(y_test, predc))
print("Precision :", precision_score(y_test, predc))
print("Recall  :", recall_score(y_test, predc))
print("F1-Score :", f1_score(y_test, predc))
```

```
Accuracy : 0.8830584707646177
Precision : 0.5894736842105263
Recall : 0.5894736842105263
F1-Score : 0.5894736842105263
```

In [ ]:

```

```

ROC AUC

In [35]:

```
from sklearn.metrics import roc_curve, auc
```

In [36]:

```
fpr, tpr, thresholds =  roc_curve(y_test, y_pred[:, 1], pos_label=1)
```

In [37]:

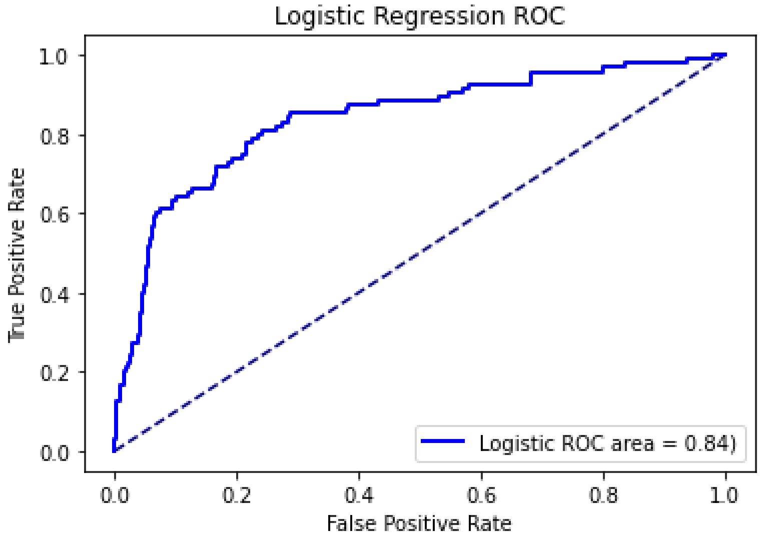
```
roc_auc = auc(fpr, tpr)
roc_auc
```

Out[37]:

```
0.8383327199116672
```

In [38]:

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.title('Logistic Regression ROC')
plt.plot(fpr, tpr, color='blue', lw=2, label='Logistic ROC area = %0.2f)' % roc_auc)
plt.legend(loc="lower right")
plt.show()
```



In [ ]:

```

```

Feature Selection

In [39]:

```
cols = ['Day Mins', 'Eve Mins', 'CustServ Calls', 'Intl Plan','VMail Message']
```

In [41]:

```
X_tr = X_train[cols]
X_te = X_test[cols]
```

In [42]:

```
logmodel = LogisticRegression()
logmodel.fit(X_tr, y_train)
```

Out[42]:

```
LogisticRegression()
```

In [44]:

```
prob = logmodel.predict_proba(X_te)
```

In [48]:

```
pred = np.where(prob[:, 1]>=0.3,1,0)
```

In [50]:

```
confusion_matrix(y_test, pred)
```

Out[50]:

```
array([[527,  45],
       [ 48,  47]])
```

In [51]:

```
print("Accuracy :", accuracy_score(y_test, pred))
print("Precision :", precision_score(y_test, pred))
print("Recall :", recall_score(y_test, pred))
print("F1-Score :", f1_score(y_test, pred))
```

```
Accuracy : 0.8605697151424287
Precision : 0.5108695652173914
Recall : 0.49473684210526314
F1-Score : 0.5026737967914439
```

In [ ]:

In [ ]:

In [55]:

```
lr_fpr, lr_tpr, _ =  roc_curve(y_test, prob[:, 1], pos_label=1)
lr_roc_auc = auc(lr_fpr, lr_tpr)
lr_roc_auc
```

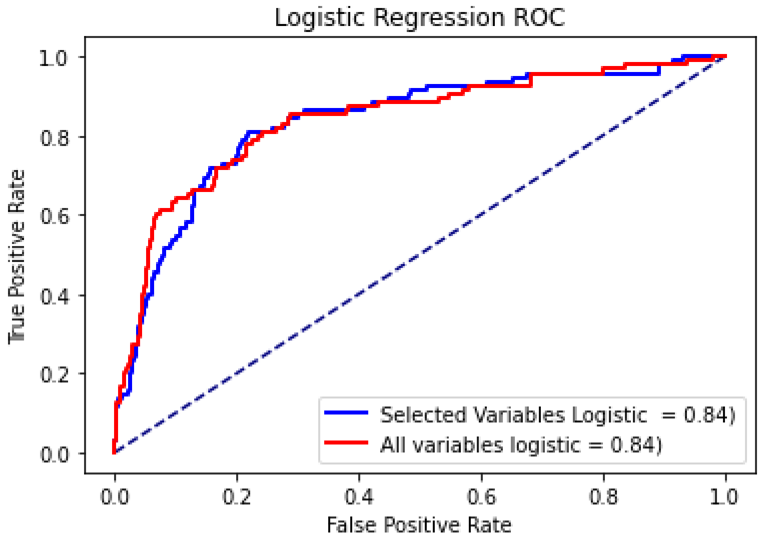
Out[55]:

```
0.8353146853146853
```

In [56]:

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')

plt.title('Logistic Regression ROC')
plt.plot(lr_fpr, lr_tpr, color='blue', lw=2, label='Selected Variables Logistic  = %
plt.plot(fpr, tpr, color='red', lw=2, label='All variables logistic = %0.2f)' % roc_
plt.legend(loc="lower right")
plt.show()
```



In [ ]:

In [ ]: