The objective of this study is to build a logistic regression model to predict the customer churn based on a set of 9 predictors. Refer below for code loading the data and establishing a random split into 70% training and 30% testing sample.

```
data = pd.read_csv("C:\\Users\\tgmce\\Downloads\\Bank Customer Churn Prediction_revised.csv")
x = data.iloc[:, :-1]
y = data.iloc[:, -1:]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30,random_state = 0)
```

After we import and split the data, we build a logistic regression model on the training data with all predictors.

```
model=sm.Logit(y_train, sm.add_constant(x_train))
lr = model.fit()
print(lr.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.435680
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                  churn   No. Observations:                 7000
Model:                          Logit   Df Residuals:                     6990
Method:                           MLE   Df Model:                            9
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                  0.1348
Time:                        02:03:48   Log-Likelihood:                -3049.8
converged:                       True   LL-Null:                       -3524.9
Covariance Type:            nonrobust   LLR p-value:                 9.177e-199
====================================================================================
                       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------
const               -3.1303      0.288    -10.869      0.000      -3.695      -2.566
credit_score        -0.0009      0.000     -2.711      0.007      -0.002      -0.000
gender              -0.5389      0.065     -8.347      0.000      -0.665      -0.412
age                  0.0715      0.003     23.475      0.000       0.066       0.077
tenure              -0.0283      0.011     -2.551      0.011      -0.050      -0.007
balance           5.012e-06   5.49e-07      9.130      0.000    3.94e-06    6.09e-06
products_number     -0.1165      0.057     -2.053      0.040      -0.228      -0.005
credit_card         -0.0186      0.070     -0.265      0.791      -0.156       0.119
active_member       -1.0000      0.068    -14.736      0.000      -1.133      -0.867
estimated_salary  6.968e-07    5.6e-07      1.244      0.214   -4.01e-07     1.8e-06
====================================================================================
```


```
model=sm.Logit(y_train, sm.add_constant(x_train))
lr = model.fit()
print(lr.summary())
```


As shown from the summary table above, the fitted linear regression function is:

- log(P(churn=1) / 1−P(churn=1)) = −3.13 − 0.0009credit_score − 0.539gender + 0.072age − 0.028tenure + 5.012 ∗ 10^−6 balance − 0.117products_number − 0.019credit_card − active_member + 6.97 ∗ 10^−7 estimated_salary

Two predictors are not significant given 0.05 significance level (See: credit_card and estimated_salary).

The coefficient of credit_score can be interpreted as for every one unit increase of customer score, the odds of churn will decrease by 0.09% (=e^−0.0009 − 1). The coefficient of active_member can be interpreted as the odds of churn for active member is 63.21% (=e^−1 − 1) lower than that for non-active member.

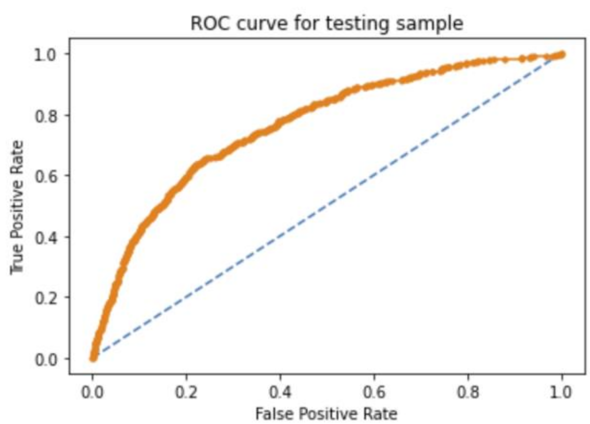
We can use the logistic regression model trained above to make prediction on both training and testing sample.

```
p_pred_train = lr.predict(sm.add_constant(x_train))
p_pred_test = lr.predict(sm.add_constant(x_test))
```

The ROC curve for the testing sample can be found below. AUC is 0.76 for the testing sample.

```
mr_probs = [0 for _ in range(len(y_test))]
mr_fpr, mr_tpr, _ = roc_curve(y_test, mr_probs)
lr_fpr, lr_tpr, _ = roc_curve(y_test, p_pred_test)
plt.plot(mr_fpr, mr_tpr, linestyle='--')
plt.plot(lr_fpr, lr_tpr, marker='.')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve for testing sample')
print('AUC for testing sample is:', metrics.auc(lr_fpr, lr_tpr))
```

```
AUC for testing sample is: 0.7648574246340937
```



The confusion matrix on the testing sample can be found below using 0.5 as the cutoff probability.

```python
from sklearn.metrics import confusion_matrix
y_pred = round(p_pred_test)
print('Confusion matrix is:', confusion_matrix(y_test, y_pred))
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
print('MR:', (fp+fn)/(tn+fp+fn+tp))
print('TPR:', tp/(tp+fn))
print('FNR:', fn/(tp+fn))
```

```
Confusion matrix is: [[2305   74]
 [ 517  104]]
MR: 0.197
TPR: 0.16747181964573268
FNR: 0.8325281803542673
```

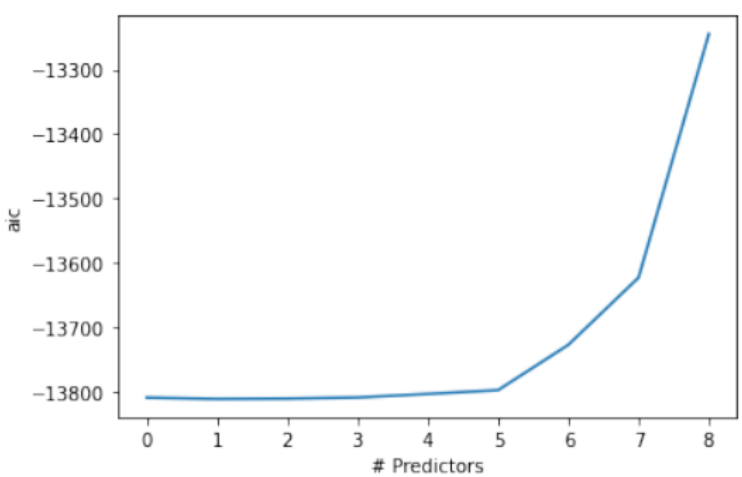
The misclassification rate is 19.7%, true positive rate is 16.75%, and false negative rate is 83.25%.

For feature selection I elected to use backward stepwise selection with the objective to minimize the value of AIC.

```python
def fit_lr(feature_set):
    model = sm.Logit(y_train, sm.add_constant(x_train[list(feature_set)]))
    lr = model.fit()
    MSE = mean_squared_error(lr.predict(sm.add_constant(x_train[list(feature_set)])),y_train)
    AIC = len(x_train)*np.log(MSE)+2*len(lr.params)
    return {"model":lr, "MSE":MSE, "AIC":AIC}

def getbest(k):
    result = []
    for combo in itertools.combinations(x_train.columns, k):
        result.append(fit_lr(combo))
    models = pd.DataFrame(result)
    best_model = models.loc[models['MSE'].argmin()]
    return best_model

picked = list(x_train.columns)
models_best = pd.DataFrame(columns=["model","MSE","AIC"])
models_best.loc[0] = fit_lr(picked)
for i in range(1,9):
    best_MSE = np.inf
    for combo in itertools.combinations(picked,len(picked)-1):
        MSE = fit_lr(list(combo))["MSE"]
        if MSE < best_MSE:
            best_MSE = MSE
            best_feature = combo
    picked = best_feature
    models_best.loc[i] = fit_lr(list(picked))
print(models_best)
plt.plot(models_best["AIC"])
plt.xlabel('# Predictors')
plt.ylabel('aic')
```



As shown, the model with 1 predictors being removed (8 predictors included) is associated with the lowest AIC. All predictors but credit_card are included in the final model.

```
print(models_best.loc[1, "model"].summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                  churn   No. Observations:                 7000
Model:                          Logit   Df Residuals:                     6991
Method:                           MLE   Df Model:                            8
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                  0.1348
Time:                        02:05:54   Log-Likelihood:                -3049.8
converged:                       True   LL-Null:                       -3524.9
Covariance Type:            nonrobust   LLR p-value:                 8.441e-200
====================================================================================
                       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------
const               -3.1437      0.284    -11.087      0.000      -3.699      -2.588
credit_score        -0.0009      0.000     -2.712      0.007      -0.002      -0.000
gender              -0.5390      0.065     -8.350      0.000      -0.666      -0.413
age                  0.0715      0.003     23.481      0.000       0.066       0.078
tenure              -0.0284      0.011     -2.558      0.011      -0.050      -0.007
balance           5.014e-06   5.49e-07      9.135      0.000    3.94e-06    6.09e-06
products_number     -0.1165      0.057     -2.052      0.040      -0.228      -0.005
active_member       -0.9996      0.068    -14.734      0.000      -1.133      -0.867
estimated_salary   6.97e-07    5.6e-07      1.244      0.214   -4.01e-07     1.8e-06
====================================================================================
```

We used the re-trained logistic regression to make prediction on the testing sample.

```
lr_backward = models_best.loc[1, "model"]
p_pred_test = lr_backward.predict(sm.add_constant(x_test[['credit_score','gender','age','tenure','balance','products_number',
mr_probs = [0 for _ in range(len(y_test))]
mr_fpr, mr_tpr, _ = roc_curve(y_test, mr_probs)
lr_fpr, lr_tpr, _ = roc_curve(y_test, p_pred_test)
print('AUC for testing sample is:', metrics.auc(lr_fpr, lr_tpr))
```

```
AUC for testing sample is: 0.7648574246340937
```

The AUC for the testing sample is 0.7649. My model has a 76.49% chance of ranking a randomly chosen positive case (e.g., customer did churn) higher than a randomly chosen negative case (e.g., customer did not churn).