Logistic Regression Example

```
import pandas as pd
import numpy as np
import scipy.optimize as opt
from sklearn import preprocessing
import matplotlib.pyplot as plt
from matplotlib import rc,font_manager

ticks_font = font_manager.FontProperties(family='Times New Roman', style='normal',
    size=12, weight='normal', stretch='normal')
ax=plt.gca()

## Loading Data ##
df=pd.read_csv('D:\Python\edx\Machine Learning\Classification\ChurnData.csv')
with open('Log_Reg.txt','a') as f:
    print(df.head(),file=f)

## Preprocessing and selection ##
df=df[['tenure', 'age', 'address', 'income', 'ed', 'employ', 'equip',   'callcard', 'wir
df['churn']=df['churn'].astype('int')
with open('Log_Reg.txt','a') as f:
    print(df.head(),file=f)
    print(df.shape,file=f)
for col in df.columns:
    with open('Log_Reg.txt','a') as f:
        print(col,file=f)

## Define X,y dataset ##
X=np.asarray(df[['tenure', 'age', 'address', 'income', 'ed', 'employ', 'equip']])
y=np.asarray(df['churn'])
with open('Log_Reg.txt','a') as f:
    print(X[0:5],file=f)
    print(y[0:5],file=f)

## Normalize dataset ##
X=preprocessing.StandardScaler().fit(X).transform(X)
with open('Log_Reg.txt','a') as f:
    print(X[0:5],file=f)

## Train_Test_Split ##
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_state=4)
with open('Log_Reg.txt','a') as f:
```

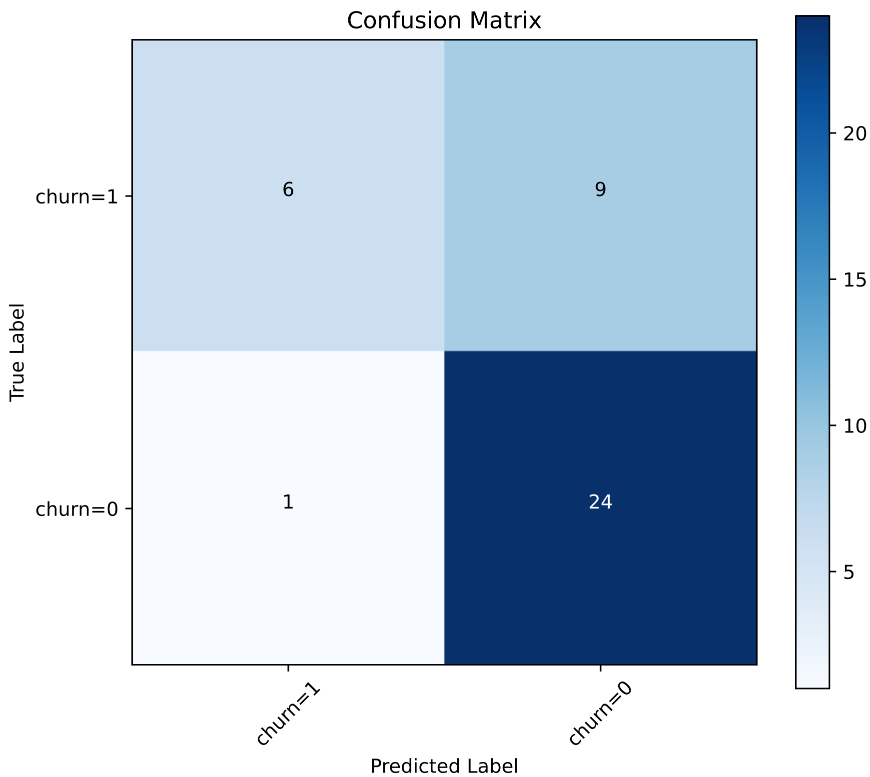
```
    print('Train set: ', X_train.shape,y_train.shape,file=f)
    print('Test set: ', X_test.shape,y_test.shape,file=f)

## Modeling using scikit-learn ##
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LogReg=LogisticRegression(C=0.01,solver='liblinear').fit(X_train,y_train) # C-inverse of

yhat=LogReg.predict(X_test)
yhat_prob=LogReg.predict_proba(X_test) # predict_proba returns estimates for all classes

### Evaluation ##

from sklearn.metrics import jaccard_score
from sklearn.metrics import accuracy_score
with open('Log_Reg.txt','a') as f:
    print('J Score: ', jaccard_score(y_test,yhat,labels=None, average='binary', sample_we
    print('Accuracy Score: ', accuracy_score(y_test,yhat),file=f)

#Note: In current version of scikit-learn 0.23.1 jaccard_similarity_score is replaced by
#differs by definition as jaccard_similarity_score is just same as accuracy_score but by
#jaccard index accuracy score and jaccard score are different. SO, if using scikit-learn

#Using Confusion matrix #

from sklearn.metrics import classification_report,confusion_matrix
import itertools

def plot_cmat(cm,classes,normalize=False,
                title='Confusion Matrix',cmap=plt.cm.Blues):
                if normalize:
                    cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
                    print('Normalized Confusion Matrix')
                else:
                    print('COnfusion matrix without Normalization')

                with open('Log_Reg.txt','a') as f:
                    print(cm,file=f)

                plt.imshow(cm,interpolation='nearest',cmap=cmap)
                plt.title(title)
                plt.colorbar()
                tick_marks=np.arange(len(classes))
                plt.xticks(tick_marks,classes,rotation=45)
                plt.yticks(tick_marks,classes)
```

```
                #For Labeling inside boxes #
                fmt='.2f' if normalize else 'd'
                threshold=cm.max()/2
                for i,j in itertools.product(range(cm.shape[0]),range(cm.shape[1])):
                    plt.text(j,i,format(cm[i,j],fmt),
                            horizontalalignment='center',
                            color='white' if cm[i,j] > threshold else 'black')
                plt.tight_layout
                plt.ylabel('True Label')
                plt.xlabel('Predicted Label')

#Confusion matrix #
c_mat=confusion_matrix(y_test,yhat,labels=[1,0])
with open('Log_Reg.txt','a') as f:
    print('Confusion matrix: \n ',c_mat,file=f)

#Confusion matrix plot#
plt.figure()
plot_cmat(c_mat,classes=['churn=1','churn=0'],normalize=False,title='Confusion Matrix')
plt.show()

# Compute classification report - Precision, Recall, F1Score and Support

with open('Log_Reg.txt','a') as f:
    print('Classification Report: \n ',classification_report(y_test,yhat),file=f)
```

Solution:

```
   tenure   age  address  income   ed  employ  equip  callcard  wireless  longmon
tollmon  equipmon  cardmon  ...  tollten  cardten  voice  pager  internet  callwait
confer  ebill  loglong  logtoll  lninc  custcat  churn
0    11.0  33.0      7.0   136.0  5.0     5.0    0.0       1.0       1.0     4.40
20.75       0.0    15.25  ...   211.45    125.0    1.0    1.0       0.0       1.0
1.0    0.0    1.482    3.033  4.913      4.0    1.0
1    33.0  33.0     12.0    33.0  2.0     0.0    0.0       0.0       0.0     9.45
0.00       0.0     0.00  ...     0.00      0.0    0.0    0.0       0.0       0.0
0.0    0.0    2.246    3.240  3.497      1.0    1.0
2    23.0  30.0      9.0    30.0  1.0     2.0    0.0       0.0       0.0     6.30
0.00       0.0     0.00  ...     0.00      0.0    0.0    0.0       0.0       0.0
1.0    0.0    1.841    3.240  3.401      3.0    0.0
3    38.0  35.0      5.0    76.0  2.0    10.0    1.0       1.0       1.0     6.05
45.00      50.1    23.25  ...  1873.05    880.0    1.0    1.0       1.0       1.0
1.0    1.0    1.800    3.807  4.331      4.0    0.0
4     7.0  35.0     14.0    80.0  2.0    15.0    0.0       1.0       0.0     7.10
22.00       0.0    23.75  ...   166.10    145.0    1.0    0.0       0.0       1.0
```

```
1.0    0.0    1.960    3.091  4.382       3.0    0.0

[5 rows x 28 columns]
   tenure   age  address  income   ed  employ  equip  callcard  wireless  churn
0    11.0  33.0      7.0   136.0  5.0     5.0    0.0       1.0       1.0      1
1    33.0  33.0     12.0    33.0  2.0     0.0    0.0       0.0       0.0      1
2    23.0  30.0      9.0    30.0  1.0     2.0    0.0       0.0       0.0      0
3    38.0  35.0      5.0    76.0  2.0    10.0    1.0       1.0       1.0      0
4     7.0  35.0     14.0    80.0  2.0    15.0    0.0       1.0       0.0      0
(200, 10)
tenure
age
address
income
ed
employ
equip
callcard
wireless
churn
[[ 11.  33.   7. 136.   5.   5.   0.]
 [ 33.  33.  12.  33.   2.   0.   0.]
 [ 23.  30.   9.  30.   1.   2.   0.]
 [ 38.  35.   5.  76.   2.  10.   1.]
 [  7.  35.  14.  80.   2.  15.   0.]]
[1 1 0 0 0]
[[-1.13518441 -0.62595491 -0.4588971   0.4751423   1.6961288  -0.58477841
  -0.85972695]
 [-0.11604313 -0.62595491  0.03454064 -0.32886061 -0.6433592  -1.14437497
  -0.85972695]
 [-0.57928917 -0.85594447 -0.261522   -0.35227817 -1.42318853 -0.92053635
  -0.85972695]
 [ 0.11557989 -0.47262854 -0.65627219  0.00679109 -0.6433592  -0.02518185
   1.16316   ]
 [-1.32048283 -0.47262854  0.23191574  0.03801451 -0.6433592   0.53441472
  -0.85972695]]
Train set:  (160, 7) (160,)
Test set:  (40, 7) (40,)
J Score:  0.375
Accuracy Score:  0.75
Confusion matrix:
  [[ 6  9]
 [ 1 24]]
[[ 6  9]
 [ 1 24]]
Classification Report:
```



	precision	recall	f1-score	support
0	0.73	0.96	0.83	25
1	0.86	0.40	0.55	15
accuracy			0.75	40
macro avg	0.79	0.68	0.69	40
weighted avg	0.78	0.75	0.72	40