Universidade Federal de Uberlândia

PGC308A Tópicos Especiais em Sistemas de Computação 2: Internet do Futuro

Rodrigo Moreira – moreira_r@outlook.com

2017-2

1 Task III: Estimating SLA Conformance and Violation from Device Statistics

1. Model Training - provide the coefficients of your model C:

   Coefficients of the model C:
   $\Theta_1 =$ **-5.67750597e-02** $\Theta_2 =$ **-4.26722323e-02**
   $\Theta_3 =$ **6.32365126e-03** $\Theta_4 =$ **-4.57938573e-07**
   $\Theta_5 =$ **3.52435923e-03** $\Theta_6 =$ **3.03572012e-04**
   $\Theta_7 =$ **-8.26688033e-02** $\Theta_8 =$ **-7.17041206e-02**
   $\Theta_9 =$ **-7.46929691e-06**

2. Accuracy of the Classifier C:

   Classification Error (ERR) - Logistic Regression-based ≈ 0.114, i.e. $\approx 11,4\%$.

3. Accuracy of the Classifier C (by considering the Naïve method):

   Classification Error (ERR) - Naive-based ≈ 0.495, i.e. $\approx 49.5\%$.

4. New classifier extending the linear regression:

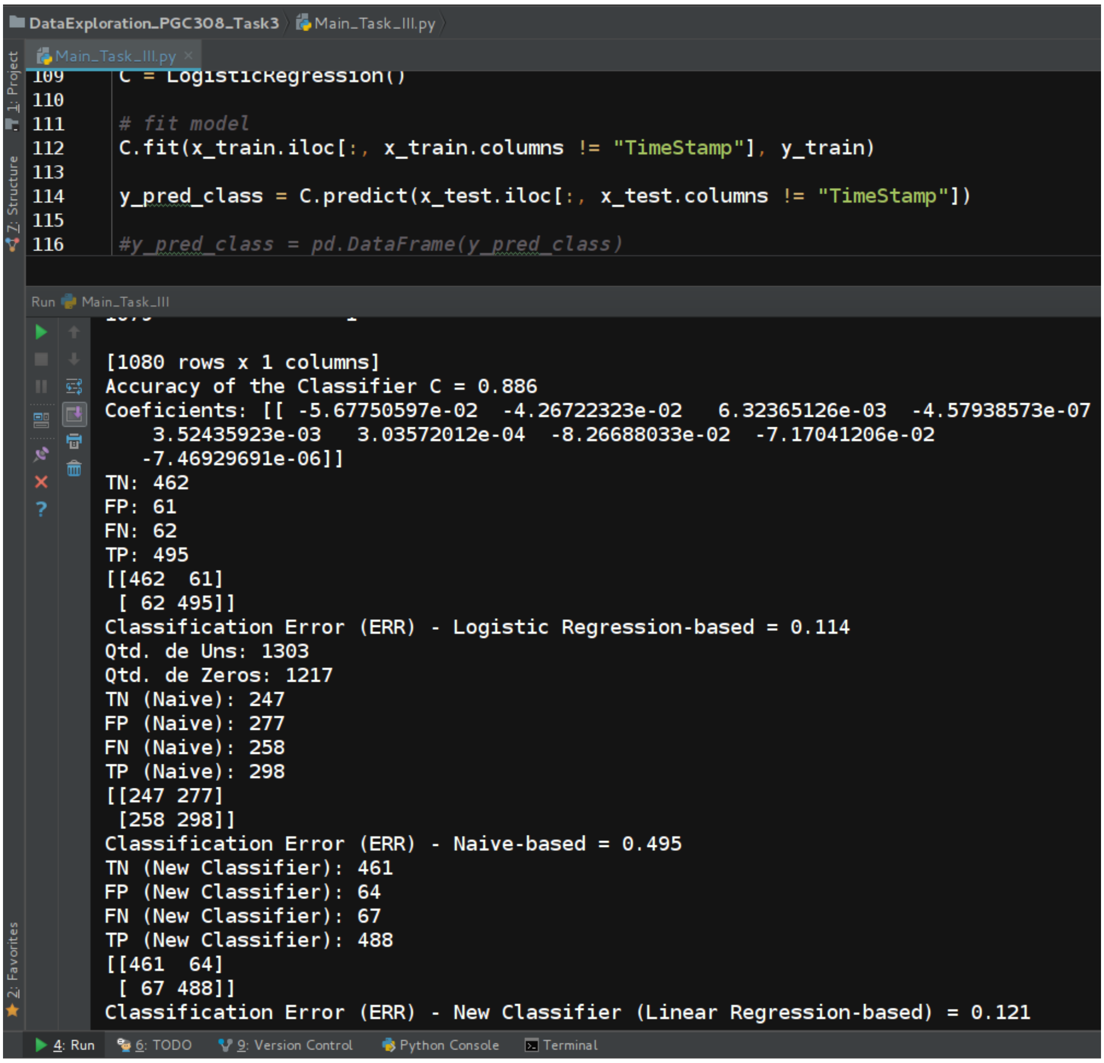
   Classification Error (ERR) - New Classifier (Linear Regression-based) ≈ 0.121, i.e. $\approx 12.1\%$.

5. Observations and conclusions based on the above work:

   The confusion matrix allows visualization of classifier performance. Allowing to check the number of correct classifications as opposed to the classifications predicted for each class. To the resolutions of the questions raised, it is possible to conclude that the accuracy of the classifier using Logistic Regression is sufficiently better than using a Naïve approach. In addition, it is possible to conclude that the accuracy, measured by the metrics (ERR) when using learning-based on Linear Regression as the learning engine for the classifier, we obtain results numerically lower then Logistic Regression. Thus, it is possible to conclude that using the Logistic Regression approach will allow higher classification rates considering the engines proposed for the classifiers.

The codes used to solve these questions are available in the following link: https://github.com/romoreira/MLN/blob/master/Main_Task_III.py

The Figure 1 depicts the output of execution of our Python Script.



```
C = LogisticRegression()

# fit model
C.fit(x_train.iloc[:, x_train.columns != "TimeStamp"], y_train)

y_pred_class = C.predict(x_test.iloc[:, x_test.columns != "TimeStamp"])

#y_pred_class = pd.DataFrame(y_pred_class)
```

```
1079                    1

[1080 rows x 1 columns]
Accuracy of the Classifier C = 0.886
Coeficients: [[ -5.67750597e-02  -4.26722323e-02   6.32365126e-03  -4.57938573e-07
    3.52435923e-03   3.03572012e-04  -8.26688033e-02  -7.17041206e-02
   -7.46929691e-06]]
TN:  462
FP:  61
FN:  62
TP:  495
[[462  61]
 [ 62 495]]
Classification Error (ERR) - Logistic Regression-based = 0.114
Qtd. de Uns: 1303
Qtd. de Zeros: 1217
TN (Naive): 247
FP (Naive): 277
FN (Naive): 258
TP (Naive): 298
[[247 277]
 [258 298]]
Classification Error (ERR) - Naive-based = 0.495
TN (New Classifier): 461
FP (New Classifier): 64
FN (New Classifier): 67
TP (New Classifier): 488
[[461  64]
 [ 67 488]]
Classification Error (ERR) - New Classifier (Linear Regression-based) = 0.121
```

Figure 1: Console Output – Pycharm IntelliJ

Task_III

November 22, 2017

```
In [4]: from __future__ import division
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn import linear_model
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import confusion_matrix
        from sklearn import metrics
        from sklearn.utils.validation import column_or_1d
        import pandas as pd
        from sklearn.cross_validation import train_test_split

        def get_dataframe():
            csv_x = pd.read_csv('./data/X.csv', sep=',', header=None)
            csv_y = pd.read_csv('./data/Y.csv', sep=',', header=None)

            # Esse trecho de codigo retira a primeira linha do DataFrame (que contem os
        nomes das colunas), cria uma novo DataFrame sem essa primeira linha,
            # depois adiciona as colunas na forma de indices
            new_header = csv_x.iloc[0]
            csv_x = csv_x[1:]
            csv_x.columns = new_header

            new_header = csv_y.iloc[0]
            csv_y = csv_y[1:]
            csv_y.columns = new_header


            csv_x['TimeStamp'] = pd.to_numeric(csv_x['TimeStamp'])
            csv_x['all_..idle'] = pd.to_numeric(csv_x['all_..idle'])
            csv_x['X..memused'] = pd.to_numeric(csv_x['X..memused'])
            csv_x['proc.s'] = pd.to_numeric(csv_x['proc.s'])
            csv_x['cswch.s'] = pd.to_numeric(csv_x['cswch.s'])
            csv_x['file.nr'] = pd.to_numeric(csv_x['file.nr'])
            csv_x['sum_intr.s'] = pd.to_numeric(csv_x['sum_intr.s'])
            csv_x['tcpsck'] = pd.to_numeric(csv_x['tcpsck'])
            csv_x['pgfree.s'] = pd.to_numeric(csv_x['pgfree.s'])
```

```
    csv_y['TimeStamp'] = pd.to_numeric(csv_y['TimeStamp'])
    csv_y['DispFrames'] = pd.to_numeric(csv_y['DispFrames'])




    return csv_x, csv_y

def dataset_headers(dataset):
    #Monto uma lista com os nomes das colunas
    return list(dataset.columns.values)

def binarize_y(y):
    # Adiciona a Y (Target) valores binarios para o SLA Conformance
    i = 0
    sla_conformance_y = np.array([])

    for i in range(len(y)):
        if y.iloc[i]['DispFrames'] >= 18:
            sla_conformance_y = np.append(sla_conformance_y, 1.0)
        else:
            sla_conformance_y = np.append(sla_conformance_y, 0.0)
        i += 1

    return sla_conformance_y


#---------Task III ---------------------

csv_x, csv_y = get_dataframe()

x_train, x_test, y_train, y_test = train_test_split(csv_x, csv_y,
test_size=0.30)

#Seto novamente a configuracao de DataFrame para nao perder a dimensao
x_train = pd.DataFrame(x_train, columns=['TimeStamp','all_..idle','X..memused','
proc.s','cswch.s','file.nr','sum_intr.s','ldavg.1','tcpsck','pgfree.s'])
x_test = pd.DataFrame(x_test, columns=['TimeStamp','all_..idle','X..memused','pr
oc.s','cswch.s','file.nr','sum_intr.s','ldavg.1','tcpsck','pgfree.s'])
y_train = pd.DataFrame(y_train, columns=['TimeStamp','DispFrames'])
y_test = pd.DataFrame(y_test, columns=['TimeStamp','DispFrames'])

y_train["SLA_Conformance"] = pd.to_numeric(binarize_y(y_train))
y_test["SLA_Conformance"] = pd.to_numeric(binarize_y(y_test))

y_train = y_train.iloc[:,y_train.columns != "TimeStamp"]
y_train = y_train.iloc[:,y_train.columns != "DispFrames"]
```

```
y_test = y_test.iloc[:,y_test.columns != "TimeStamp"]
y_test = y_test.iloc[:,y_test.columns != "DispFrames"]


y_train = column_or_1d(y_train, warn=False)

#Instantiate model
C = LogisticRegression()

# fit model
C.fit(x_train.iloc[:, x_train.columns != "TimeStamp"], y_train)

y_pred_class = C.predict(x_test.iloc[:, x_test.columns != "TimeStamp"])

#y_pred_class = pd.DataFrame(y_pred_class)
print("Accuracy of the Classifier C = %.3f" % metrics.accuracy_score(y_test,
y_pred_class))

print("Coeficients: "+np.array2string(C.coef_))

#print("BLA: %s " % y_test['SLA_Conformance'].values)

y_pred_class = pd.DataFrame(y_pred_class)

#y_test.to_csv("y_test.csv", sep='\t')
#y_pred_class.to_csv("y_pred_class.csv", sep='\t')

TN, FP, FN, TP = metrics.confusion_matrix(y_test, y_pred_class).ravel()

print("TN: %d " % TN)
print("FP: %d " % FP)
print("FN: %d " % FN)
print("TP: %d " % TP)

print(metrics.confusion_matrix(y_test, y_pred_class))

m = len(y_test)
ERR = 1 - (TP.astype(float) + TN.astype(float))/m

print("Classification Error (ERR) - Logistic Regression-based = %.3f" % ERR)


#_________________________________________________________
#---------Classifier based in a Naive Method--------------
#_________________________________________________________

csv_x, csv_y = get_dataframe()
```

```
x_train, x_test, y_train, y_test = train_test_split(csv_x, csv_y,
test_size=0.30)

#Seto novamente a configuracao de DataFrame para nao perder a dimensao
x_train = pd.DataFrame(x_train, columns=['TimeStamp','all_..idle','X..memused','
proc.s','cswch.s','file.nr','sum_intr.s','ldavg.1','tcpsck','pgfree.s'])
x_test = pd.DataFrame(x_test, columns=['TimeStamp','all_..idle','X..memused','pr
oc.s','cswch.s','file.nr','sum_intr.s','ldavg.1','tcpsck','pgfree.s'])
y_train = pd.DataFrame(y_train, columns=['TimeStamp','DispFrames'])
y_test = pd.DataFrame(y_test, columns=['TimeStamp','DispFrames'])

y_train.to_csv("y_train_dispframes.csv", sep='\t')

y_train["SLA_Conformance"] = binarize_y(y_train)
y_test["SLA_Conformance"] = binarize_y(y_test)

y_train = y_train.iloc[:,y_train.columns != "TimeStamp"]
y_train = y_train.iloc[:,y_train.columns != "DispFrames"]

y_test = y_test.iloc[:,y_test.columns != "TimeStamp"]
y_test = y_test.iloc[:,y_test.columns != "DispFrames"]

#print(y_train)

#Ajuste de configuracao utilizada no metodo fit - treina o x apenasa com a saida
binaria de y
y_train = column_or_1d(y_train, warn=False)

#Instantiate model
C_naive = LogisticRegression()

# fit model
C_naive.fit(x_train.iloc[:, x_train.columns != "TimeStamp"], y_train)

qtd_uns = 0
qtd_zeros = 0
i = 0

for i in range(len(y_train)):
    if(y_train[i] == 1):
        qtd_uns += 1

    else:
        qtd_zeros += 1
y_train = pd.DataFrame(y_train)
y_train.to_csv("y_train.csv", sep='\t')

print("Qtd. de Uns: %d" % qtd_uns)
print("Qtd. de Zeros: %d" % qtd_zeros)

p = qtd_uns / len(y_train)
y_pred_class = np.array([])
for _ in range(len(x_test)):
    #print("Probability")
    #print(qtd_uns/len(y_train))
    #print("Choice")
    #print(np.random.binomial(1, p))
    y_pred_class = np.append(y_pred_class, np.random.binomial(1, p))

y_pred_class = pd. DataFrame(y_pred_class)
TN, FP, FN, TP = metrics.confusion_matrix(y_test, y_pred_class).ravel()
print("TN (Naive): %d " % TN)
print("FP (Naive): %d " % FP)
print("FN (Naive): %d " % FN)
print("TP (Naive): %d " % TP)

print(metrics.confusion_matrix(y_test, y_pred_class))

m = len(y_test)
ERR = 1 - (TP.astype(float) + TN.astype(float))/m

print("Classification Error (ERR) - Naive-based = %.3f" % ERR)

#_________________________________________________________
#---------Build a new Classifier--------------
#_________________________________________________________

csv_x, csv_y = get_dataframe()

x_train, x_test, y_train, y_test = train_test_split(csv_x, csv_y,
test_size=0.30)

#Seto novamente a configuracao de DataFrame para nao perder a dimensao
x_train = pd.DataFrame(x_train, columns=['TimeStamp','all_..idle','X..memused','
proc.s','cswch.s','file.nr','sum_intr.s','ldavg.1','tcpsck','pgfree.s'])
x_test = pd.DataFrame(x_test, columns=['TimeStamp','all_..idle','X..memused','pr
oc.s','cswch.s','file.nr','sum_intr.s','ldavg.1','tcpsck','pgfree.s'])
y_train = pd.DataFrame(y_train, columns=['TimeStamp','DispFrames'])
y_test = pd.DataFrame(y_test, columns=['TimeStamp','DispFrames'])

#Binarizo o y_test
y_test["SLA_Conformance"] = binarize_y(y_test)
y_test = y_test.iloc[:,y_test.columns != "TimeStamp"]
y_test = y_test.iloc[:,y_test.columns != "DispFrames"]

regr = linear_model.LinearRegression()
```

```
regr.fit(x_train.iloc[:, x_train.columns != "TimeStamp"], y_train.iloc[:,
y_train.columns != "TimeStamp"])
y_pred = regr.predict(x_test.iloc[:, x_test.columns != "TimeStamp"])

#Binarizar a saida da prdicao baseada em Regressao para montar a matriz de
confusao
y_pred = pd.DataFrame(y_pred)
sla_pred_bin = np.array([])
i = 0
for i in range(len(y_pred)):
    if (y_pred.iloc[i].astype(float) >= 18).bool():
        sla_pred_bin = np.append(sla_pred_bin, 1)
        i += 1
    else:
        sla_pred_bin = np.append(sla_pred_bin, 0)
        i += 1

sla_pred_bin = pd.DataFrame(sla_pred_bin)
sla_pred_bin['SLA_Conformance'] = sla_pred_bin

#print(sla_pred_bin.shape)
sla_pred_bin = column_or_1d(sla_pred_bin['SLA_Conformance'], warn=False)
#print(sla_pred_bin.shape)

#print(y_test.shape)
y_test = column_or_1d(y_test['SLA_Conformance'], warn=False)
#print(y_test.shape)

TN, FP, FN, TP = metrics.confusion_matrix(y_test, sla_pred_bin).ravel()
print("TN (New Classifier): %d " % TN)
print("FP (New Classifier): %d " % FP)
print("FN (New Classifier): %d " % FN)
print("TP (New Classifier): %d " % TP)

print(metrics.confusion_matrix(y_test, sla_pred_bin))

m = len(y_test)
ERR = 1 - (TP.astype(float) + TN.astype(float))/m

print("Classification Error (ERR) - New Classifier (Linear Regression-based) =
%.3f" % ERR)
```

```
Accuracy of the Classifier C = 0.877
Coeficients: [[ -5.67887233e-02  -3.01543040e-02  -8.90274167e-04  -2.38493838e-06
    3.59119631e-03   2.04249079e-04  -8.41125892e-02  -5.72565722e-02
   -7.68648567e-06]]
TN: 463
FP: 72
```

```
FN: 61
TP: 484
[[463  72]
 [ 61 484]]
Classification Error (ERR) - Logistic Regression-based = 0.123
Qtd. de Uns: 1311
Qtd. de Zeros: 1209
TN (Naive): 244
FP (Naive): 288
FN (Naive): 270
TP (Naive): 278
[[244 288]
 [270 278]]
Classification Error (ERR) - Naive-based = 0.517
TN (New Classifier): 447
FP (New Classifier): 70
FN (New Classifier): 47
TP (New Classifier): 516
[[447  70]
 [ 47 516]]
Classification Error (ERR) - New Classifier (Linear Regression-based) = 0.108
```