Vandre.Task_III

November 21, 2017

```
In [1]: # Task III - Estimating SLA Conformance and Violation from Device Statistics
        import pandas as pd
        import numpy as np
        import sklearn.svm as svm
        import matplotlib.pyplot as plt
        import matplotlib.patches as mpatches
        from sklearn import datasets, linear_model, preprocessing
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import mean_absolute_error
        from pandas_ml import ConfusionMatrix
```

```
In [2]: X = pd.read_csv('../data/X.csv')
        Y = pd.read_csv('../data/Y.csv')
```

```
In [3]: # 1. Model Training - use Logistic Regression to train a classifier C
        # with the training set.
        X = X.iloc[:, X.columns != 'TimeStamp']
        Y = Y.iloc[:, Y.columns != 'TimeStamp']

        X_train, X_test, y_train, y_test = train_test_split(X, Y, train_size=0.7,
        test_size=0.3)

        X_train_int = X_train.astype('int')
        X_test_int = X_test.astype('int')
        y_train_int = y_train.astype('int')
        y_test_int = y_test.astype('int')

        y_train_bool = y_train_int['DispFrames'] >= 18
        y_test_bool = y_test_int['DispFrames'] >= 18

        lr = linear_model.LogisticRegression()
        model = lr.fit(X_train_int, y_train_bool)

        y_pred = lr.predict(X_test_int)
        y_true = y_test_bool.values.ravel()
```

```
In [4]: # Provide the coefficients (0, ..., 9) of your model C. (0 is the offset.)
        print 'Coefficients:', lr.coef_
```

```
Coefficients: [[ -7.03388772e-02  -3.93219130e-02   3.66521643e-03
-1.25802942e-05
    4.13081999e-03   2.21754981e-04  -8.77979774e-02  -5.80615276e-02
   -8.43599033e-06]]
```

```
In [5]: # (b) Accuracy of Model M - compute the estimation error of M over the test set.
        # Explained variance score: 1 is perfect prediction
        print 'Accuracy score: %.4f' % model.score(X_test_int, y_true)
```

```
Accuracy score: 0.8806
```

```
In [6]: # 2. Accuracy of the Classifiers C - Compute the classification error (ERR) on
        # the test set for C. For this, you first compute the confusion matrix,
        # which includes the four numbers True Positives (TP), True Negatives (TN),
        # False Positives (FN), and False Negatives (FN). We define the classification
        # error as ERR = 1  (TP+TN)/m , whereby m is the number of observations in
        # the test set. A true positive is an m observation that is correctly classified
        # by the classifier as conforming to the SLA; a true negative is an observation
        # that is correctly classified by the classifier as violating the SLA.

        m = len(y_true)
        cnf_matrix = ConfusionMatrix(y_true, y_pred)

        print("Confusion Matrix:\n%s\n" % cnf_matrix)
        print "Stats:\n", cnf_matrix.print_stats(), '\n'

        cls_error = 1.0 - (float(cnf_matrix.TP + cnf_matrix.TN) / m)
        print("Classification Error: %.4f" % cls_error)
```

```
Confusion Matrix:
Predicted  False  True  __all__
Actual
False        455    71      526
True          58   496      554
__all__      513   567     1080

Stats:
population: 1080
P: 554
N: 526
PositiveTest: 567
NegativeTest: 513
TP: 496
TN: 455
FP: 71
FN: 58
TPR: 0.895306859206
```

```
TNR: 0.865019011407
PPV: 0.874779541446
NPV: 0.88693957115
FPR: 0.134980988593
FDR: 0.125220458554
FNR: 0.104693140794
ACC: 0.880555555556
F1_score: 0.884924174844
MCC: 0.76102217277
informedness: 0.760325870613
markedness: 0.761719112596
prevalence: 0.512962962963
LRP: 6.63283673158
LRN: 0.121029872654
DOR: 54.8033025741
FOR: 0.11306042885
None

Classification Error: 0.1194
```

```
In [7]: # 3. As a baseline for C, use a naive method which relies on Y values only,
        # as follows. For each x  X, the naive classifier predicts a value True with
        # probability p and False with probability 1  p. p is the fraction of Y values
        # that conform with the SLA. Compute p over the training set and the
        # classification error for the naive classifier over the test set.

        m = len(y_true)
        p = float(np.sum(y_train_int >= 18)) / len(y_train_int)

        naive = np.random.choice([True, False], size=(m), p=[p, (1-p)])
        cnf_matrix = ConfusionMatrix(y_true, naive)

        print("Confusion Matrix:\n%s\n" % cnf_matrix)
        print "Stats:\n", cnf_matrix.print_stats(), '\n'

        cls_error = 1.0 - (float(cnf_matrix.TP + cnf_matrix.TN) / m)
        print("Classification Error: %.4f" % cls_error)
```

```
Confusion Matrix:
Predicted  False  True  __all__
Actual
False        237   289      526
True         260   294      554
__all__      497   583     1080

Stats:
population: 1080
```

```
P: 554
N: 526
PositiveTest: 583
NegativeTest: 497
TP: 294
TN: 237
FP: 289
FN: 260
TPR: 0.530685920578
TNR: 0.450570342205
PPV: 0.504288164666
NPV: 0.476861167002
FPR: 0.549429657795
FDR: 0.495711835334
FNR: 0.469314079422
ACC: 0.491666666667
F1_score: 0.517150395778
MCC: -0.0187971267376
informedness: -0.0187437372171
markedness: -0.0188506683325
prevalence: 0.512962962963
LRP: 0.965885101121
LRN: 1.04160002437
DOR: 0.927309023157
FOR: 0.523138832998
None

Classification Error: 0.5083
```

In [8]:
```
# 4. Build a new classifier by extending the linear regression function
# developed in Task II with a check on the output, i.e., the Video Frame Rate.
# If the frame rate for a given X is above the SLA threshold, then the Y label
# of the classifier is set to conformance, otherwise to violation. Compute the
# new classifier over the training set and the classification error for this
# new classifier over the test set.

lm = linear_model.LinearRegression()
model = lm.fit(X_train, y_train.DispFrames)
y_pred = lm.predict(X_test)

cnf_matrix = ConfusionMatrix(y_true, y_pred >= 18)

print("Confusion Matrix:\n%s\n" % cnf_matrix)
print "Stats:\n", cnf_matrix.print_stats(), '\n'

cls_error = 1.0 - (float(cnf_matrix.TP + cnf_matrix.TN) / m)
print("Classification Error: %.4f" % cls_error)
```

```
Confusion Matrix:
Predicted  False  True  __all__
Actual
False        442    84      526
True          44   510      554
__all__      486   594     1080

Stats:
population: 1080
P: 554
N: 526
PositiveTest: 594
NegativeTest: 486
TP: 510
TN: 442
FP: 84
FN: 44
TPR: 0.920577617329
TNR: 0.84030418251
PPV: 0.858585858586
NPV: 0.909465020576
FPR: 0.15969581749
FDR: 0.141414141414
FNR: 0.0794223826715
ACC: 0.881481481481
F1_score: 0.88850174216
MCC: 0.764457935601
informedness: 0.760881799838
markedness: 0.768050879162
prevalence: 0.512962962963
LRP: 5.76456936565
LRN: 0.094516229152
DOR: 60.9902597403
FOR: 0.0905349794239
None

Classification Error: 0.1185
```

```
In [9]: # 5. Formulate your observations and conclusions based on the above work.

        # Both the logistic and linear classifiers produced close results and
        # similar classification errors around 0.12, whereas the naive classifier
        # did a poor job at predicting and had a classification error around 0.5.
```