FordAsleep

July 10, 2017

1 Stay Alert! The Ford Challenge

by Scott Josephson

Driving while distracted, fatigued or drowsy may lead to accidents. Activities that divert the driver's attention from the road ahead, such as engaging in a conversation with other passengers in the car, making or receiving phone calls, sending or receiving text messages, eating while driving or events outside the car may cause driver distraction. Fatigue and drowsiness can result from driving long hours or from lack of sleep.

The data for this Kaggle challenge shows the results of a number of "trials", each one representing about 2 minutes of sequential data that are recorded every 100 ms during a driving session on the road or in a driving simulator. The trials are samples from some 100 drivers of both genders, and of different ages and ethnic backgrounds. The files are structured as follows:

The first column is the Trial ID - each period of around 2 minutes of sequential data has a unique trial ID. For instance, the first 1210 observations represent sequential observations every 100ms, and therefore all have the same trial ID The second column is the observation number - this is a sequentially increasing number within one trial ID The third column has a value X for each row where

```
X = 1     if the driver is alert

X = 0     if the driver is not alert
```

The next 8 columns with headers P1, P2 , ..., P8 represent physiological data;

The next 11 columns with headers E1, E2, ..., E11 represent environmental data;

The next 11 columns with headers V1, V2, ..., V11 represent vehicular data;

1.1 Import Libraries

```
In [1]: import numpy as np
        import pandas as pd

        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score, classification_report
```

1.2 Get the Data

Read in the fordtrain.csv file and set it to a data frame called ford_train.
** Split the data into training set and testing set using train_test_split**

```
In [2]: ford_train = pd.read_csv('fordtrain.csv')
```

Check the head of ad_data

```
In [3]: ford_train.head()
```

```
Out[3]:    TrialID  ObsNum  IsAlert       P1        P2    P3       P4        P5   P6  \
        0        0       0        0  34.7406   9.84593  1400  42.8571  0.290601  572
        1        0       1        0  34.4215  13.41120  1400  42.8571  0.290601  572
        2        0       2        0  34.3447  15.18520  1400  42.8571  0.290601  576
        3        0       3        0  34.3421   8.84696  1400  42.8571  0.290601  576
        4        0       4        0  34.3322  14.69940  1400  42.8571  0.290601  576

                P7   ...        V2   V3        V4  V5    V6  V7    V8  V9  V10      V11
        0  104.895   ...     0.175  752  5.99375   0  2005   0  13.4   0    4  14.8004
        1  104.895   ...     0.455  752  5.99375   0  2007   0  13.4   0    4  14.7729
        2  104.167   ...     0.280  752  5.99375   0  2011   0  13.4   0    4  14.7736
        3  104.167   ...     0.070  752  5.99375   0  2015   0  13.4   0    4  14.7667
        4  104.167   ...     0.175  752  5.99375   0  2017   0  13.4   0    4  14.7757

        [5 rows x 33 columns]
```

```
In [4]: ford_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 604329 entries, 0 to 604328
Data columns (total 33 columns):
TrialID    604329 non-null int64
ObsNum     604329 non-null int64
IsAlert    604329 non-null int64
P1         604329 non-null float64
P2         604329 non-null float64
P3         604329 non-null int64
P4         604329 non-null float64
P5         604329 non-null float64
P6         604329 non-null int64
P7         604329 non-null float64
P8         604329 non-null int64
E1         604329 non-null float64
E2         604329 non-null float64
E3         604329 non-null int64
E4         604329 non-null int64
E5         604329 non-null float64
E6         604329 non-null int64
```

```
E7          604329 non-null int64
E8          604329 non-null int64
E9          604329 non-null int64
E10         604329 non-null int64
E11         604329 non-null float64
V1          604329 non-null float64
V2          604329 non-null float64
V3          604329 non-null int64
V4          604329 non-null float64
V5          604329 non-null int64
V6          604329 non-null int64
V7          604329 non-null int64
V8          604329 non-null float64
V9          604329 non-null int64
V10         604329 non-null int64
V11         604329 non-null float64
dtypes: float64(14), int64(19)
memory usage: 152.2 MB
```

2 Logistic Regression

Now it's time to do a train test split, and train our model!

Choose columns that you want to train on!

```
In [19]: X_train, X_test, y_train, y_test = train_test_split(ford_train.drop('IsAlert',axis=1)
                                                              test_size=0.30,random_state=101)
```

** Train and fit a logistic regression model on the training set.**

```
In [21]: logmodel = LogisticRegression()
```

```
In [22]: logmodel.fit(X_train, y_train)
```

```
Out[22]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False)
```

2.1 Predictions and Evaluations

** Now predict values for the testing data.**

```
In [23]: predictions = logmodel.predict(X_test)
```

** Create a classification report for the model.**

```
In [25]: print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

          0       0.82      0.73      0.77     76334
          1       0.82      0.88      0.85    104965

avg / total       0.82      0.82      0.82    181299
```