Regularized Logistic Regression sklearn

December 3, 2020

```
[100]: from sklearn.linear_model import LogisticRegression
       from sklearn.model_selection import train_test_split
       from sklearn import metrics
       import pandas as pd
       import numpy as np
```

```
[67]: data = pd.read_csv("./Data/caesarian.csv")
      data.iloc[0:3,:]
```

```
[67]:    Age  Delivery number  Delivery time  Blood of Pressure  Heart Problem  \
      0   22                1              0                  2              0
      1   26                2              0                  1              0
      2   26                2              1                  1              0

         Caesarian
      0          0
      1          1
      2          0
```

```
[125]: def test(df, split, penalty="l2"):
           data = pd.get_dummies(df, drop_first=True)
           X = data.iloc[:,0:len(data.columns)-1]
           y = data.iloc[:,len(data.columns)-1]
           X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
        ↪random_state=42)
           lr = LogisticRegression(penalty=penalty,max_iter=10000).fit(X_train,␣
        ↪y_train)
           accuracy = sum(y_test==lr.predict(X_test))/len(y_test)
           s = lr.score(X_test, y_test)
           auc = metrics.roc_auc_score(y_test, lr.predict_proba(X_test)[:,1])
           return lr, auc, accuracy
```

```
[123]: data = pd.get_dummies(data, drop_first=True)
       X = data.iloc[:,0:len(data.columns)-1]
       y = data.iloc[:,len(data.columns)-1]
       X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
        ↪random_state=42)
```

```
lr = LogisticRegression(penalty="l2",max_iter=10000).fit(X_train, y_train)
accuracy = sum(y_test==lr.predict(X_test))/len(y_test)
s = lr.score(X_test, y_test)
auc = metrics.roc_auc_score(y_test, lr.predict_proba(X_test)[:,1])
```

```
[187]: data = pd.read_csv("./Data/caesarian.csv")
```

```
[189]: lg, auc, acc = test(data, 0.75, penalty = "none")
       print("AUC:",auc,"Accuracy:",acc)
```

```
AUC: 0.7135416666666667 Accuracy: 0.7
```

```
[190]: lg, auc, acc = test(data, 0.75)
       print("AUC:",auc,"Accuracy:",acc)
```

```
AUC: 0.6927083333333333 Accuracy: 0.7
```

```
[191]: data = pd.read_csv("Data/monks-1.test",delimiter=' ',header=None)
       data = data.drop([0,8],axis=1)
       cols = data.columns.tolist()
       cols = cols[+1:] + cols[:+1]
       data = data[cols]
```

```
[192]: lg, auc, acc = test(data, 0.75, penalty = "none")
       print("AUC:",auc,"Accuracy:",acc)
```

```
AUC: 0.7347781217750259 Accuracy: 0.7129629629629629
```

```
[193]: lg, auc, acc = test(data, 0.75)
       print("AUC:",auc,"Accuracy:",acc)
```

```
AUC: 0.7347781217750258 Accuracy: 0.7129629629629629
```

```
[194]: data = pd.read_csv("Data/credit-screening/crx.data")
```

```
[195]: lg, auc, acc = test(data, 0.75, penalty = "none")
       print("AUC:",auc,"Accuracy:",acc)
```

```
AUC: 0.8029221653649339 Accuracy: 0.8034682080924855
```

```
[197]: lg, auc, acc = test(data, 0.75)
       print("AUC:",auc,"Accuracy:",acc)
```

```
AUC: 0.920549420953407 Accuracy: 0.8439306358381503
```

```
[198]: data = pd.read_csv("Data/framingham.csv")
```

```
[199]: lg, auc, acc = test(data, 0.75, penalty = "none")
       print("AUC:",auc,"Accuracy:",acc)
```

```
AUC: 0.9357530864197531 Accuracy: 0.8765027322404372
```

```
[200]: lg, auc, acc = test(data, 0.75)
       print("AUC:",auc,"Accuracy:",acc)
```

```
AUC: 0.9360493827160494 Accuracy: 0.8797814207650273
```

```
[204]: data = pd.read_csv("Data/default of credit card clients.csv")
```

```
[205]: pd.get_dummies(data, drop_first=True)
```

```
[205]:          ID  LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  \
       0          1      20000    2          2         1   24      2      2     -1
       1          2     120000    2          2         2   26     -1      2      0
       2          3      90000    2          2         2   34      0      0      0
       3          4      50000    2          2         1   37      0      0      0
       4          5      50000    1          2         1   57     -1      0     -1
       …        …        …  …          …       …  …     …    …      …
       29995  29996     220000    1          3         1   39      0      0      0
       29996  29997     150000    1          3         2   43     -1     -1     -1
       29997  29998      30000    1          2         2   37      4      3      2
       29998  29999      80000    1          3         1   41      1     -1      0
       29999  30000      50000    1          2         1   46      0      0      0

              PAY_4  …  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  \
       0         -1  …          0          0          0         0       689
       1          0  …       3272       3455       3261         0      1000
       2          0  …      14331      14948      15549      1518      1500
       3          0  …      28314      28959      29547      2000      2019
       4          0  …      20940      19146      19131      2000     36681
       …        …  …        …        …         …       …
       29995      0  …      88004      31237      15980      8500     20000
       29996     -1  …       8979       5190          0      1837      3526
       29997     -1  …      20878      20582      19357         0         0
       29998      0  …      52774      11855      48944     85900      3409
       29999      0  …      36535      32428      15313      2078      1800

              PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  default payment next month
       0             0         0         0         0                           1
       1          1000      1000         0      2000                           1
       2          1000      1000      1000      5000                           0
       3          1200      1100      1069      1000                           0
       4         10000      9000       689       679                           0
       …           …       …       …       …                       …
```

```
29995     5003      3047      5000      1000                              0
29996     8998       129         0         0                              0
29997    22000      4200      2000      3100                              1
29998     1178      1926     52964      1804                              1
29999     1430      1000      1000      1000                              1

[30000 rows x 25 columns]
```

[202]:
```
lg, auc, acc = test(data, 0.75, penalty = "none")
print("AUC:",auc,"Accuracy:",acc)
```

```
AUC: 0.6501707782984041 Accuracy: 0.7830666666666667
```

[203]:
```
lg, auc, acc = test(data, 0.75)
print("AUC:",auc,"Accuracy:",acc)
```

```
AUC: 0.6501443010428375 Accuracy: 0.7830666666666667
```

[ ]: