Python Tutorial 3 Exercises with Solutions

February 17, 2020

This is the solution of "Python Tutorial 3 Exercises" for Prof. Xin Tong's DSO 530 class at the University of Southern California in spring 2020.

1. With the same stock market data in *Python Tutorial 3*, train the model using logistic regression with data **before 2004**, and test with data **in 2004** (do not use 2005 at all in either training and testing).

1) For predictor variables, only use *Lag1* and *Lag2*.

2) Use both 0.50 and 0.49 as the threshold for $P(Y = 1|X = x)$.

3) Using both of the above decision thresholds, construct the *confusion matrix* on the test data

Anwser:

[1]:
```
import pandas as pd
import numpy as np
smarket = pd.read_csv('smarket.csv')
smarket['Up'] = np.where(smarket['Direction'] == 'Up', 1, 0)
```

Because *smarket* data is ordered by *Year*, we can use *tail()* function to return the last n(default 5) rows from the dataset in order to check if we get rid of the data in 2005.

[2]:
```
smarket.tail()
```

[2]:

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction	Up
1245	2005	0.422	0.252	-0.024	-0.584	-0.285	1.88850	0.043	Up	1
1246	2005	0.043	0.422	0.252	-0.024	-0.584	1.28581	-0.955	Down	0
1247	2005	-0.955	0.043	0.422	0.252	-0.024	1.54047	0.130	Up	1
1248	2005	0.130	-0.955	0.043	0.422	0.252	1.42236	-0.298	Down	0
1249	2005	-0.298	0.130	-0.955	0.043	0.422	1.38254	-0.489	Down	0

We create a dataset *smarket_without_2005* without the data in 2005. Then we could split the training and test data as we did in *Python Tutorial 3*.

[3]:
```
smarket_without_2005 = smarket[smarket['Year'] < 2005]
smarket_without_2005.tail()
```

[3]:

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction	Up
993	2004	0.046	0.342	0.904	0.038	-0.749	0.9561	-0.431	Down	0
994	2004	-0.431	0.046	0.342	0.904	0.038	0.9220	0.715	Up	1

```
995  2004  0.715 -0.431  0.046  0.342  0.904  0.9830 -0.007      Down   0
996  2004 -0.007  0.715 -0.431  0.046  0.342  0.9259  0.008        Up   1
997  2004  0.008 -0.007  0.715 -0.431  0.046  0.8298 -0.134      Down   0
```

[4]:
```
# split the dataset into training and testdata
X = smarket_without_2005[['Lag1', 'Lag2']]
y = smarket_without_2005['Up']

train_bool = smarket_without_2005['Year'] < 2004

X_test = X[~train_bool]
y_test = y[~train_bool]
```

[5]:
```
# train the logistic regression model using training data
import statsmodels.formula.api as smf
result = smf.logit('Up ~ Lag1 + Lag2', data=smarket_without_2005, subset =␣
 ↪train_bool).fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.692045
         Iterations 3
```

[5]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:                     Up   No. Observations:                  746
Model:                          Logit   Df Residuals:                      743
Method:                           MLE   Df Model:                            2
Date:                Mon, 17 Feb 2020   Pseudo R-squ.:                0.001404
Time:                        10:29:31   Log-Likelihood:                -516.27
converged:                       True   LL-Null:                       -516.99
Covariance Type:            nonrobust   LLR p-value:                    0.4839
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0333      0.073     -0.454      0.650      -0.177       0.110
Lag1          -0.0524      0.054     -0.971      0.332      -0.158       0.053
Lag2          -0.0396      0.054     -0.734      0.463      -0.145       0.066
==============================================================================
"""
```

[6]:
```
# obtain the predicted probabilities of the stock market
result_prob = result.predict(X_test)
```

[7]:
```
from sklearn.metrics import confusion_matrix
```

[8]:
```
# choose 0.5 as threshold and construct the confusion matrix
result_pred1 = (result_prob > 0.5)
confusion_matrix(y_test, result_pred1)
```

[8]:
```
array([[ 89,  23],
       [110,  30]], dtype=int64)
```

<threshold = 0.50>	Down(result_pred1)	Up(result_pred1)
Down(y_test)	89	23
Up(y_test)	110	30

[9]:
```
# choose 0.49 as threshold and construct the confusion matrix
result_pred2 = (result_prob > 0.49)
confusion_matrix(y_test, result_pred2)
```

[9]:
```
array([[62, 50],
       [66, 74]], dtype=int64)
```

<threshold = 0.49>	Down(result_pred1)	Up(result_pred1)
Down(y_test)	62	50
Up(y_test)	66	74

We can calculate the accuracy, type I error rate and type II error rate as follows (the question doesn't ask for it).

[10]:
```
print("threshold = 0.50:")
print("\taccuracy: ", np.mean(result_pred1 == y_test))
print("\ttype I error rate: ", 23/(89+23))
print("\ttype II error rate: ", 110/(110+30))

print("\nthreshold = 0.49:")
print("\taccuracy: ", np.mean(result_pred2 == y_test))
print("\ttype I error rate: ", 50/(50+62))
print("\ttype II error rate: ", 66/(66+74))
```

```
threshold = 0.50:
        accuracy:  0.4722222222222222
        type I error rate:  0.20535714285714285
        type II error rate:  0.7857142857142857
threshold = 0.49:
        accuracy:  0.5396825396825397
        type I error rate:  0.44642857142857145
        type II error rate:  0.4714285714285714
```

We can see that accuracy, type I and type II errors change with the different threshold choices.