4.6.2 Logistic Regression

May 18, 2018

In [1]:
```python
# conventional way to import pandas
import pandas as pd
# conventional way to import seaborn
import seaborn as sns
# conventional way to import numpy
import numpy as np

from sklearn import metrics, linear_model
import matplotlib.pyplot as plt

data = pd.read_csv("https://raw.github.com/vincentarelbundock/Rdatasets/master/csv/ISLR
data.head()
```

Out[1]:

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
1	2001	0.381	-0.192	-2.624	-1.055	5.010	1.1913	0.959	Up
2	2001	0.959	0.381	-0.192	-2.624	-1.055	1.2965	1.032	Up
3	2001	1.032	0.959	0.381	-0.192	-2.624	1.4112	-0.623	Down
4	2001	-0.623	1.032	0.959	0.381	-0.192	1.2760	0.614	Up
5	2001	0.614	-0.623	1.032	0.959	0.381	1.2057	0.213	Up

In [2]:
```python
import statsmodels.api as sm
from scipy import stats
from patsy import dmatrices
y, X = dmatrices('Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume', data, return_type = 'data
print(y)
```

```
C:\Users\entvex\Anaconda3\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The
  from pandas.core import datetools


     Direction[Down]  Direction[Up]
1                0.0            1.0
2                0.0            1.0
3                1.0            0.0
4                0.0            1.0
5                0.0            1.0
6                0.0            1.0
7                1.0            0.0
```

8	0.0	1.0
9	0.0	1.0
10	0.0	1.0
11	1.0	0.0
12	1.0	0.0
13	0.0	1.0
14	0.0	1.0
15	1.0	0.0
16	0.0	1.0
17	1.0	0.0
18	0.0	1.0
19	1.0	0.0
20	1.0	0.0
21	1.0	0.0
22	1.0	0.0
23	0.0	1.0
24	1.0	0.0
25	1.0	0.0
26	0.0	1.0
27	1.0	0.0
28	1.0	0.0
29	1.0	0.0
30	1.0	0.0
...	...	...
1221	0.0	1.0
1222	0.0	1.0
1223	0.0	1.0
1224	0.0	1.0
1225	0.0	1.0
1226	0.0	1.0
1227	1.0	0.0
1228	0.0	1.0
1229	1.0	0.0
1230	0.0	1.0
1231	0.0	1.0
1232	1.0	0.0
1233	0.0	1.0
1234	1.0	0.0
1235	1.0	0.0
1236	0.0	1.0
1237	0.0	1.0
1238	0.0	1.0
1239	0.0	1.0
1240	1.0	0.0
1241	1.0	0.0
1242	1.0	0.0
1243	1.0	0.0
1244	0.0	1.0

```
1245              0.0            1.0
1246              0.0            1.0
1247              1.0            0.0
1248              0.0            1.0
1249              1.0            0.0
1250              1.0            0.0

[1250 rows x 2 columns]
```

We are using y.iloc[:,1] to set Direction

```
In [3]: logit = sm.GLM(y.iloc[:,1],X, family=sm.families.Binomial())

        # fit the model
        result = logit.fit()
```

```
In [4]: print (result.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:           Direction[Up]   No. Observations:                 1250
Model:                             GLM   Df Residuals:                     1243
Model Family:                 Binomial   Df Model:                            6
Link Function:                   logit   Scale:                             1.0
Method:                           IRLS   Log-Likelihood:                -863.79
Date:                 Sat, 03 Mar 2018   Deviance:                       1727.6
Time:                         15:45:43   Pearson chi2:                 1.25e+03
No. Iterations:                      4
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.1260      0.241     -0.523      0.601      -0.598       0.346
Lag1          -0.0731      0.050     -1.457      0.145      -0.171       0.025
Lag2          -0.0423      0.050     -0.845      0.398      -0.140       0.056
Lag3           0.0111      0.050      0.222      0.824      -0.087       0.109
Lag4           0.0094      0.050      0.187      0.851      -0.089       0.107
Lag5           0.0103      0.050      0.208      0.835      -0.087       0.107
Volume         0.1354      0.158      0.855      0.392      -0.175       0.446
==============================================================================
```

This displays the probabilities for the market going up for the traning data.

```
In [5]: result.predict()[0:11]
```

```
Out[5]: array([ 0.50708413,  0.48146788,  0.48113883,  0.51522236,  0.51078116,
                0.50695646,  0.49265087,  0.50922916,  0.51761353,  0.48883778,
                0.4965211 ])
```

Setting the predtion value so if it is above 0.5 then the market goes up if not then it goes down.

```
In [6]: predict_label = pd.DataFrame(np.zeros(shape=(1250,1)), columns = ['label'])
        predict_label.iloc[result.predict()>0.5] = 1
```

Then we create a confusion_matrix. So we can see how many we got right. https://www.wikiwand.com/en/Confusion_matrix

```
In [7]: from sklearn.metrics import confusion_matrix
        confusion_matrix(y.iloc[:,1], predict_label.iloc[:,0])

Out[7]: array([[145, 457],
               [141, 507]], dtype=int64)
```

The diagonal elements of the confusion matrix indicate correct predictions, while the off-diagonals represent incorrect predictions. In this case, logistic regression correctly predicted the movement of the market 52.2% of the time

```
In [8]: np.mean(y.iloc[:,1].values == predict_label.iloc[:,0].values) # to get accuracy

Out[8]: 0.52159999999999995
```

In order to better assess the accuracy of the logistic regression model in this setting, we can fit the model using part of the data, and then examine how well it predicts the held out data. This will yield a more realistic error rate, in the sense that in practice we will be interested in our model's performance not on the data that we used to fit the model, but rather on days in the future for which the market's movements are unknown.

```
In [9]: Smarket_2005 = data.query('Year >= 2005')
        Smarket_train = data.query('Year < 2005')
```

We will use the training dataset to build the logistic regression model

```
In [10]: y_train, X_train = dmatrices('Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume', Smarket_trai
         y_test, X_test = dmatrices('Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume', Smarket_2005,

In [11]: logit = sm.GLM(y_train.iloc[:,1], X_train, family=sm.families.Binomial())
         print( logit.fit().summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:           Direction[Up]   No. Observations:                  998
Model:                             GLM   Df Residuals:                      991
Model Family:                 Binomial   Df Model:                            6
Link Function:                   logit   Scale:                             1.0
Method:                           IRLS   Log-Likelihood:                -690.55
Date:                 Sat, 03 Mar 2018   Deviance:                       1381.1
Time:                         15:45:43   Pearson chi2:                     998.
No. Iterations:                      4
==============================================================================
```

```
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.1912      0.334      0.573      0.567      -0.463       0.845
Lag1          -0.0542      0.052     -1.046      0.295      -0.156       0.047
Lag2          -0.0458      0.052     -0.884      0.377      -0.147       0.056
Lag3           0.0072      0.052      0.139      0.889      -0.094       0.108
Lag4           0.0064      0.052      0.125      0.901      -0.095       0.108
Lag5          -0.0042      0.051     -0.083      0.934      -0.104       0.096
Volume        -0.1163      0.240     -0.485      0.628      -0.586       0.353
==============================================================================
```

```
In [12]: preds = logit.fit().predict(X_test)
         predict_label = pd.DataFrame(np.zeros(shape=(X_test.shape[0],1)), columns = ['label']
         threshold = 0.5
         mark = (preds > threshold).reset_index(drop=True)
         predict_label.loc[mark] = 1
         confusion_matrix(y_test.iloc[:,1], predict_label.iloc[:,0])

Out[12]: array([[77, 34],
                [97, 44]], dtype=int64)

In [13]: np.mean(y_test.iloc[:,1].reset_index(drop=True)==predict_label.iloc[:,0].reset_index(

Out[13]: 0.48015873015873017
```

Notice that we have trained and tested our model on two completely separate data sets: training was performed using only the dates before 2005, and testing was performed using only the dates in 2005. Finally, we compute the predictions for 2005 and compare them to the actual movements of the market over that time period. The results are rather disappointing: the test error rate is 1 - 48% = 52 %, which is worse than random guessing! Of course this result is not all that surprising, given that one would not generally expect to be able to use previous days' returns to predict future market performance.ű The retrain of the model with Lag1 and Lag2 will be similar to previous steps (I will omit those). Another way to deal with logistics regression is to change the threshold value from 0.5 to others. There is an example below with threshold 0.45

```
In [14]: preds = logit.fit().predict(X_test)
         predict_label = pd.DataFrame(np.zeros(shape=(X_test.shape[0],1)), columns = ['label']
         threshold = 0.45
         predict_label.loc[(preds > threshold).reset_index(drop=True)] = 1
         confusion_matrix(y_test.iloc[:,1], predict_label.iloc[:,0])
         np.mean(y_test.iloc[:,1].reset_index(drop=True)==predict_label.iloc[:,0].reset_index(

Out[14]: 0.5674603174603174

In [15]: y_train, X_train = dmatrices('Direction~Lag1+Lag2', Smarket_train, return_type = 'data
         y_test, X_test = dmatrices('Direction~Lag1+Lag2', Smarket_2005, return_type = 'datafra
         logit = sm.Logit(y_train.iloc[:,1], X_train)
         preds = logit.fit().predict(X_test)
```

```
predict_label = pd.DataFrame(np.zeros(shape=(X_test.shape[0],1)), columns = ['label']
threshold = 0.5
confusion_matrix(y_test.iloc[:,1], predict_label.iloc[:,0])
np.mean(y_test.iloc[:,1].reset_index(drop=True)==predict_label.iloc[:,0].reset_index(
```

```
Optimization terminated successfully.
         Current function value: 0.692085
         Iterations 3
```

Out[15]: 0.44047619047619047

Because 1 - 0.44 = 0.56 there is a 0.56 chance of an error.