index

March 7, 2022

1 Logistic Regression in scikit-learn - Lab

1.1 Introduction

In this lab, you are going to fit a logistic regression model to a dataset concerning heart disease. Whether or not a patient has heart disease is indicated in the column labeled `'target'`. 1 is for positive for heart disease while 0 indicates no heart disease.

1.2 Objectives

In this lab you will:

- Fit a logistic regression model using scikit-learn

1.3 Let's get started!

Run the following cells that import the necessary functions and import the dataset:

[3]:
```
# Import necessary functions
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import seaborn as sns
```

[2]:
```
# Import data
df = pd.read_csv('heart.csv')
df.head()
```

[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2			
4	57	0	0	120	354	0	1	163	1	0.6	2			

```
3   0     2       1
4   0     2       1
```

1.4 Define appropriate `X` and `y`

Recall the dataset contains information about whether or not a patient has heart disease and is indicated in the column labeled `'target'`. With that, define appropriate `X` (predictors) and `y` (target) in order to model whether or not a patient has heart disease.

[4]:
```
# Split the data into target and predictors
y = df["target"]
X = df.drop(columns = ["target"], axis = 1)
```

1.5 Normalize the data

Normalize the data (`X`) prior to fitting the model.

[5]:
```
# Your code here
X = X.apply(lambda i: (i - min(i)) / (max(i) - min(i)), axis = 0)
X.head()
```

[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	\
0	0.708333	1.0	1.000000	0.481132	0.244292	1.0	0.0	0.603053	0.0	
1	0.166667	1.0	0.666667	0.339623	0.283105	0.0	0.5	0.885496	0.0	
2	0.250000	0.0	0.333333	0.339623	0.178082	0.0	0.0	0.770992	0.0	
3	0.562500	1.0	0.333333	0.245283	0.251142	0.0	0.5	0.816794	0.0	
4	0.583333	0.0	0.000000	0.245283	0.520548	0.0	0.5	0.702290	1.0	

	oldpeak	slope	ca	thal
0	0.370968	0.0	0.0	0.333333
1	0.564516	0.0	0.0	0.666667
2	0.225806	1.0	0.0	0.666667
3	0.129032	1.0	0.0	0.666667
4	0.096774	1.0	0.0	0.666667

1.6 Train- test split

- Split the data into training and test sets
- Assign 25% to the test set
- Set the `random_state` to 0

[6]:
```
# Split the data into training and test sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25
                                                    , random_state = 0)
```

1.7 Fit a model

- Instantiate `LogisticRegression`
  - Make sure you don't include the intercept
  - set `C` to a very large number such as `1e12`
  - Use the `'liblinear'` solver
- Fit the model to the training data

[7]:
```
# Instantiate the model
logreg = LogisticRegression(fit_intercept=False, C = 1e12, solver = 'liblinear')


# Fit the model
logreg.fit(X_train, y_train)
```

[7]:
```
LogisticRegression(C=1000000000000.0, fit_intercept=False, solver='liblinear')
```

1.8 Predict

Generate predictions for the training and test sets.

[9]:
```
# Generate predictions
y_hat_train = logreg.predict(X_train)
y_hat_test = logreg.predict(X_test)
```

1.9 How many times was the classifier correct on the training set?

[11]:
```
# Your code here
equal_train = np.abs(y_train - y_hat_train)
equal_train.value_counts(normalize=True)
```

[11]:
```
0    0.854626
1    0.145374
Name: target, dtype: float64
```

1.10 How many times was the classifier correct on the test set?

[12]:
```
# Your code here
equal_test = np.abs(y_test - y_hat_test)
equal_test.value_counts(normalize=True)
```

[12]:
```
0    0.815789
1    0.184211
Name: target, dtype: float64
```

1.11 Analysis

Describe how well you think this initial model is performing based on the training and test performance. Within your description, make note of how you evaluated performance as compared to your previous work with regression.

[ ]:
```
# Your analysis here
'''
For he training set, our model can predict the target by around 85% of the time
and for the test set, this value is around 81%.
'''
```

1.12 Summary

In this lab, you practiced a standard data science pipeline: importing data, split it into training and test sets, and fit a logistic regression model. In the upcoming labs and lessons, you'll continue to investigate how to analyze and tune these models for various scenarios.