README.md

Logistic Regression in scikit-learn - Lab

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.

Objectives

In this lab you will:

- Fit a logistic regression model using scikit-learn

Let's get started!

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

```
# 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 data
df = pd.read_csv('heart.csv')
df.head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

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

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.

```
# Split the data into target and predictors
y = df['target']
```

```
X = df.drop(columns=['target'], axis=1)
```

Normalize the data

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

```
X = X.apply(lambda x : (x - x.min()) /(x.max() - x.min()), axis=0)
X.head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	age	sex	cp	trestbps	chol	fbs	restecg	thalac
0	0.708333	1.0	1.000000	0.481132	0.244292	1.0	0.0	0.6030
1	0.166667	1.0	0.666667	0.339623	0.283105	0.0	0.5	0.8854
2	0.250000	0.0	0.333333	0.339623	0.178082	0.0	0.0	0.7709
3	0.562500	1.0	0.333333	0.245283	0.251142	0.0	0.5	0.8167
4	0.583333	0.0	0.000000	0.245283	0.520548	0.0	0.5	0.7022

Train- test split

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

```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

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

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

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

```
LogisticRegression(C=1000000000000.0, class_weight=None, dual=False,
                   fit_intercept=False, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

Predict

Generate predictions for the training and test sets.

```
# Generate predictions
y_hat_train = logreg.predict(X_train)
y_hat_test = logreg.predict(X_test)
```

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

```
# We could subtract the two columns. If values or equal, difference will be zero. Th
residuals = np.abs(y_train - y_hat_train)
print(pd.Series(residuals).value_counts())
print('----------------------------------')
print(pd.Series(residuals).value_counts(normalize=True))
# 194 correct, ~ 85% accuracy
```

```
0    194
1     33
Name: target, dtype: int64
-----------------------------------
0    0.854626
1    0.145374
Name: target, dtype: float64
```

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

```
# We could subtract the two columns. If values or equal, difference will be zero. Th
residuals = np.abs(y_test - y_hat_test)
print(pd.Series(residuals).value_counts())
print('-----------------------------------')
print(pd.Series(residuals).value_counts(normalize=True))
# 62 correct, ~ 82% accuracy
```

```
0    62
1    14
Name: target, dtype: int64
-----------------------------------
0    0.815789
1    0.184211
Name: target, dtype: float64
```

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.

```
"""
Answers will vary. In this instance, our model has 85% accuracy on the train set and
You can also see that our model has a reasonably even number of False Positives and
with slightly more False Positives for both the training and testing validations.
"""
```

```
'\nAnswers will vary. In this instance, our model has 85% accuracy on the train
set and 83% on the test set. \nYou can also see that our model has a reasonably
even number of False Positives and False Negatives, \nwith slightly more False
Positives for both the training and testing validations.\n'
```

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.

Releases

No releases published

Packages

No packages published

Contributors 6

Languages

- **Jupyter Notebook** 100.0%