Cmd 1

Cmd 2

Logistic Regression Lab 1

Objectives:

1. Develop a single-variable logistic regression model.
2. Develop a multi-variable logistic regression model.

Cmd 3

```
%run ../../Includes/Classroom-Setup
```

Command took 1.18 minutes -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Mounting course-specific datasets to **/mnt/training**...
Datasets are already mounted to **/mnt/training** from **s3a://databricks-corp-training/common**

```
Out[2]: DataFrame[]
res1: Boolean = false
res2: Boolean = false
```

OK

Cmd 4

Setup

Load the Data

The `Includes/Classroom-Setup` notebook has made an aggregate table of data available to us via the Metastore associated with our workspace. We can load the data as a pandas dataframe using the cell below.

This command loads the table using the Metastore reference. The `.toPandas()` method converts the Spark DataFrame to a Pandas DataFrame. We will use the Pandas DataFrame with Scikit-Learn throughout this Module.

Cmd 5

```
ht_agg_spark_df = spark.read.table("ht_agg")
ht_agg_pandas_df = ht_agg_spark_df.toPandas()
```

▸ (1) Spark Jobs
▸ ht_agg_spark_df: pyspark.sql.dataframe.DataFrame = [device_id: string, mean_bmi: double ... 5 more fields]

Command took 1.45 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 6

Framing a Business Problem

Over the next few labs, we will use supervised machine learning to answer a new business question:

> Given a users fitness profile, can we predict the lifestyle of a user?

Like the regression problem we previously solved, our **inputs** will be fitness profile information. This is, however, a classification problem and will have a different **output**, lifestyle.

Cmd 7

The Scikit-Learn `estimator` API

Once more, we will use the sklearn **estimator** API.

The good news is that we use the exact same pattern for classification as we did for regression.

```
estimator.fit(features, target)
estimator.score(features, target)
```

Cmd 8

Demonstration

Single-Variable Logistic Regression

First, we'll import our estimator of choice, a predictor called Logistic Regression.

Cmd 9

```
from sklearn.linear_model import LogisticRegression
```

💡1

Command took 0.62 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 10

Then, we'll instantiate or create an instance of our estimator.

Cmd 11

```
lr = LogisticRegression(max_iter=10000)
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 12

Create Feature Vectors

😉 sklearn wants the shape of our data to be a matrix for our feature(s) and the shape of our target to be a vector. This is why you will see two square brackets around our feature - a matrix - and a single set of square brackets around our target - a vector.

Cmd 13

```
X = ht_agg_pandas_df[['mean_bmi']]
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 14

Create Target Vector

An additional step, not required when perform the linear regression, is necessary to encode our target vector when performing a logistic regression.

This has to do with the way the lifestyle lables are stored.

Cmd 15

```
ht_agg_pandas_df["lifestyle"].unique()
```

```
Out[11]: array(['Cardio Enthusiast', 'Athlete', 'Sedentary', 'Weight Trainer'],
      dtype=object)
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 16

⚠️ Each lifestyle is recorded as a string value.

sklearn models can only work on numerical values. For this reason, it is required to numerically encode our lifestyle values.

We will use an sklearn transformer to do this encoding.

An sklearn transformer is like an sklearn estimator except rather than using it to `.predict()` or `.score()`, we will use it to `.transform()`

```
estimator.fit(data)
estimator.transform(data)
```

Cmd 17

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
lifestyle = ht_agg_pandas_df['lifestyle']
le.fit(lifestyle)
y = le.transform(lifestyle)
```

Command took 0.03 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 18

Fit the Model

Next, fit our model, using the same `.fit(feature, target)` pattern we learned earlier.

The model will learn the relationship between features and target, i.e. we will "train or fit the model".

Cmd 19

```
lr.fit(X, y)
```

```
Out[13]: LogisticRegression(max_iter=10000)
```

Command took 0.32 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 20

Evaluate the model

Finally, use the `.score()` method to evaluate the single-variable model.

Note that a classifier estimator in sklearn uses accuracy for scoring by default.

Cmd 21

```
lr.score(X, y)
```

```
Out[14]: 0.417
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 22

Your Turn

Exercise 1: Single-Variable Logistic Regression

Fit a single-variable logistic model for each of the remaining feature.

1. prepare a feature matrix for each of these features:
   - `mean_bmi`
   - `mean_active_heartrate`
   - `mean_resting_heartrate`

- `mean_vo2`

2. fit a single-variable logistic model for each of these features
3. evaluate using `.score()` each of these models and print the result

Cmd 23

```
# ANSWER
X_bmi = ht_agg_pandas_df[['mean_bmi']]
X_active_heartrate = ht_agg_pandas_df[['mean_active_heartrate']]
X_resting_heartrate = ht_agg_pandas_df[['mean_resting_heartrate']]
X_vo2 = ht_agg_pandas_df[['mean_vo2']]

lr_bmi = LogisticRegression(max_iter=10000)
lr_active_heartrate = LogisticRegression(max_iter=10000)
lr_resting_heartrate = LogisticRegression(max_iter=10000)
lr_vo2 = LogisticRegression(max_iter=10000)

lr_bmi.fit(X_bmi, y)
lr_active_heartrate.fit(X_active_heartrate, y)
lr_resting_heartrate.fit(X_resting_heartrate, y)
lr_vo2.fit(X_vo2, y)

print("bmi:               ", lr_bmi.score(X_bmi, y))
print("active_heartrate:  ", lr_active_heartrate.score(X_active_heartrate, y))
print("resting_heartrate: ", lr_resting_heartrate.score(X_resting_heartrate, y))
print("vo2:               ", lr_vo2.score(X_vo2, y))
```

```
bmi:               0.417
active_heartrate:  0.576
resting_heartrate: 0.586
vo2:               0.557
```

Command took 1.24 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 24

Question: Which of these single-variable models is the best at predicting lifestyle?

Cmd 25

Demonstration

Multiple-Variable Logistic Regression

Our next set of models will use more that one feature and but still have a single target.

Cmd 26

Display results from previous models

Before we train this new model, let's display the results from the previous models for comparison.

Cmd 27

```
print("bmi:               ", lr_bmi.score(X_bmi, y))
print("active_heartrate:  ", lr_active_heartrate.score(X_active_heartrate, y))
print("resting_heartrate: ", lr_resting_heartrate.score(X_resting_heartrate, y))
print("vo2:               ", lr_vo2.score(X_vo2, y))
```

```
bmi:               0.417
active_heartrate:  0.576
resting_heartrate: 0.586
vo2:               0.557
```

Command took 0.05 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 28

```
X_bmi_act_hr = ht_agg_pandas_df[['mean_bmi', 'mean_active_heartrate']]
lr_bmi_act_hr = LogisticRegression(max_iter=10000)
lr_bmi_act_hr.fit(X_bmi_act_hr, y)
print("bmi_act_hr: ", lr_bmi_act_hr.score(X_bmi_act_hr, y))
```

```
bmi_act_hr:  0.5826666666666667
```

Command took 0.58 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:36 PM on My Cluster

Cmd 29

Your Turn

Exercise 2: Multi-Variable Logistic Regression

😎 Note that this two feature model performs better than any of the single feature models.

Fit four multiple-variable logistic models.

1. prepare a feature matrix
2. fit a logistic model for each of feature matrix
3. evaluate each model using `.score()` and print the result

🦸 Did you try any models with more than two features? Multiple-variable logistic regression models can use any or all of the features.

Cmd 30

```
# ANSWER
X_1 = ht_agg_pandas_df[['mean_active_heartrate', 'mean_resting_heartrate']]
X_2 = ht_agg_pandas_df[['mean_active_heartrate', 'mean_vo2']]
X_3 = ht_agg_pandas_df[['mean_active_heartrate', 'mean_bmi', 'mean_vo2']]
X_4 = ht_agg_pandas_df[['mean_active_heartrate', 'mean_bmi', 'mean_vo2', 'mean_resting_heartrate']]

lr_1 = LogisticRegression(max_iter=10000)
lr_2 = LogisticRegression(max_iter=10000)
lr_3 = LogisticRegression(max_iter=10000)
lr_4 = LogisticRegression(max_iter=10000)
```

```
11
12 lr_1.fit(X_1, y)
13 lr_2.fit(X_2, y)
14 lr_3.fit(X_3, y)
15 lr_4.fit(X_4, y)
16
17 print("model 1: ", lr_1.score(X_1, y))
18 print("model 2: ", lr_2.score(X_2, y))
19 print("model 3: ", lr_3.score(X_3, y))
20 print("model 4: ", lr_4.score(X_4, y))
```

```
model 1:  0.605
model 2:  0.5963333333333334
model 3:  0.6006666666666667
model 4:  0.6043333333333333
```

Command took 6.86 seconds -- by tjamesbu@gmail.com at 4/2/2021, 12:10:37 PM on My Cluster

Cmd 31

Which of these models is the best at predicting lifestyle?

Cmd 32

© 2021 Databricks, Inc. All rights reserved.
Apache, Apache Spark, Spark and the Spark logo are trademarks of the Apache Software Foundation.

Privacy Policy | Terms of Use | Support

Shift+Enter to run