Cmd 1

databricks Academy

Cmd 2

Linear Regression Lab 1

Objectives:
1. Develop a single-variable linear regression model.
2. Develop a multi-variable linear regression model.

In this lab, we'll do a quick demonstration of single-variable and multi-variable linear regression using Python and Scikit-Learn.

After each demonstration, you'll have the opportunity to complete the exercises yourself.

Cmd 3

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

Command took 3.38 minutes -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 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.

We can load the data as a Pandas DataFrame using the cell below. 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 2.56 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 PM on My Cluster

Cmd 6

Framing a Business Problem

We have spoken frequently about the entire data science process starting with a good question.

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

> Given a users fitness profile, can we predict the average number of steps they are likely to take each day?

Here, our **inputs** will be fitness profile information and our **output** will be the average number of daily steps. The fitness profile information consists of average daily measurements of BMI, VO2, and resting and active heartrates.

We will perform supervised learning to develop a function to map these inputs to average daily steps.

Cmd 7

Scikit-Learn

Overview

One of the most popular libraries for doing machine learning in Python.

Scikit-Learn features:
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

The Scikit-Learn `estimator` API

The main API for performing machine learning with sklearn is the **estimator** API.

An estimator is any object that learns from data; it may be a
- A **predictor**
  - classification algorithm

- regression algorithm
- clustering algorithm
- A **transformer** that extracts/filters useful features from raw data

Fitting a predictor model with sklearn

- All estimator objects expose a `.fit()` method.
- For supervised learning, this looks like `predictor.fit(features, target)`

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

Evaluating a model with sklearn

- All predictor objects expose a `.score()` method
- For supervised learning, this looks like `predictor.score(features, target)`
- sklearn provides a built-in metric depending upon whether a classification or regression algorithm is being used
- For classification, `predictor.score(features, target)`, uses the accuracy metric
- For regression, `predictor.score(features, target)`, uses the R2 metric

Cmd 8

Demonstration

Single-Variable Linear Regression

Our first set of models will have a single independent variable (or single feature) and a single dependent variable (or single target).

A way to think about the relationship between feature and target is to put them both into a sentence, "for a [feature] of [value], we would predict that this user would have [value] [target]".

In our case , we might have an assumption that the feature `mean_bmi` is predictive of our target `mean_steps`, so our sentence could read:

> "For a mean BMI of 20, we would predict that this user would have 4000 mean steps."

Our intution and domain knowledge can help us discern predictive features.

Setting up Linear Regression

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

Cmd 9

```python
from sklearn.linear_model import LinearRegression
```

1

Command took 1.45 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 PM on My Cluster

Cmd 10

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

Cmd 11

```python
lr = LinearRegression()
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 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

```python
X = ht_agg_pandas_df[['mean_bmi']]
y = ht_agg_pandas_df['mean_steps']
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 PM on My Cluster

Cmd 14

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 15

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

```
Out[11]: LinearRegression()
```

Command took 0.32 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 PM on My Cluster

Cmd 16

Evaluate the model

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

Cmd 17

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

```
Out[12]: 0.206022518486479
```

Command took 0.06 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 PM on My Cluster

Cmd 18

Your Turn

Exercise 1: Single-Variable Linear Regression

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

Cmd 19

```
# 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 = LinearRegression()
lr_active_heartrate = LinearRegression()
lr_resting_heartrate = LinearRegression()
lr_vo2 = LinearRegression()

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:                0.206022518486479
active_heartrate:   0.6678701074913512
resting_heartrate:  0.6238073213314257
vo2:                0.5349213499717118
```

Command took 0.07 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 PM on My Cluster

Cmd 20

Demonstration

Multiple-Variable Linear Regression

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

We can apply similar logic in forming a sentence to describe the relationship "for a [feature1] of [value1] and a [feature2] of [value2], we would predict that this user would have [value] [target]".

e.g.

> "For a mean BMI of 20 and a mean active heartrate of 125, we would predict that this user would have 9500 mean steps."

Let's try this model out.

Cmd 21

```
ht_agg_pandas_df.mean_active_heartrate.sample()
```

```
Out[14]: 2752    134.573656
Name: mean_active_heartrate, dtype: float64
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 PM on My Cluster

Cmd 22

Display results from previous models

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

Cmd 23

```
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:                0.206022518486479
active_heartrate:   0.6678701074913512
resting_heartrate:  0.6238073213314257
vo2:                0.5349213499717118
```

Command took 0.07 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 PM on My Cluster

Cmd 24

Train new multiple-variable linear regression

Train the new model using both `mean_bmi` and `mean_active_heartrate` as predictors.

Cmd 25

```
X_bmi_act_hr = ht_agg_pandas_df[['mean_bmi', 'mean_active_heartrate']]
lr_bmi_act_hr = LinearRegression()
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:  0.7062981576686536
```

Command took 0.06 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 PM on My Cluster

Cmd 26

Your Turn

Exercise 2: Multi-Variable Linear Regression

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

Fit four multiple-variable linear models.

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

💡 **Hint:** Did you try any models with more than two features? Multiple-variable linear models can use any or all of the features.

Cmd 27

```
# 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 = LinearRegression()
lr_2 = LinearRegression()
lr_3 = LinearRegression()
lr_4 = LinearRegression()

lr_1.fit(X_1, y)
lr_2.fit(X_2, y)
lr_3.fit(X_3, y)
lr_4.fit(X_4, y)

print("model 1: ", lr_1.score(X_1, y))
print("model 2: ", lr_2.score(X_2, y))
print("model 3: ", lr_3.score(X_3, y))
print("model 4: ", lr_4.score(X_4, y))

```

```
model 1:  0.7168433557294125
model 2:  0.6876258568196303
model 3:  0.7424100264786863
model 4:  0.7831864974961588
```

Command took 0.10 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:28:02 PM on My Cluster

Cmd 28

Shift+Enter to run