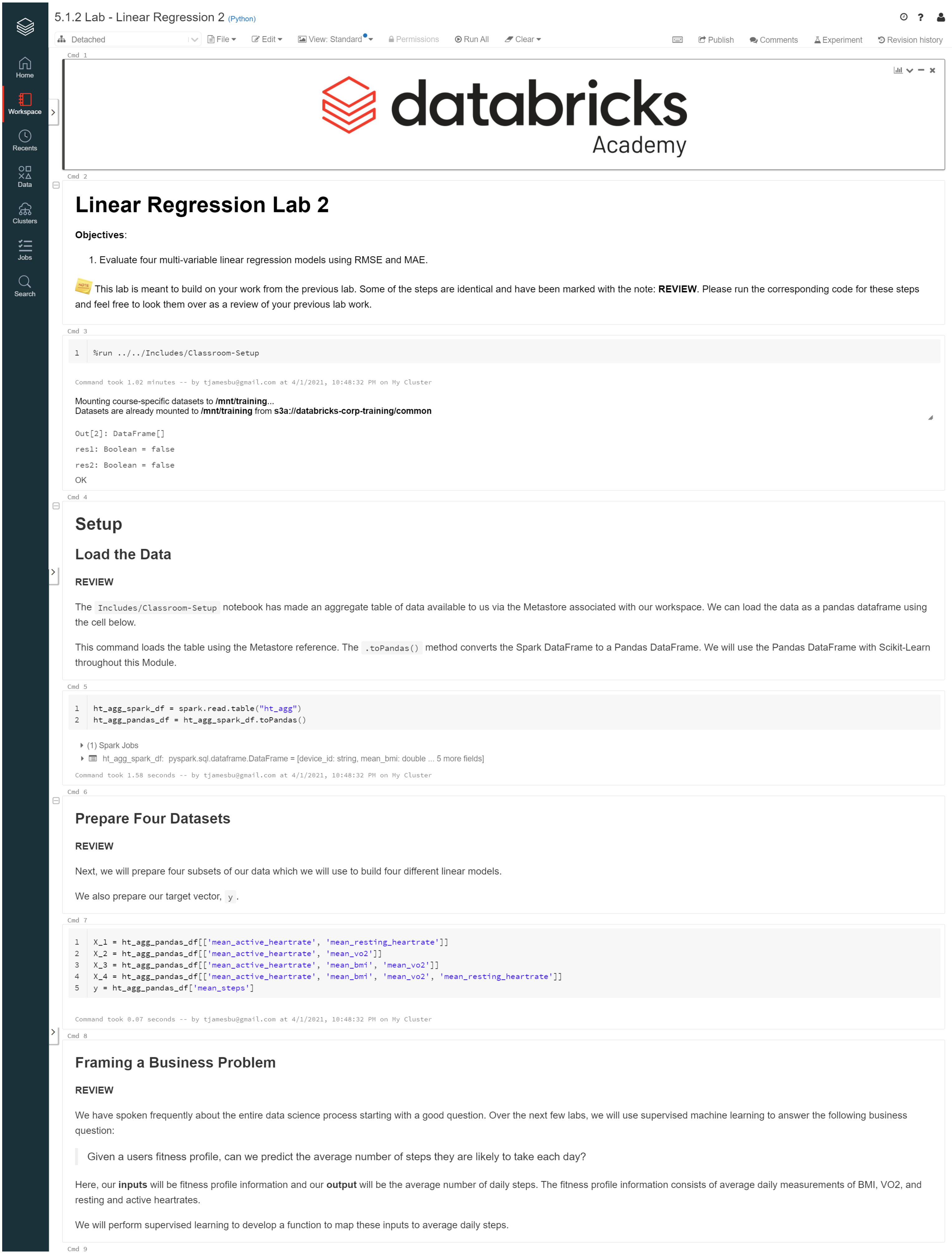

5.1.2 Lab - Linear Regression 2 (Python)

Cmd 1

databricks
Academy

Cmd 2

Linear Regression Lab 2

Objectives:

1. Evaluate four multi-variable linear regression models using RMSE and MAE.

NOTE This lab is meant to build on your work from the previous lab. Some of the steps are identical and have been marked with the note: **REVIEW**. Please run the corresponding code for these steps and feel free to look them over as a review of your previous lab work.

Cmd 3

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

```
Command took 1.02 minutes -- by tjamesbu@gmail.com at 4/1/2021, 10:48:32 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

REVIEW

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 1.58 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:48:32 PM on My Cluster
```

Cmd 6

Prepare Four Datasets

REVIEW

Next, we will prepare four subsets of our data which we will use to build four different linear models.

We also prepare our target vector, `y`.

Cmd 7

```
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']]
y = ht_agg_pandas_df['mean_steps']
```

```
Command took 0.07 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:48:32 PM on My Cluster
```

Cmd 8

Framing a Business Problem

REVIEW

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 9

Demonstration

Multi-Variable Linear Regression

REVIEW

Fit four multiple-variable linear models, one for each datasubset.

Cmd 10

```
from sklearn.linear_model import LinearRegression
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)
```

```
Out[9]: LinearRegression()
```

Command took 0.57 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:48:32 PM on My Cluster

Cmd 11

Evaluate a Multi-variable Model using RMSE and MAE

Finally, we evaulate our models. We do so using the RMSE and MAE metrics.

To use these metrics, we need to

1. generate a vector of precictions using `estimator.predict()`
2. pass actual and predicted values to the metric as `metric(actual, predicted)`
3. do this for both the ing and testing data

Cmd 12

```
from sklearn.metrics import mean_squared_error, mean_absolute_error

y_1_predicted = lr_1.predict(X_1)

print("mse: ", mean_squared_error(y, y_1_predicted))
print("mae: ", mean_absolute_error(y, y_1_predicted))
```

```
mse:  2522409.8032970093
mae:  1256.7168470610088
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:48:33 PM on My Cluster

Cmd 13

MSE vs. RMSE

Note that our metrics, mse and mae are on different scales. Let's take the square root of the mse to put them on the same scale.

Cmd 14

```
import numpy as np
rmse_1 = np.sqrt(mean_squared_error(y, y_1_predicted))
mae_1 = mean_absolute_error(y, y_1_predicted)

print("model 1: rmse: ", rmse_1)
print("model 1: mae: ", mae_1)
```

```
model 1: rmse:  1588.2096219633634
model 1: mae:  1256.7168470610088
```

Command took 0.03 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:48:33 PM on My Cluster

Cmd 15

Your Turn

Exercise 1: Generate Predictions

Perform the train-test split on the remaining data subsets:

1. use the following subsets:
   - `X_2` , `X_3` , `X_4`

Cmd 16

```
# ANSWER
y_2_predicted = lr_2.predict(X_2)
y_3_predicted = lr_3.predict(X_3)
y_4_predicted = lr_4.predict(X_4)
```

Command took 0.06 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:48:33 PM on My Cluster

Cmd 17

Exercise 2: Evaluate Our Models

1. Use the `mean_squared_error` and `mean_absolute_error` metrics
2. don't forget to take the square root of the mean squared error
3. use the following subset splits:
   - `X_2` , `X_3` , `X_4`

Cmd 18

```
# ANSWER
rmse_2 = np.sqrt(mean_squared_error(y, y_2_predicted))
mae_2 = mean_absolute_error(y, y_2_predicted)
rmse_3 = np.sqrt(mean_squared_error(y, y_3_predicted))
mae_3 = mean_absolute_error(y, y_3_predicted)
rmse_4 = np.sqrt(mean_squared_error(y, y_4_predicted))
mae_4 = mean_absolute_error(y, y_4_predicted)

print("model 1: rmse: ", rmse_1)
print("model 1: mae: ", mae_1)
print("model 2: rmse: ", rmse_2)
print("model 2: mae: ", mae_2)
print("model 3: rmse: ", rmse_3)
print("model 3: mae: ", mae_3)
print("model 4: rmse: ", rmse_4)
print("model 4: mae: ", mae_4)
```

```
model 1: rmse:  1588.2096219633634
model 1: mae:  1256.7168470610088
model 2: rmse:  1668.138029591898
model 2: mae:  1345.7471379237868
model 3: rmse:  1514.812644556499
model 3: mae:  1241.5379110431395
model 4: rmse:  1389.7529579102875
model 4: mae:  1105.374072516157
```

Command took 0.07 seconds -- by tjamesbu@gmail.com at 4/1/2021, 10:48:33 PM on My Cluster

Cmd 19

Question: Which of these models is best at predicting mean steps?

Cmd 20

Shift+Enter to run