Cmd 1

databricks Academy

Cmd 2

Model Generalization

Objectives:

1. Perform a train-test split on data.
2. Evaluate four multi-variable linear regression models using RMSE and MAE.

Additionally, you will be asked to consider overfitting and underfitting of the models based upon these results.

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 4.39 minutes -- by tjamesbu@gmail.com at 4/2/2021, 10:13:37 AM 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 2.60 seconds -- by tjamesbu@gmail.com at 4/2/2021, 10:13:37 AM on My Cluster

Cmd 6

Prepare Four Datasets

REVIEW

Next, we will prepare four subsets of our, used as in the previous lab 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 4/2/2021, 10:13:37 AM 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

Perform the Train-Test Split

Next, we will split one of our four subsets of feature data and our target data into training and testing data.

Cmd 10

```
from sklearn.model_selection import train_test_split

X_1_train, X_1_test, y_train, y_test = train_test_split(X_1, y)
```

Command took 0.54 seconds -- by tjamesbu@gmail.com at 4/2/2021, 10:13:37 AM on My Cluster

Cmd 11

Your Turn

Exercise 1: Perform the Train-Test Split

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

1. use the helper function `train_test_split`
2. split the following subsets:
   - `X_2` , `X_3` , `X_4`

Cmd 12

```
# ANSWER
X_2_train, X_2_test, y_train, y_test = train_test_split(X_2, y)
X_3_train, X_3_test, y_train, y_test = train_test_split(X_3, y)
X_4_train, X_4_test, y_train, y_test = train_test_split(X_4, y)
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/2/2021, 10:13:37 AM on My Cluster

Cmd 13

Exercise 2: Multi-Variable Linear Regression

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

Cmd 14

```
# ANSWER
from sklearn.linear_model import LinearRegression
lr_1 = LinearRegression()
lr_2 = LinearRegression()
lr_3 = LinearRegression()
lr_4 = LinearRegression()

lr_1.fit(X_1_train, y_train)
lr_2.fit(X_2_train, y_train)
lr_3.fit(X_3_train, y_train)
lr_4.fit(X_4_train, y_train)
```

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

Command took 0.12 seconds -- by tjamesbu@gmail.com at 4/2/2021, 10:13:37 AM on My Cluster

Cmd 15

Demonstration

Evaluate a Multi-variable Model using RMSE and MAE

Finally, we evaulate the training and test 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 training and testing data

Cmd 16

```
from sklearn.metrics import mean_squared_error, mean_absolute_error

y_train_1_predicted = lr_1.predict(X_1_train)
y_test_1_predicted = lr_1.predict(X_1_test)

print("training mse: ", mean_squared_error(y_train, y_train_1_predicted))
print("test mse:     ", mean_squared_error(y_test, y_test_1_predicted))
print("training mae: ", mean_absolute_error(y_train, y_train_1_predicted))
print("test mae:     ", mean_absolute_error(y_test, y_test_1_predicted))
```

```
training mse:  8992897.95158175
test mse:      8669062.119108276
training mae:  2632.8136910341573
test mae:      2570.7583049004747
```

Command took 0.06 seconds -- by tjamesbu@gmail.com at 4/2/2021, 10:13:37 AM on My Cluster

Cmd 17

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 18

```python
import numpy as np
train_1_rmse = np.sqrt(mean_squared_error(y_train, y_train_1_predicted))
test_1_rmse = np.sqrt(mean_squared_error(y_test, y_test_1_predicted))
train_1_mae = mean_absolute_error(y_train, y_train_1_predicted)
test_1_mae = mean_absolute_error(y_test, y_test_1_predicted)

print("model 1: training rmse: ", train_1_rmse)
print("model 1: training mae: ", train_1_mae)
print("model 1: test rmse:     ", test_1_rmse)
print("model 1: test mae:     ", test_1_mae)
```

```
model 1: training rmse:  2998.816091657131
model 1: training mae:  2632.8136910341573
model 1: test rmse:      2944.3271080347504
model 1: test mae:      2570.7583049004747
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/2/2021, 10:13:37 AM on My Cluster

Cmd 19

Your Turn

Exercise 3: Generate Predictions

1. use the following subset splits:
   - `X_2_test`, `X_3_test`, `X_4_test`
   - `X_2_train`, `X_3_train`, `X_4_train`

Cmd 20

```python
# ANSWER
y_train_2_predicted = lr_2.predict(X_2_train)
y_test_2_predicted = lr_2.predict(X_2_test)
y_train_3_predicted = lr_3.predict(X_3_train)
y_test_3_predicted = lr_3.predict(X_3_test)
y_train_4_predicted = lr_4.predict(X_4_train)
y_test_4_predicted = lr_4.predict(X_4_test)
```

Command took 0.06 seconds -- by tjamesbu@gmail.com at 4/2/2021, 10:13:37 AM on My Cluster

Cmd 21

Exercise 4: 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_test`, `X_3_test`, `X_4_test`
   - `X_2_train`, `X_3_train`, `X_4_train`

Cmd 22

```python
# ANSWER
train_2_rmse = np.sqrt(mean_squared_error(y_train, y_train_2_predicted))
train_2_mae = mean_absolute_error(y_train, y_train_2_predicted)
test_2_rmse = np.sqrt(mean_squared_error(y_test, y_test_2_predicted))
test_2_mae = mean_absolute_error(y_test, y_test_2_predicted)

train_3_rmse = np.sqrt(mean_squared_error(y_train, y_train_3_predicted))
train_3_mae = mean_absolute_error(y_train, y_train_3_predicted)
test_3_rmse = np.sqrt(mean_squared_error(y_test, y_test_3_predicted))
test_3_mae = mean_absolute_error(y_test, y_test_3_predicted)

train_4_rmse = np.sqrt(mean_squared_error(y_train, y_train_4_predicted))
train_4_mae = mean_absolute_error(y_train, y_train_4_predicted)
test_4_rmse = np.sqrt(mean_squared_error(y_test, y_test_4_predicted))
test_4_mae = mean_absolute_error(y_test, y_test_4_predicted)

print("model 1: training rmse: ", train_1_rmse)
print("model 1: training mae: ", train_1_mae)
print("model 1: test rmse:     ", test_1_rmse)
print("model 1: test mae:     ", test_1_mae)
print("model 2: training rmse: ", train_2_rmse)
print("model 2: training mae: ", train_2_mae)
print("model 2: test rmse:     ", test_2_rmse)
print("model 2: test mae:     ", test_2_mae)
print("model 3: training rmse: ", train_3_rmse)
print("model 3: training mae: ", train_3_mae)
print("model 3: test rmse:     ", test_3_rmse)
print("model 3: test mae:     ", test_3_mae)
print("model 4: training rmse: ", train_4_rmse)
print("model 4: training mae: ", train_4_mae)
print("model 4: test rmse:     ", test_4_rmse)
print("model 4: test mae:     ", test_4_mae)
```

```
model 1: training rmse:  2998.816091657131
model 1: training mae:  2632.8136910341573
model 1: test rmse:      2944.3271080347504
model 1: test mae:      2570.7583049004747
model 2: training rmse:  2996.369502541238
model 2: training mae:  2628.3903929471385
model 2: test rmse:      2940.738953186546
model 2: test mae:      2572.798316810965
model 3: training rmse:  2998.0402989440868
model 3: training mae:  2631.043587592155
model 3: test rmse:      2942.3870591161963
model 3: test mae:      2570.2000854660528
model 4: training rmse:  1391.566538450658
model 4: training mae:  1107.1935461167373
model 4: test rmse:      1385.0730569885372
model 4: test mae:      1108.0837486125008
```

Command took 0.07 seconds -- by tjamesbu@gmail.com at 4/2/2021, 10:13:37 AM on My Cluster

Cmd 23

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

Question: Do any of the models show signs of overfitting?

Cmd 24

Shift+Enter to run