



Objectives:

```
1 %run ../../Includes/Classroom-Setup
```

Command took 1.01 minutes -- by tjamesbu@gmail.com at 4/2/2021, 1:49:20 PM on My Cluster

Datasets are already mounted to `/mnt/training` from `s3a://databricks-corp-training/common`

```
Out[2]: DataFrame[]

res1: Boolean = false

res2: Boolean = false

OK
```

Load the Data

The `Includes/Classroom-Setup` notebook has made an aggregate table of the 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.

```
1 ht_agg_spark_df = spark.read.table("ht_agg")
2 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]

Next, we will prepare four subsets of our, used as in the previous lab to build four different logistic regression models.

We also prepare our target vector, y .

```
1 from sklearn.preprocessing import LabelEncoder
2
3 X_1 = ht_agg_pandas_df[['mean_active_hearttrate', 'mean_resting_hearttrate']]
4 X_2 = ht_agg_pandas_df[['mean_active_hearttrate', 'mean_vo2']]
5 X_3 = ht_agg_pandas_df[['mean_active_hearttrate', 'mean_bmi', 'mean_vo2']]
6 X_4 = ht_agg_pandas_df[['mean_active_hearttrate', 'mean_bmi', 'mean_vo2', 'mean_resting_hearttrate']]
7
8 le = LabelEncoder()
9 lifestyle = ht_agg_pandas_df['lifestyle']
10 le.fit(lifestyle)
11 y = le.transform(lifestyle)
```

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.

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

Cmd 10

```
1 from sklearn.model_selection import train_test_split
2
3 X_1_train, X_1_test, y_train, y_test = train_test_split(X_1, y)
```

Command took 0.84 seconds -- by t.jamesbu@gmail.com at 4/2/2021, 1:49:20 PM 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

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

Command took 0.83 seconds -- by t.jamesbu@gmail.com at 4/2/2021, 1:49:20 PM on My Cluster

Cmd 13

Exercise 2: Multi-Variable Logistic Regression

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

Cmd 14

```
1 # ANSWER
2 from sklearn.linear_model import LogisticRegression
3 lr_1 = LogisticRegression(max_iter=10000)
4 lr_2 = LogisticRegression(max_iter=10000)
5 lr_3 = LogisticRegression(max_iter=10000)
6 lr_4 = LogisticRegression(max_iter=10000)
7
8 lr_1.fit(X_1_train, y_train)
9 lr_2.fit(X_2_train, y_train)
10 lr_3.fit(X_3_train, y_train)
11 lr_4.fit(X_4_train, y_train)
```

Out[11]: LogisticRegression(max_iter=10000)

Command took 2.63 seconds -- by t.jamesbu@gmail.com at 4/2/2021, 1:49:20 PM on My Cluster

Cmd 15

Demonstration

Evaluate a Multi-variable Model using accuracy and a confusion matrix

Finally, we evaluate our models. We do so using the accuracy metric and a confusion matrix.

To use these metrics, we need to

1. generate a vector of predictions 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

```
1 from sklearn.metrics import accuracy_score, confusion_matrix
2
3 y_train_1_predicted = lr_1.predict(X_1_train)
4 y_test_1_predicted = lr_1.predict(X_1_test)
5
6 print("training accuracy: ", accuracy_score(y_train, y_train_1_predicted))
7 print("test accuracy:      ", accuracy_score(y_test, y_test_1_predicted))
8 print("training confusion matrix")
9 print(confusion_matrix(y_train, y_train_1_predicted))
10 print("")
11 print("test confusion matrix")
12 print(confusion_matrix(y_test, y_test_1_predicted))
```

```
training accuracy:  0.3622222222222222
test accuracy:      0.332
training confusion matrix
[[ 0 628  0  0]
 [ 0 815  0  0]
 [ 0 228  0  0]
 [ 0 579  0  0]]
```

```
test confusion matrix
[[ 0 231  0  0]
 [ 0 249  0  0]
 [ 0  84  0  0]
 [ 0 186  0  0]]
```

Command took 0.84 seconds -- by t.jamesbu@gmail.com at 4/2/2021, 1:49:20 PM on My Cluster

Cmd 17

Question: What do you notice about the results?

Cmd 18

Your Turn

Exercise 3: Generate Predictions

1. use the following subset splits:

- `X_1_test`, `X_2_test`, `X_3_test`, `X_4_test`
- `X_1_train`, `X_2_train`, `X_3_train`, `X_4_train`

Cmd 19

```
1 # ANSWER
2 y_train_1_predicted = lr_1.predict(X_1_train)
3 y_test_1_predicted = lr_1.predict(X_1_test)
4 y_train_2_predicted = lr_2.predict(X_2_train)
5 y_test_2_predicted = lr_2.predict(X_2_test)
6 y_train_3_predicted = lr_3.predict(X_3_train)
7 y_test_3_predicted = lr_3.predict(X_3_test)
8 y_train_4_predicted = lr_4.predict(X_4_train)
9 y_test_4_predicted = lr_4.predict(X_4_test)
```

Command took 0.86 seconds -- by tjamesbu@gmail.com at 4/2/2021, 1:49:20 PM on My Cluster

Cmd 20

Exercise 4: Evaluate Our Models

1. Use the `accuracy_score` and `confusion_matrix` 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 21

```
1 # ANSWER
2 train_1_accuracy = accuracy_score(y_train, y_train_1_predicted)
3 train_1_conf_mat = confusion_matrix(y_train, y_train_1_predicted)
4 test_1_accuracy = accuracy_score(y_test, y_test_1_predicted)
5 test_1_conf_mat = confusion_matrix(y_test, y_test_1_predicted)
6
7 train_2_accuracy = accuracy_score(y_train, y_train_2_predicted)
8 train_2_conf_mat = confusion_matrix(y_train, y_train_2_predicted)
9 test_2_accuracy = accuracy_score(y_test, y_test_2_predicted)
10 test_2_conf_mat = confusion_matrix(y_test, y_test_2_predicted)
11
12 train_3_accuracy = accuracy_score(y_train, y_train_3_predicted)
13 train_3_conf_mat = confusion_matrix(y_train, y_train_3_predicted)
14 test_3_accuracy = accuracy_score(y_test, y_test_3_predicted)
15 test_3_conf_mat = confusion_matrix(y_test, y_test_3_predicted)
16
17 train_4_accuracy = accuracy_score(y_train, y_train_4_predicted)
18 train_4_conf_mat = confusion_matrix(y_train, y_train_4_predicted)
19 test_4_accuracy = accuracy_score(y_test, y_test_4_predicted)
20 test_4_conf_mat = confusion_matrix(y_test, y_test_4_predicted)
21
22 print("model 1: training accuracy: ", train_1_accuracy)
23 print("model 1: training confusion matrix: ")
24 print(train_1_conf_mat)
25 print(" ")
26 print("model 1: test accuracy:      ", test_1_accuracy)
27 print("model 1: test confusion matrix: ")
28 print(test_1_conf_mat)
29 print(" ")
30 print("model 2: training accuracy: ", train_2_accuracy)
31 print("model 2: training confusion matrix: ")
32 print(train_2_conf_mat)
33 print(" ")
34 print("model 2: test accuracy:      ", test_2_accuracy)
35 print("model 2: test confusion matrix: ")
36 print(test_2_conf_mat)
37 print(" ")
38 print("model 3: training accuracy: ", train_3_accuracy)
39 print("model 3: training confusion matrix: ")
40 print(train_3_conf_mat)
41 print(" ")
42 print("model 3: test accuracy:      ", test_3_accuracy)
43 print("model 3: test confusion matrix: ")
44 print(test_3_conf_mat)
45 print(" ")
46 print("model 4: training accuracy: ", train_4_accuracy)
47 print("model 4: training confusion matrix: ")
48 print(train_4_conf_mat)
49 print(" ")
50 print("model 4: test accuracy:      ", test_4_accuracy)
51 print("model 4: test confusion matrix: ")
52 print(test_4_conf_mat)
53 print(" ")
```

```
model 1: training accuracy:  0.3622222222222222
model 1: training confusion matrix:
[[ 0 628  0  0]
 [ 0 815  0  0]
 [ 0 228  0  0]
 [ 0 579  0  0]]
```

```
model 1: test accuracy:      0.332
model 1: test confusion matrix:
[[ 0 231  0  0]
 [ 0 249  0  0]
 [ 0  84  0  0]
 [ 0 186  0  0]]
```

```
model 2: training accuracy:  0.3622222222222222
```

model 2: training confusion matrix:

```
[[ 0 628  0  0]
 [ 0 815  0  0]
 [ 0 228  0  0]
 [ 0 579  0  0]]
```

Command took 0.05 seconds -- by t.jamesbu@gmail.com at 4/2/2021, 1:49:20 PM on My Cluster

Cmd 22

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

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

Cmd 23

© 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