Cmd 1

Cmd 2

Decision Tree Lab

Objective:

1. Perform a train-test split on data.
2. Evaluate four multi-variable decision tree models using accuracy and a confusion matrix.

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

Cmd 3

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

Command took 1.12 minutes -- by tjamesbu@gmail.com at 4/2/2021, 2:23:25 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

```
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]

Command took 1.00 second -- by tjamesbu@gmail.com at 4/2/2021, 2:23:25 PM on My Cluster

Cmd 6

Prepare Four Datasets and Target

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

We also prepare our target vector, `y`.

Cmd 7

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

1

Command took 0.52 seconds -- by tjamesbu@gmail.com at 4/2/2021, 2:23:26 PM on My Cluster

Cmd 8

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 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 0.04 seconds -- by tjamesbu@gmail.com at 4/2/2021, 2:23:26 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

```
# 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 0.03 seconds -- by tjamesbu@gmail.com at 4/2/2021, 2:23:26 PM on My Cluster

Cmd 13

Exercise 2: Multi-Variable Decision Tree

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

Cmd 14

```
# ANSWER
from sklearn.tree import DecisionTreeClassifier
dt_1 = DecisionTreeClassifier()
dt_2 = DecisionTreeClassifier()
dt_3 = DecisionTreeClassifier()
dt_4 = DecisionTreeClassifier()

dt_1.fit(X_1_train, y_train)
dt_2.fit(X_2_train, y_train)
dt_3.fit(X_3_train, y_train)
dt_4.fit(X_4_train, y_train)
```

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

Command took 0.16 seconds -- by tjamesbu@gmail.com at 4/2/2021, 2:23:26 PM on My Cluster

Cmd 15

Demonstration

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

Finally, we evaulate 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 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 accuracy_score, confusion_matrix

y_train_1_predicted = dt_1.predict(X_1_train)
y_test_1_predicted = dt_1.predict(X_1_test)

print("training accuracy: ", accuracy_score(y_train, y_train_1_predicted))
print("test accuracy:     ", accuracy_score(y_test, y_test_1_predicted))
print("training confusion matrix")
print(confusion_matrix(y_train, y_train_1_predicted))
print("")
print("test confusion matrix")
print(confusion_matrix(y_test, y_test_1_predicted))
```

```
training accuracy:  1.0
test accuracy:      0.2866666666666667
training confusion matrix
[[637   0   0   0]
 [  0 801   0   0]
 [  0   0 245   0]
 [  0   0   0 567]]

test confusion matrix
[[59 74 30 59]
 [66 99 43 55]
 [20 20 11 16]
 [51 81 20 46]]
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/2/2021, 2:23:26 PM on My Cluster

Cmd 17

Question: What do you notice about the results?

Cmd 18

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

```
# ANSWER
y_train_1_predicted = dt_1.predict(X_1_train)
y_test_1_predicted = dt_1.predict(X_1_test)
y_train_2_predicted = dt_2.predict(X_2_train)
y_test_2_predicted = dt_2.predict(X_2_test)
y_train_3_predicted = dt_3.predict(X_3_train)
y_test_3_predicted = dt_3.predict(X_3_test)
y_train_4_predicted = dt_4.predict(X_4_train)
y_test_4_predicted = dt_4.predict(X_4_test)
```

Command took 0.04 seconds -- by tjamesbu@gmail.com at 4/2/2021, 2:23:26 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

```
# ANSWER
train_1_accuracy = accuracy_score(y_train, y_train_1_predicted)
train_1_conf_mat = confusion_matrix(y_train, y_train_1_predicted)
test_1_accuracy = accuracy_score(y_test, y_test_1_predicted)
test_1_conf_mat = confusion_matrix(y_test, y_test_1_predicted)

train_2_accuracy = accuracy_score(y_train, y_train_2_predicted)
train_2_conf_mat = confusion_matrix(y_train, y_train_2_predicted)
test_2_accuracy = accuracy_score(y_test, y_test_2_predicted)
test_2_conf_mat = confusion_matrix(y_test, y_test_2_predicted)

train_3_accuracy = accuracy_score(y_train, y_train_3_predicted)
train_3_conf_mat = confusion_matrix(y_train, y_train_3_predicted)
test_3_accuracy = accuracy_score(y_test, y_test_3_predicted)
test_3_conf_mat = confusion_matrix(y_test, y_test_3_predicted)

train_4_accuracy = accuracy_score(y_train, y_train_4_predicted)
train_4_conf_mat = confusion_matrix(y_train, y_train_4_predicted)
test_4_accuracy = accuracy_score(y_test, y_test_4_predicted)
test_4_conf_mat = confusion_matrix(y_test, y_test_4_predicted)

print("model 1: training accuracy: ", train_1_accuracy)
print("model 1: training confusion matrix: ")
print(train_1_conf_mat)
print(" ")
print("model 1: test accuracy:     ", test_1_accuracy)
print("model 1: test confusion matrix:     ")
print(test_1_conf_mat)
print(" ")
print("model 2: training accuracy: ", train_2_accuracy)
print("model 2: training confusion matrix: ")
print(train_2_conf_mat)
print(" ")
print("model 2: test accuracy:     ", test_2_accuracy)
print("model 2: test confusion matrix:     ")
print(test_2_conf_mat)
print(" ")
print("model 3: training accuracy: ", train_3_accuracy)
print("model 3: training confusion matrix: ")
print(train_3_conf_mat)
print(" ")
print("model 3: test accuracy:     ", test_3_accuracy)
print("model 3: test confusion matrix:     ")
print(test_3_conf_mat)
print(" ")
print("model 4: training accuracy: ", train_4_accuracy)
print("model 4: training confusion matrix: ")
print(train_4_conf_mat)
print(" ")
print("model 4: test accuracy:     ", test_4_accuracy)
print("model 4: test confusion matrix:     ")
print(test_4_conf_mat)
print(" ")
```

```
model 1: training accuracy:  1.0
model 1: training confusion matrix:
[[637   0   0   0]
 [  0 801   0   0]
 [  0   0 245   0]
 [  0   0   0 567]]

model 1: test accuracy:      0.2866666666666667
model 1: test confusion matrix:
[[59 74 30 59]
 [66 99 43 55]
 [20 20 11 16]
 [51 81 20 46]]

model 2: training accuracy:  1.0
model 2: training confusion matrix:
[[637   0   0   0]
 [  0 801   0   0]
```

```
[  0   0 245   0]
[  0   0   0 567]]
```

Command took 0.06 seconds -- by tjamesbu@gmail.com at 4/2/2021, 2:23:26 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

Shift+Enter to run