Supervised & Unsupervised Learning in Spark R

Lily Cheng

2/8/2021

Step 1. Preparation

1.1 Load libraries

```
library(sparklyr)
```

```
## Warning: package 'sparklyr' was built under R version 3.6.2
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
library(dbplot)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

1.2 Make a connection to the local spark cluster.

```
sc <- spark_connect(master = "local", version = "2.3")
```

1.3 Load data to the spark cluster. file name: house_price.csv

```
df <- spark_read_csv(sc, "house_price.csv", header = TRUE)
```

Step 2. EDA

2.1 Take a look at the data.

```
glimpse(df)
```

```
## Observations: ??
```

```
## Variables: 8
## Database: spark_connection
## $ no        <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,...
## $ sold_year <int> 2009, 2007, 2016, 2002, 2014, 2008, 2000, 2004, 2017, 201...
## $ age       <int> 21, 4, 18, 13, 25, 11, 23, 15, 32, 14, 22, 12, 29, 7, 8, ...
## $ store     <int> 9, 2, 3, 2, 5, 9, 5, 8, 5, 3, 4, 6, 3, 6, 4, 8, 8, 2, 4, ...
## $ distance  <int> 6, 3, 7, 2, 8, 3, 10, 9, 5, 0, 5, 4, 9, 7, 2, 0, 9, 4, 10...
## $ x_coord   <int> 84, 86, 90, 80, 81, 88, 83, 88, 82, 90, 85, 81, 85, 81, 8...
## $ y_coord   <int> 121, 121, 120, 128, 122, 126, 128, 124, 124, 126, 127, 12...
## $ price     <int> 14264, 12032, 13560, 12029, 14157, 14287, 13721, 14010, 1...
```

2.2 Check missing value for all features.

```
df %>% mutate(age = ifelse(is.na(age), "missing", age),
              store = ifelse(is.na(store), "missing", store))
```

```
## # Source: spark<?> [?? x 8]
##       no sold_year age   store distance x_coord y_coord price
##    <int>     <int> <chr> <chr>    <int>   <int>   <int> <int>
##  1     0      2009 21    9            6      84     121 14264
##  2     1      2007 4     2            3      86     121 12032
##  3     2      2016 18    3            7      90     120 13560
##  4     3      2002 13    2            2      80     128 12029
##  5     4      2014 25    5            8      81     122 14157
##  6     5      2008 11    9            3      88     126 14287
##  7     6      2000 23    5           10      83     128 13721
##  8     7      2004 15    8            9      88     124 14010
##  9     8      2017 32    5            5      82     124 14689
## 10     9      2017 14    3            0      90     126 12670
## # ... with more rows
```

2.3 Drop the first column of the data.

```
df1 <- df %>%
  select(-no)
```

2.4 Split data into train and test (size = 0.05) set.

```
data_splits <- sdf_random_split(df1, training = 0.95, testing = 0.05, seed = 42)
df_train <- data_splits$training
df_test <- data_splits$testing
```

2.5 Get numerical summaries of age and distance of the training data.

```
sdf_describe(df_train, cols = c("age", "distance"))
```

```
## # Source: spark<?> [?? x 3]
##   summary age                distance
##   <chr>   <chr>              <chr>
## 1 count   4771               4771
## 2 mean    18.885768182770907 4.911548941521693
## 3 stddev  11.347556222101494 3.146015870944592
## 4 min     0                  0
```

```
## 5 max     38                 10
```

Step 3. Supervised Learning

3.1 Fit training data with decision tree.

```
lr <- ml_decision_tree(
  df_train, price ~ .
)
```

3.2 Make prediction on test data.

```
pred <- ml_predict(lr, df_test)
```

3.3 Evaluate the prediction (report r2).

```
ml_regression_evaluator(pred, label_col = "price", metric_name = "r2")
```

```
## [1] 0.8207503
```

Step 4. Pipelines

4.1 Build a transformer for turning all features into vectors.

```
ft_vector_assembler(
    input_cols = c("sold_year", "age", "store",
                   "distance", "x_coord", 'y_coord'),
    output_col = "features"
  )
```

4.2 Build an estimator that estimates feature mean and standard deviation.

```
ft_standard_scaler(input_col = "features", output_col = "features_scaled",
                     with_mean = TRUE)
```

4.3 Create a pipeline with transformer, estimator, and decision tree.

```
pipeline <- ml_pipeline(sc)%>%
  ft_vector_assembler(
    input_cols = c("sold_year", "age", "store",
                   "distance", "x_coord", 'y_coord'),
    output_col = "features"
  )%>%
  ft_standard_scaler(input_col = "features", output_col = "features_scaled",
                     with_mean = TRUE) %>%
  ml_decision_tree_regressor(features_col = "features_scaled",
                             label_col = "price"
  )
```

4.4 Fit the pipeline model with training data.

```
pipeline_model <- ml_fit(pipeline, df_train)
```

4.5 Make prediction on test data using the pipeline model and evaluate the prediction (report r2).

```
pred <- ml_predict(pipeline_model, df_test)
ml_regression_evaluator(pred, label_col = "price", metric_name = "r2")
```

```
## [1] 0.8207503
```

4.6 Save pipeline model.

```
model_dir <- file.path("spark_model")
ml_save(pipeline_model, model_dir, overwrite = TRUE)
```

```
## Model successfully saved.
```

4.7 Reload pipeline model.

```
model_reload <- ml_load(sc, model_dir)
```

4.8 Extract one estimator from the pipeline.

```
ml_stage(model_reload, "decision_tree_regressor")
```

```
## DecisionTreeRegressionModel (Transformer)
## <decision_tree_regressor__920d106d_f153_4d1a_8104_7581b10e399d>
##  (Parameters -- Column Names)
##   features_col: features_scaled
##   label_col: price
##   prediction_col: prediction
##  (Transformer Info)
##   depth: <function>
##   feature_importances: <function>
##   num_features:  int 6
##   num_nodes: <function>
```

4.9 Disconnect to the spark cluster

```
spark_disconnect(sc)
```

Step 5. Unsupervised Learning

5.1 Make a connection to the local spark cluster.

```
sc <- spark_connect(master = "local", version = "2.3")
```

5.2 Load `iris` dataset into spark.

```
iris_tbl <- copy_to(sc, iris, "iris", overwrite = TRUE)
```

5.3 Take a look at the data.

```
iris_tbl
```

```
## # Source: spark<iris> [?? x 5]
##    Sepal_Length Sepal_Width Petal_Length Petal_Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <chr>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with more rows
```

5.4 Check missing value for chosen features (Petal_Length, Petal_Width).

```
iris_tbl %>% mutate(Petal_Width = ifelse(is.na(Petal_Width), "missing", Petal_Width),
              Petal_Length = ifelse(is.na(Petal_Length), "missing", Petal_Length))
```

```
## # Source: spark<?> [?? x 5]
##    Sepal_Length Sepal_Width Petal_Length Petal_Width Species
##           <dbl>       <dbl> <chr>        <chr>       <chr>
##  1          5.1         3.5 1.4          0.2         setosa
##  2          4.9         3   1.4          0.2         setosa
##  3          4.7         3.2 1.3          0.2         setosa
##  4          4.6         3.1 1.5          0.2         setosa
##  5          5           3.6 1.4          0.2         setosa
##  6          5.4         3.9 1.7          0.4         setosa
##  7          4.6         3.4 1.4          0.3         setosa
##  8          5           3.4 1.5          0.2         setosa
##  9          4.4         2.9 1.4          0.2         setosa
## 10          4.9         3.1 1.5          0.1         setosa
## # ... with more rows
```

5.4 Check missing value for chosen features (Petal_Length, Petal_Width).

```
iris_tbl %>% mutate(Petal_Width = ifelse(is.na(Petal_Width), "missing", Petal_Width),
              Petal_Length = ifelse(is.na(Petal_Length), "missing", Petal_Length))
```

```
## # Source: spark<?> [?? x 5]
##    Sepal_Length Sepal_Width Petal_Length Petal_Width Species
##           <dbl>       <dbl> <chr>        <chr>       <chr>
##  1          5.1         3.5 1.4          0.2         setosa
##  2          4.9         3   1.4          0.2         setosa
```

```
##  3          4.7         3.2 1.3          0.2         setosa
##  4          4.6         3.1 1.5          0.2         setosa
##  5          5           3.6 1.4          0.2         setosa
##  6          5.4         3.9 1.7          0.4         setosa
##  7          4.6         3.4 1.4          0.3         setosa
##  8          5           3.4 1.5          0.2         setosa
##  9          4.4         2.9 1.4          0.2         setosa
## 10          4.9         3.1 1.5          0.1         setosa
## # ... with more rows
```

5.5 Fit a model using k-means model with 3 clusters.

```
kmeans_model <- iris_tbl %>%
  ml_kmeans(k = 3, features = c("Petal_Length", "Petal_Width"))
```

```
## Warning in spark_param_deprecated("compute_cost"): The 'compute_cost' parameter
## is deprecated in Spark 3.x
```

```
## Warning in spark_param_deprecated("compute_cost"): The 'compute_cost' parameter
## is deprecated in Spark 3.x
```

```
kmeans_model
```

```
## K-means clustering with 3 clusters
##
## Cluster centers:
##   Petal_Length Petal_Width
## 1     4.292593    1.359259
## 2     1.462000    0.246000
## 3     5.626087    2.047826
##
## Within Set Sum of Squared Errors =  31.41289
```

5.6 Predict the associated class.

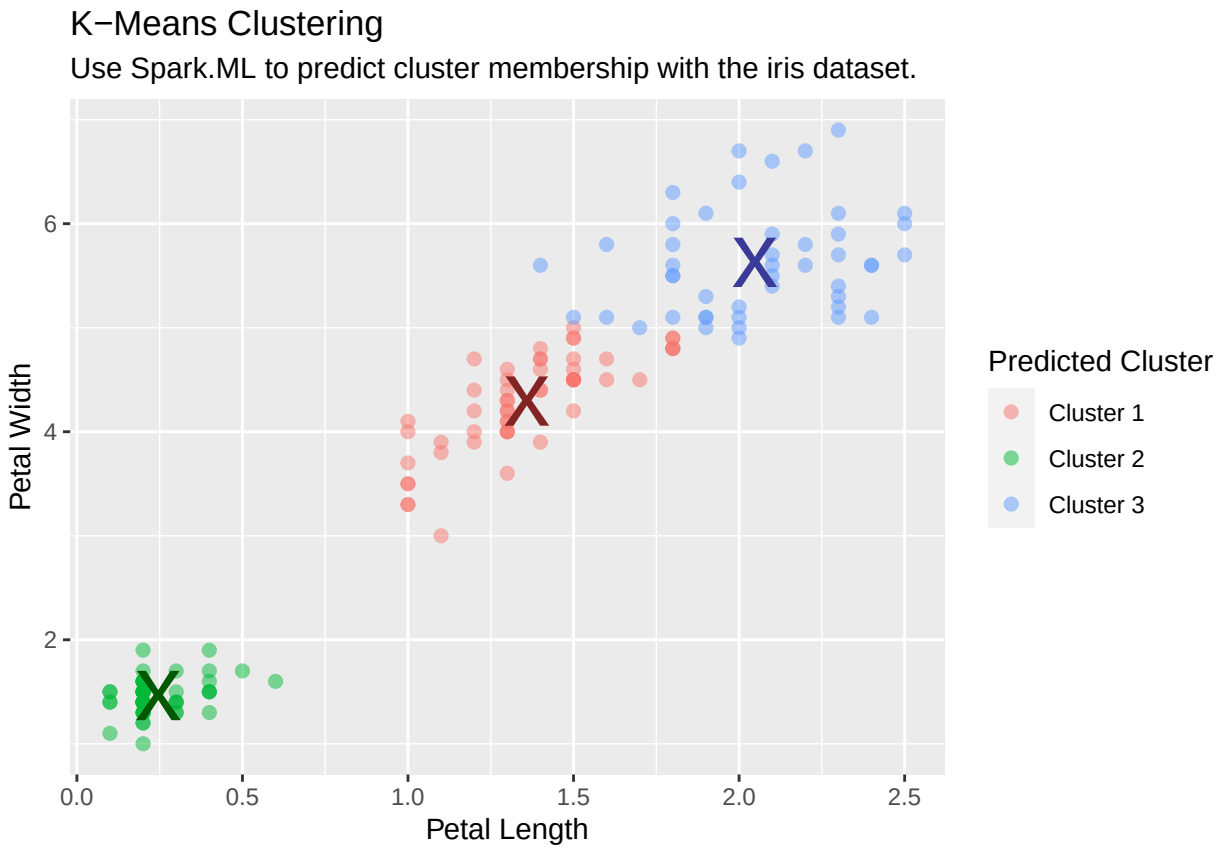
```
predicted <- ml_predict(kmeans_model, iris_tbl) %>%
  collect
table(predicted$Species, predicted$prediction)
```

```
##
##                0  1  2
##   setosa       0 50  0
##   versicolor  48  0  2
##   virginica    6  0 44
```

5.7 Plot cluster membership.

```
ml_predict(kmeans_model) %>%
  collect() %>%
  ggplot(aes(Petal_Length, Petal_Width)) +
  geom_point(aes(Petal_Width, Petal_Length, col = factor(prediction + 1)),
             size = 2, alpha = 0.5) +
  geom_point(data = kmeans_model$centers, aes(Petal_Width, Petal_Length),
             col = scales::muted(c("red", "green", "blue")),
```

```
            pch = 'x', size = 12) +
  scale_color_discrete(name = "Predicted Cluster",
                       labels = paste("Cluster", 1:3)) +
  labs(
    x = "Petal Length",
    y = "Petal Width",
    title = "K-Means Clustering",
    subtitle = "Use Spark.ML to predict cluster membership with the iris dataset."
  )
```



5.8 Disconnect to the spark cluster.

```
spark_disconnect(sc)
```