Ciencia de Datos en Spark con *sparklyr* : : GUÍA RÁPIDA

Conectar

DATABRICKS CONNECT (v2)

1. Abre tu archivo .Renviron:

```
usethis::edit_r_environ()
```

2. Enel archivo .Renviron agregue la dirección URL y el token del host de Databricks (PAT):
   - DATABRICKS_HOST = [Tu dirección URL]
   - DATABRICKS_TOKEN = [Tu token PAT]
3. Instalar extensión: `install.packages("pysparklyr")`
4. Abrir conexión:

```
sc <- spark_connect(
cluster_id = "[Your cluster's ID]",
method = "databricks_connect"
)
```

= Soportado en Databricks Connect v2

CLÚSTER INDEPENDIENTE

1. Instale RStudio Server en uno de los nodos existentes o en un servidor de la misma LAN
2. Abrir una conexión

```
spark_connect(master="spark://host:port",
   version = "3.2",
   spark_home = [path to Spark])
```

CLIENTE DE YARN

1. Instalación de RStudio Server en un nodo perimetral
2. Busque la ruta de acceso al directorio principal de Spark del clúster, normalmente es "/usr/lib/spark"
3. Ejemplo de configuración básica

```
conf <- spark_config()
conf$spark.executor.memory <- "300M"
conf$spark.executor.cores <- 2
conf$spark.executor.instances <- 3
conf$spark.dynamicAllocation.enabled<-"false"
```

4. Abrir una conexión

```
sc <- spark_connect(master = "yarn",
          spark_home = "/usr/lib/spark/",
          version = "2.1.0", config = conf)
```

CLÚSTER DE YARN

1. Asegúrese de tener copias de los archivos yarn-site.xml y hive-site.xml en el servidor RStudio
2. Apunte las variables de entorno a las rutas correctas

```
Sys.setenv(JAVA_HOME="[Path]")
Sys.setenv(SPARK_HOME ="[Path]")
Sys.setenv(YARN_CONF_DIR ="[Path]")
```

3. Abrir una conexión

```
sc <- spark_connect(master = "yarn-cluster")
```

KUBERNETES

1. Utilice lo siguiente para obtener el host y el puerto

```
system2("kubectl", "cluster-info")
```

2. Abrir connexión

```
sc <- spark_connect(config =
    spark_config_kubernetes(
     "k8s://https://[HOST]>:[PORT]",
     account = "default",
     image = "docker.io/owner/repo:version"
   ))
```

MODO LOCAL

No se requiere clúster. Usar para solo para aprendizaje

1. Instale una versión local de Spark: `spark_install()`
2. Abrir conexión

```
sc <- spark_connect(master="local")
```

NUBE

Azure - `spark_connect(method = "synapse")`

Qubole- `spark_connect(method = "qubole")`

Importar

Importar datos en Spark, no en R

LEER UN ARCHIVO EN SPARK

Argumentos que se aplican a todas las funciones:
sc, name, path, options=list(), repartition=0, memory=TRUE, overwrite=TRUE

CSV	spark_read_csv(header = TRUE, columns=NULL, infer_schema=TRUE, delimiter = ",", quote= "\"", escape = "\\", charset = "UTF-8", null_value = NULL)
JSON	spark_read_json()
PARQUET	spark_read_parquet()
TEXT	spark_read_text()
DELTA	spark_read_delta()

DE UNA TABLA

dplyr::tbl(scr, …) - Crea una referencia a la tabla sin cargar sus datos en la memoria

dbplyr::in_catalog() - Habilita una dirección de tabla de tres partes

```
x <- tbl(sc,in_catalog("catalog", "schema", "table"))
```

Importar
- Desde R (**copy_to()**)
- Un archivo (**spark_read_**)
- Una Hive table (**tbl()**)

Manipular
- Verbos **dplyr**
- Comandos **tidyr**
- Transformador de características (**ft_**)
- SQL directo de Spark (**DBI**)

Visualizar
- Recopile el resultado, grafique en R

Modelado
- Spark MLlib (**ml_**)
- Extensión H2O

Communicar
Recolecte los resultados usando R y comunique con **Quarto**

R for Data Science, Wickham, Çetinkaya-Rundel, Grolemund

MARCO DE DATOS DE R EN SPARK

dplyr::**copy_to**(dest, df, name)

Apache Arrow acelera la transferencia de datos entre R y Spark. Para usarlo, simplemente cargue la biblioteca

```
library(sparklyr)
library(arrow)
```

Manipular

VERBOS DPLYR

Se traduce en instrucciones SQL de Spark

```
copy_to(sc, mtcars) |>
    mutate(trm = ifelse(am == 0,
           "auto", "man")) |>
group_by(trm) |>
summarise_all(mean)
```

TIDYR

pivot_longer() - Contraer varias columnas en dos.

pivot_wider() - Expanda dos columnas en varias.

nest() / unnest() - Convierta grupos de celdas en columnas de lista y viceversa.

unite() / separate() - Divida una sola columna en varias columnas y viceversa.

fill() - Rellene NA con el valor anterior

TRANSFORMADORES DE CARACTERÍSTICAS

ft_binarizer() – Asigna valores basándose en un umbral

ft_bucketizer() - De columna numérica a columna discretizada

ft_count_vectorizer() - Extrae un vocabulario de un documento

ft_discrete_cosine_transform() - Transformada discreta de coseno 1D de un vector real

ft_elementwise_product() -Producto elemental entre 2 cols

ft_hashing_tf() - Asigna una secuencia de términos a sus frecuencias de términos mediante el truco de hash.

ft_idf() - Calcule la frecuencia inversa de documentos (IDF) dada una colección de documentos.

ft_imputer() - El estimador de imputación para completar los valores faltantes, utiliza la media o la mediana de las columnas.

ft_index_to_string() - Indexar etiquetas de nuevo para etiquetar como cadenas

ft_interaction() - Toma las columnas Double y Vector y genera un vector aplanado de sus interacciones de entidades.

ft_max_abs_scaler() - Cambie la escala de cada entidad individualmente al rango [-1, 1]

ft_min_max_scaler() - Cambiar la escala de cada entidad a un rango común [mín., máx.] linealmente

ft_ngram() - Convierte la matriz de cadenas de entrada en una matriz de n-gramas

ft_bucketed_random_projection_lsh()
ft_minhash_lsh() - Funciones hash sensibles a la localidad para la distancia euclidiana y la distancia de Jaccard (MinHash)

Ciencia de Datos en Spark con *sparklyr* : : GUÍA RÁPIDA

ft_normalizer() - Normalizar un vector para que tenga una norma unitaria usando la norma p dada

ft_one_hot_encoder()- Continuo a vectores binarios

ft_pca() - Proyecte vectores a un espacio dimensional inferior de los k componentes principales superiores.

ft_quantile_discretizer() - Continuo a valores categóricos agrupados.

ft_regex_tokenizer() - Extrae tokens mediante el patrón de expresiones regulares proporcionado para dividir el texto.

ft_robust_scaler() - Elimina la mediana y escala de acuerdo con la escala estándar.

ft_standard_scaler() - Elimina la media y escala a la varianza unitaria Uso de estadísticas de resumen de columnas

ft_stop_words_remover() - Filtra las palabras vacías de la entrada

ft_string_indexer() - Columna de etiquetas en una columna de índices de etiquetas.

ft_tokenizer() - Convierte a minúsculas y luego lo divide por espacios en blanco

ft_vector_assembler() - Combinar vectores en un solo vector de fila

ft_vector_indexer() - Indexación de columnas de entidades categóricas en un conjunto de datos de Vector

ft_vector_slicer() - Toma un vector de características y genera un nuevo vector de características con una submatriz de las características originales

ft_word2vec() - Word2Vec transforma una palabra en un código

Visualizar

DPLYR + GGPLOT2

```
group_by(cyl) |>
summarise(mpg_m = mean(mpg)) |>
collect() |>
ggplot() +
geom_col(aes(cyl, mpg_m))
```

Resumir en Spark

Recopilar resultados en R

Crear gráfica

Modelado

REGRESIÓN

ml_linear_regression() - Regresión lineal.

ml_aft_survival_regression() - Modelo de regresión de supervivencia paramétrico denominado modelo de tiempo de falla acelerado (AFT)

ml_generalized_linear_regression() - GLM

ml_isotonic_regression() - Utiliza el algoritmo de infractores adyacentes al grupo en paralelo.

ml_random_forest_regressor() - Regresión con bosques aleatorios.

CLASIFICACIÓN

ml_linear_svc() - Clasificación mediante máquinas de vectores de soporte lineales

ml_logistic_regression() - Regresión logística

ml_multilayer_perceptron_classifier() - Basado en el Perceptrón Multicapa.

ml_naive_bayes() - Es compatible con Multinomial NB, que puede manejar datos discretos finitamente soportados

ml_one_vs_rest() - Reducción de Multiclase, realiza la reducción utilizando la estrategia de uno contra todos.

ÁRBOL

ml_decision_tree_classifier()|ml_decision_tree()|ml_decision_tree_regressor() - Clasificación y regresión mediante árboles de decisión

ml_gbt_classifier()|ml_gradient_boosted_trees() | ml_gbt_regressor() - Clasificación binaria y regresión mediante árboles potenciados por gradiente

ml_random_forest_classifier() - Clasificación y regresión mediante bosques aleatorios.

ml_feature_importances() | ml_tree_feature_importance() - Importancia de las características para los modelos de árbol

AGRUPAMIENTO

ml_bisecting_kmeans() - Un algoritmo de bisección de k-medias basado en el artículo

ml_lda() | ml_describe_topics() | ml_log_likelihood() | ml_log_perplexity() | ml_topics_matrix() - Modelo de tema LDA diseñado para documentos de texto.

ml_gaussian_mixture() - Maximización de expectativas para modelos de mezclas gaussianas (GMM) multivariantes

ml_kmeans() | ml_compute_cost() |ml_compute_silhouette_measure() - Agrupación en clústeres con soporte para k-means

ml_power_iteration() - Para agrupar vértices de un grafo dadas similitudes por pares como propiedades de borde.

RECOMENDACIÓN

ml_als() | ml_recommend() - Recomendación mediante la factorización de matrices de mínimos cuadrados alternos

EVALUACIÓN

ml_clustering_evaluator() - Evaluador de clústeres

ml_evaluate() - Métricas de rendimiento de proceso

ml_binary_classification_evaluator() | ml_binary_classification_eval() | ml_classification_eval() - Conjunto de funciones para calcular las métricas de rendimiento de los modelos de predicción.

PATRÓN FRECUENTE

ml_fpgrowth() | ml_association_rules() | ml_freq_itemsets() - Un algoritmo paralelo de crecimiento de FP para minar conjuntos de elementos frecuentes.

ml_freq_seq_patterns() | ml_prefixspan() - Algoritmo PrefixSpan para minar conjuntos de elementos frecuentes.

ESTADÍSTICAS

ml_summary() - Extrae una métrica del objeto de resumen de un modo de Spark MLl

ml_corr() - Calcular matriz de correlación

CARACTERÍSTICA

ml_chisquare_test(x,features,label) - Prueba de independencia de Pearson para característica-etiqueta

ml_default_stop_words() - Carga las palabras vacías predeterminadas para el idioma especificado

UTILIDADES

ml_call_constructor() - Identifica el constructor de ML de Sparklyr asociado para la JVM

ml_model_data() - Extrae datos asociados a un modelo de Spark ML

ml_standardize_formula() - Genera una cadena de fórmula a partir de las entradas del usuario

ml_uid() - Extrae el UID de un objeto de ML.

Canalizaciones de ML

Cree fácilmente modelos formales de Spark Pipeline con R. Guarde la canalización en Sacala nativo. No tendrá dependencias de R.

INICIALIZAR Y ENTRENAR

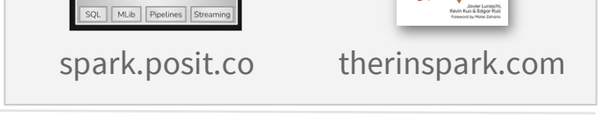
ml_pipeline() - Inicializa una nueva canalización de Spark

ml_fit() - Entrena el modelo y genera un modelo de canalización de Spark.

GUARDAR Y RECUPERAR

ml_save() - Guarda en un formato que puede ser leído por Scala y PySpark .

ml_read() - Lee el objeto Spark en sparklyr.



spark.posit.co/guides/pipelines

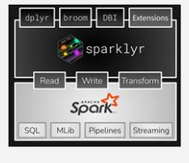
R distribuido

Ejecute código R arbitrario a escala dentro del clúster con **spark_apply()**. Útil cuando se necesita una funcionalidad que solo está disponible en R, y para resolver "problemas vergonzosamente paralelos"

spark_apply(x, f, columns = NULL, memory = TRUE, group_by = NULL, name = NULL, barrier = NULL, fetch_result_as_sdf = TRUE**)**

```
copy_to(sc, mtcars) |>
  spark_apply(
    nrow, # R only function
    group_by = "am",
    columns = "am double, x long"
    )
```

Más información



spark.posit.co

therinspark.com

posit™