Data Science in Spark with sparklyr :: CHEAT SHEET

Intro

sparklyr is an R interface for Apache Spark™. It enables us to write all of our analysis code in R, but have the actual processing happen inside Spark clusters. Easily manipulate and model large-scale using R and Spark via sparklyr.

Import

READ A FILE INTO SPARK

Arguments that apply to all functions: sc, name, path, options=list(), repartition=0, memory=True, overwrite=True:

- spark_read_csv() header = TRUE, columns=TRUE, inferSchema=TRUE, delimiter = “;”, quote = “”, escape = “\”, charset = “UTF-8”, null_value = NULL
- spark_read_json()
- spark_read_parquet()
- spark_read_orc()
- spark_read_libsvm()
- spark_read_text()
- spark_read_delta()
- spark_read_avro()

R DATA FRAME INTO SPARK

dplyr::copy_to(dest, df, name)

DPLYR VERBS

Translates into Spark SQL statements:

copy_to(sc, mtcars) %>%
mutate(trm = ifelse(am == 0, “auto”, “man”)) %>%
group_by(trm) %>%
summarise_all(mean)

TIDYR

- pivot_longer() - Collapse several columns into two.
- pivot_wider() - Expand two columns into several.
- nest() / unnest() - Convert groups of cells into list-columns, and vice versa.
- unite() / separate() - Split a single column into several columns, and vice versa.
- fill() - Fill NA with the previous value

FEATURE TRANSFORMERS

- ft_binarizer() - Assigned values based on threshold
- ft_bucketizer() - Numeric column to discretized column
- ft_count_vectorizer() - Extracts a vocabulary from document
- ft_discrete_cosine_transform() - 1D discrete cosine transform of a real vector
- ft_elementwise_product() - Element-wise product between 2 cols
- ft_hashing_tfidf() - Maps a sequence of terms to their term frequencies using the hashing trick
- ft_idf() - Compute the Inverse Document Frequency (IDF) given a collection of documents.
- ft_imputer() - Imputation estimator for completing missing values, uses the mean or the median of the columns.
- ft_index_to_string() - Index labels back to label as strings
- ft_interaction() - Takes in Double and Vector columns and outputs a flattened vector of their feature interactions
- ft_max_abs_scaler() - Rescale each feature individually to range [-1, 1]
- ft_min_max_scaler() - Rescale each feature to a common range [min, max] linearly
- ft_minhash_lsh() - Hashing functions for Euclidean distance and Jaccard distance (MinHash)
- ft_normalizer() - Project vectors to a lower dimensional space of top k principal components.
- ft_pca() - Project vectors to a lower dimensional space of top k principal components.
- ft_quantile_discretizer() - Continuous to binned categorical values.
- ft_regex_tokenizer() - Extracts tokens from input and scales according to standard scale.
- ft_robust_scaler() - Removes the median and scaling to unit variance using column summary statistics
- ft_stop_words_remover() - Filters out stop words from input
- ft_string_indexer() - Column of labels into a column of label indices.
- ft_tokenizer() - Converts to lowercase and then splits it by white spaces
- ft_vector_assembler() - Combine vectors into single row-vector
- ft_vector_indexer() - Indexing categorical feature columns in a dataset of Vector
- ft_vector_slicer() - Takes a feature vector and outputs a new feature vector with a subarray of the original features
- ft_word2vec() - Word2Vec transforms a word into a code

Visualize

- dplyr + ggplot2

- copy_to(sc, mtcars) %>%
  group_by(by=cyl) %>%
  summarise(mpg = mean(mpg)) %>%
  collect() %>%
  ggplot() +
  geom_col(aes(cyl, mpg, n))

Communicate

Collect results into R share using RMarkdown

R for Data Science, Grolemund & Wickham

RStudio is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at spark.rstudio.com • sparklyr 1.7.5 • Updated: 2022-02
Data Science in Spark with sparklyr: CHEAT SHEET

## Modeling

### REGRESSION
- `ml_linear_regression()` - Linear regression.
- `ml_aft_survival_regression()` - Parametric survival regression model named accelerated failure time (AFT) model.
- `ml_generalized_linear_regression()` - GLM
- `ml_isotonic_regression()` - Currently implemented using parallelized pool adjacent violators algorithm. Only univariate (single feature) algorithm supported
- `ml_random_forest_regression()` - Regression using random forests.

### CLASSIFICATION
- `ml_logistic_regression()` - Logistic regression
- `ml_multilayer_perceptron_classifier()` - Classification model based on the Multilayer Perceptron.
- `ml_naive_bayes()` - It supports Multinomial NB which can handle finitely supported discrete data
- `ml_one_vs_rest()` - Reduction of Multiclass Classification to Binary Classification. Performs classification to Binary Classification. Performs

### FEATURE
- `ml_chisquare_test(x,features,label)` - Pearson's independence test for every feature against the label
- `ml_default_stop_words()` - Loads the default stop words for the given language

### STATS
- `ml_summary()` - Extracts a metric from the summary object of a Spark ML model
- `ml_corr()` - Compute correlation matrix

### RECOMMENDATION
- `ml_also()` | `ml_recommend()` - Recommendation using Alternating Least Squares matrix factorization

### EVALUATION
- `ml_evaluate()` - Computes a scoring metric between the model predictions and actual values.
- `ml_binary_classification_evaluator()` - Evaluation of a binary classifier model.
- `ml_binary_classification_evaluator()` - Evaluation of a binary classifier model.
- `ml_classification_evaluator()` - Evaluation of a classifier model.
- `ml_classification_model()` - Evaluation of a classification model.
- `ml_linear_regression_model()` - Evaluation of a linear regression model.
- `ml_multilayer_perceptron_classification_model()` - Evaluation of a multilayer perceptron classification model.
- `ml_random_forest_classification_model()` - Evaluation of a random forest classification model.
- `ml_random_forest_regression_model()` - Evaluation of a random forest regression model.
- `ml_regression_model()` - Evaluation of a regression model.
- `ml_svm_classification_model()` - Evaluation of a support vector classification model.
- `ml_svm_regression_model()` - Evaluation of a support vector regression model.
- `ml_tree_classification_model()` - Evaluation of a decision tree classification model.
- `ml_tree_regression_model()` - Evaluation of a decision tree regression model.
- `ml_xgboost_classification_model()` - Evaluation of an xgboost classification model.
- `ml_xgboost_regression_model()` - Evaluation of an xgboost regression model.

### UTILITIES
- `ml_call_constructor()` - Identifies the associated sparklyr ML constructor for the JVM
- `ml_model_data()` - Extracts data associated with a Spark ML model
- `ml_standardize_formula()` - Generates a formula string from user inputs, to be used in `ml_model` constructor
- `ml_uid()` - Extracts the UID of an ML object.

### ML Pipelines

```
<table>
<thead>
<tr>
<th>Pipeline Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ml_pipeline()</td>
<td>Initializes a new Spark Pipeline</td>
</tr>
<tr>
<td>ml_fit()</td>
<td>Trains the model, outputs a Spark Pipeline Model.</td>
</tr>
</tbody>
</table>
```

### SQL AND DPLYR
- `ft_sql_transformer()` - SQL query transformer for a dplyr data frame.
- `ft_dplyr_transformer()` - Transforms Spark data frames using dplyr.
- `ft_bucketizer()` - Bins a continuous feature into a desired number of buckets.

### CHEAT SHEET

**Data Science in Spark with** sparklyr**: CHEAT SHEET**