PySpark ML 2.4 Quick Reference Guide

What is Apache Spark ML?

- Machine Learning and statistics library for the Apache Spark analytics platform
- It includes everything needed for feature engineering, model training, evaluation, selection, deployment and monitoring of production machine learning models

PySpark ML Object Types

- **Transformers:** transform the data in a data agnostic way
- **Estimators:** can fit a model based on data, but do not transform data
- **Models:** are fitted estimators and can transform data (names end in "Model")

PySpark ML Modules

- **param:** module contains utilities for reusing and storing model parameters
- **feature:** module is used for feature engineering (scaling, imputation, encoders, etc.) and automatic feature selection algorithms (ChiSqSelector)
- **classification:** module contains classification algorithms such as Support Vector Machines, Logistic Regression, Decision Trees, Gradient Boosted Trees, Random Forests, Naive Bayes, Multilayer Perceptron (MLP), and OneVsRest
- **clustering:** module contains algorithms for clustering or separating data into homogeneous groups
- **linalg:** module for performing linear algebra on vectors and matrices such as dot product, size, norm, squared distance
- **recommendation:** module containing collaborative filtering recommendation algorithms. Currently the only supported type is Alternating Least Squares.
- **regression:** module contains regression algorithms such as Linear Regression, Generalized Linear Regression, Isotonic Regression, Decision Trees, Gradient Boosted Trees, and Random Forests
- **image:** image import and analysis tools
- **stat:** module containing statistical interference tools such as ChiSquareTest, and correlation algorithms like Spearman and Pearson
- **tuning:** contains tools for tuning performance through cross validated model comparisons and parameter tuning
- **evaluation:** contains tools for assessing performance of regression, binary and multinomial classification models
- **fpm:** is a module used to mine for frequent itemsets. This module contains the FP-Growth algorithm used for affinity and market basket analysis.
- **util:** contains a number of special tools

PySpark ML parm Module

- **Params:** parameters with self-contained documentation that can be reused
- **TypeConverters:** methods for common type conversion

PySpark ML feature Module

Adaptors

- **Rformula / RformulaModel:** implements transforms required for fitting a dataset against an R model formula, using a limited subset of the R operators
- **SQLTransformer:** implements transforms defined by a SQL statement

Aggregators

- **BucketedRandomProjectionLSH / BucketedRandomProjectionLSHModel:** Performs Bucketed Locality Sensitive Hashing on a column; fitted model can be used to for finding nearest neighbours or performing similarity joins
- **HashingTF:** maps a sequence of terms to their Term Frequencies using Hashing
- **IDF / IDFModel:** Compute the Inverse Document Frequency of a collection of documents
- **MinHashLSH / MinHashLSHModel:** Locality sensitive Hashing using the Min-Hash scheme for similarity measurement

Cleaners

- **Imputer / ImputerModel:** imputation estimator for completing missing values, using various techniques such as mean or median imputation
- **StopWordsRemover:** filters out stop words from input

Discretizers

- **Binarizer:** Binarize a column of continuous features given a threshold
- **Bucketizer:** maps continuous features to feature buckets
- **QuantileDiscretizer:** takes a column with continuous features and outputs a column with binned categorical features based on approximate quantiles

Encoders

- **FeatureHasher:** Projects a set of categorical or numeric features into a feature vector
- **IndexToString:** maps a column of indices back to a new column of corresponding string values
- **Ngram:** converts the input array of strings into an array of n-grams
- **OneHotEncoderEstimator / OneHotEncoderModel:** maps a column of category indices to a column of binary vectors
- **StringIndexer / StringIndexerModel:** label indexer that maps a string column of labels to an ML column of label indices
- **Word2Vec / Word2VecModel:** trains a model to transform a word into a code for further natural language processing

Expanders and Reducers

- **DCT:** feature transformer that produces the discrete cosine transform of a vector
- **PCA / PCAModel:** Principal Components Analysis trains a model to project vectors to a lower dimensional space of the top k principal components
- **PolynomialExpansion:** performs feature expansion in a polynomial space

Feature Selectors

- **ChiSqSelector / ChiSqSelectorModel:** Chi-Squared feature selection, selects categorical features to use for predicting a categorical label

Scalers

- **ElementwiseProduct:** scales each column of the dataset by a scalar multiplier by a provided "weight" vector
- **MaxAbsScaler / MaxAbsScalerModel:** rescales each feature individually to range [-1, 1] by dividing through the largest maximum absolute value in each feature
- **MinMaxScaler / MinMaxScalerModel:** rescales each feature individually to a common range [min, max] linearly using summary statistics
- **Normalizer:** normalizes a vector to have unit norm using the given p-norm
- **StandardScaler / StandardScalerModel:** standardizes features by removing mean and scaling to unit variance using summary statistics

Tokenizers

- **RegexTokenizer:** extracts tokens using provided regex pattern once or repeatedly
- **Tokenizer:** converts the input string to lowercase and then splits by white spaces

Vectorizers

- **CountVectorizer / CountVectorizerModel:** extracts vocabulary from document collections and generates a vector representation
- **VectorAssembler:** merges multiple columns into a vector column
- **VectorIndexer / VectorIndexerModel:** indexing categorical feature columns in a dataset of Vector
- **VectorSlicer:** takes a feature vector and outputs a new feature vector with a subarray of the original features
- **VectorSizeHint:** Adds size metadata to a vector column

PySpark ML 2.4 Quick Reference Guide

PySpark ML classification Module

- **LinearSVC / LinearSVCModel:** a Linear Support Vector Machine (SVM) binary classifier
- **LogisticRegression / LogisticRegressionModel** Logistic regression supports binomial and multinomial logistic logistic regression
- **LogisticRegressionSummary:** Logistic Regression Results for a given model
- **LogisticRegressionTrainingSummary:** provides Multinomial Logistic Regression Training summary results
- **BinaryLogisticRegressionSummary:** Binary Logistic regression results for model
- **BinaryLogisticRegressionTrainingSummary:** provides iteration information on Binary Logistic Regression model training
- **DecisionTreeClassifier / DecisionTreeClassificationModel:** Decision tree learning algorithm for classification, supports binary and multiclass labels, and continuous and categorical features
- **GBTClassifier / GBTClassificationModel:** Gradient-Boosted Trees (GBTs) learning algorithm for classification, supports binary labels, and continuous and categorical features
- **RandomForestClassifier / RandomForestClassificationModel:** Random Forest learning algorithm for classification, supporting binary and multiclass labels, and continuous and categorical features
- **NaiveBayes / NaiveBayesModel:** Naive Bayes classifiers supports both Multinomial NB and Bernoulli NB
- **MultilayerPerceptronClassifier / MultilayerPerceptronClassificationModel:** Multilayer Perceptron algorithm, where each layer has sigmoid activation function, and has a softmax output layer
- **OneVsRest / OneVsRestModel:** Learning algorithm that reduces Multiclass labels to binary classification, by leveraging a one against all strategy

PySpark ML clustering Module

- **BisectingKMeans / BisectingKMeansModel:** hierarchical model iteratively divides and bisects them using k-means
- **BisectingKMeansSummary:** Bisecting K-Means clustering results for a model
- **KMeans / KMeansModel:** clustering with a k-means++ like initialization
- **GaussianMixture/GaussianMixtureModel:** performs expectation maximization for multivariate Gaussian Mixture Models
- **GaussianMixtureSummary:** Gaussian mixture clustering results for a model.
- **LDA / LDAModel:** Latent Dirichlet Allocation, a text document topic model
- **LocalLDAModel:** Local (non-distributed) model fitted by LDA
- **DistributedLDAModel:** Distributed model fitted by LDA
- **PowerIterationClustering:** Power Iteration Clustering (PIC), a scalable graph clustering algorithm

PySpark ML linalg Module

- **Vector:** allows vectors to be converted into an NumPy ndarray
- **Vectors:** Factory methods for working with vectors
- **DenseVector:** dense vector represented by a value array (stored as NumPy array)
- **SparseVector:** simple sparse vector
- **Matrix:** allows its elements in a NumPy ndarray
- **DenseMatrix:** column-major dense NumPy matrix
- **SparseMatrix:** sparse matrix stored in SciPy CSC format
- **Matrices:** factory methods for working with matricies

PySpark ML recommendation Module

- **ALS / ALSModel:** Alternating Least Squares matrix factorization, used for user recommendations

PySpark ML regression Module

- **AFTSurvivalRegression / AFTSurvivalRegressionModel:** Accelerated Failure Time Survival Regression, fits a parametric survival regression model based on the Weibull distribution of the survival time
- **DecisionTreeRegressor / DecisionTreeRegressionModel:** tree learning algorithm, supports continuous and categorical features
- **GBTRegressor /GBTRegressionModel:** Gradient-Boosted Trees learning algorithm for regression, supports continuous and categorical features
- **GeneralizedLinearRegression / GeneralizedLinearRegressionModel:** Generalized Linear Model supports link functions from the gaussian, binomial, poisson, gamma, and tweedie families
- **GeneralizedLinearRegressionSummary:** Generalized linear regression results
- **GeneralizedLinearRegressionTraining Summary:** Generalized linear regression training results
- **IsotonicRegression / IsotonicRegressionModel:** isotonic / monotonic regression constrains regression model to be non-decreasing and as close to observations as possible
- **LinearRegression / LinearRegressionModel:** Linear Regression models support multiple types of regularization including OLS, L1, L2, and L2 + L1
- **LinearRegressionSummary:**
- **LinearRegressionTrainingSummary:** Linear Regression results evaluated
- **RandomForestRegressor / RandomForestRegressionModel:** Random Forest learning algorithm for regression, supports continuous and categorical features

PySpark ML image Module

- **ImageSchema:** Access to PySpark ML image tools including the bulk reader and conversion of images to numpy array

PySpark ML stat Module

- **ChiSquareTest:** Pearson's independence test for every feature against the label, produces contingency matrix with Chi-squared statistic
- **Correlation:** correlation matrix using Pearson (default), or Spearman methods
- **KolmogorovSmirnovTest: a** two-sided Kolmogorov Smirnov (KS) test for data sampled from a continuous distribution
- **Summarizer: t**ools for vectorized statistics on MLlib Vectors

PySpark ML tuning Module

- **ParamGridBuilder:** builder for a param grid used in grid search model selection
- **CrossValidator / CrossValidatorModel:** K-fold cross validation model selection by splitting the data into non-overlapping randomly partitioned folds which are used as separate training and test datasets
- **TrainValidationSplit / TrainValidationSplitModel:** similar to CrossValidator, but only splits once

PySpark ML evaluation Module

- **BinaryClassificationEvaluator:** evaluator for binary classification using metrics such as "areaUnderPR"
- **ClusteringEvaluator:** evaluate clustering results using metrics such as "silhouette"
- **MulticlassClassificationEvaluator:** evaluator for multiclass classification using metrics such as "accuracy"
- **RegressionEvaluator:** evaluator for regression using metrics such as "r2" and "mae"

PySpark ML fpm Module

- **FPGrowth / FPGrowthModel:** FP-growth algorithm for frequent itemsets

ML Pipeline API Components

- **Transformers:** transformers are tools that transform one dataset into another
- **Estimators:** estimators are tools that fit models to data
- **Models:** tools for models that are fitted by estimators
- **Pipeline / PipelineModel:** pipeline acts as an estimator, consisting of a sequence of stages, each of which is either an Estimator or a Transformer; Pipeline.fit() executes the stages in order