

Big Data Meets Machine Learning

Apache Spark MLlib

MLlib

MLlib

Spark
Streaming

Graphx

...

Spark Dataframe

Spark Core (RDD)

Machine Learning Algorithms

- Supervised learning
 - Given a set of features and labels
 - Builds a model that predicts the label from the features
 - E.g., classification and regression
- Unsupervised learning
 - Given a set of features without labels
 - Finds interesting patterns or underlying structure
 - E.g., clustering and association mining

Overview of MLlib

- Simple primitives
- Basic Statistics
- Extractors, transformations
- Estimators
- Evaluators
- Model tuning

spark.mllib Vs spark.ml

- spark.mllib
 - RDD-based library which is now in maintenance mode
 - ~~Will be deprecated in Spark 3.x~~
 - Not recommended to use
- spark.ml
 - Dataframe-based API
 - Recommended
 - Replaces (almost) everything in the RDD-API
- Be aware when searching online on which API is used

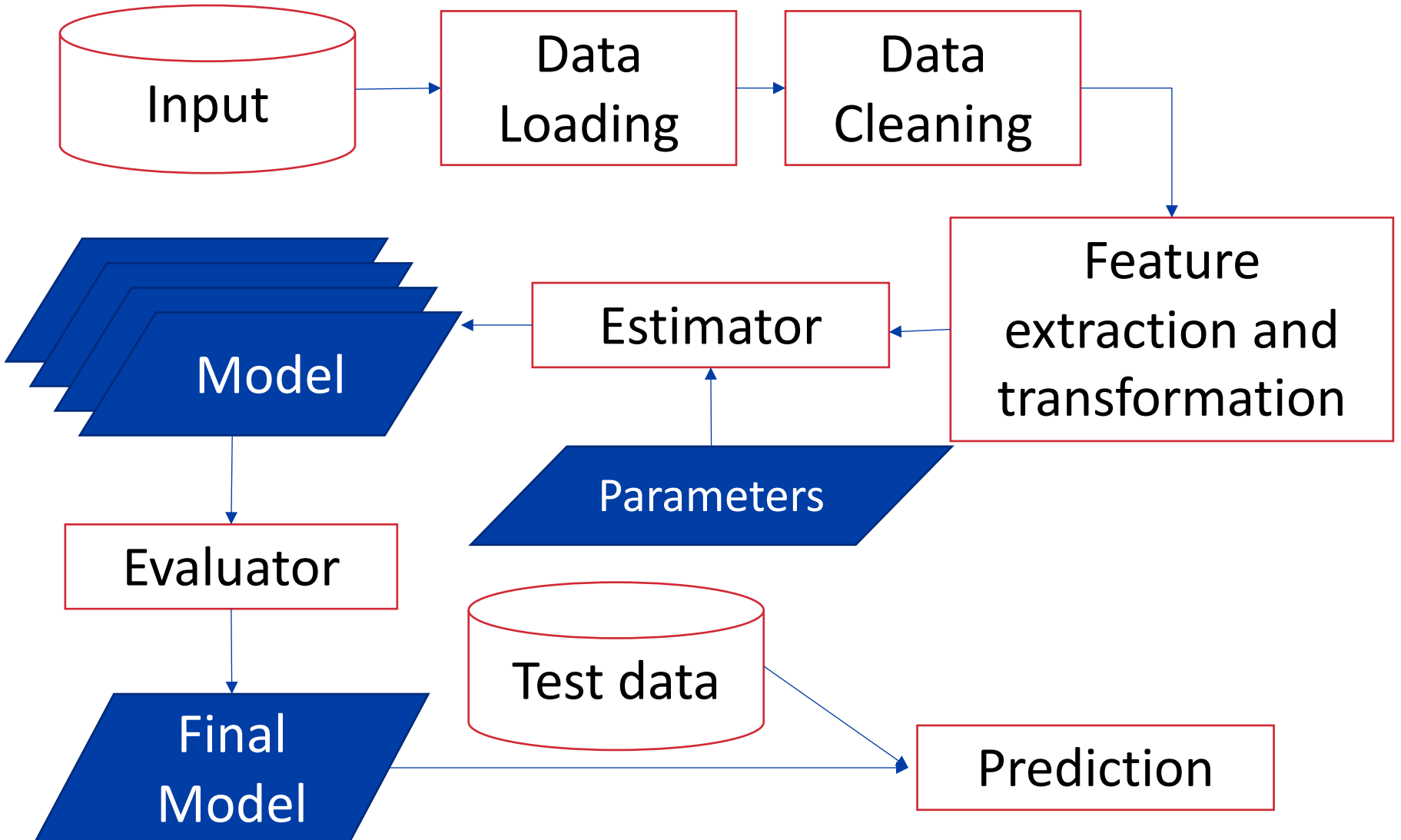
Simple Primitives

- Local Vector (Data Type)
 - To represent features
 - Example: (1.2, 0.0, 0.0, 3.4)
 - Dense vector [1.2, 0.0, 0.0, 3.4]
 - Sparse vector [0, 3], [1.2, 3.4]
- Local Matrix (Data Type)
 - Dense and Sparse
- `Dataframe.randomSplit`
 - Randomly splits an input dataset
 - Helps in building training and test sets

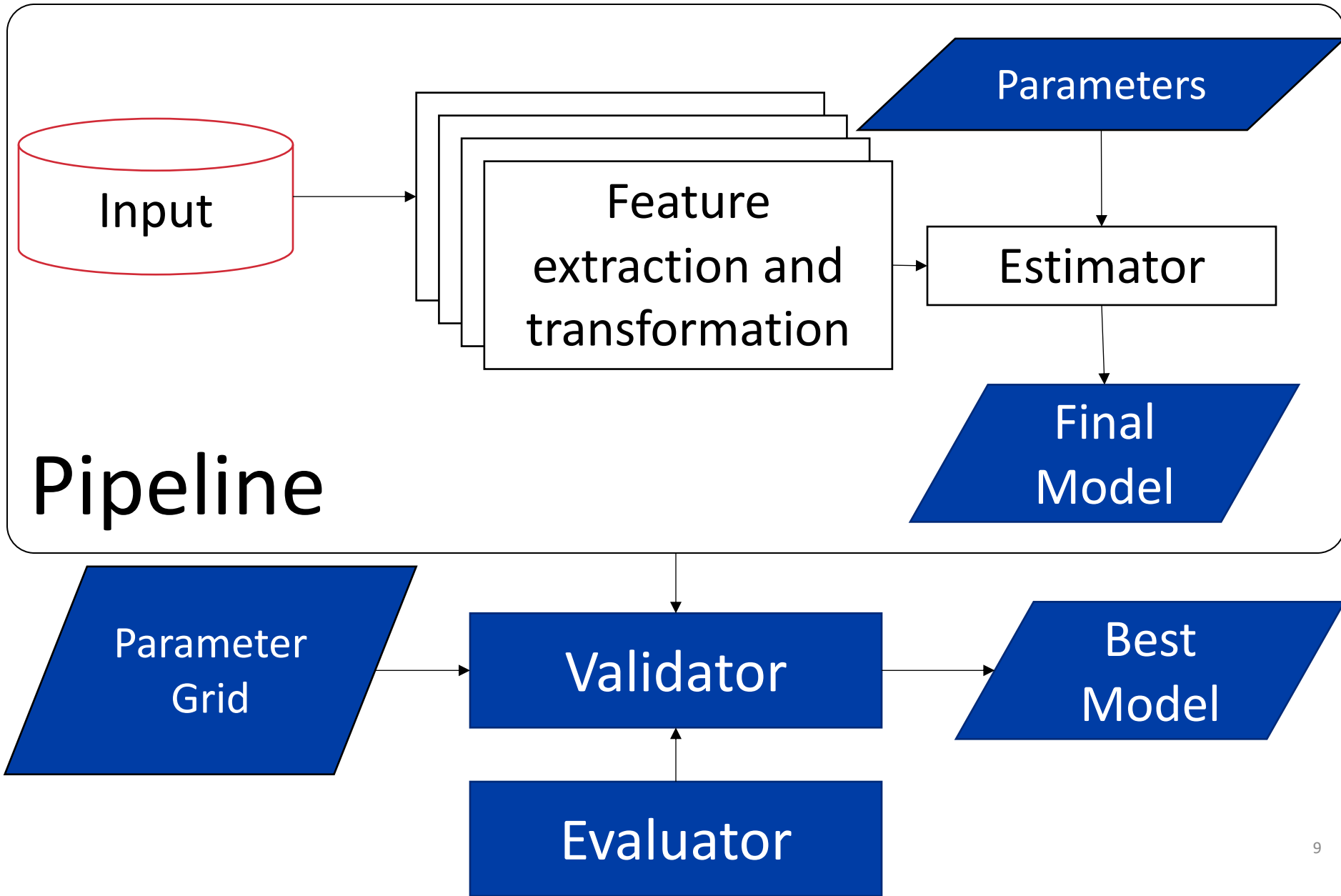
Basic Statistics

- Column statistics
 - Minimum, Maximum, count, ... etc.
- Correlation
 - Pearson's and Spearman's correlation
- Hypothesis testing
 - Chi-square Test χ^2

ML Stages



ML Pipeline



Transformations

- Used in feature extraction, dimensionality reduction, or schema transformation
- Text transformations
- Encoding
- Normalization
- Hashing

TF-IDF

- Term Frequency-Inverse Document Frequency
- A measure of the importance of a term in a document
- TF: Count of a term in a document
- DF: Number of documents that contain a term
- $IDF(t, D) = \log \frac{|D|+1}{DF(t,D)+1}$
- $TFIDF(t, D) = TF(t, d) \cdot IDF(t, D)$
- Classes: HashingTF, CountVectorizer

Word2Vec

- Converts each sequence of words to a fixed-size vector
- Similar sequences of words are supposed to be mapped to nearby vectors using this model

Other Text Transformers

- `Tokenizer`: Extracts words (tokens) from text
- `StopWordRemover`: Removes common words, e.g., a, the, an
- *n*-gram: Given a sequence of words, it generates subsequences of length *n*
- `StringIndexer`: Converts each unique string, e.g., label or class, to a numeric value
- `IndexToString`: Converts each integer value to a `String` value using a lookup table

Encoders

- PCA (Principal Component Analysis)
 - Reduces number of dimensions to a set of uncorrelated dimensions (components)
- DiscreteCosineTransform (DCT)
 - Frequency analysis
- OneHotEncoder: Converts categorical values to a vector with one bit set for the category

Numeric Transformers

- Binarizer: Converts numerical values to (0/1) based on a threshold
- Bucketizer: Converts continuous values to a set of $n+1$ buckets based on n thresholds
- QuantileDiscretizer: Places numeric values into buckets based on quantiles
- Normalizer: normalizes each vector to have unit norm. For example,
$$\begin{aligned} & [4.0 \quad 10.0 \quad 2.0] \\ & \rightarrow [0.25 \quad 0.625 \quad 0.125] \end{aligned}$$
- MinMaxScaler: Scales each feature in a vector to a standard scale, e.g., $[0.0, 1.0]$

Other Transformers

- **Imputer**: Replaces missing values by a number or the mean
- **VectorAssembler**: Combines multiple attributes into a vector attribute
- **VectorSlicer**: Extracts a subarray of a long vector
- **SQLTransformer**: Applies an SQL query on the input dataset

Applying Transformers

- Simple transformers
 - Can be applied by looking at each individual record
 - E.g., Bucketizer, or VectorAssembler
 - Applied by calling the transform method
 - E.g., `outdf = model.transform(indf)`
- Holistic transformers
 - Need to see the entire dataset first before they can work
 - e.g., MinMaxScaler, HashingTF, StringIndexer
 - To apply them, you need to call fit then transform
 - e.g., `outdf = model.fit(indf).transform(indf)`

Estimators

- An estimator is a machine learning algorithm that fits a model on the data
- Classification
 - Classifies data points into discrete points (categories)
- Regression
 - Estimates a continuous numeric
- Clustering
 - Groups similar records together into clusters
- Collaborative filtering (Recommendation)
 - Predicts (missing) user ratings for items
- Frequent Pattern Mining

Classification and regression

- Supervised learning algorithms
- Classification
 - Logistic regression
 - Decision tree
 - Naïve Bayes
 - ...
- Regression
 - Linear regression
 - Decision tree regression
 - Random forest regression
 - ...

Clustering

- Unsupervised learning method
- K-means clustering. Clustering based on distance between vectors
- Latent Dirichlet allocation (LDA). Groups vectors based on some latent (hidden) variables
- Bisecting k-means. Hierarchical clustering
- Gaussian Mixture Model (GMM). Breaks down data distribution into multiple Gaussian distributions

Evaluators

- An Evaluator takes a model and produces numeric values that measure the goodness of the model for a specific dataset
- BinaryClassificationEvaluator evaluates binary classifiers using precision, recall, F-measure, area under ROC curve, ... etc.
- MulticlassClassificationEvaluator evaluates multiclass classifiers using confusion matrix, accuracy, precision, recall ... etc.

Evaluators

- ClusteringEvaluator evaluates clustering algorithms using sum of squared distances
- RegressionEvaluator evaluates regression models using Mean Squared Error (MSE), Root Mean Squared Error (RMSE) ... etc.

Validators

- Each model has its own parameters that are usually no intuitive to tune
- A validator takes a pipeline, an evaluator, and a set of parameters and it tries all possible combinations of parameters to find the best model, i.e., the model that gives the best numeric evaluation metric
- Examples, CrossValidator and TrainValidationSplit

Further Reading

- Documentation
 - <http://spark.apache.org/docs/latest/ml-guide.html>
- MLlib paper
 - X. Meng et al, “MLlib: Machine Learning in Apache Spark”, Journal of Machine Learning Research 17:34:1-34:7 (2016)