Spark MLLib machine learning

- Connect to batch/ real time streaming sources
- Data to be cleansed and transformed into a stream, stored in memory
- Models can or cannot be built in real time depending on the requirement and availability of libraries
- Do predictions in batch/real time
- Receive, store and communicate with external system

Dataset → Pandas → Sk-learn ... → Model → Deployment
CSV
JSON
RDBMS
Pre Processing
Model building
Tuning/HPO ...
Inferencing (pred)
IoT's GPS → predictive
Streaming analytics
Spark Dataframe → Spark MLLIB → ... → Pipeline
Model

Spark ML Lib

- **Packages** – spark.mllib, spark.ml
- **Data types** – Local vector, Labelled point
- **Local vector** – vector of double values – Dense and Sparse
- e.g, a vector (1.0, 0.0, 3.0) can be represented in dense format as [1.0, 0.0, 3.0] or in sparse format as (3, [0, 2], [1.0, 3.0])
- **Labeled point** – contains the target variable(outcome) and list of features (predictors) ()
- **Process** - Load data into RDD **->** Transform RDD – Filter, Type conversion, centering, Scaling, etc. **->** Convert to labeled point **->** Split training and testing **->** Create model -> Tune -> Perform predictions

ML Pipelines

Spam / Not spam
feature engineering → TF-IDF → Logistic regression → prediction
Shortcut of complete ML pipeline in a single line of code.

- Standard APIs for machine learning algorithms to make it easier to combine multiple algorithms into a single pipeline, or workflow.
- Inspired from SciKit-learn where multiple libraries of pre-processing, feature engineering and modelling are combined to give an output in using single line of code.
- Three major parts:

1. **DataFrame** – to hold the dataset
2. **Transformer** – Apply Functions/Models to datasets (may add columns), .transform() — function applied
3. **Estimator** – Converts a dataset into a Transformer, e.g. training the datasets using .fit() → Resource intensive

action
fit() — training starts.

- Sometimes pipelines may also contain params which are the args passed to a transformer.

TF — TOS → Tensorflow on Spark
→ Horovod on Spark.

Feature engineering in Spark

- **Extraction**: Extracting features from “raw” data.
- TF-IDF - Feature vectorization method widely used in text mining to reflect the importance of a term to a document in the corpus.
- **TF**: Both HashingTF and CountVectorizer can be used to generate the term frequency vectors.
- HashingTF is a Transformer which takes sets of terms and converts those sets into fixed-length feature vectors.
- **IDF** is an Estimator which is fit on a dataset and produces an IDFModel. The IDFModel takes feature vectors (generally created from HashingTF or CountVectorizer) and scales each column.
- Word2Vec -Estimator which takes sequences of words representing documents and trains a Word2VecModel. The model maps each word to a unique fixed-size vector. The Word2VecModel transforms each document into a vector using the average of all words in the document
- CountVectorizer -CountVectorizer and CountVectorizerModel aim to help convert a collection of text documents to vectors of token counts.

https://spark.apache.org/docs/2.4.0/ml-features.html#pca

TF-IDF

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$\text{TF-IDF} = \text{TF}(t, d) \times \text{IDF}(t)$$

Term frequency

Number of times term t appears in a doc, d

Inverse document frequency

$$\log \frac{1 + n}{1 + df(d, t)} + 1$$

n ← # of documents

df(d, t) ← Document frequency of the term t

$$\log_{10} \frac{1+2}{1+2} + 1$$

$$= \log \frac{3}{3} + 1$$

$$= \log 1 + 1$$

$$= 0 + 1$$

$$= 1.$$

$$= 2 \times 1 = 2.$$

Transformation: Scaling, converting, or modifying features

- Tokenizer
- StopWordsRemover
- n-gram
- Normalizer
- StandardScaler
- Bucketizer
- SQLTransformer
- Imputer

Selection: Selecting a subset from a larger set of features

Name, id, tpw ...

> 5

(select * from df where id > 5)

- VectorSlicer - Transformer that takes a feature vector and outputs a new feature vector with a sub-array of the original features. It is useful for extracting features from a vector column.

  → rdd.map

- Rformula - selects columns specified by an R model formula. Currently supports a limited subset of the R operators, including '~', '.', ':', '+', and '-'.

  R developers working on Spark MLlib

- ChiSqSelector - It operates on labeled data with categorical features. ChiSqSelector uses the Chi-Squared test of independence to decide which features to choose.

Spam
not Spam
Not Sp
may be spam
may be NS.

Spam vs.

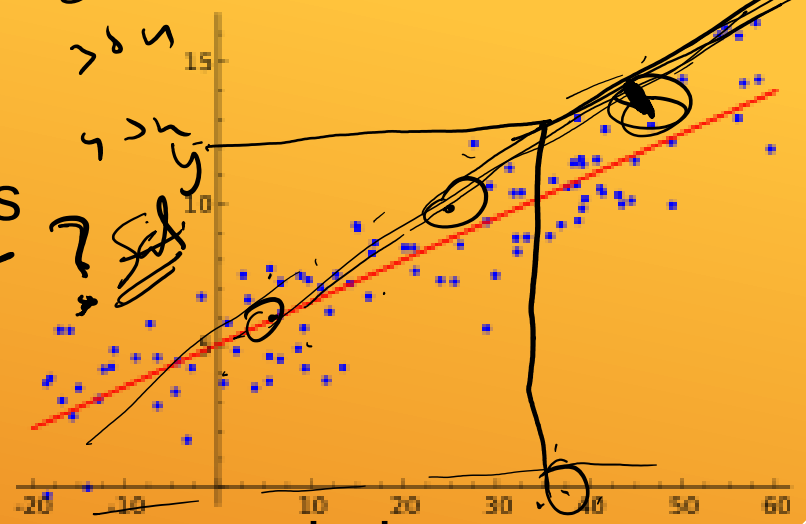
More..

- **Correlation**
- Correlation is any of a broad class of statistical relationships involving dependence, though in common usage it most often refers to how close two variables are to having a linear relationship with each other.
- Correlation computes the correlation matrix for the input Dataset of Vectors using the specified method. The output will be a DataFrame that contains the correlation matrix of the column of vectors.

- **Hypothesis testing**
- Determine whether a result is statistically significant, whether this result occurred by chance or not.

Linear regression

- Estimate value of dependent variables from the values of independent variables with some correlation.
- Draw the best line to fit plotted points.



Example -

- Convert input string to vector & drop columns which are not needed
- Create rDD vectors.dense(atList(x).toFloat)
- Create labelled vectors & drop low correlation vectors (should be continuous)
- Run LR model on training data Print coefficients, intercepts, summary, lr.fit(trainingdata)
- Predict
- Evaluate using MSE (average((predicted value – actual value) ** 2, should be low), R square should be high as close to 1 as possible

Mean Squared Error

K-Means Clustering

- Attempts to split data into K- groups that are closest to K centroids
- Un Supervised learning – uses only position of each data point
- Randomly pick K centroids -> Assign each data point to the centroid it’s closest to
- Re-compute centroids based on average position of each centroid’s points
- Iterate until points stop changing assignment to centroids
- If you want to predict the cluster for new points, just find the centroid they are closest to

Classification

- **Binary Classification** is the task of predicting a binary label. E.g., is an email spam or not spam?
- Supervised learning – uses only position of each data point
- For binary classification problems, the algorithm outputs a binary logistic regression model. Given a new data point, denoted by xx, the model makes predictions by applying the logistic function

Model selection and hyper parameter tuning

- An important task in ML is *model selection*, or using data to find the best model or parameters for a given task.
- **Estimator**: algorithm or Pipeline to tune
- Set of **ParamMaps** : parameters to choose from, sometimes called a “parameter grid” to search over

-Split the input data into separate training and test datasets.

-For each (training, test) pair, they iterate through the set of ParamMaps

-Identify the best ParamMap, CrossValidator finally re-fits the Estimator using the best ParamMap and the entire dataset.

- **Evaluator**: metric to measure how well a fitted Model does on held-out test data i.e evaluate the Model’s performance using the Evaluator.
- RegressionEvaluator , a BinaryClassificationEvaluator for binary data, or a MulticlassClassificationEvaluator

Thank you!