Distributed Data Systems

DIANE WOODBRIDGE, PH.D

Announcement

Individual Assignment

- HW 4 - March 3rd (Extended)
  - Please do not plagiarize!! We're almost towards to the end of the module.

Group Assignment

- Task 3 - March 10th

Quiz 3

- March 7 - 9 AM

Contents

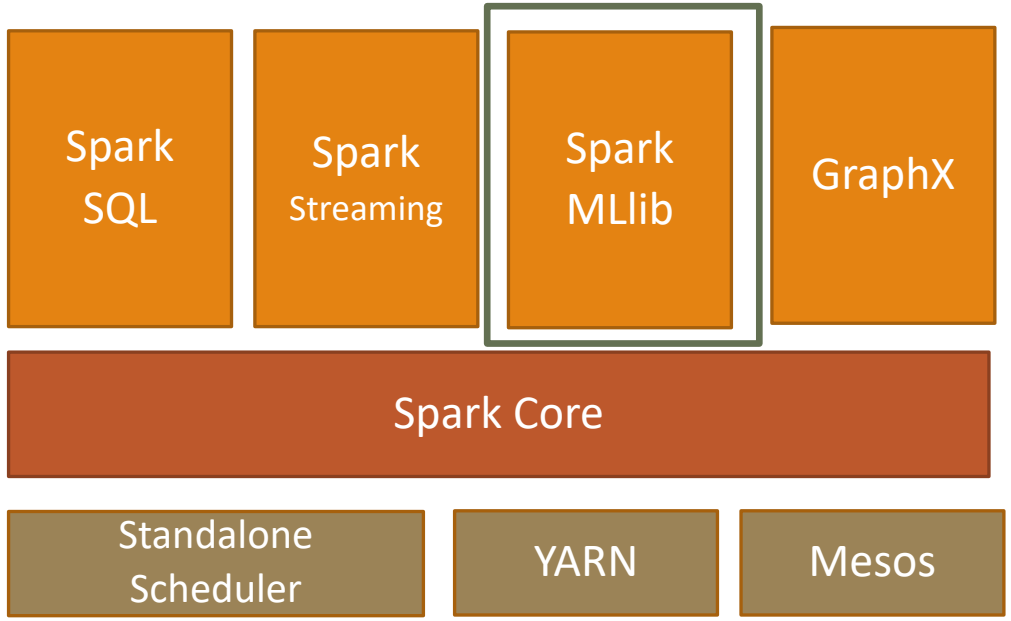
Machine Learning With Spark (Spark ML)

- Main Components
- Algorithms
- Logistic Regression
- Decision Tree
- Random Forest
- K-Mean Clustering

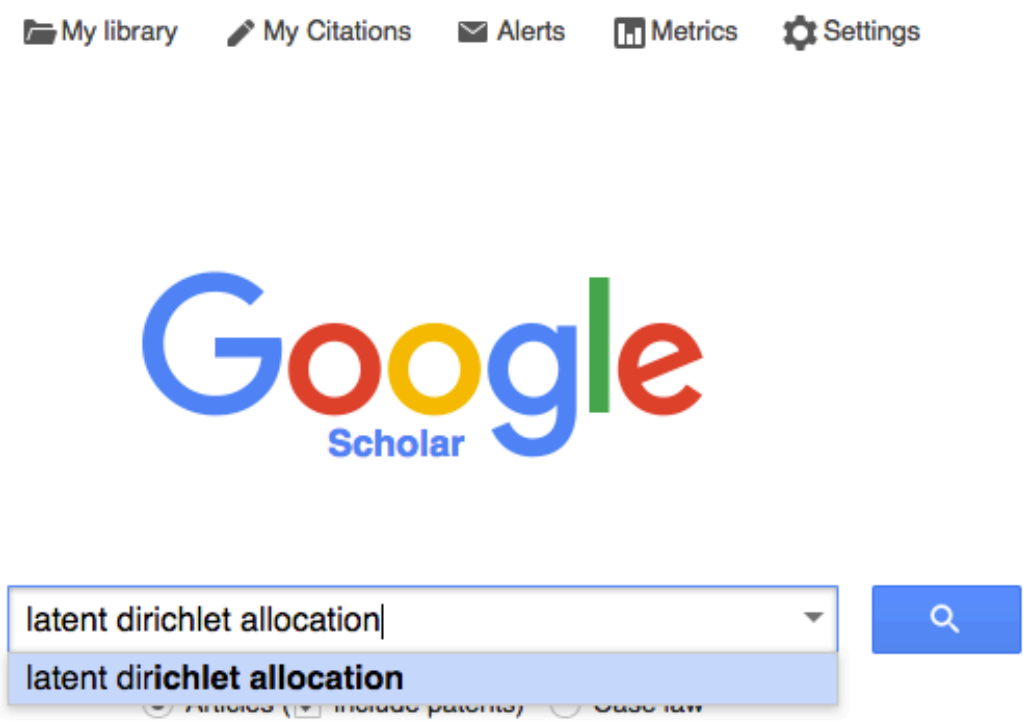
Machine Learning with Spark

Designed to run in parallel on clusters.

Spark lets you perform machine learning tasks all in the same systems, using the same API.

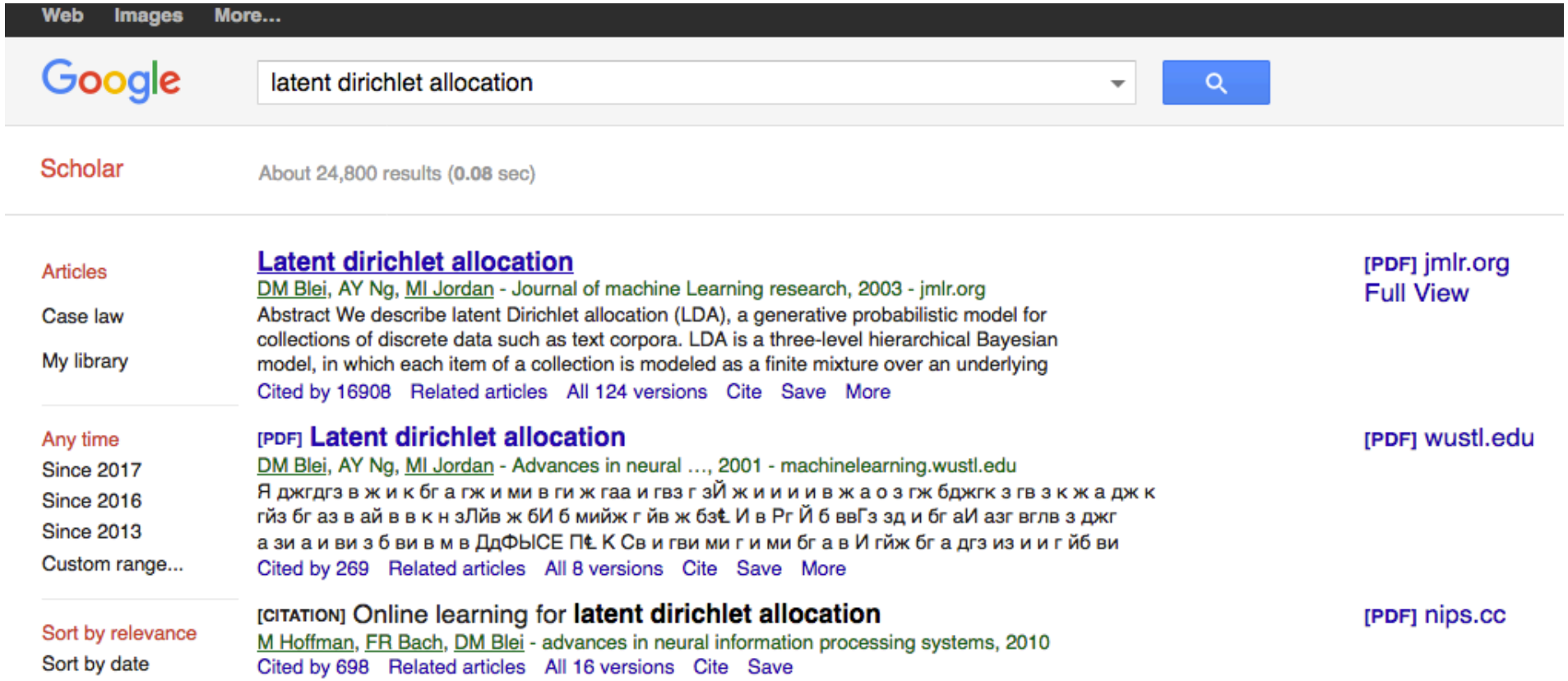


Algorithm Details?



Stand on the shoulders of giants

Algorithm Details?



Machine Learning with Spark

MLlib RDD-based API

- Use RDD.
- As of Spark 2.0, the RDD-based APIs in the spark.mllib package have entered maintenance mode. The primary Machine Learning API for Spark is now the DataFrame-based API in the spark.ml package.

MLlib DataFrame-based API

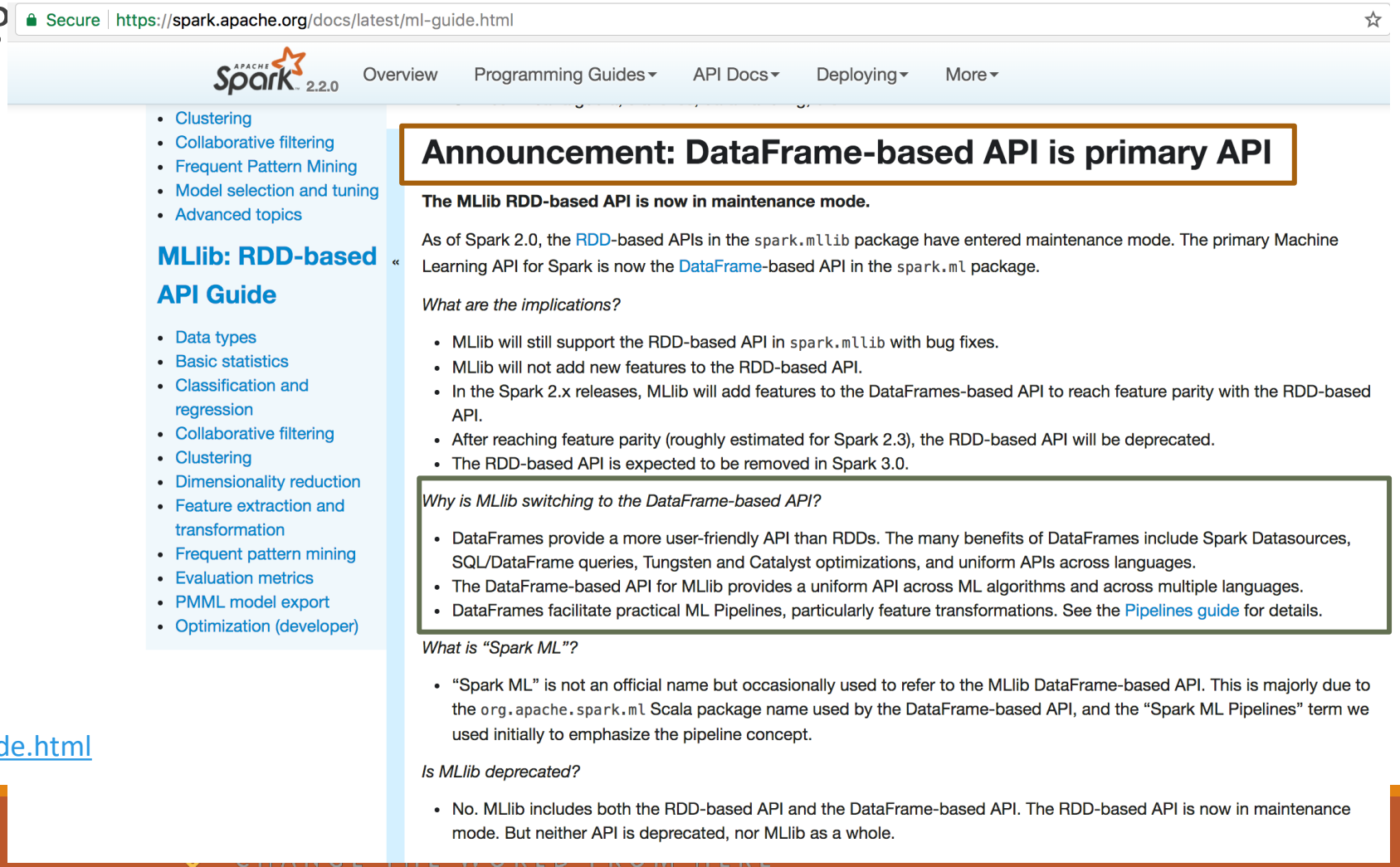
- New (Spark v 1.2).
- Support Pipelines of estimators, transformer and evaluators.
- Use DataFrame.

Some of the algorithms are not included because they were not designed for parallel platforms.

https://spark.apache.org/docs/latest/ml-guide.html

Machine Learning with Spark

RDD-based MLlib ?



https://spark.apache.org/docs/latest/ml-guide.html

Contents

Machine Learning With Spark (Spark ML)

- **Main Components**
- Algorithms
- Logistic Regression
- Decision Tree
- Random Forest
- K-Mean Clustering

Spark ML

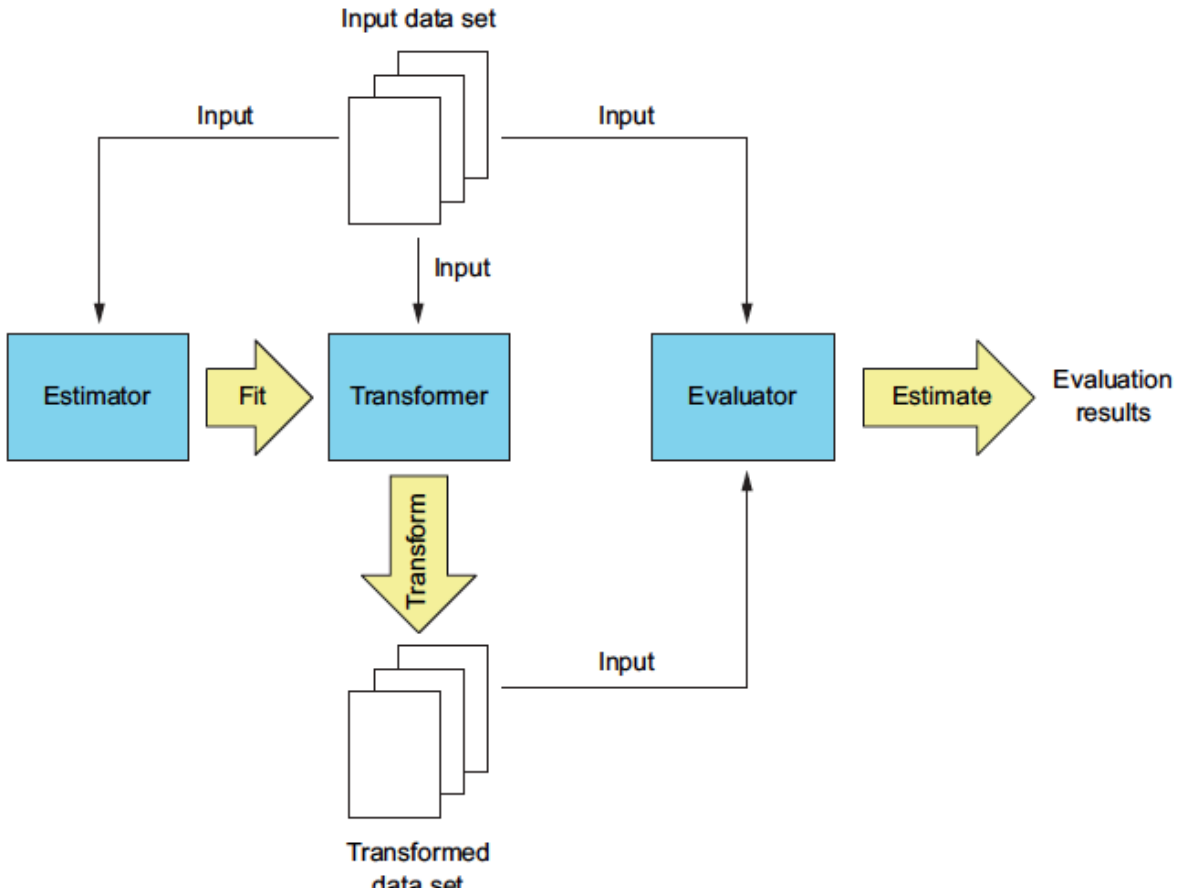
Uses DataFrame and Dataset.

- Dataset : A strongly typed collection of objects (This includes DataFrame.).

Main Components (5)

- Transformers
- Estimators
- Evaluators

- ML Parameters
- ML Pipeline



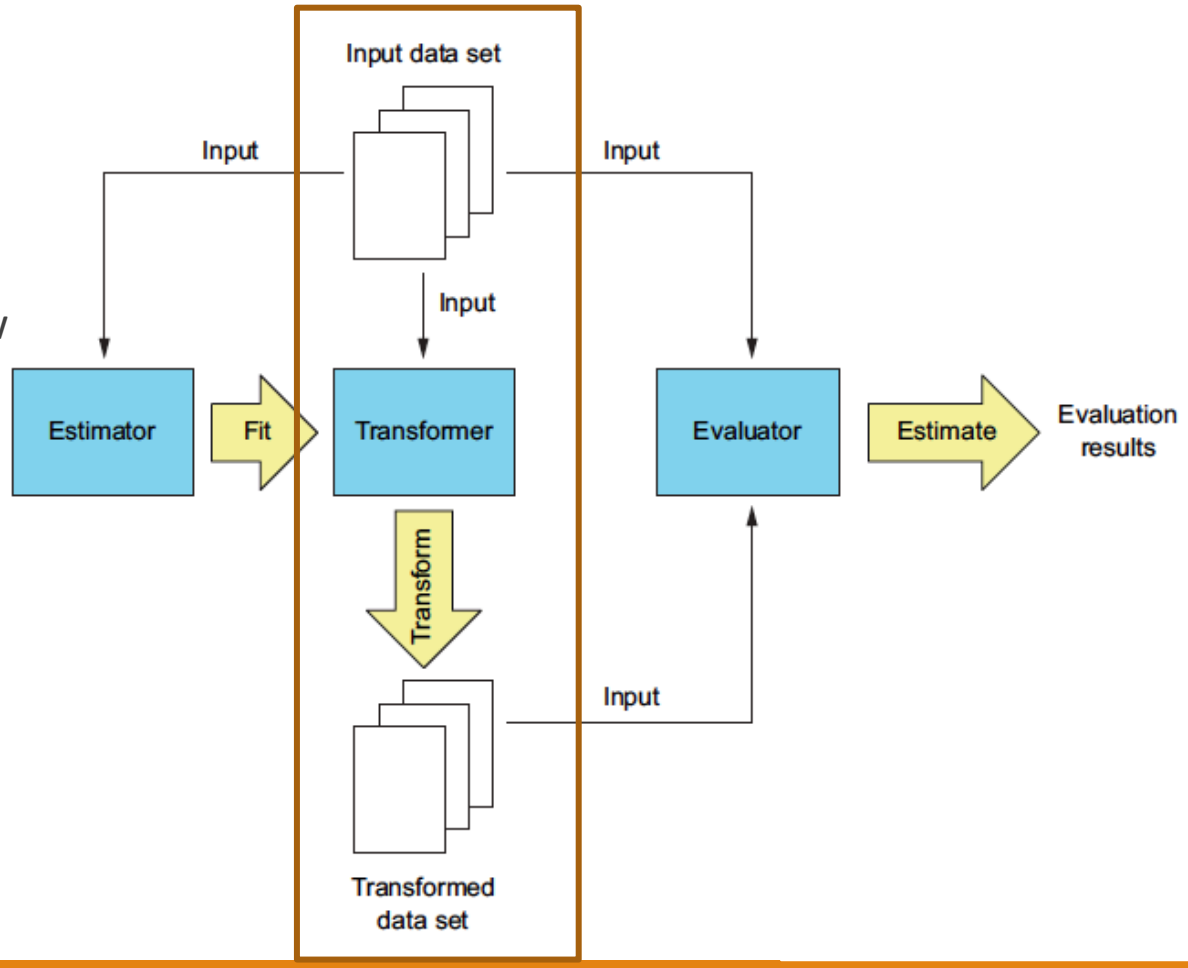
https://spark.apache.org/docs/latest/api/java/org/apache/spark/sql/Dataset.html
https://spark.apache.org/docs/latest/ml-pipeline.html

Spark ML

Main Components (5)

- **Transformers**
  - Convert a dataset to another.
  - Types
    1) Feature transformer – take a data frame. Output a data frame with new columns like feature vectors.
    2) Learning model – take a data frame and output a data frame with predicted labels.
  - **`transform()`** : takes DataFrame and optional parameters.
- Estimators
- Evaluators

- ML Parameters
- ML Pipeline

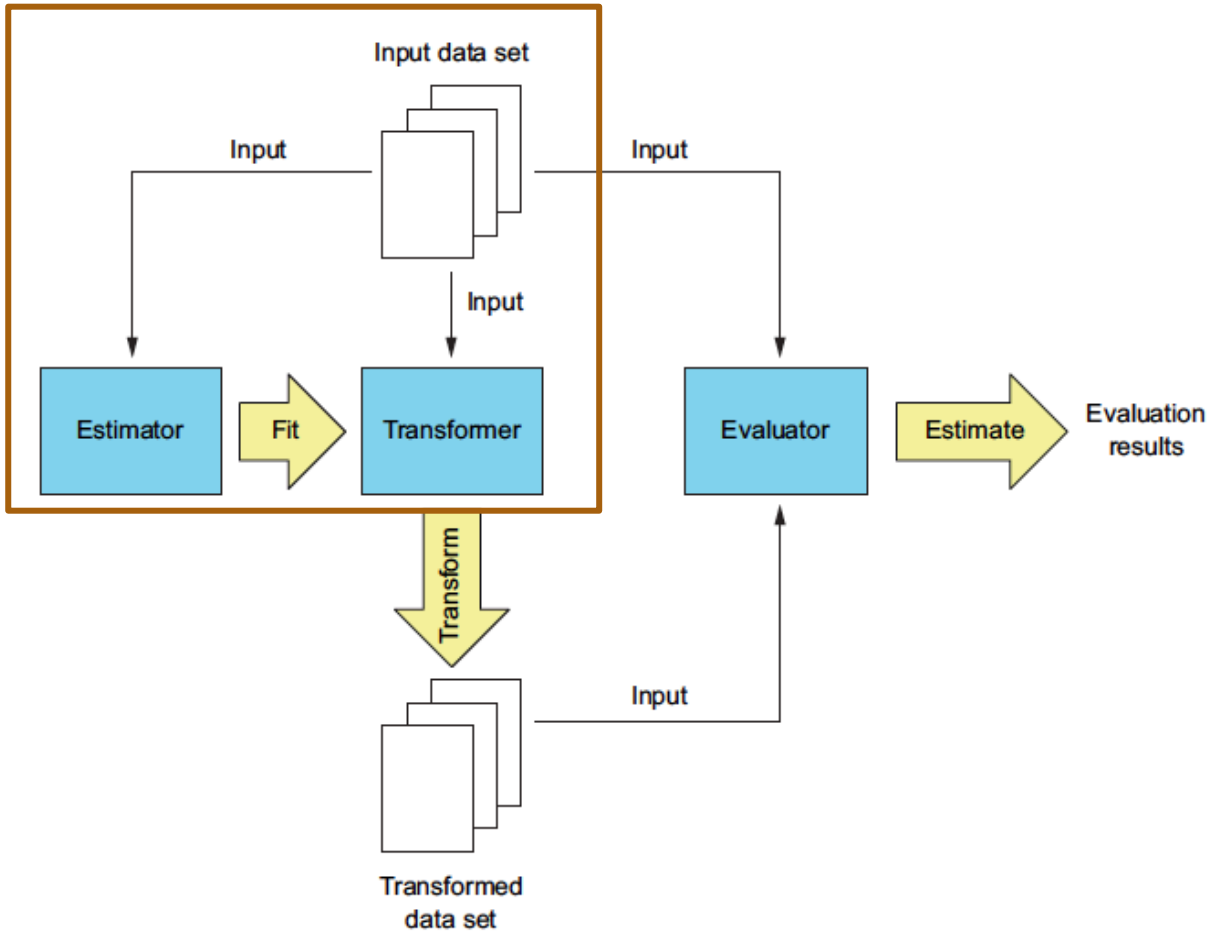


https://spark.apache.org/docs/latest/ml-pipeline.html

Spark ML

Main Components (5)

- Transformers
- **Estimators**
  - Algorithms that produce transformers by fitting on a dataset.
    - Ex. Linear regression produces a linear regression model with fitted weights and an intercepts, which is a transformer.
  - **`fit()`** : takes a DataFrame and parameters.
- Evaluators
- ML Parameters
- ML Pipeline

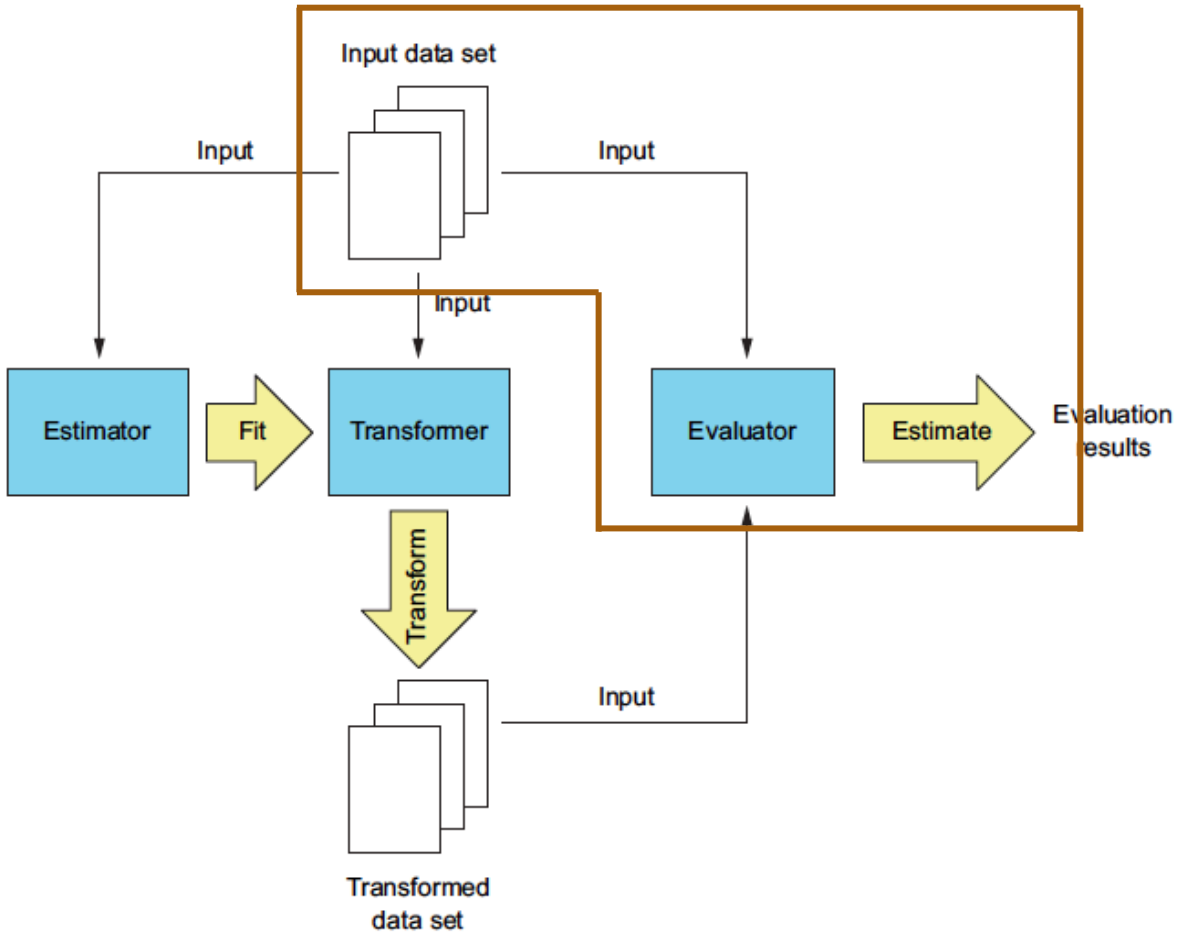


https://spark.apache.org/docs/latest/ml-pipeline.html

Spark ML

Main Components (5)

- Transformers
- Estimators
- **Evaluators**
  - Evaluate the performance of a model.
  - `Evaluator()`
- ML Parameters
- ML Pipeline



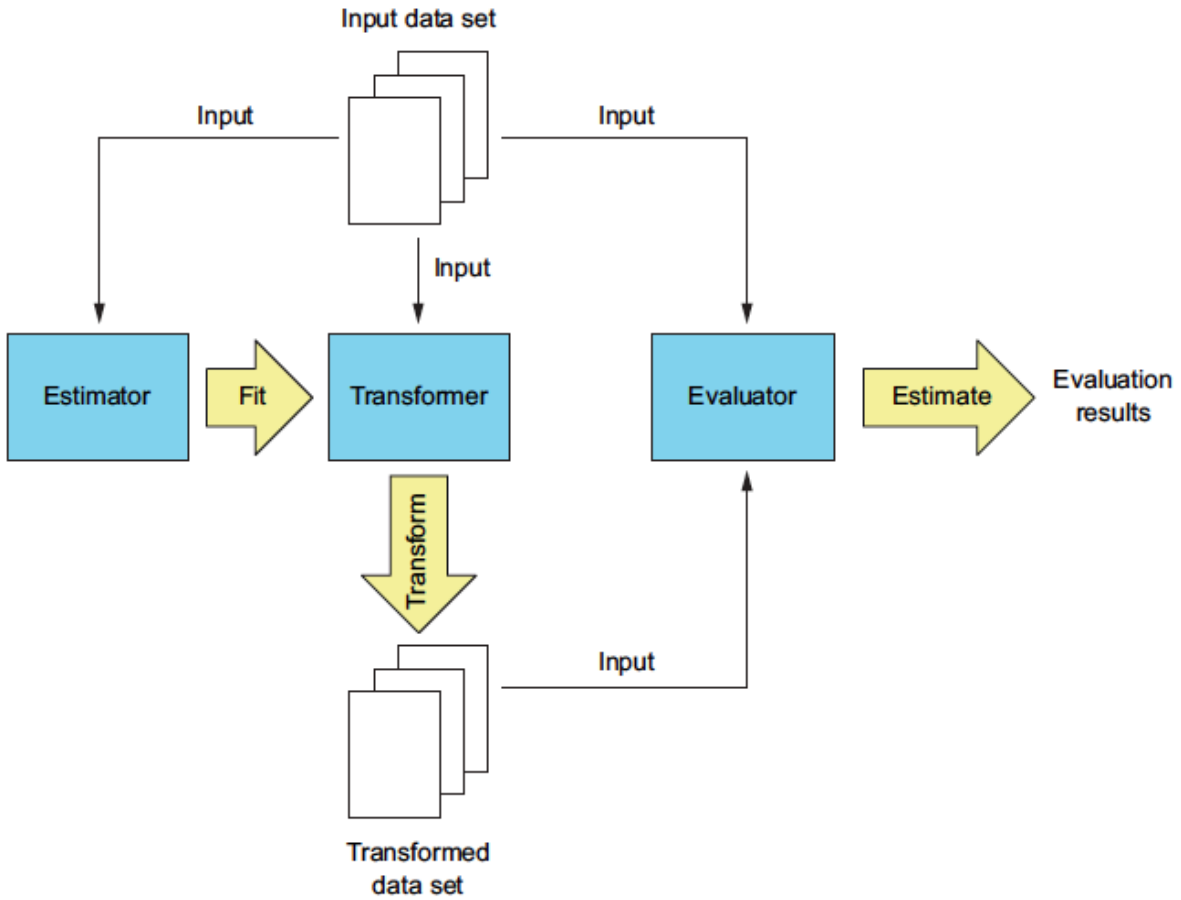
https://spark.apache.org/docs/latest/ml-pipeline.html

Spark ML

Main Components (5)

- Transformers
- Estimators
- Evaluators

- **ML Parameters**
  - Specify parameters for estimators and transformers .
  - Also can use **`ParamGridBuilder()`** for choosing the model produced by the best-performing set of parameters in **`CrossValidator().`**
- ML Pipeline



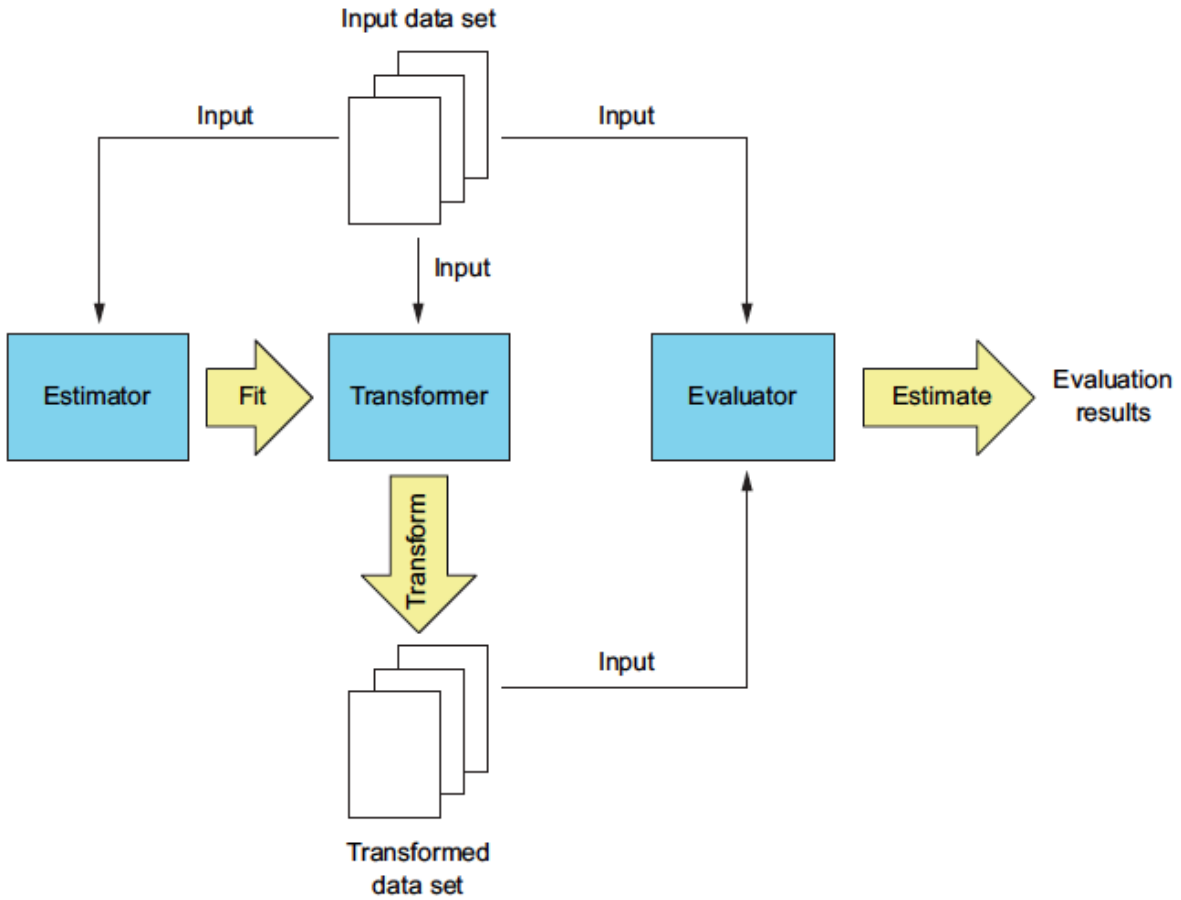
https://spark.apache.org/docs/latest/ml-pipeline.html

Spark ML

Main Components (5)

- Transformers
- Estimators
- Evaluators
- ML Parameters

- **ML Pipeline** (**`PipelineModel()`**)
  - In machine learning, the same steps are often repeated with slightly different parameters to find the best results.
  - A Pipeline chains multiple Transformers and Estimators together to specify an ML workflow and runs in order.



https://spark.apache.org/docs/latest/ml-pipeline.html

Contents

Machine Learning With Spark (Spark ML)

- Main Components
- **Algorithms**
- Logistic Regression
- Decision Tree
- Random Forest
- K-Mean Clustering

Spark ML

Algorithms

- Feature Extractors, Transformers, Selector and Locality Sensitive Hashing.
  - Feature Extractors : TF-IDF, Word2Vec, CountVectorizer, FeatureHasher
  - Feature Transformer : Tokenize, StopWordsRemover, n-gram, Binarizer, PCA, PolynomialExpansion, Discrete Cosine Transform (DCT), StringIndexer, IndexToString, OneHotEncoder , OneHotEncoderEstimator, VectorIndexer, Interaction, Normalizer, StandardScaler, MinMaxScaler, MaxAbsScaler, Bucketizer, ElementwiseProduct, SQLTransformer, VectorAssembler, VectorSizeHint, QuantileDiscretizer, Imputer
  - Feature Selectors : VectorSlicer, Rformula, ChiSqSelector
  - Locality Sensitive Hashing :
    - LSH Operations : Feature Transformation, Approximate Similarity Join, Approximate Nearest Neighbor Search
    - LSH Algorithms : Bucketed Random Projection for Euclidean Distance, MinHash for Jaccard Distance

https://spark.apache.org/docs/latest/ml-features.html

Spark ML

Algorithms

- Classification and Regression
  - Classification : Logistic regression, Decision tree classifier, Random forest classifier, Gradient-boosted tree classifier, Multilayer perceptron classifier, One-vs-Rest classifier (a.k.a. One-vs-All), Naive Bayes, Linear Support Vector Machine
  - Regression: Linear regression, Generalized linear regression, Decision tree regression, Random forest regression, Gradient-boosted tree regression, Survival regression , Isotonic regression
  - Decision trees
  - Tree Ensembles: Random Forests, Gradient-Boosted Trees (GBTs)
- Clustering : K-means, Latent Dirichlet allocation (LDA), Bisecting k-means, Gaussian Mixture Model (GMM)
- Collaborative filtering for Recommender Systems
- Frequent Pattern Mining : FP-Growth, PrefixSpan

https://spark.apache.org/docs/latest/ml-classification-regression.html

Spark ML

Data Sets

- Adult Data Set : Prediction task is to determine whether a person makes over 50K a year.
  - 48842 instances.
  - Attribute Information:
    - 14 attributes.
      >50K, <=50K.
      age: continuous.
      workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
      fnlwgt: continuous.
      education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
      education-num: continuous.
      marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
      occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
      relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
      race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
      sex: Female, Male.
      capital-gain: continuous.
      capital-loss: continuous.
      hours-per-week: continuous.
      native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba,….

https://archive.ics.uci.edu/ml/datasets

Contents

Machine Learning With Spark (Spark ML)

- Main Components
- Algorithms
- **Logistic Regression**
- Decision Tree
- Random Forest
- K-Mean Clustering

Spark ML – Example 1

Logistic Regression Example.

- Adult Data Set : **Prediction task is to determine whether a person makes over 50K a year.**
  - 48842 instances.
  - Attribute Information:
    - 14 attributes.
      >50K, <=50K.
      age: continuous.
      workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
      fnlwgt: continuous.
      education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
      education-num: continuous.
      marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
      occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
      relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
      race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
      sex: Female, Male.
      capital-gain: continuous.
      capital-loss: continuous.
      hours-per-week: continuous.
      native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba,….

Spark ML – Example 1

Logistic Regression Example.

- Use adult.dat to generate a regression model.

1. Create an RDD.
2. Convert the RDD to DataFrame.
3. Clean the data.
   1) Missing data imputation.
   2) Convert strings to categorical values.
4. Divide the dataset into training and validation sets.
5. Train the model.
6. Interpret the model parameters.
7. Evaluate the model.

Spark ML – Example 1

LogisticRegression ( )

- Input
  1. features - Feature vector.
  2. label - Label to predict.
- Output
  - Coefficient and intercept of the model.

https://spark.apache.org/docs/latest/ml-classification-regression.html#logistic-regression

Spark ML – Example 1

Logistic Regression Example.

- Use adult.dat to generate a regression model.

1. **Create an RDD.**
2. Convert the RDD to DataFrame.
3. Clean the data.
   1) Missing data imputation.
   2) Convert strings to categorical values.
4. Divide the dataset into training and validation sets.
5. Train the model.
6. Interpret the model parameters.
7. Evaluate the model.

Spark ML – Example 1

Logistic Regression Example.

- Use adult.dat to generate a regression model.

1. Create an RDD.

```
def toDoubleSafe(v):
    try:
        return float(v)
    except ValueError:
        return v #if it is not a float type return as a string.
```

```
#load and convert the data
census_raw = sc.textFile("../adult.raw", 4).map(lambda x:  x.split(", "))
census_raw = census_raw.map(lambda row:  [toDoubleSafe(x) for x in row])
```

```
census_raw.take(1)
```

```
[[39.0,
  u'State-gov',
  77516.0,
  u'Bachelors',
  u'Never-married',
  u'Adm-clerical',
  u'Not-in-family',
  u'White',
  u'Male',
  2174.0,
```

Spark ML – Example 1

Logistic Regression Example.

- Use adult.dat to generate a regression model.

1. Create an RDD.
2. **Convert the RDD to DataFrame.**
3. Clean the data.
   1) Missing data imputation.
   2) Convert strings to categorical values.
4. Divide the dataset into training and validation sets.
5. Train the model.
6. Interpret the model parameters.
7. Evaluate the model.

Spark ML – Example 1

Logistic Regression Example.

2. Convert the RDD to DataFrame
   - Define schema.

```
from pyspark.sql.types import *
adultschema = StructType([
    StructField("age",DoubleType(),True),
    StructField("workclass",StringType(),True),
    StructField("fnlwgt",DoubleType(),True),
    StructField("education",StringType(),True),
    StructField("marital_status",StringType(),True),
    StructField("occupation",StringType(),True),
    StructField("relationship",StringType(),True),
    StructField("race",StringType(),True),
    StructField("sex",StringType(),True),
    StructField("capital_gain",DoubleType(),True),
    StructField("capital_loss",DoubleType(),True),
    StructField("hours_per_week",DoubleType(),True),
    StructField("native_country",StringType(),True),
    StructField("income",StringType(),True)
])
```

Spark ML – Example 1

Logistic Regression Example.

2. Convert the RDD to DataFrame
   - Create Data Frame.

```
dfraw = ss.createDataFrame(census_raw, adultschema)
```

```
dfraw.show(10)
```

```
+---+----------------+-------+---------+------------------+----------------+-----------
-+-----+------+-----------+-----------+-------------+--------------+------+
| age|       workclass|  fnlwgt|education|      marital_status|       occupation| relationshi
p| race|   sex|capital_gain|capital_loss|hours_per_week|native_country|income|
+---+----------------+-------+---------+------------------+----------------+-----------
-+-----+------+-----------+-----------+-------------+--------------+------+
|39.0|       State-gov| 77516.0|Bachelors|       Never-married|     Adm-clerical|Not-in-famil
y|White|  Male|      2174.0|         0.0|          40.0| United-States| <=50K|
|50.0|Self-emp-not-inc| 83311.0|Bachelors|  Married-civ-spouse|  Exec-managerial|      Husban
d|White|  Male|         0.0|         0.0|          13.0| United-States| <=50K|
|38.0|         Private|215646.0|  HS-grad|            Divorced|Handlers-cleaners|Not-in-famil
y|White|  Male|         0.0|         0.0|          40.0| United-States| <=50K|
|53.0|         Private|234721.0|     11th|  Married-civ-spouse|Handlers-cleaners|      Husban
d|Black|  Male|         0.0|         0.0|          40.0| United-States| <=50K|
```

Spark ML – Example 1

Logistic Regression Example.

- Use adult.dat to generate a regression model.

1. Create an RDD.
2. Convert the RDD to DataFrame.
3. **Clean the data.**
   1) Missing data imputation.
   2) Convert strings to categorical values.
4. Divide the dataset into training and validation sets.
5. Train the model.
6. Interpret the model parameters.
7. Evaluate the model.

Spark ML – Example 1

Logistic Regression Example.

- Use adult.dat to generate a regression model.

1. Create an RDD.
2. Convert the RDD to DataFrame.
3. Clean the data.
   1) **Missing data imputation.**
   2) Convert strings to categorical values.
4. Divide the dataset into training and validation sets.
5. Train the model.
6. Interpret the model parameters.
7. Evaluate the model.

Spark ML – Example 1

Logistic Regression Example.

3. Clean the data.
   - Missing Data Imputation
     - I simply imputed the common value, although I prefer to use classification/ regression for missing data imputation.

```
#Check the most commonly used vals.
dfraw.groupBy(dfraw["workclass"]).count().orderBy("count",ascending=False).show()
dfraw.groupBy(dfraw["occupation"]).count().orderBy("count",ascending=False).show()
dfraw.groupBy(dfraw["native_country"]).count().orderBy("count",ascending=False).show()
```

```
+----------------+-----+
|       workclass|count|
+----------------+-----+
|         Private|33906|
|Self-emp-not-inc| 3862|
|       Local-gov| 3136|
|               ?| 2799|
|       State-gov| 1981|
|  Self-emp-inc| 1695|
|   Federal-gov| 1432|
|   Without-pay|   21|
+----------------+-----+
```

```
+-----------------+-----+
|       occupation|count|
+-----------------+-----+
|  Prof-specialty| 6172|
|    Craft-repair| 6112|
| Exec-managerial| 6086|
|    Adm-clerical| 5611|
|           Sales| 5504|
|   Other-service| 4923|
|Machine-op-inspct| 3022|
+-----------------+-----+
```

```
+------------------+-----+
|    native_country|count|
+------------------+-----+
|     United-States|43832|
|            Mexico|  951|
|                 ?|  857|
|       Philippines|  295|
|           Germany|  206|
|       Puerto-Rico|  184|
|            Canada|  182|
+------------------+-----+
```

Spark ML – Example 1

Logistic Regression Example.

3. Clean the data.
   - Missing Data Imputation
     - I simply imputed the common value, although I prefer to use classification/ regression for missing data imputation.

       `.replace(to_replace, value, subset=None)`
       - `to_replace` : Value to be replaced.
       - `value` : Value to use to replace holes.
       - `subset` : Optional list of column names to consider.

```
#Missing data imputation - Impute the most common row for "?".
dfrawrp = dfraw.replace(["?"], ["Private"], ["workclass"])
dfrawrpl = dfrawrp.replace(["?"], ["Prof-specialty"], ["occupation"])
dfrawnona = dfrawrpl.replace(["?"], ["United-States"], ["native_country"])
```

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html

Spark ML – Example 1

Logistic Regression Example.

- Use adult.dat to generate a regression model.

1. Create an RDD.
2. Convert the RDD to DataFrame.
3. Clean the data.
   1) Missing data imputation.
   2) **Convert strings to categorical values.**
4. Divide the dataset into training and validation sets.
5. Train the model.
6. Interpret the model parameters.
7. Evaluate the model.

Spark ML – Example 1

Logistic Regression Example.

3. Clean the data.
   - Convert Strings to Categorical Values.

```
from pyspark.sql.types import *
adultschema = StructType([
    StructField("age",DoubleType(),True),
    StructField("capital_gain",DoubleType(),True),
    StructField("capital_loss",DoubleType(),True),
    StructField("education",StringType(),True),
    StructField("fnlwgt",DoubleType(),True),
    StructField("hours_per_week",DoubleType(),True),
    StructField("income",StringType(),True),
    StructField("marital_status",StringType(),True),
    StructField("native_country",StringType(),True),
    StructField("occupation",StringType(),True),
    StructField("race",StringType(),True),
    StructField("relationship",StringType(),True),
    StructField("sex",StringType(),True),
    StructField("workclass",StringType(),True),
])
```

Many algorithms cannot handle string.

➔ Numeric Encoding?

- When male (0) and female (1), female > male?

➔ One-hot encoding : One columns is expanded to as many columns as the number of distinct values in the column and only one of the columns contains a 1 and all the others are 0s per row.

Marital status
Separated
Divorced
Widowed
Married
Widowed
Separated
Never married
Married

→

Married	Divorced	Separated	Widowed	Never Married
0	0	1	0	0
0	1	0	0	0
0	0	0	1	0
1	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	0	0	1
1	0	0	0	0

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html

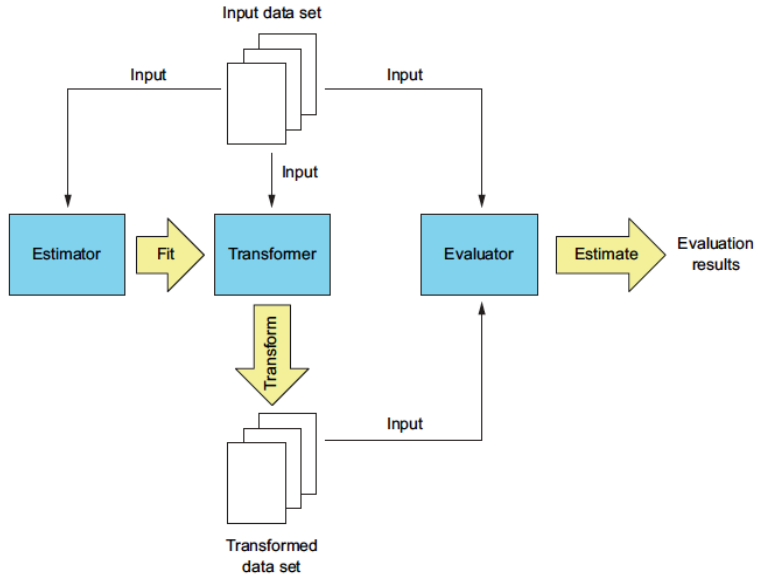
Spark ML – Example 1

Logistic Regression Example.

3. Clean the data.
   - Convert Strings to Categorical Values.
     1. String Indexer:
        - Convert string categorical values into integer indexes.
        - Takes a DataFrame and fits a **`StringIndexerModel(inputCol, outputCol)`** and used it for transformation.
     2. One-hot encoding - **`OneHotEncoder(inputCols, outputCols)`**:
        - Expand a column to as many columns as there are distinct strings in it and only one column contains a 1 and others are 0.
        - Create a new column as a one-hot-encoded sparse vector. (Replace a column with a vector.)
     3. **`VectorAssembler(inputCols, outputCol)`**
        - Merge all the new vectors and the original columns into a single vector.
        - Useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression and decision trees.

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html

Spark ML – Example 1



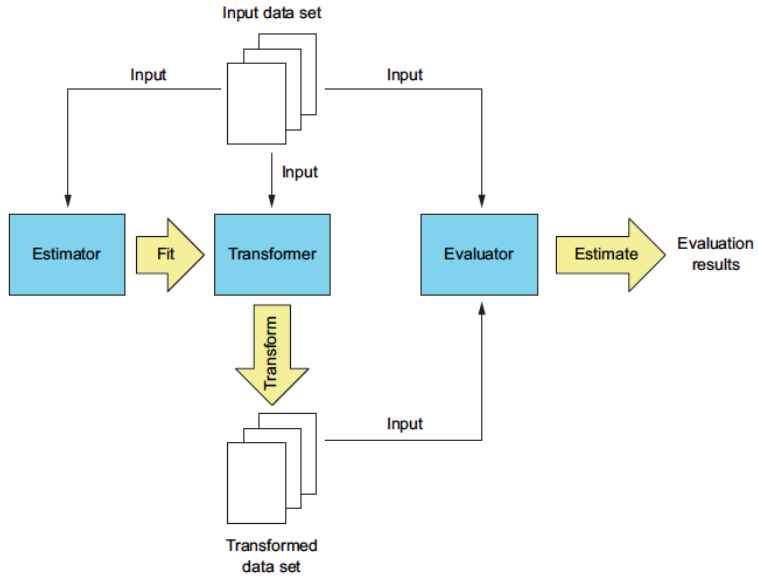
Logistic Regression Example.

3. Clean the data.
   - Convert Strings to Categorical Values.
     1. String Indexer:
        - Convert string categorical values into integer indexes.
        - Takes a DataFrame and fits a **`StringIndexerModel(inputCol, outputCol)`** and used it for transformation.

```
#converting strings to numeric values
def indexStringColumns(df, cols):
    from pyspark.ml.feature import StringIndexer
    #variable newdf will be updated several times
    newdf = df
    for c in cols:
        si = StringIndexer(inputCol=c, outputCol=c+"-num")
        sm = si.fit(newdf)
        newdf = sm.transform(newdf).drop(c)
        newdf = newdf.withColumnRenamed(c+"-num", c)
    return newdf

dfnumeric = indexStringColumns(dfrawnona, ["workclass", "education", "marital_status", "occupation", "relationship", "r
```

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html

Spark ML – Example 1



Logistic Regression Example.

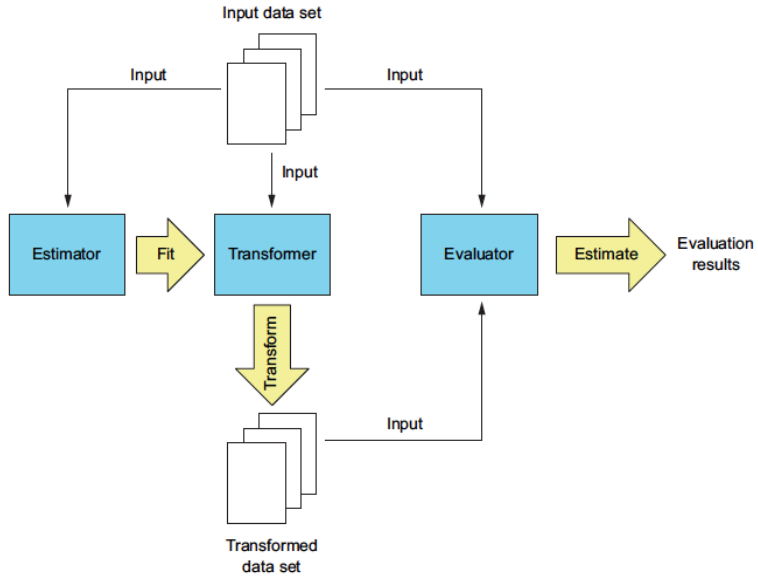
3. Clean the data.
   - Convert Strings to Categorical Values.
     1. String Indexer:
        - Convert string categorical values into integer indexes.
        - Takes a DataFrame and fits a **`StringIndexerModel(inputCol, outputCol)`** and used it for transformation.

Before

```
dfraw.show()
```

```
+----+------------+------------+------------+-------+--------------+------+------------------+--------------+-----
-----------+------------------+-------------+------+----------------+
| age|capital_gain|capital_loss|   education|fnlwgt|hours_per_week|income|
marital_status|native_country|       occupation|               race| relationship|   sex|       workclass|
+----+------------+------------+------------+-------+--------------+------+------------------+--------------+-----
-----------+------------------+-------------+------+----------------+
|39.0|      2174.0|         0.0|   Bachelors|77516.0|          40.0| <=50K|      Never-married| United-States|
Adm-clerical|             White|Not-in-family|  Male|       State-gov|
|50.0|         0.0|         0.0|   Bachelors|83311.0|          13.0| <=50K| Married-civ-spouse| United-States|
Exec-managerial|             White|      Husband|  Male|Self-emp-not-inc|
|38.0|         0.0|         0.0|     HS-grad|215646.0|          40.0| <=50K|           Divorced| United-
States|Handlers-cleaners|             White|Not-in-family|  Male|        Private|
|53.0|         0.0|         0.0|        11th|234721.0|          40.0| <=50K| Married-civ-spouse| United-
States|Handlers-cleaners|             Black|      Husband|  Male|        Private|
```

Spark ML – Example 1



Logistic Regression Example.

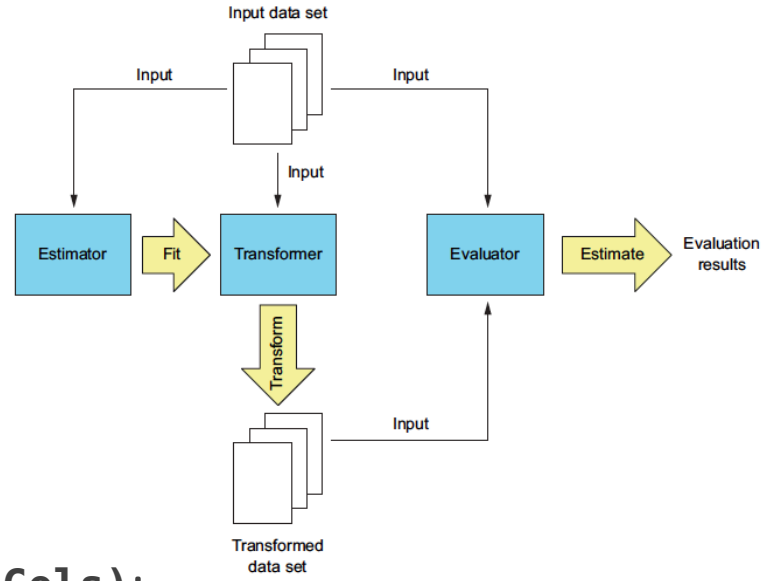
3. Clean the data.
   - Convert Strings to Categorical Values.
     1. String Indexer:
        - Convert string categorical values into integer indexes.
        - Takes a DataFrame and fits a **`StringIndexerModel(inputCol, outputCol)`** and used it for transformation.

After

```
dfnumeric.show()
```

```
+----+------------+------------+--------+--------------+---------+---------+--------------+----------+------------+---
---+---+--------------+------+
| age|capital_gain|capital_loss|  fnlwgt|hours_per_week|workclass|education|marital_status|occupation|relationship|ra
ce|sex|native_country|income|
+----+------------+------------+--------+--------------+---------+---------+--------------+----------+------------+---
---+---+--------------+------+
|39.0|      2174.0|         0.0| 77516.0|          40.0|      3.0|      2.0|           1.0|       3.0|         1.0| 0
.0|0.0|           0.0|   0.0|
|50.0|         0.0|         0.0| 83311.0|          13.0|      1.0|      2.0|           0.0|       2.0|         0.0| 0
.0|0.0|           0.0|   0.0|
|38.0|         0.0|         0.0|215646.0|          40.0|      0.0|      0.0|           2.0|       8.0|         1.0| 0
.0|0.0|           0.0|   0.0|
|53.0|         0.0|         0.0|234721.0|          40.0|      0.0|      5.0|           0.0|       8.0|         0.0| 1
.0|0.0|           0.0|   0.0|
```

Spark ML – Example 1



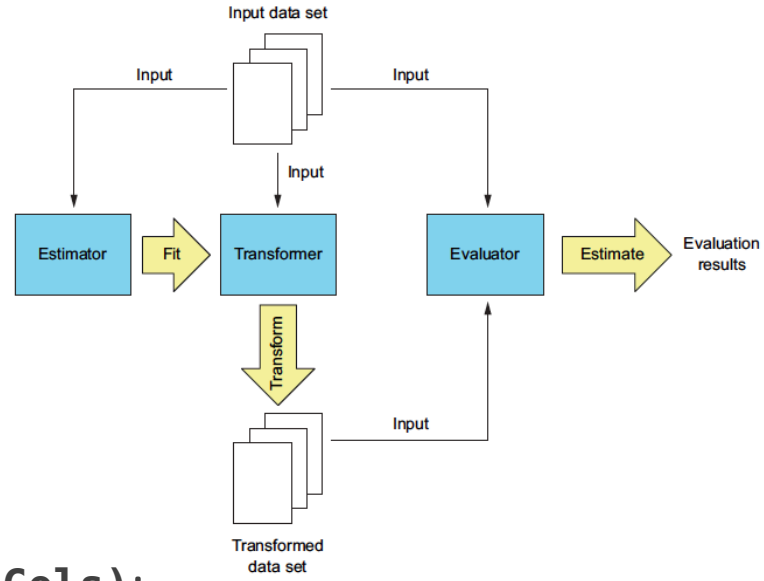
Logistic Regression Example.

3. Clean the data.
   - Convert Strings to Categorical Values.
     2. One-hot encoding - **`OneHotEncoder(inputCols, outputCols)`**:
        - Expand a column to as many columns as there are distinct values in it and only one column contains a 1 and others are 0.
        - Create a new column as a one-hot-encoded sparse vector. (Replace a column with a vector.)

```
from pyspark.ml.feature import OneHotEncoder

def oneHotEncodeColumns(df, cols):
    newdf = df
    for c in cols:
        ohe = OneHotEncoder(inputCol=c, outputCol=c+"-onehot", dropLast=False)
        ohe_model = ohe.fit(newdf)

        newdf = ohe_model.transform(newdf).drop(c)
        newdf = newdf.withColumnRenamed(c+"-onehot", c)
    return newdf

dfhot = oneHotEncodeColumns(dfnumeric, ["workclass", "education",
                                        "marital_status", "occupation",
                                        "relationship", "race", "native_country"])
```

The last category is not included by default.

Spark ML – Example 1



Logistic Regression Example.

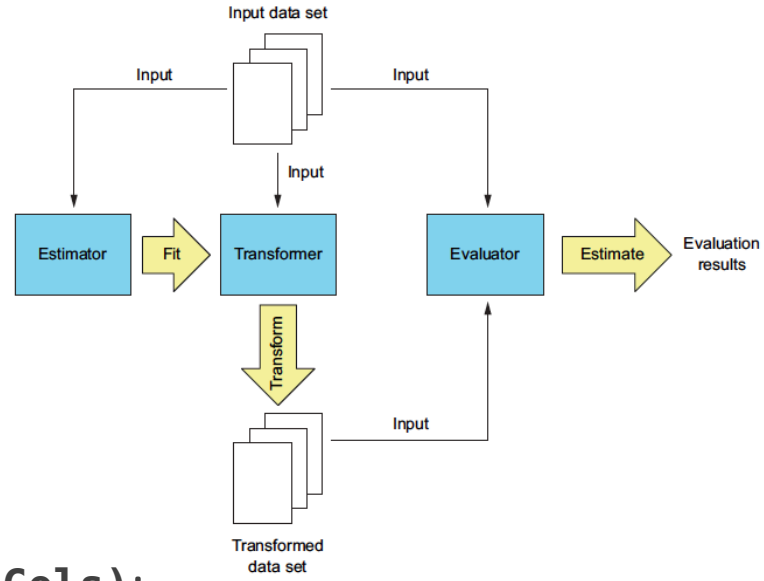
3. Clean the data.
   - Convert Strings to Categorical Values.
     2. One-hot encoding - **`OneHotEncoder(inputCols, outputCols)`**:
        - Expand a column to as many columns as there are distinct values in it and only one column contains a 1 and others are 0.
        - Create a new column as a one-hot-encoded sparse vector. (Replace a column with a vector.)

Before

```
dfnumeric.show()
```

```
+----+------------+------------+--------+--------------+---------+---------+--------------+----------+------------+--
--+---+--------------+------+
| age|capital_gain|capital_loss|  fnlwgt|hours_per_week|workclass|education|marital_status|occupation|relationship|ra
ce|sex|native_country|income|
+----+------------+------------+--------+--------------+---------+---------+--------------+----------+------------+--
--+---+--------------+------+
|39.0|      2174.0|         0.0| 77516.0|          40.0|      3.0|      2.0|           1.0|       3.0|         1.0| 0
.0|0.0|           0.0|   0.0|
|50.0|         0.0|         0.0| 83311.0|          13.0|      1.0|      2.0|           0.0|       2.0|         0.0| 0
.0|0.0|           0.0|   0.0|
|38.0|         0.0|         0.0|215646.0|          40.0|      0.0|      0.0|           2.0|       8.0|         1.0| 0
.0|0.0|           0.0|   0.0|
```

Spark ML – Example 1



Logistic Regression Example.

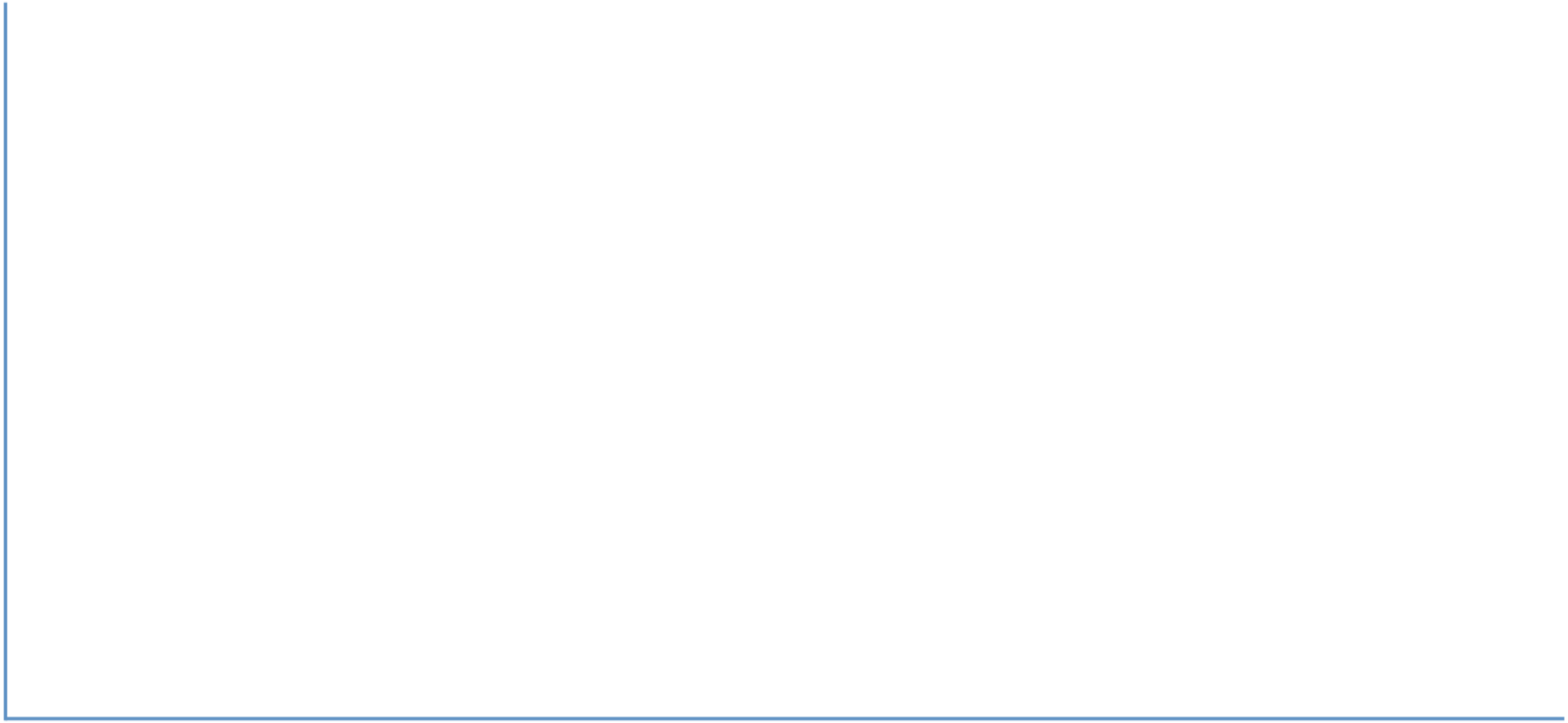
3. Clean the data.
   - Convert Strings to Categorical Values.
     2. One-hot encoding - **`OneHotEncoder(inputCols, outputCols)`**:
        - Expand a column to as many columns as there are distinct values in it and only one column contains a 1 and others are 0.
        - Create a new column as a one-hot-encoded sparse vector. (Replace a column with a vector.)

After

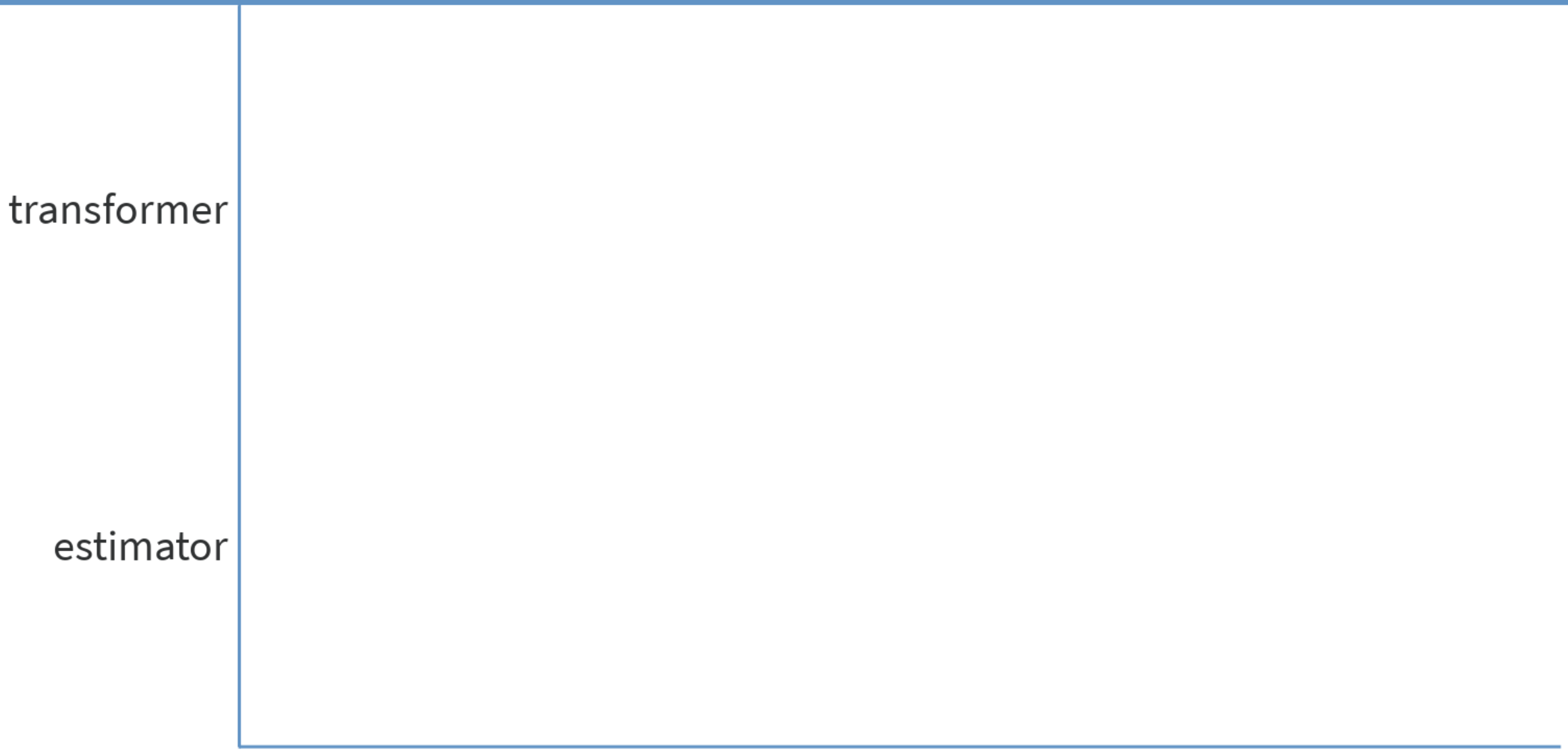
```
dfhot.show()
```

```
+----+------------+------------+--------+--------------+---+------+-------------+---------------+--------------+-----
---------+-------------+-------------+---------------+
| age|capital_gain|capital_loss|  fnlwgt|hours_per_week|sex|income|    workclass|      education|marital_status|    o
ccupation| relationship|         race| native_country|
+----+------------+------------+--------+--------------+---+------+-------------+---------------+--------------+-----
---------+-------------+-------------+---------------+
|39.0|      2174.0|         0.0| 77516.0|          40.0|0.0|   0.0|(8,[3],[1.0])| (16,[2],[1.0])| (7,[1],[1.0])|(14,[
3],[1.0])|(6,[1],[1.0])|(5,[0],[1.0])| (41,[0],[1.0])|
|50.0|         0.0|         0.0| 83311.0|          13.0|0.0|   0.0|(8,[1],[1.0])| (16,[2],[1.0])| (7,[0],[1.0])|(14,[
2],[1.0])|(6,[0],[1.0])|(5,[0],[1.0])| (41,[0],[1.0])|
|38.0|         0.0|         0.0|215646.0|          40.0|0.0|   0.0|(8,[0],[1.0])| (16,[0],[1.0])| (7,[2],[1.0])|(14,[
8],[1.0])|(6,[1],[1.0])|(5,[0],[1.0])| (41,[0],[1.0])|
```

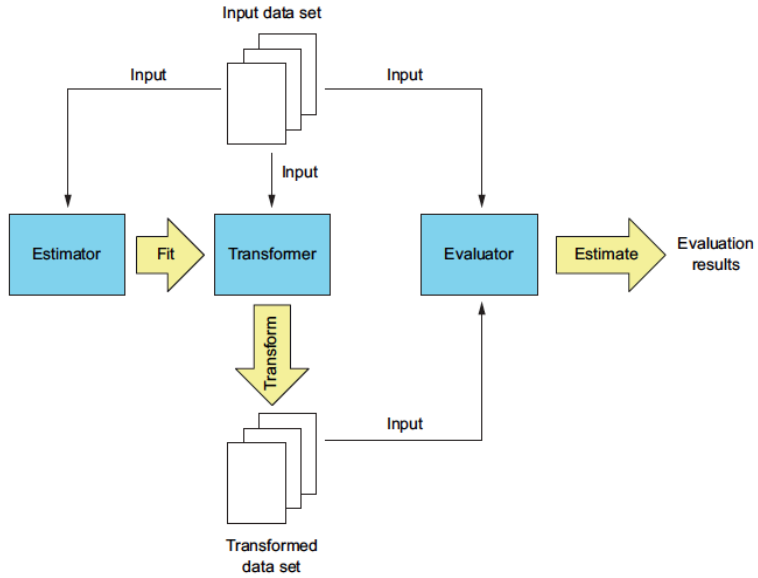
Is OneHotEncoder a transformer or estimator?



Is one-hot-coding transformer or estimator?



Spark ML – Example 1



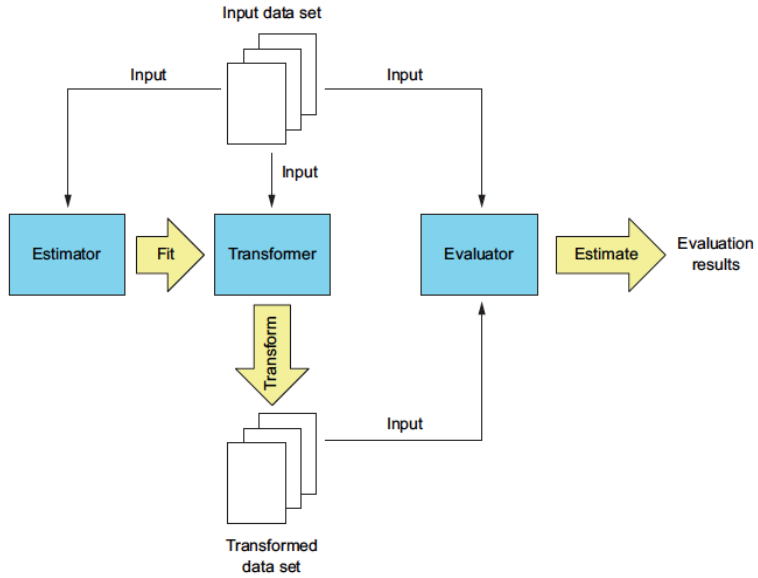
Logistic Regression Example.

3. Clean the data.
   - Convert Strings to Categorical Values.
     3. **`VectorAssembler(inputCols, outputCol)`**
        - Merge all the new vectors and the original columns into a single vector.
        - Useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression and decision trees.
   - ML algorithms work with two columns called **features** and **label** by default.

```
# Merging the data with Vector Assembler.
from pyspark.ml.feature import VectorAssembler
input_cols=["age","capital_gain","capital_loss","fnlwgt","hours_per_week","sex","workclass","education","marital_status
va = VectorAssembler(outputCol="features", inputCols=input_cols)
lpoints = va.transform(dfhot).select("features", "income").withColumnRenamed("income", "label")
```

http://spark.apache.org/docs/latest/ml-features.html#vectorassembler

Spark ML – Example 1



Logistic Regression Example.

3. Clean the data.
   - Convert Strings to Categorical Values.
     3. **`VectorAssembler(inputCols, outputCol)`**
        - Merge all the new vectors and the original columns into a single vector.
        - Useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression and decision trees.
   - ML algorithms work with two columns called **features** and **label** by default.

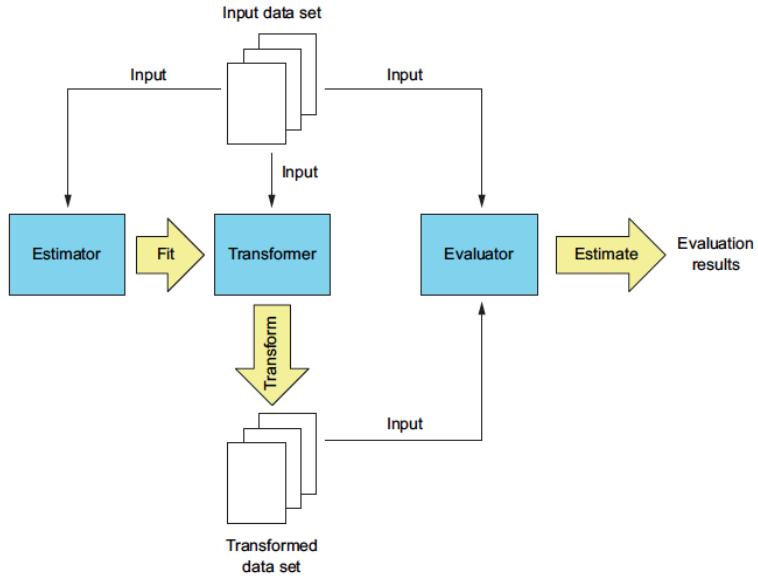
Before

```
dfhot.show()
```

```
+----+------------+------------+--------+--------------+---+------+------------+--------------+--------------+-----
---------+-------------+-------------+---------------+
| age|capital_gain|capital_loss|  fnlwgt|hours_per_week|sex|income|   workclass|     education|marital_status|    o
ccupation| relationship|         race| native_country|
+----+------------+------------+--------+--------------+---+------+------------+--------------+--------------+-----
---------+-------------+-------------+---------------+
|39.0|      2174.0|         0.0| 77516.0|          40.0|0.0|   0.0|(8,[3],[1.0])| (16,[2],[1.0])| (7,[1],[1.0])|(14,[
                                                                  ,[1.0])| (16,[2],[1.0])| (7,[0],[1.0])|(14,[
```

http://spark.apache.org/docs/latest/ml-features.html#vectorassembler

Spark ML – Example 1



Logistic Regression Example.

3. Clean the data.
   - Convert Strings to Categorical Values.
     3. **`VectorAssembler(inputCols, outputCol)`**
        - Merge all the new vectors and the original columns into a single vector.
        - Useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression and decision trees.
   - ML algorithms work with two columns called **features** and **label** by default.

After

```
lpoints.show()
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|(103,[0,1,3,4,9,1...|  0.0|
|(103,[0,3,4,7,16,...|  0.0|
```

http://spark.apache.org/docs/latest/ml-features.html#vectorassembler

Spark ML – Example 1

Logistic Regression Example.

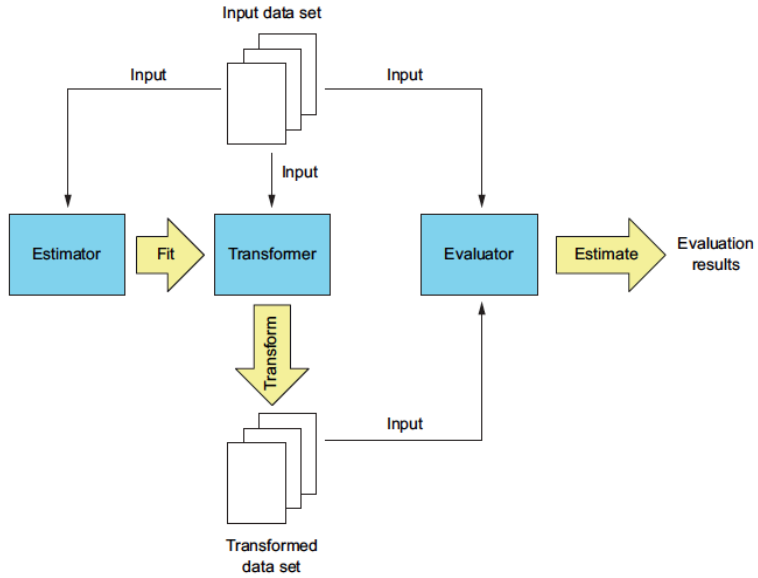
- Use adult.dat to generate a regression model.

1. Create an RDD.
2. Convert the RDD to DataFrame.
3. Clean the data.
   1) Missing data imputation.
   2) Convert strings to categorical values.
4. **Divide the dataset into training and validation sets.**
5. Train the model.
6. Interpret the model parameters.
7. Evaluate the model.

Spark ML – Example 1

Logistic Regression Example.

4. Divide the dataset into training and validation sets.




```
#Divide the dataset into training and vaildation sets.
splits = lpoints.randomSplit([0.8, 0.2])

adulttrain = splits[0].cache()
adultvalid = splits[1].cache()
```

WHY??

http://spark.apache.org/docs/latest/ml-features.html#vectorassembler

Why do we use .cache() for ML models?

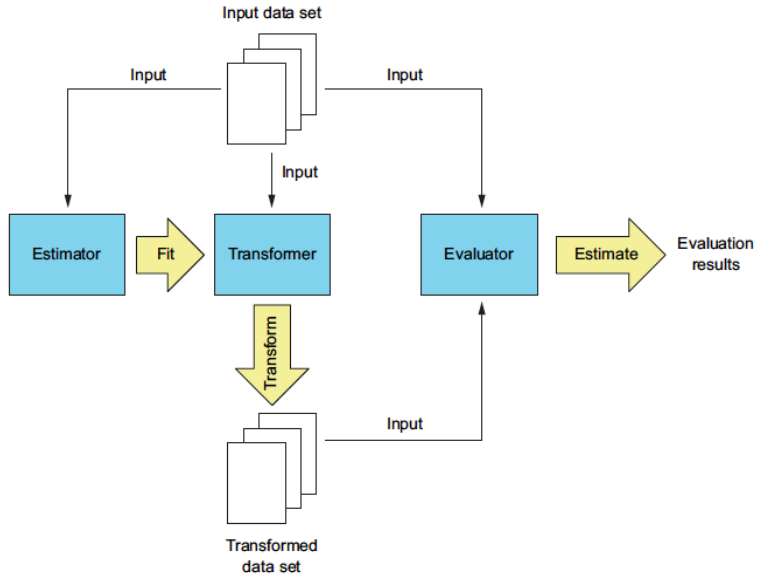
Spark ML – Example 1

Logistic Regression Example.

- Use adult.dat to generate a regression model.

1. Create an RDD.
2. Convert the RDD to DataFrame.
3. Clean the data.
   1) Missing data imputation.
   2) Convert strings to categorical values.
4. Divide the dataset into training and validation sets.
5. **Train the model.**
6. Interpret the model parameters.
7. Evaluate the model.

Spark ML – Example 1



Logistic Regression Example.

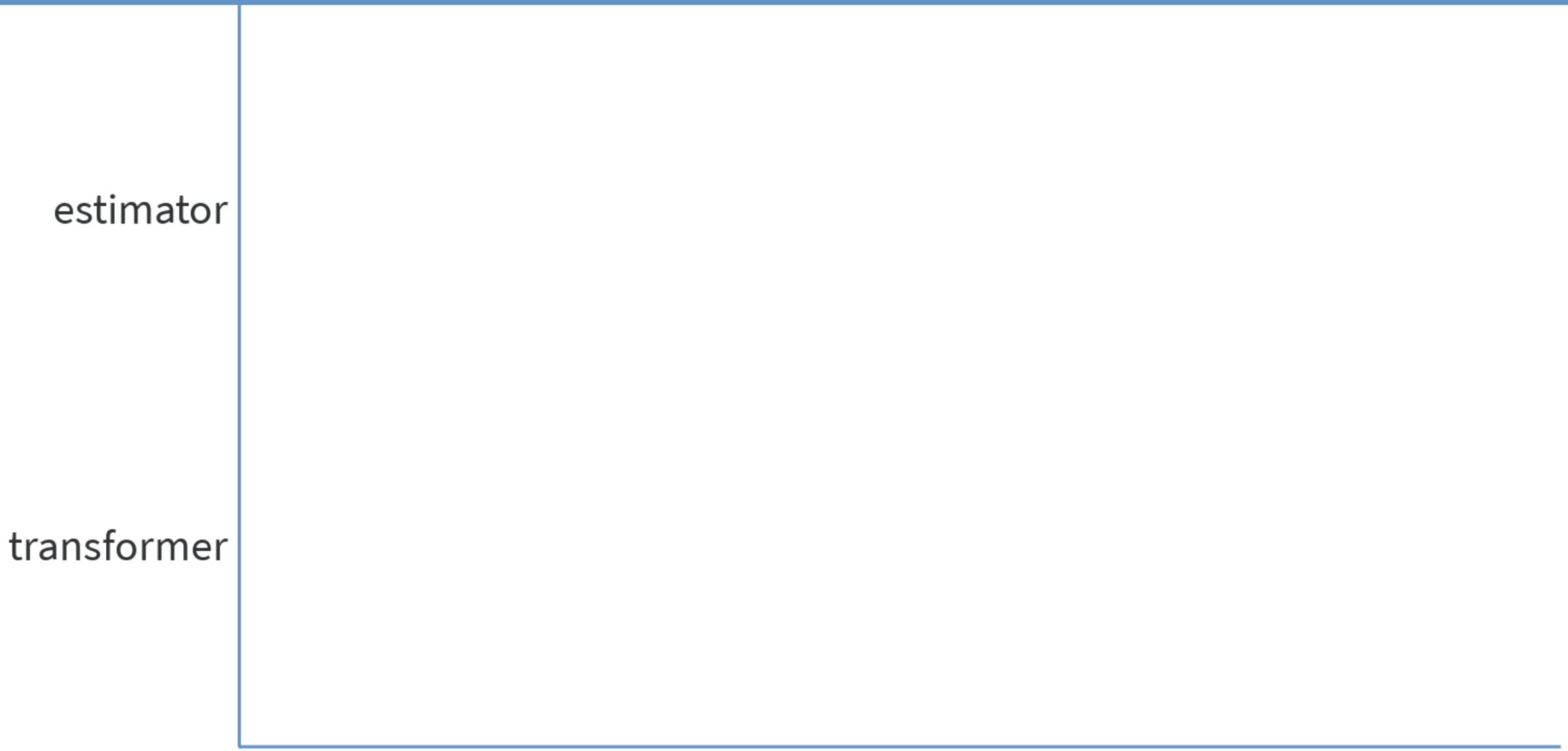
5. Train the model.
   - Use Spark ML's `LogisticRegression`
     - Set parameters - regParam, maxIter, fitIntercept.
     - Call `fit()` passing in a DataFrame.

```
#Train the model.
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(regParam=0.01, maxIter=1000, fitIntercept=True)
lrmodel = lr.fit(adulttrain)
```

ML Parameters (More in the documentation)

https://spark.apache.org/docs/latest/ml-classification-regression.html#logistic-regression
https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.classification.LogisticRegression

Is lr (Logistic Regression) an estimator or a transformer?



Spark ML – Example 1

Logistic Regression Example.

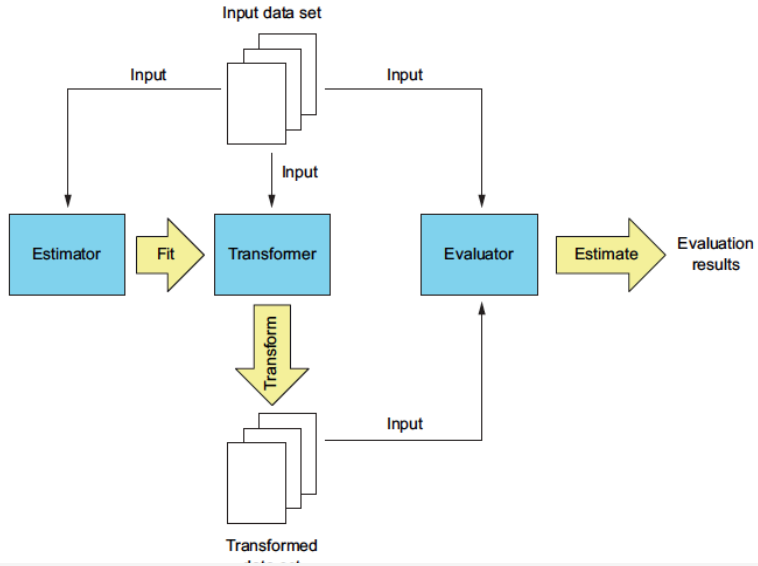
- Use adult.dat to generate a regression model.

1. Create an RDD.
2. Convert the RDD to DataFrame.
3. Clean the data.
   1) Missing data imputation.
   2) Convert strings to categorical values.
4. Divide the dataset into training and validation sets.
5. Train the model.
6. **Interpret the model parameters.**
7. Evaluate the model.

Spark ML – Example 1

Logistic Regression Example.

6. Interpret the model parameters.




```
#Interpret the model parameters
print(lrmodel.coefficients)
print(lrmodel.intercept)
```

```
[0.0200635956757,0.000142117910422,0.000549680557822,7.67939881134e-07,0.0270580225506,-0.527453343079,0.00966356226161,-0.34291667775,0.0395917194118,-0.141854478053,0.280058560177,0.600089208761,-0.387298971983,-0.64014643248,-0.344749447097,0.000956755374673,0.753174821018,1.08931162847,0.153992779575,-0.918979066595,0.173732711249,-1.11489212755,-1.35296560719,1.64107789548,-1.23499994442,-0.611294148404,1.52228819816,-1.21910537157,-1.55424341573,-1.63381426706,0.807335596657,-0.675633544989,-0.273023613173,-0.298616483069,-0.259763608797,-0.174182110048,0.892905297572,0.223649546869,0.0406062034707,0.674242185475,-0.0362094985227,0.196109637517,-0.747375692485,-0.271198569957,-0.134722895846,-0.616817018666,-0.910237980391,0.446199946491,0.351709412347,-0.895293985467,0.589665239528,0.471866510172,-0.0964050040335,-0.817006143056,-0.325652759335,1.32289221548,-0.580583289363,0.194206452698,-0.591199710438,0.447122319076,-0.00743155608916,-0.106991845472,0.403034553484,-0.392782163748,0.0315347018095,0.338559425942,0.749846528935,-0.445684932018,-1.01727162353,0.207976345143,0.730583078689,-1.04443802189,0.0421863262162,-0.286338264139,0.134407218438,-0.636935242122,-1.21433121715,-0.0680550984543,0.365524483517,0.0388005241512,-0.476740741173,0.199492582103,-0.564159833277,-0.541838707731,-0.925450854006,0.656209481137,0.886629033942,-0.885074619199,-0.784944915792,0.888763451506,-0.840846549927,1.15928732002,-0.959851499168,-0.578807741163,-0.557760905544,-0.0365170688454,0.474330934434,-1.14851367742,0.0968734548632,-0.177597852545,0.231903507463,-0.37008350593,-0.0334950683076]
-4.33068546155
```

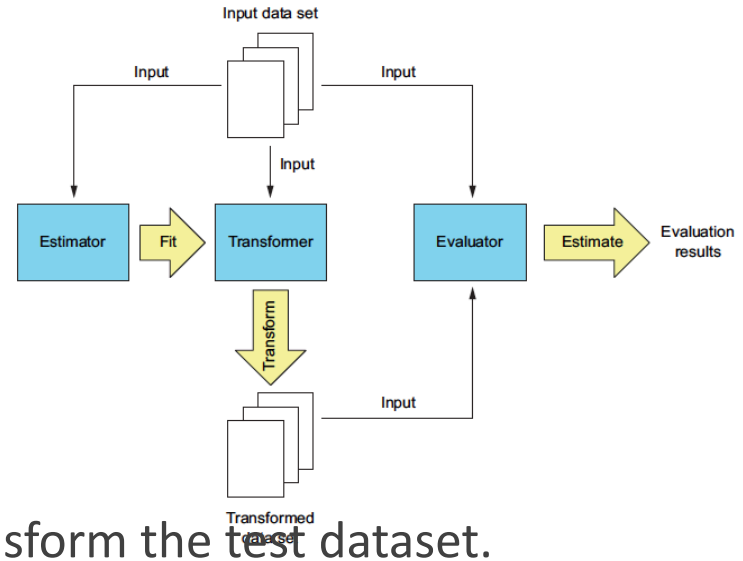
Spark ML – Example 1

Logistic Regression Example.

- Use adult.dat to generate a regression model.

1. Create an RDD.
2. Convert the RDD to DataFrame.
3. Clean the data.
   1) Missing data imputation.
   2) Convert strings to categorical values.
4. Divide the dataset into training and validation sets.
5. Train the model.
6. Interpret the model parameters.
7. **Evaluate the model.**

Spark ML – Example 1



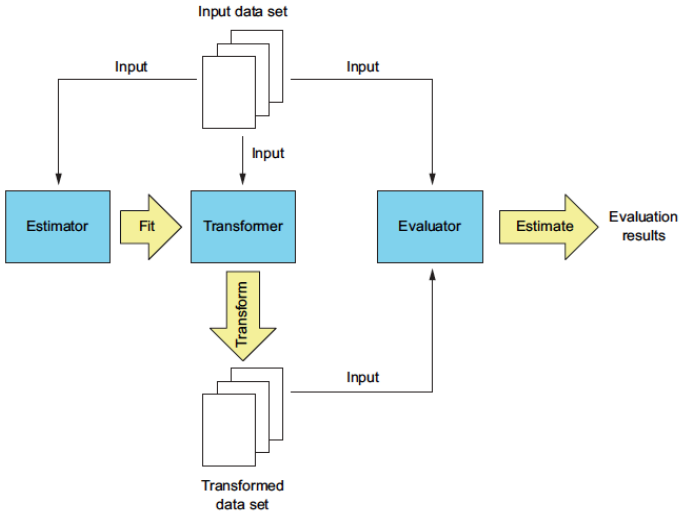
Logistic Regression Example.

7. Evaluate classification models.
   - First use lrmodel (the developed linear regression model) to transform the test dataset.
   - Then use `BinaryClassificationEvaluator()` to evaluate the performance.

```
#Evaluate models using test dataset.
validpredicts = lrmodel.transform(adultvalid)
validpredicts.show()
```

```
+--------------------+-----+--------------------+--------------------+----------+
|            features|label|       rawPrediction|         probability|prediction|
+--------------------+-----+--------------------+--------------------+----------+
|(103,[0,1,3,4,5,6...|  0.0|[0.65761126454865...|[0.65872358820700...|       0.0|
|(103,[0,1,3,4,5,6...|  1.0|[-0.9590664998231...|[0.27706513608652...|       1.0|
|(103,[0,1,3,4,5,6...|  0.0|[3.68197930538441...|[0.97544502461658...|       0.0|
|(103,[0,1,3,4,5,6...|  0.0|[4.09627501272418...|[0.98363765626483...|       0.0|
|(103,[0,1,3,4,5,6...|  1.0|[0.96345154510005...|[0.72381232948775...|       0.0|
|(103,[0,1,3,4,5,6...|  0.0|[2.28984133291450...|[0.90803220068241...|       0.0|
|(103,[0,1,3,4,5,6...|  0.0|[2.71279447036546...|[0.93777740821339...|       0.0|
|(103,[0,1,3,4,5,6...|  0.0|[4.64042974795558...|[0.99043875182615...|       0.0|
|(103,[0,1,3,4,5,6...|  0.0|[-2.2054793310857...|[0.09925952135483...|       1.0|
|(103,[0,1,3,4,5,6...|  1.0|[-0.3346962461104...|[0.41709839480886...|       1.0|
```

Spark ML – Example 1



Logistic Regression Example.

7. Evaluate classification models.
   - First use lrmodel (the developed linear regression model) to transform the test dataset.
   - Then use `BinaryClassificationEvaluator()` to evaluate the performance.

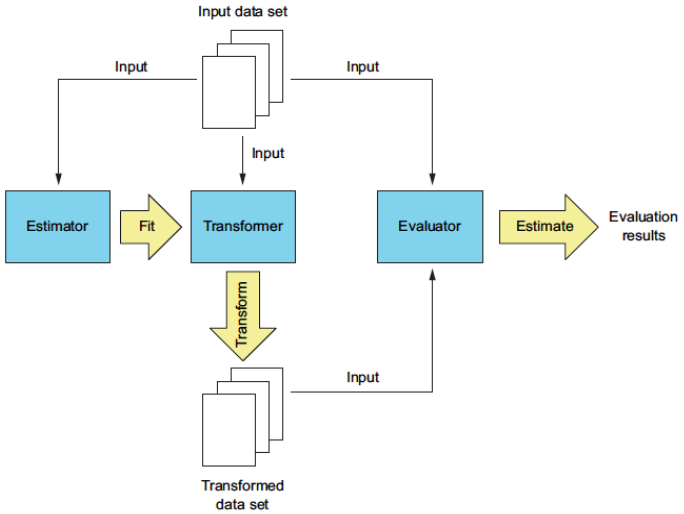
```
#Evaluate models using test dataset.
validpredicts = lrmodel.transform(adultvalid)
validpredicts.show()
```

```
+--------------------+-----+--------------------+Row(rawPrediction=DenseVector([0.6636, -0.6636])),
|            features|label|       rawPrediction|Row(rawPrediction=DenseVector([-0.1135, 0.1135])),
+--------------------+-----+--------------------+Row(rawPrediction=DenseVector([0.9454, -0.9454])),
|(103,[0,1,3,4,5,6...|  0.0|[0.65761126454865...|Row(rawPrediction=DenseVector([0.6486, -0.6486])),
|(103,[0,1,3,4,5,6...|  1.0|[-0.9590664998231...|Row(rawPrediction=DenseVector([-0.948, 0.948])),
|(103,[0,1,3,4,5,6...|  0.0|[3.68197930538441...|Row(rawPrediction=DenseVector([4.3695, -4.3695])),
|(103,[0,1,3,4,5,6...|  0.0|[4.09627501272418...|Row(rawPrediction=DenseVector([5.2266, -5.2266])),
|(103,[0,1,3,4,5,6...|  1.0|[0.96345154510005...|
|(103,[0,1,3,4,5,6...|  0.0|[2.28984133291450...|
|(103,[0,1,3,4,5,6...|  0.0|[2.71279447036546...|
|(103,[0,1,3,4,5,6...|  0.0|[4.64042974795558...|[0.99043875182615...|         0.0|
|(103,[0,1,3,4,5,6...|  1.0|[-0.3346962461104...|[0.41709839480886...|         1.0|
```

rawPrediction : log-odds that a sample doesn't/does belong to the category.

https://spark.apache.org/docs/latest/api/java/index.html?org/apache/spark/ml/evaluation/BinaryClassificationEvaluator.html

Spark ML – Example 1



Logistic Regression Example.

7. Evaluate classification models.
   - First use lrmodel (the developed linear regression model) to transform the test dataset.
   - Then use `BinaryClassificationEvaluator()` to evaluate the performance.

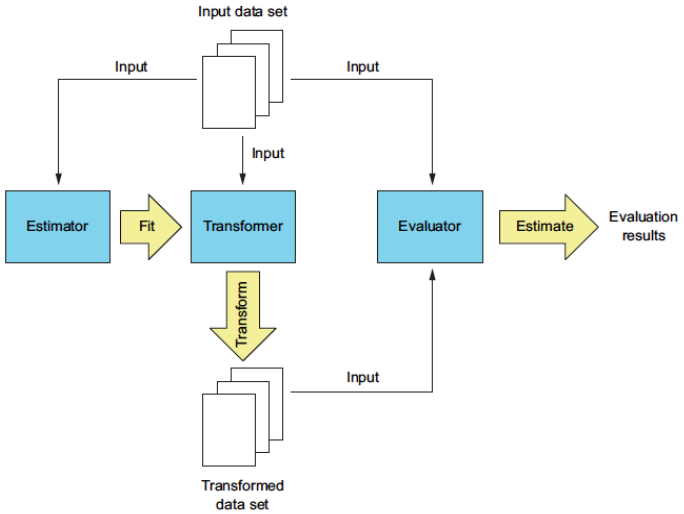
```
#Evaluate models using test dataset.
validpredicts = lrmodel.transform(adultvalid)
validpredicts.show()
```

```
+--------------------+-----+--------------------+--------------------+----------+
Row(probability=DenseVector([0.6601, 0.3399])),|        probability|prediction|
Row(probability=DenseVector([0.4716, 0.5284])),+--------------------+----------+
Row(probability=DenseVector([0.7202, 0.2798])),|[0.65872358820700...|       0.0|
Row(probability=DenseVector([0.6567, 0.3433])),|[0.27706513608652...|       1.0|
Row(probability=DenseVector([0.2793, 0.7207])),|[0.97544502461658...|       0.0|
Row(probability=DenseVector([0.9875, 0.0125])),|[0.98363765626483...|       0.0|
                                               |[0.72381232948775...|       0.0|
                                               |[0.90803220068241...|       0.0|
                                               |[0.93777740821339...|       0.0|
                                               |[0.99043875182615...|       0.0|
|(103,[0,1,3,4,5,6...|  1.0|[-0.3346962461104...|[0.41709839480886...|       1.0|
```

probability : the probability that the sample is in the category.

https://spark.apache.org/docs/latest/api/java/index.html?org/apache/spark/ml/evaluation/BinaryClassificationEvaluator.html

Spark ML – Example 1



Logistic Regression Example.

7. Evaluate classification models.
   - First use lrmodel (the developed linear regression model) to transform the test dataset.
   - Then use `BinaryClassificationEvaluator()` to evaluate the performance.

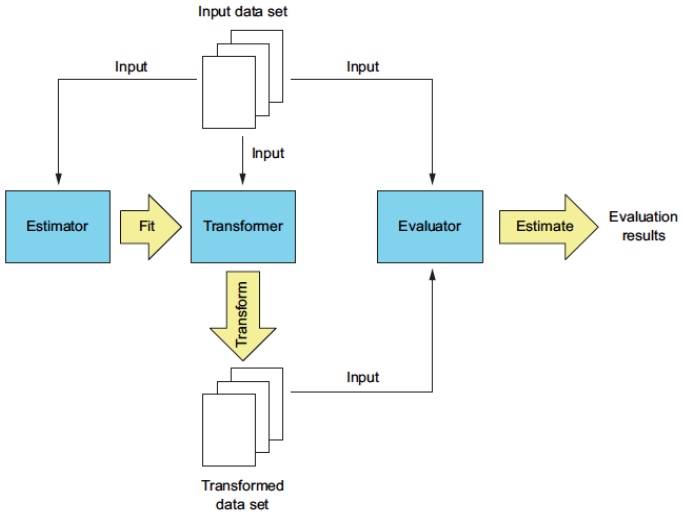
```
#Evaluate the model. default metric : Area Under ROC
from pyspark.ml.evaluation import BinaryClassificationEvaluator
bceval = BinaryClassificationEvaluator()
print (bceval.getMetricName() +":" + str(bceval.evaluate(validpredicts)))
```

```
areaUnderROC:0.901769284628039
```

https://spark.apache.org/docs/latest/mllib-evaluation-metrics.html

https://spark.apache.org/docs/latest/api/java/index.html?org/apache/spark/ml/evaluation/BinaryClassificationEvaluator.html

Spark ML – Example 1



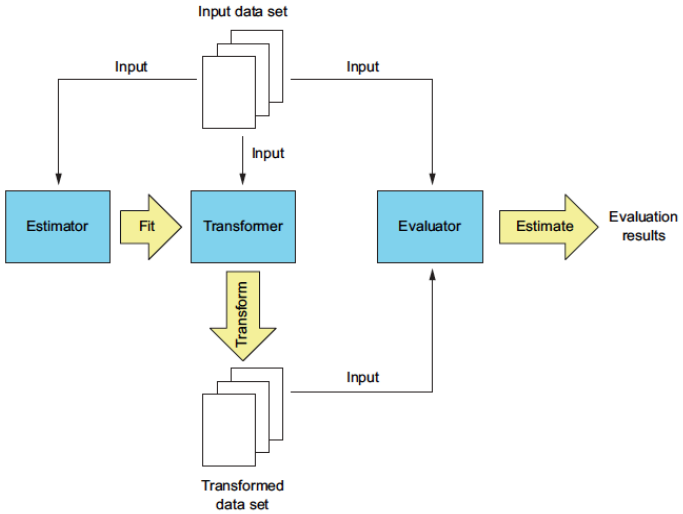
Logistic Regression Example.

7. Evaluate classification models.
   - First use lrmodel (the developed linear regression model) to transform the test dataset.
   - Then use `BinaryClassificationEvaluator()` to evaluate the performance.

```
#Evaluate the model. metric : Area Under PR
bceval.setMetricName("areaUnderPR")
print (bceval.getMetricName() +":" + str(bceval.evaluate(validpredicts)))
```

```
areaUnderPR:0.7441970509619391
```

Spark ML – Example 1

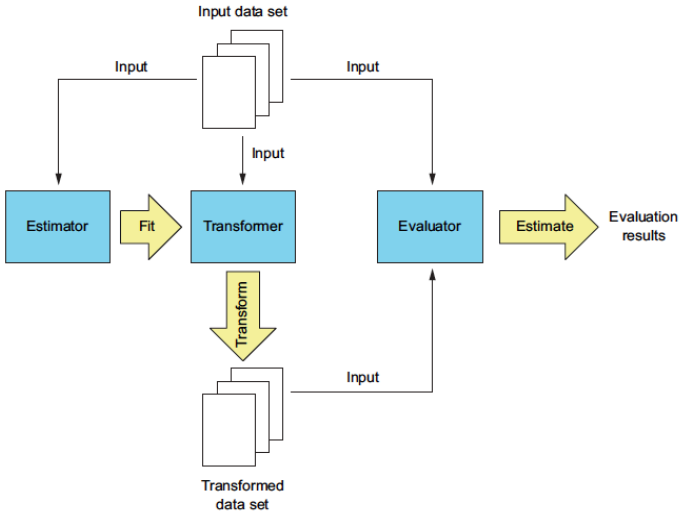


Logistic Regression Example.

* n-fold cross-validation
  - Validate the performance of the model more reliably.
    - Divide the dataset into n subsets of equal sizes and train n models excluding a different subset each time and train all n models.
    - Calculate the mean error for all n models.
    - Choose the set of parameters with the smallest average error.

```
# n-fold validation and the results.
from pyspark.ml.tuning import CrossValidator
from pyspark.ml.tuning import ParamGridBuilder
cv = CrossValidator().setEstimator(lr).setEvaluator(bceval).setNumFolds(5)
paramGrid = ParamGridBuilder().addGrid(lr.maxIter, [1000]).addGrid(lr.regParam, [0.0001, 0.001, 0.005, 0.01, 0.05, 0.1,
cv.setEstimatorParamMaps(paramGrid)
cvmodel = cv.fit(adulttrain)
```

Spark ML – Example 1



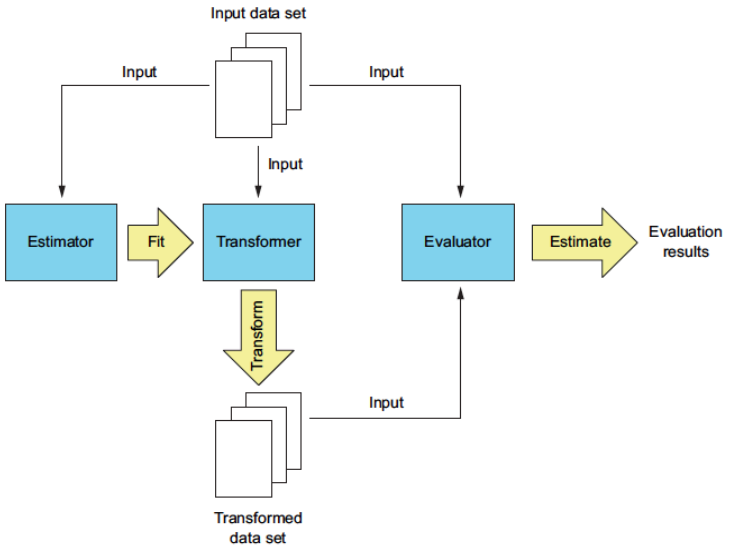
Logistic Regression Example.

* n-fold cross-validation
  - **`CrossValidator(estimator, estimatorParamMaps, evaluator)`**
    - Takes estimator, evaluator and number of folds to use.
    - Takes several parameters in setEstimatorParamMaps()
      - **`ParamGridBuilder()`** – combinations of parameters and their values.

ML ParameterGridBuilder.

```
# n-fold validation and the results.
from pyspark.ml.tuning import CrossValidator
from pyspark.ml.tuning import ParamGridBuilder
cv = CrossValidator().setEstimator(lr).setEvaluator(bceval).setNumFolds(5)
paramGrid = ParamGridBuilder().addGrid(lr.maxIter, [1000]).addGrid(lr.regParam, [0.0001, 0.001, 0.005, 0.01, 0.05, 0.1,
cv.setEstimatorParamMaps(paramGrid)
cvmodel = cv.fit(adulttrain)
```

Spark ML – Example 1



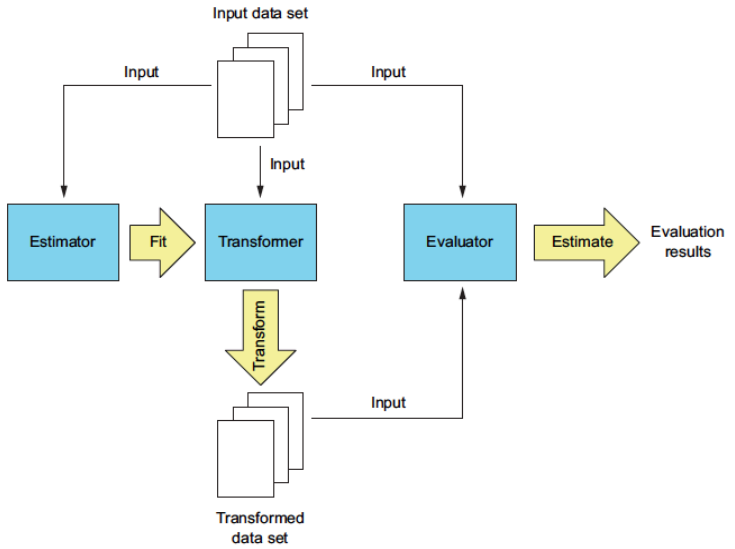
Logistic Regression Example.

* n-fold cross-validation
  - **`CrossValidator(estimator, estimatorParamMaps, evaluator)`**

```
print(cvmodel.bestModel.coefficients)
print(cvmodel.bestModel.intercept)
print(cvmodel.bestModel._java_obj.getMaxIter())
print(cvmodel.bestModel._java_obj.getRegParam())
```

```
[0.021934920773056855,0.00030850400512423955,0.0006103684605904492,7.364208158083989e-07,0.03025707481419387,-0.73911
26434980373,-0.41515148054361445,-0.9085212422581654,-0.43714493065830334,-0.566912208253489,-0.2640193838222534,0.17
874854493322762,-1.002539250858143,-3.016038561311831,-0.6144720759629188,-0.20702644617498422,0.5634544957535148,0.9
916079557291828,-0.04625270847297603,-1.376304368602061,-0.006769168908898807,-1.5303480503018725,-2.00105860915664
5,1.6717431297724037,-1.7249989941412194,-1.033176310185123 5,1.60515841451851,-1.5802786396497435,-2.228985770963519,
-2.880257726790732,1.3003323613573357,-1.4752977398706573,-1.0430617947144387,-1.1086213828740774,-0.908216131474305
1,-0.8854042982867404,1.6345770768213554,-0.01974376178802173,-0.15620052611969132,0.5100655051317843,-0.255930838683
2983,0.01655204275448495,-1.0835233601323067,-0.5081685773304384,-0.34947118011101364,-0.8446862314587074,-1.16009144
60191314,0.3549003581911287,0.15613930188120895,-1.7959338744180884,0.4563749243045909,-0.4562651236433301,0.15254703
6189287,-0.9611087052980742,-0.004127105143589575,0.7380876119309817,-0.7919609877812239,-0.9685525314990799,-1.87407
5702077144,-0.995013595011799,-1.03901589497575,-1.2686645915008945,-0.5163500064607778,-2.110893969375521,-1.1845216
083457977,-0.9416126214571209,-0.7469597390800851,-1.5887361966684377,-2.309808259907228,-0.7954827188408711,-0.50838
51549010496,-2.299587290861013,-1.2137476389896529,-1.4825389896815764,-1.1601722419800475,-2.0928621888320666,-3.548
138798793347,-0.7470158710302366,-0.5064666292351641,-1.1196187901384622,-1.3105214930153253,-1.2556122791267421,-2.9
95661869878692,-1.793085715839111,-1.4901723268578941,-0.041990025645952764,0.1762080380088669,-2.455974449863295,-1.
98550272839025,-0.24859686847368176,-3.101831351723714,-0.23462144690717848,-2.2952405374058387,-1.630512397795353,-
2.0060832007694342,-1.1309945575430274,-0.9863461669921886,-4.738822215280169,-0.6625666353939309,-0.938416751841870
8,-0.47164679238652496,-1.1973346428384923,-0.7182528682365087]
-1.386090304179345
1000
0.0001
```

Spark ML – Example 1



Logistic Regression Example.

* n-fold cross-validation
  - **`CrossValidator(estimator, estimatorParamMaps, evaluator)`**
    - You can access the selected model through `bestModel`.

```
BinaryClassificationEvaluator().evaluate(cvmodel.bestModel.transform(adultvalid))
```

```
0.9039037104278022
```

```
BinaryClassificationEvaluator().setMetricName("areaUnderPR")\
                               .evaluate(cvmodel.bestModel.transform(adultvalid))
```

```
0.7571472121238862
```

```
BinaryClassificationEvaluator().setMetricName("areaUnderROC")\
                               .evaluate(cvmodel.bestModel.transform(adultvalid))
```

```
0.9039037104277987
```

Contents

Machine Learning With Spark (Spark ML)

- Main Components
- Algorithms
- Logistic Regression
- Decision Tree
- Random Forest
- K-Mean Clustering

Reference

Spark Online Documentation : http://spark.apache.org/docs/latest/

Karau, Holden, et al. *Learning spark: lightning-fast big data analysis*. O'Reilly Media, Inc., 2015.

Zecevic, Petar, et al. Spark in Action, Manning, 2016.

Appendix

Loading/Writing Data using SparkSQL

Data Sources

1. S3
   - Will use s3a to access data in S3 with configurations.
     - s3a: Hadoop's client which offers high-performance IO against Amazon S3 object store.
     - Configurations
       - JAR files - add aws-java-sdk-bundle and hadoop-aws jars.

```
ss = SparkSession.builder.config("spark.jars.packages",
        "com.amazonaws:aws-java-sdk-bundle:1.11.901,
         org.apache.hadoop:hadoop-aws:3.3.1").getOrCreate()
```

       - Add AWS Access Key and Secret Key.

```
ss._jsc.hadoopConfiguration().set("fs.s3a.access.key","ACCESS_KEY")
ss._jsc.hadoopConfiguration().set("fs.s3a.secret.key","SECRET_KEY")
```

   - Use "s3a://bucket/path" to read/write data in a bucket.

https://hadoop.apache.org/docs/current/hadoop-aws/tools/hadoop-aws/
https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk

Example 1

Read data from world_bank_project.json and write back as parquet from/to usfca-msan694, a s3 bucket.

Expected Output

```
|{52b213b38594d8a2...|      1999|            November|2013-11-12T00:00:00Z|FEDERAL DEMOCRAT
I...|2018-07-07T00:00:00Z|Federal Democrati...|         ET|Federal Democrati...|          Eth
iopia|Project Informati...|                        C|       0|          0|P129828| 130000000|M
INISTRY OF EDUCA...|Investment Projec...|              IN|      550000000|[{Education, 4
6},...|[{EX, Education},...| [Human development]|[{8, Human develo...|       8,11|      PE|
IBRD/IDA|             L|{The development ...|Ethiopia General ...|[{28-AUG-2013, PI...|
IDA|             Active|             Africa|[{Primary educati...|{Primary educatio...|{Seco
ndary educat...|{Public administr...|{Tertiary educati...|[{EP, Primary edu...|ET,BS,ES,EP|
IBRD|Active|                   N|{Education for al...|[{65, Education f...|         65|130000
000|   130000000|http://www.worldb...|
|{52b213b38594d8a2...|      2015|            November|2013-11-04T00:00:00Z|GOVERNMENT OF TU
N...|               null|Republic of Tunis...|         TN| Republic of Tunisia|           Tu
nisia|Project Informati...|                        C| 4700000|          0|P144674|         0|
MINISTRY OF FINANCE|Specific Investme...|              IN|        5700000|[{Public Adminis
t...|[{BX, Public Admi...|[Economic managem...|[{1, Economic man...|        1,6|      RE|Reci
pient Execute...|             L|                null|TN: DTF Social Pr...|[{29-MAR-2013, P
I...|              OTHER|             Active|Middle East and N...|[{Public administ...|{Pub
lic administr...|{General public a...|               null|                null|[{BS, Public
admi...|      BZ,BS|  IBRD|Active|                  N|{Other economic m...|[{24, Other econ
o...|      54,24|        0|     4700000|http://www.worldb...|
|{52b213b38594d8a2...|      2014|            November|2013-11-01T00:00:00Z|MINISTRY OF FINA
N...|               null|         Tuvalu!$!TV|         TV|              Tuvalu|            T
uvalu|Resettlement Plan...|                        B|       0|          0|P145310|   6060000|M
INISTRY OF TRANS...|Investment Projec...|              IN|        6060000|[{Transportatio
n,...|[{TX, Transportat...|[Trade and integr...|[{5, Trade and in...|   5,2,11,6|      PE|
```

Example 1

Read data from world_bank_project.json and write back as parquet from/to usfca-msan694, a s3 bucket.

```
from pyspark.sql import SparkSession
from pyspark.sql.types import *

#https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-aws/3.3.1 for pyspark 3.2.0
ss = SparkSession.builder.config("spark.jars.packages", "com.amazonaws:aws-java-sdk-bundle:1.11
                                                         org.apache.hadoop:hadoop-aws:3.3.1").g
```

```
ss._jsc.hadoopConfiguration().set("fs.s3a.access.key","ACCESS_KEY")
ss._jsc.hadoopConfiguration().set("fs.s3a.secret.key","SECRET_KEY")
```

Example 1

Read data from world_bank_project.json and write back as parquet from/to usfca-msan694, a s3 bucket.

```
world_bank = ss.read.json("s3a://usfca-msan694/world_bank_project.json")
```

executed in 6.75s, finished 10:54:31 2021-01-31

```
world_bank.show(5)
```

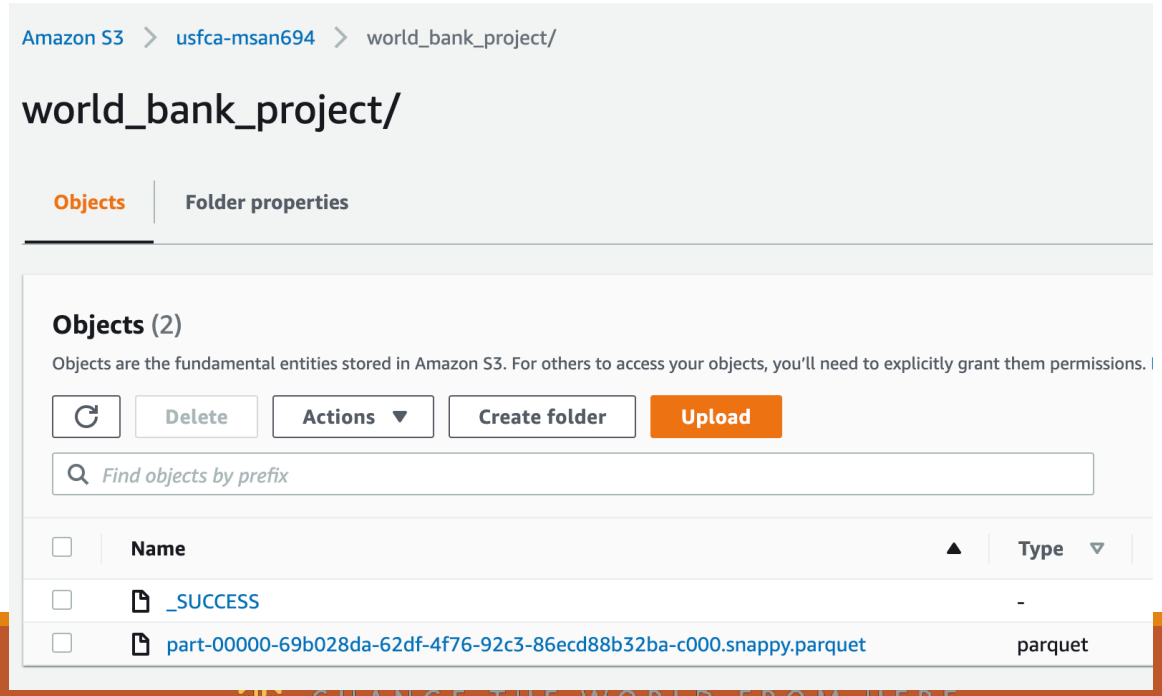
executed in 709ms, finished 10:54:32 2021-01-31

```
+--------------------+----------+--------------------+--------------------+--------------------
-+--------------------+--------------------+-----------+--------------------+----------------
-+--------------------+-----------------------+--------+-----------+-------+----------+-----
--------------+--------------------+---------------+---------------+--------------------+--
-----------------+--------------------+--------------------+-----------+--------+-----------
---------+--------------+--------------------+--------------------+--------------------+----
----------------+--------------------+--------------------+--------------------+------------
-------+--------------------+--------------------+--------------------+--------------------+-
----------+------+------+--------------------+--------------------+--------------------+-----
------+---------+------------+--------------------+
|                 _id|approvalfy|board_approval_month|  boardapprovaldate|            borrowe
r|        closingdate|    country_namecode|countrycode|         countryname|  countryshortnam
e|               docty|envassesmentcategorycode|grantamt|ibrdcommamt|     id|idacommamt|
impagency|        lendinginstr|lendinginstrtype|lendprojectcost| majorsector_percent|   mjsec
tor_namecode|            mjtheme|    mjtheme_namecode|mjthemecode|prodline|        prodlinet
ext|productlinetype|    project_abstract|        project_name|         projectdocs|projectfin
ancialtype|projectstatusdisplay|          regionname|              sector|             sector
1|             sector2|             sector3|             sector4|     sector_namecode| sector
code|source|status|supplementprojectflg|              theme1|      theme_namecode|  themecode
```

Example 1

Read data from world_bank_project.json and write back as parquet from/to usfca-msan694, a s3 bucket.

```
world_bank.write.parquet("s3a://usfca-msan694/world_bank_project")
```



Loading/Writing Data using SparkSQL

Data Sources

- Relational Databases and Other DB (MongoDB) with JDBC
  - Postgres and others (MySQL) using JDBC
    - Spark can load data from any relational databases that supports Java Database Connectivity (JDBC) including MySQL, Postgres, and other systems.
      - Ex. Postgres JDBC jar : https://jdbc.postgresql.org/download.html

Contents

Loading/Writing Data using SparkSQL

- Data Sources

1. S3
2. **Relational Databases and Other DB (MongoDB) with JDBC**

Machine Learning With Spark (Spark ML)

- Main Components
- Algorithms
- Logistic Regression
- Decision Tree
- Random Forest
- K-Mean Clustering

Loading/Writing Data using SparkSQL

Loading Data with SparkSQL

2. MongoDB
   - Provide the mongo-spark-connector package as a packages parameter value.

     `org.mongodb.spark:mongo-spark-connector_2.12:VERSION_NUMBER`

   - To read as a DataFrame, provide mongodb.input.uri and its URI, DB and collection.
   - To write a DataFrame, provide mongodb.output.uri and its URI, DB and collection.

```
spark = SparkSession \
    .builder \
    .appName("myApp") \
    .config("spark.jars.packages", "org.mongodb.spark:mongo-spark-connector_2.12:2.4.0")\
    .config("spark.mongodb.input.uri", "mongodb+srv://students:usfmsds694@example.m9n4r.mong
    .config("spark.mongodb.output.uri", "mongodb+srv://students:usfmsds694@example.m9n4r.mor
    .config("spark.network.timeout", "3600s")\
    .getOrCreate()
```

Loading/Writing Data using SparkSQL

Loading Data with SparkSQL

2. MongoDB
   To read as a DataFrame provide mongodb.input.uri and its URI, DB and collection.
   - Reading with "`com.mongodb.spark.sql.DefaultSource`"
     - `ss.read.format("com.mongodb.spark.sql.DefaultSource").load()`
   - To write a dataframe, provide mongodb.output.uri and its URI, DB and collection.

```
df.write.format("com.mongodb.spark.sql.DefaultSource")\
        .mode("overwrite")\
        .option("database","db_name")\
        .option("collection", "collection_name")\
        .save()
```

Example 2

1) Read from MongoDB using mongodb+srv://user1:user1@day8.qh1ug.mongodb.net/msds697.business

2) Filter business in “CA” and write to the filtered_business in MongoDB.

Expected Output

_id	business	city	state	street	supervisor	zip
{6213bdd228b3f46a...	Tournahu George L	San Francisco	CA	3301 Broderick St	2	94123
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	2225 Jerrold Ave	10	94124
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	2225 Jerrold Ave	9	94124
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	180 New Montgomer...	3	94105
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	180 New Montgomer...	6	94105
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	540 Powell St	3	94108
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	540 Powell St	6	94108
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	460 Townsend St	6	94107
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	460 Townsend St	10	94107
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	1835-49 Van Ness Ave	5	94109
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	1835-49 Van Ness Ave	3	94109
{6213bdd228b3f46a...	Stephens Institut...	San Francisco	CA	1835-49 Van Ness Ave	6	94109

Example 2

1) Read from MongoDB using mongodb+srv://user1:user1@day8.qh1ug.mongodb.net/msds697.business

2) Filter business in “CA” and write to the filtered_business in MongoDB.

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("myApp") \
    .config("spark.jars.packages", "org.mongodb.spark:mongo-spark-connector_2.12:2.4.0")\
    .config("spark.mongodb.input.uri", "mongodb+srv://students:usfmsds694@example.m9n4r.mongodb
    .config("spark.mongodb.output.uri", "mongodb+srv://students:usfmsds694@example.m9n4r.mongod
    .config("spark.network.timeout", "3600s")\
    .getOrCreate()
```

Example 2

1) Read from MongoDB using mongodb+srv://user1:user1@day8.qh1ug.mongodb.net/msds697.business

```
df = spark.read.format('com.mongodb.spark.sql.DefaultSource').load()
```

Example 2

2) Filter business in “CA” and write to the filtered_business in MongoDB.

```
filtered_df = df.filter("state == 'CA'")
```

```
filtered_df.show()
```

```
+--------------------+--------------------+-------------+-----+--------------------+----------+-----+
|                 _id|            business|         city|state|              street|supervisor|  zip|
+--------------------+--------------------+-------------+-----+--------------------+----------+-----+
|{6213bdd228b3f46a...|   Tournahu George L|San Francisco|   CA|   3301 Broderick St|         2|94123|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|    2225 Jerrold Ave|        10|94124|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|    2225 Jerrold Ave|         9|94124|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|180 New Montgomer...|         3|94105|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|180 New Montgomer...|         6|94105|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|       540 Powell St|         3|94108|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|       540 Powell St|         6|94108|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|     460 Townsend St|         6|94107|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|     460 Townsend St|        10|94107|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|1835-49 Van Ness Ave|         5|94109|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|1835-49 Van Ness Ave|         3|94109|
|{6213bdd228b3f46a...|Stephens Institut...|San Francisco|   CA|1835-49 Van Ness Ave|         6|94109|
```

Loading/Writing Data using SparkSQL

Data Sources

- Relational Databases
  - Postgres and others (MySQL) using JDBC
    - Spark can load data from any relational databases that supports Java Database Connectivity (JDBC) including MySQL, Postgres, and other systems.
      - Download a Postgres JDBC jar from https://jdbc.postgresql.org/download.html
      - Include .jar on extraClassPath

```
ss =
SparkSession.builder.config('spark.driver.extraClassPath','postgresql-42.2.18.jar').getOrCreate()
```

      - Read : Use ss.read.jdbc(url, table, properties)
      - Write: Use df.write.jdbc(url, table, properties)

http://spark.apache.org/docs/latest/sql-programming-guide.html
https://jdbc.postgresql.org/download.html
https://spark.apache.org/docs/latest/sql-data-sources-jdbc.html

Loading/Writing Data using SparkSQL

Data Sources

- Relational Databases
  - Postgres and others (MySQL) using JDBC
    - Read : Use ss.read.jdbc(url, table, properties)

```
endpoint = 'msds694.cmxsootjz10m.us-west-2.rds.amazonaws.com'
database = 'postgres'
table = 'activity_code'
properties = {'user': 'students', 'password': 'msdsstudents'}
url = 'jdbc:postgresql://%s/%s'%(endpoint, database)

activity_code = ss.read.jdbc(url=url, table=table, properties=properties)
```

    - Write: Use df.write.jdbc(url, table, properties)

```
table = 'business'
business_df.write.jdbc(url=url, table=table, properties=properties)
```

http://spark.apache.org/docs/latest/sql-programming-guide.html
https://jdbc.postgresql.org/download.html
https://spark.apache.org/docs/latest/sql-data-sources-jdbc.html

Example

Using

- Endpoint : msds694.cmxsootjz10m.us-west-2.rds.amazonaws.com
- Database : postgres
- User : students
- Password : msdsstudents

Read data from the activity_code table.

Create a dataframe using filtered_registered_business_sf.csv and write to the business table.

Example

Using

- Endpoint : msds694.cmxsootjz10m.us-west-2.rds.amazonaws.com
- Database : postgres
- User : students
- Password : msdsstudents

```
from pyspark.sql import SparkSession
from pyspark.sql.types import *

ss = SparkSession.builder.config('spark.driver.extraClassPath',
                                 'postgresql-42.2.18.jar').getOrCreate()
```

executed in 2.83s, finished 10:31:01 2021-01-31

```
endpoint = 'msds694.cmxsootjz10m.us-west-2.rds.amazonaws.com'
database = 'postgres'
table = 'activity_code'
properties = {'user': 'students', 'password': 'msdsstudents'}
url = 'jdbc:postgresql://%s/%s' % (endpoint, database)
```

Example

Using

- Endpoint : msds694.cmxsootjz10m.us-west-2.rds.amazonaws.com
- Database : postgres
- User : students
- Password : msdsstudents

Read data from the activity_code table.

```
activity_code = ss.read.jdbc(url=url, table=table, properties=properties)
activity_code.show()
```

executed in 2.84s, finished 10:31:04 2021-01-31

```
+--------------------+----+
|            activity|code|
+--------------------+----+
|             Walking|   A|
|             Jogging|   B|
|              Stairs|   C|
|             Sitting|   D|
|            Standing|   E|
```

Example

Using

- Endpoint : msds694.cmxsootjz10m.us-west-2.rds.amazonaws.com
- Database : postgres
- User : students
- Password : msdsstudents

Create a dataframe using filtered_registered_business_sf.csv and write to the business table.

```
table = 'business'
business_df.write.jdbc(url=url, table=table, properties=properties)
```