Distributed Computing

DIANE WOODBRIDGE, PH.D

Announcement

Individual Assignment

- HW 4 - March 5th (Extended)
  - Please do not plagiarize!! We’re almost towards to the end of the module.

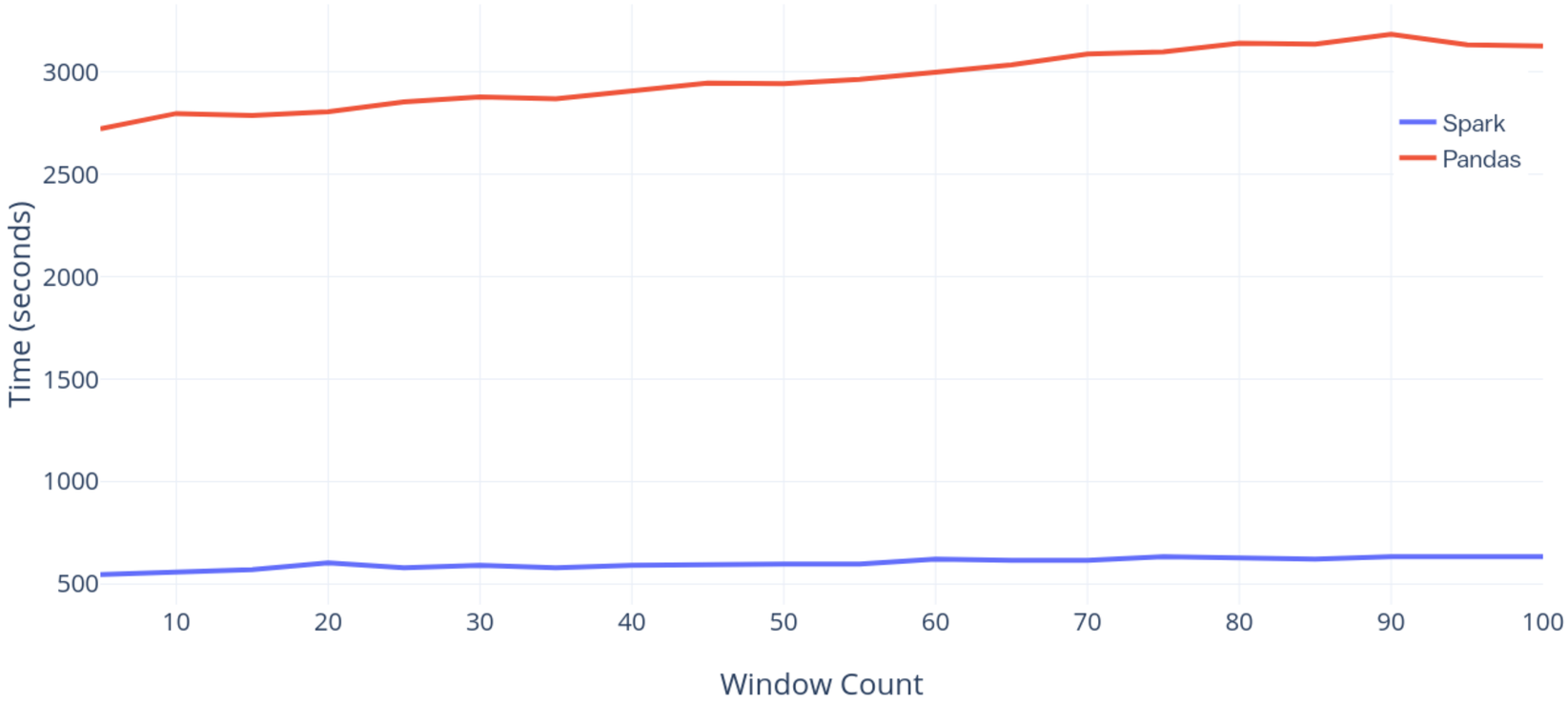
Group Assignment

- Task 3 - March 10th

Quiz 3

- March 7 - 9 AM

Spark ML vs Pandas for Data Preprocessing



Woodbridge, Diane Myung-Kyung, and Kevin Bengtson Wong. "A scalable medication intake monitoring system." In *Big Data in Psychiatry# x0026; Neurology*, pp. 217-240. Academic Press, 2021.

Contents

Machine Learning With Spark (Spark ML)

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

Spark ML

Data Sets

- Optical Recognition of Handwritten Digits Data Sets.
  - 10992 instances.
  - Attribute Information:
    - 16 pixels with intensity values in the range 0..100.
    - The last attribute is the class code 0..9

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

Contents

Machine Learning With Spark (Spark ML)

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

Spark ML – Example 1

Decision Tree

- Pros
  - Do not require data normalization, can handle numerical/categorical values, and work with missing values.
- Cons
  - Prone to overfitting and is sensitive to the input data.

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

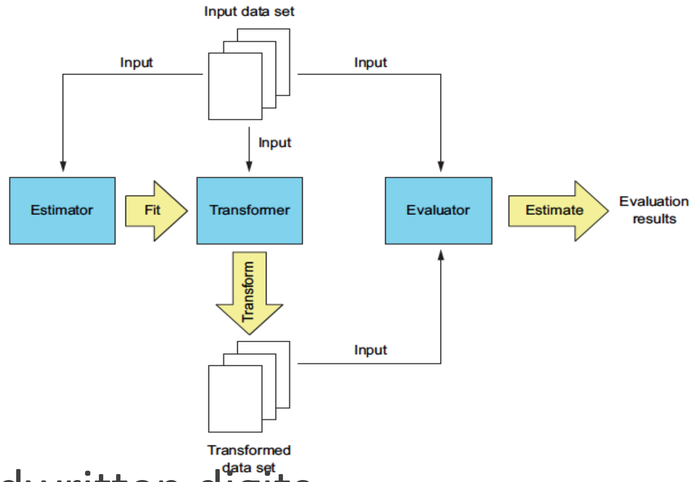
Spark ML – Example 1

Decision Tree

- `DecisionTreeClassifier()` - Binary Decision Tree classifier.
  - Input
    1. `features` - Feature vector.
    2. `label` - Label to predict.
  - Output
    1. prediction – Predicted label.
    2. rawPrediction - Vector of length # classes, with the counts of training instance labels at the tree node which makes the prediction.
    3. probability - Vector of length # classes equal to rawPrediction normalized to a multinomial distribution

https://spark.apache.org/docs/2.1.0/ml-classification-regression.html#decision-trees

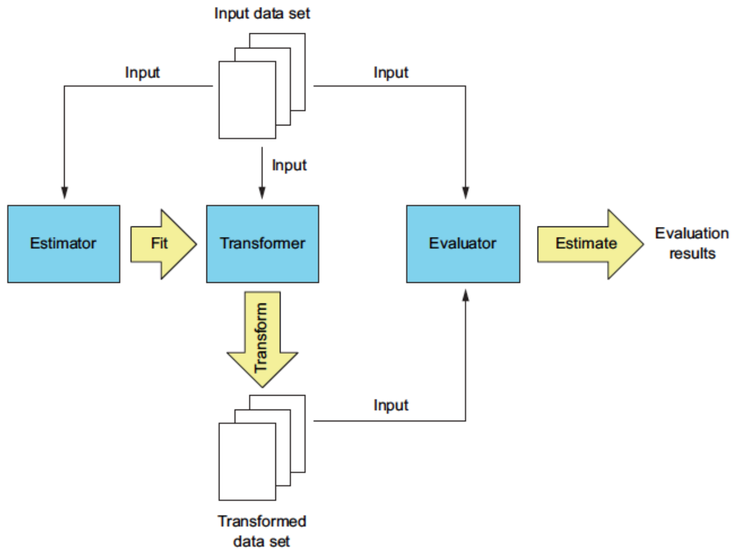
Spark ML – Example 1



Decision Tree Example

- Optical Recognition of Handwritten Digits Data Sets. : Classify handwritten digits.
  - 10992 instances.
  - Attribute Information:
    - 16 pixels with intensity values in the range 0..100.
    - The last attribute is the class code (label) 0..9

Spark ML – Example 1

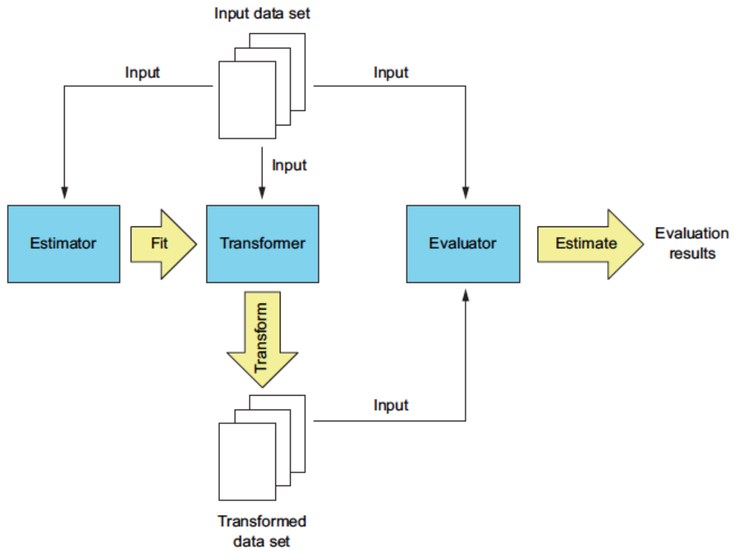


Decision Tree Example

1. Create an RDD.
2. Convert the RDD to DataFrame.
3. Clean the data.
4. Split Data into training and test sets.
5. Train the data.
6. Evaluate the model.

And then create a pipeline.

Spark ML – Example 1



Decision Tree Example

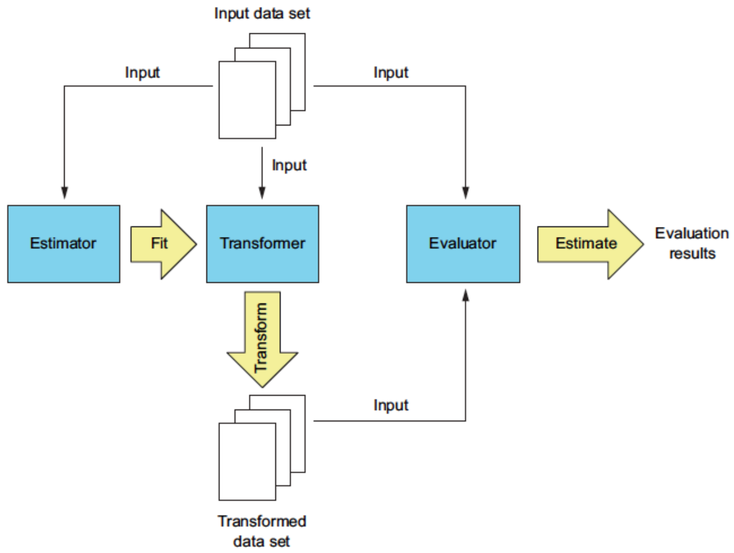
1. Create an RDD.

```
#Load the data and create an RDD (16 pixels and label)
pen_raw = sc.textFile("../Data/penbased.dat", 4)\
            .map(lambda x:  x.split(", "))\
            .map(lambda row: [float(x) for x in row])
pen_raw.take(1)
```

executed in 1.74s, finished 08:34:07 2021-02-13

```
[[47.0,
  100.0,
  27.0,
  81.0,
  57.0,
  37.0,
  26.0,
  0.0,
  0.0,
  23.0,
  56.0,
  53.0,
  100.0,
  90.0,
  40.0,
  98.0,
  8.0]]
```

Spark ML – Example 1



Decision Tree Example

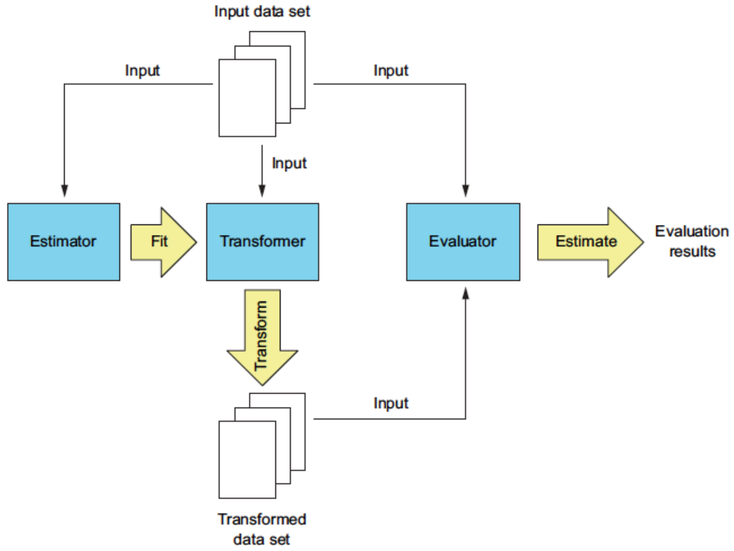
2. Convert the RDD to DataFrame

```
#Create a DataFrame
from pyspark.sql.types import *

penschema = StructType([
    StructField("pix1",DoubleType(),True),
    StructField("pix2",DoubleType(),True),
    StructField("pix3",DoubleType(),True),
    StructField("pix4",DoubleType(),True),
    StructField("pix5",DoubleType(),True),
    StructField("pix6",DoubleType(),True),
    StructField("pix7",DoubleType(),True),
    StructField("pix8",DoubleType(),True),
    StructField("pix9",DoubleType(),True),
    StructField("pix10",DoubleType(),True),
    StructField("pix11",DoubleType(),True),
    StructField("pix12",DoubleType(),True),
    StructField("pix13",DoubleType(),True),
    StructField("pix14",DoubleType(),True),
    StructField("pix15",DoubleType(),True),
    StructField("pix16",DoubleType(),True),
    StructField("label",DoubleType(),True)
])

dfpen = ss.createDataFrame(pen_raw, penschema)
```

Spark ML – Example 1



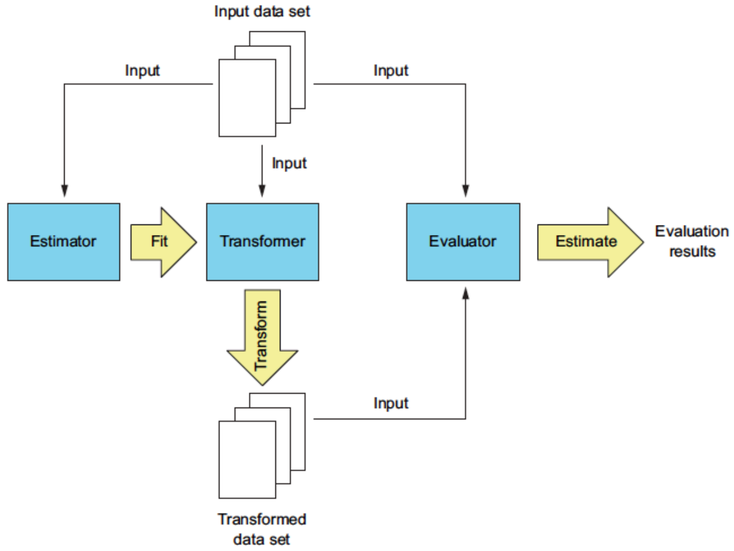
Decision Tree Example

2. Convert the RDD to DataFrame

```
dfpen.show()
```

```
+-----+-----+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| pix1| pix2|pix3| pix4| pix5| pix6| pix7| pix8| pix9|pix10|pix11|pix12|pix13|pix14|pix15|pix16|label|
+-----+-----+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 47.0|100.0|27.0| 81.0| 57.0| 37.0| 26.0|  0.0|  0.0| 23.0| 56.0| 53.0|100.0| 90.0| 40.0| 98.0|  8.0|
|  0.0| 89.0|27.0|100.0| 42.0| 75.0| 29.0| 45.0| 15.0| 15.0| 37.0|  0.0| 69.0|  2.0|100.0|  6.0|  2.0|
|  0.0| 57.0|31.0| 68.0| 72.0| 90.0|100.0|100.0| 76.0| 75.0| 50.0| 51.0| 28.0| 25.0| 16.0|  0.0|  1.0|
|  0.0|100.0| 7.0| 92.0|  5.0| 68.0| 19.0| 45.0| 86.0| 34.0|100.0| 45.0| 74.0| 23.0| 67.0|  0.0|  4.0|
|  0.0| 67.0|49.0| 83.0|100.0|100.0| 81.0| 80.0| 60.0| 60.0| 40.0| 40.0| 33.0| 20.0| 47.0|  0.0|  1.0|
|100.0|100.0|88.0| 99.0| 49.0| 74.0| 17.0| 47.0|  0.0| 16.0| 37.0|  0.0| 73.0| 16.0| 20.0| 20.0|  6.0|
|  0.0|100.0| 3.0| 72.0| 26.0| 35.0| 85.0| 35.0|100.0| 71.0| 73.0| 97.0| 65.0| 49.0| 66.0|  0.0|  4.0|
|  0.0| 39.0| 2.0| 62.0| 11.0|  5.0| 63.0|  0.0|100.0| 43.0| 89.0| 99.0| 36.0|100.0|  0.0| 57.0|  0.0|
| 13.0| 89.0|12.0| 50.0| 72.0| 38.0| 56.0|  0.0|  4.0| 17.0|  0.0| 61.0| 32.0| 94.0|100.0|100.0|  5.0|
| 74.0| 87.0|31.0|100.0|  0.0| 69.0| 62.0| 64.0|100.0| 79.0|100.0| 38.0| 84.0|  0.0| 18.0|  1.0|  9.0|
| 48.0| 96.0|62.0| 65.0| 88.0| 27.0| 21.0|  0.0| 21.0| 33.0| 79.0| 67.0|100.0|100.0|  0.0| 85.0|  8.0|
|100.0|100.0|72.0| 99.0| 36.0| 78.0| 34.0| 54.0| 79.0| 47.0| 64.0| 13.0| 19.0|  0.0|  0.0|  2.0|  5.0|
```

Spark ML – Example 1



Decision Tree Example

3. Clean the data.

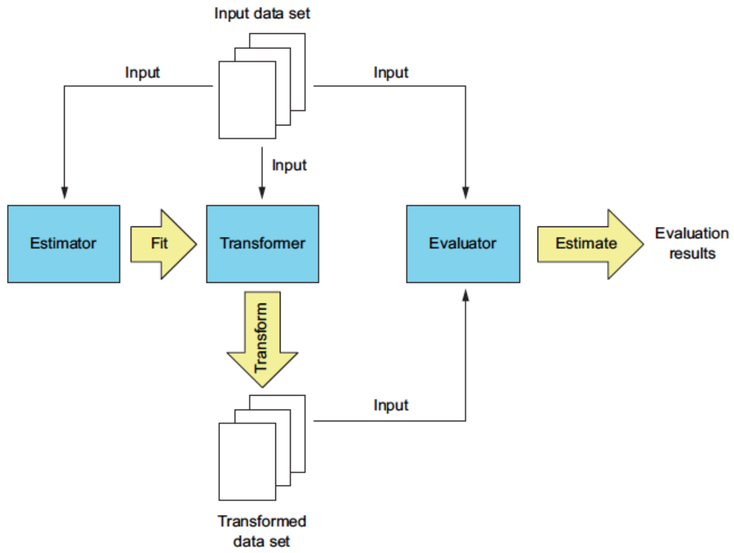
- Create a feature vector.

```
# Merging the data with Vector Assembler.
from pyspark.ml.feature import VectorAssembler
va = VectorAssembler(outputCol="features", inputCols=dfpen.columns[0:-1])
penlpoints = va.transform(dfpen).select("features", "label")
```

```
penlpoints.show()
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[47.0,100.0,27.0,...|  8.0|
|[0.0,89.0,27.0,10...|  2.0|
|[0.0,57.0,31.0,68...|  1.0|
|[0.0,100.0,7.0,92...|  4.0|
```

Spark ML – Example 1



Decision Tree Example

4. Split Data into training and test sets.

```
# Create Training and Test data.
pendtsets = penlpoints.randomSplit([0.8, 0.2])
pendttrain = pendtsets[0].cache()
pendtvalid = pendtsets[1].cache()
```

When poll is active, respond at **pollev.com/msds**

Text **MSDS** to **37607** once to join

df.cache() improves efficiency for iterative algorithms.

True

False

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

When poll is active, respond at **pollev.com/msds**

Text **MSDS** to **37607** once to join

df.cache() improves efficiency for iterative algorithms.

True

False

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Poll locked. Responses not accepted.

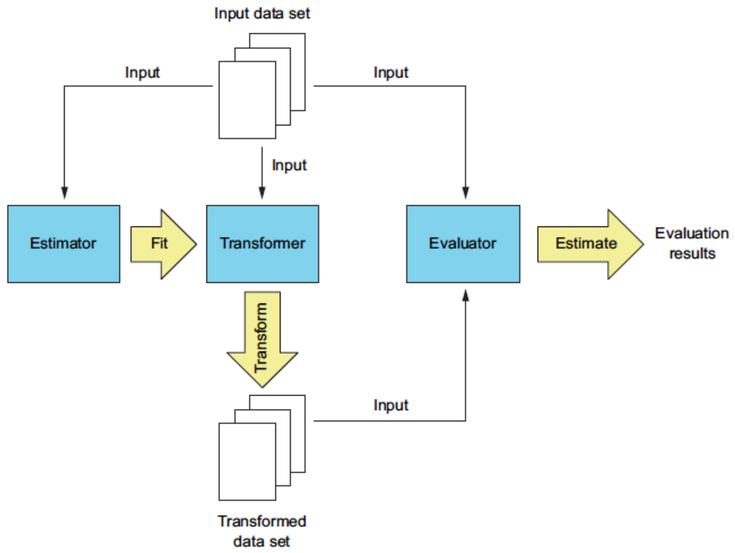
df.cache() improves efficiency for iterative algorithms.

True

False

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Spark ML – Example 1



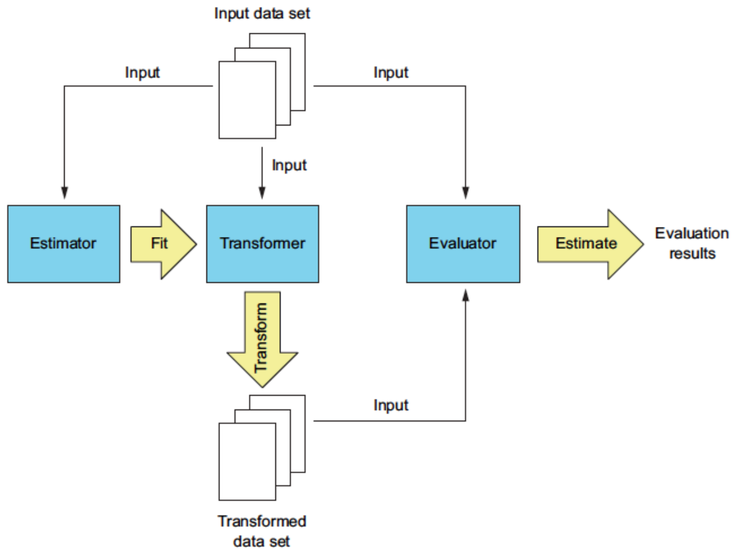
Decision Tree Example

5. Train the data.
   - Parameters for DecisionTreeClassifier
     - `maxDepth` : maximum tree depth (default : 5).
     - `maxBins` : maximum number of bins when binning continuous features (default : 32).
     - `minInstancesPerNode` : minimum number of dataset samples each branch needs to have after a split (default : 1).
     - `minInfoGain` : minimum information gain for a split (default : 0).

```
# Train the data.
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(maxDepth=20, maxBins= 32, minInstancesPerNode=1, minInfoGain = 0)
dtmodel = dt.fit(pendttrain)
```

https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.classification.DecisionTreeClassificationModel

Spark ML – Example 1



Decision Tree Example

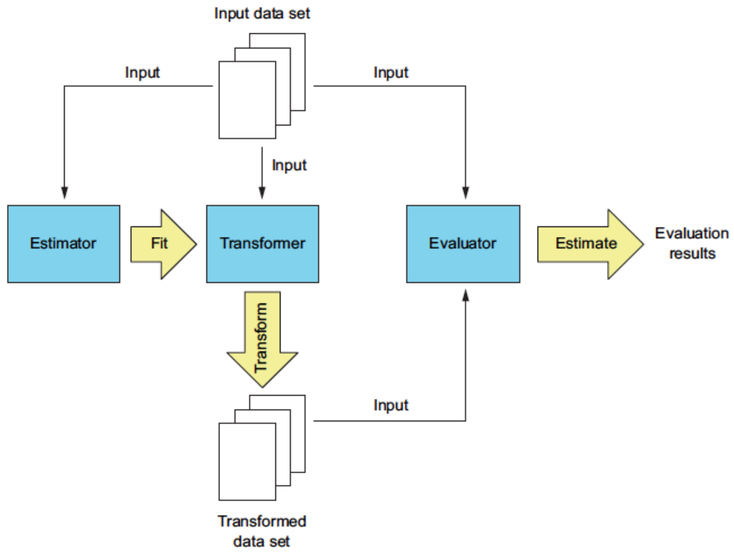
5. Train the data.
   - See the trained model.

```
dtmodel.featureImportances
```

executed in 19ms, finished 08:34:13 2021-02-13

```
SparseVector(16, {0: 0.0892, 1: 0.062, 2: 0.0248, 3: 0.1012, 4: 0.0993, 5: 0.0084, 6: 0.0292,
7: 0.0211, 8: 0.0227, 9: 0.1318, 10: 0.1269, 11: 0.0068, 12: 0.0403, 13: 0.0282, 14: 0.0572,
15: 0.1512})
```

Spark ML – Example 1



Decision Tree Example

5. Train the data.
   - See the trained model.

```
print(dtmodel.toDebugString)
```

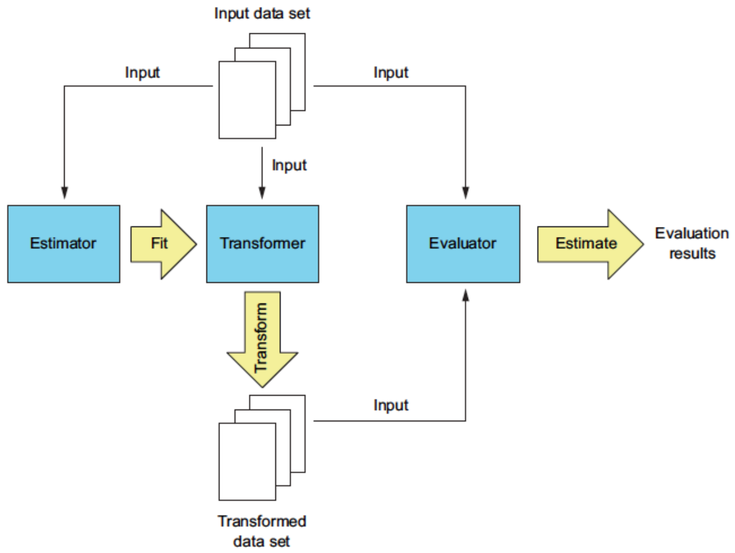
executed in 7ms, finished 08:34:13 2021-02-13

```
DecisionTreeClassificationModel: uid=DecisionTreeClassifier_d5b751448e0a, depth=20, numNodes=
589, numClasses=10, numFeatures=16
  If (feature 15 <= 51.5)
   If (feature 4 <= 41.5)
    If (feature 9 <= 16.5)
     If (feature 14 <= 63.5)
      Predict: 6.0
     Else (feature 14 > 63.5)
      If (feature 12 <= 55.5)
       If (feature 0 <= 38.5)
        Predict: 1.0
       Else (feature 0 > 38.5)
        Predict: 8.0
      Else (feature 12 > 55.5)
       If (feature 11 <= 9.5)
        If (feature 0 <= 20.5)
         Predict: 2.0
        Else (feature 0 > 20.5)
         Predict: 6.0
```

Spark ML – Example 1

Decision Tree Example

6. Evaluate the model.




```
#Test data.
dtpredicts = dtmodel.transform(pendtvalid)
```

Spark ML – Example 1

Decision Tree Example

6. Evaluate the model.

- MulticlassClassificationEvaluator()
  - Expects two input columns: prediction and label.
  - Calculates accuracy by defaults and supports f1, weightedPrecision/Recall, accuracy (default), etc.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
# expects two input columns: prediction and label.

# f1|accuracy(defulat)|weightedPrecision|weightedRecall|weightedTruePositiveRate|
# weightedFalsePositiveRate|weightedFMeasure|truePositiveRateByLabel|
# falsePositiveRateByLabel|precisionByLabel|recallByLabel|fMeasureByLabel|
# logLoss|hammingLoss
metric_name = "f1"

metrics = MulticlassClassificationEvaluator()\
                .setLabelCol("label")\
                .setPredictionCol("prediction")
metrics.setMetricName(metric_name)

metrics.evaluate(dtpredicts)
```

executed in 181ms, finished 08:34:14 2021-02-13

```
0.9585890253909967
```

https://spark.apache.org/docs/latest/api/java/org/apache/spark/ml/evaluation/MulticlassClassificationEvaluator.html
https://spark.apache.org/docs/latest/mllib-evaluation-metrics.html

Spark ML – Example 1

Decision Tree Example

6. Evaluate the model with n-fold validation.
   - Using 5 fold validation, find the best model's maxDepth among 5,10,15,20,25 and 30.

```
# n-fold validation and the results.
from pyspark.ml.tuning import CrossValidator
from pyspark.ml.tuning import ParamGridBuilder
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator()\
  |             .setLabelCol("label")\
                .setPredictionCol("prediction")
#ParamGridBuilder() – combinations of parameters and their values.
dt = DecisionTreeClassifier()
paramGrid = ParamGridBuilder().addGrid(dt.maxDepth, [5,10,15,20,25,30]).build()

cv = CrossValidator(estimator=dt,
                    evaluator=evaluator,
                    numFolds=5,
                    estimatorParamMaps=paramGrid)

cvmodel = cv.fit(pendttrain)
dtpredicts = cvmodel.bestModel.transform(pendtvalid)

print("Best Max Depth : %s" % cvmodel.bestModel.getMaxDepth)
print("Accuracy : %s" % evaluator.evaluate(dtpredicts))
```

Spark ML – Example 1

Decision Tree Example

6. Evaluate the model.
   - Check the confusion matrix using mllib (RDD-based)

```
from pyspark.mllib.evaluation import MulticlassMetrics

#prediction and label
prediction_label = rfpredicts.select("prediction", "label").rdd

metrics = MulticlassMetrics(prediction_label)

confusionMetrics = metrics.confusionMatrix()

print("Confusion Metrics = \n%s" % confusionMetrics)
```

executed in 211ms, finished 08:34:53 2021-02-13

```
Confusion Metrics =
DenseMatrix([[223.,   0.,   0.,   0.,   0.,   0.,   1.,   0.,   4.,   0.],
             [  0., 197.,  12.,   3.,   0.,   0.,   2.,   0.,   0.,   0.],
             [  0.,   4., 200.,   0.,   1.,   0.,   1.,   1.,   0.,   0.],
             [  1.,   0.,   1., 185.,   0.,   0.,   0.,   1.,   0.,   3.],
             [  0.,   1.,   0.,   0., 216.,   0.,   1.,   0.,   0.,   2.],
             [  0.,   0.,   0.,   3.,   1., 176.,   2.,   2.,   2.,   9.],
             [  0.,   1.,   0.,   0.,   3.,   1., 192.,   1.,   0.,   0.],
             [  0.,   1.,   3.,   1.,   0.,   2.,   0., 189.,   0.,   0.],
             [  4.,   0.,   0.,   0.,   0.,   0.,   0.,   1., 169.,   0.],
             [  0.,   0.,   0.,   3.,   0.,   3.,   0.,   1.,   0., 178.]])
```

Contents

Machine Learning With Spark (Spark ML)

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

Spark ML

Pipeline

- Make it easier to combine multiple algorithms into a single pipeline, or workflow
- Stages : An ordered array of either a Transformer or an Estimator
  - These stages are run in order
    - For Transformer stages, the `transform()` method is called on the DataFrame
    - For Estimator stages, the `fit()` method is called to produce a Transformer
  - A `Pipeline` is an **estimator** and produces a `PipelineModel`.

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

Spark ML – Example 1 Pipeline Model

Decision Tree Example

- Create a pipeline of VectorAssembler and DecisionTreeClassifier to create a decision tree classifier model.

```
# Create Training and Test data.
pendtsets = dfpen.randomSplit([0.8, 0.2])
pendttrain = pendtsets[0].cache()
pendtvalid = pendtsets[1].cache()
```

```
# Transformer - Vector Assembler.
from pyspark.ml.feature import VectorAssembler
va = VectorAssembler(outputCol="features", inputCols=dfpen.columns[0:-1]) #except the last col.
```

```
# Estimator - DecisionTreeClassifier which creates a transformer (Decision Tree Classifier model)
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(maxDepth=20, maxBins= 32, minInstancesPerNode=1, minInfoGain = 0)
```

```
# Fit the pipeline to training documents.
from pyspark.ml import Pipeline
pipeline = Pipeline(stages=[va,dt])
```

Spark ML – Example 1 Pipeline Model

Decision Tree Example

- Create a pipeline of VectorAssembler and DecisionTreeClassifier to create a decision tree classifier model.
- Cross validate with estimator=pipeline and build a model.

```
from pyspark.ml.tuning import CrossValidator
from pyspark.ml.tuning import ParamGridBuilder
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator()
#ParamGridBuilder() – combinations of parameters and their values.
paramGrid = ParamGridBuilder().addGrid(dt.maxDepth, [5,10,15,20,25,30])\
                              .build()
cv = CrossValidator(estimator=pipeline,
                    evaluator=evaluator,
                    numFolds=5,
                    estimatorParamMaps=paramGrid)
```

Spark ML – Example 1 Pipeline Model

Decision Tree Example

- Create a pipeline of VectorAssembler and DecisionTreeClassifier to create a decision tree classifier model.
- Cross validate with estimator=pipeline and build a model.
- Evaluate the performance.

```
print("Best Max Depth : %s" % cvmodel.bestModel.getMaxDepth)
print("Accuracy : %s" % MulticlassClassificationEvaluator()
      .evaluate(dtpredicts))
```

executed in 42.7s, finished 10:21:29 2021-02-13

```
Best Max Depth : <bound method _DecisionTreeParams.getMaxDepth of DecisionTreeClassificationM
odel: uid=DecisionTreeClassifier_aa6b9bf3b156, depth=20, numNodes=589, numClasses=10, numFeat
ures=16>
Accuracy : 0.9585890253909967
```

Spark ML – Example 1 Pipeline Model

Decision Tree Example

- Create a pipeline of VectorAssembler and DecisionTreeClassifier to create a decision tree classifier model.
- Cross validate with estimator=pipeline and build a model.
- Evaluate the performance.

```
#prediction and label
prediction_label = dtpredicts.select("prediction", "label").rdd

metrics = MulticlassMetrics(prediction_label)

confusionMetrics = metrics.confusionMatrix()

print("Confusion Metrics = \n%s" % confusionMetrics)
```

executed in 201ms, finished 10:21:30 2021-02-13

```
Confusion Metrics =
DenseMatrix([[223.,   0.,   0.,   0.,   0.,   0.,   1.,   0.,   4.,   0.],
             [  0., 197.,  12.,   3.,   0.,   0.,   2.,   0.,   0.,   0.],
             [  0.,   4., 200.,   0.,   1.,   0.,   1.,   1.,   0.,   0.],
             [  1.,   0.,   1., 185.,   0.,   0.,   0.,   1.,   0.,   3.],
             [  0.,   1.,   0.,   0., 216.,   0.,   1.,   0.,   0.,   2.],
             [  0.,   0.,   0.,   3.,   1., 176.,   2.,   2.,   2.,   9.],
             [  0.,   1.,   0.,   0.,   3.,   1., 192.,   1.,   0.,   0.],
             [  0.,   1.,   3.,   1.,   0.,   2.,   0., 189.,   0.,   0.],
             [  4.,   0.,   0.,   0.,   0.,   0.,   0.,   1., 169.,   0.],
             [  0.,   0.,   0.,   3.,   0.,   3.,   0.,   1.,   0., 178.]])
```

Contents

Machine Learning With Spark (Spark ML)

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

Spark ML – Example 2

Random Forest

- Ensemble learning method
  - Train a certain number of decision trees on data randomly sampled from the original data.
    - Avoid overfitting and find a global optima that particular decision tree cannot find on their own.
- Performs well on high dimensional datasets.

Spark ML – Example 2

Random Forest

- `RandomForestClassifier`
  - Parameters (See more parameters on the documentation)
    - `numTrees` : The number of trees to train. Default : 20
    - `featureSubsetStrategy`
      - `all` – use all features.
      - `onethird` – randomly selects 1/3 of the features.
      - `sqrt` – randomly select sqrt(number of features).
      - `log2` – randomly select log2(number of features).
      - `auto` – sqrt for classification and onethird for regression (default)
  - Make sure to have enough memory (configuration)

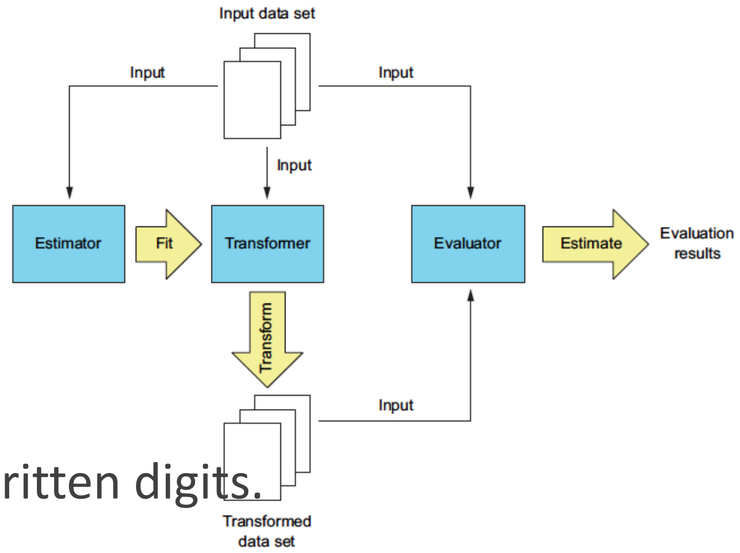
https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.classification.RandomForestClassifier

Spark ML – Example 2

Random Forest

- `RandomForestClassifier`
  - Input
    1. features - Feature vector.
    2. label - Label to predict.
  - Output
    1. prediction – Predicted label.
    2. rawPrediction - Vector of length # classes, with the counts of training instance labels at the tree node which makes the prediction.
    3. probability - Vector of length # classes equal to rawPrediction normalized to a multinomial distribution.

Spark ML – Example 2



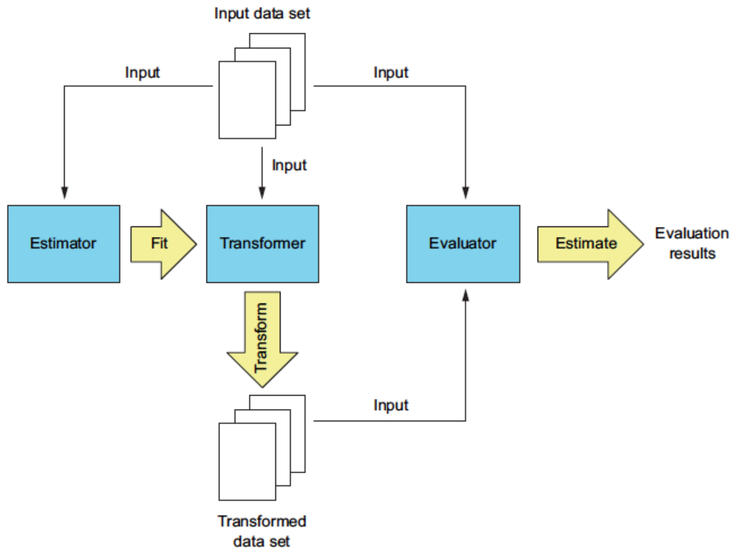
Random Forest Example

- Optical Recognition of Handwritten Digits Data Sets. : Classify handwritten digits.
  - 10992 instances.
  - Attribute Information:
    - 16 pixels with intensity values in the range 0..100.
    - The last attribute is the class code 0..9

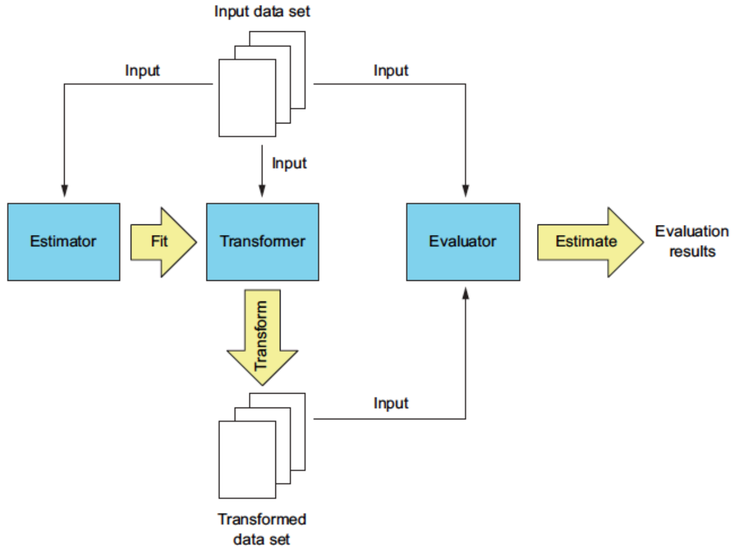
Spark ML – Example 2

Random Forest Example

1. Create an RDD.
2. Convert the RDD to DataFrame.
3. Clean the data.
4. Train.
5. Evaluate the valid dataset.



Spark ML – Example 2



Random Forest Example

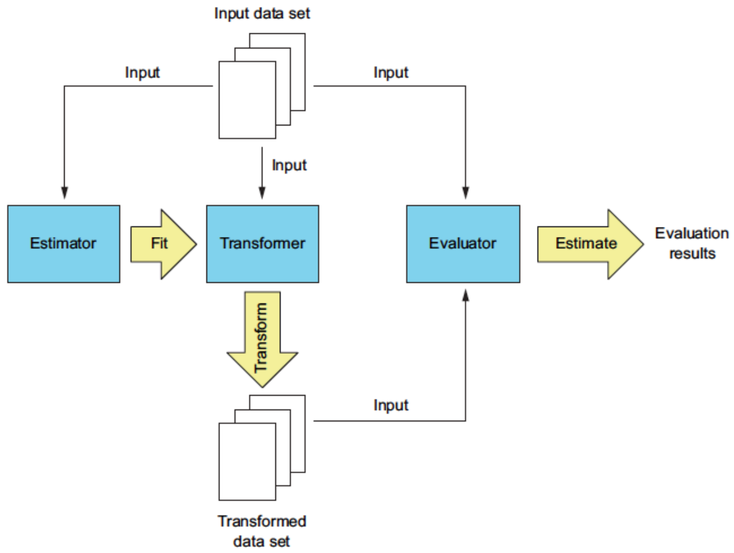
1. Create an RDD.
2. Convert the RDD to DataFrame.
3. Clean the data.

```
# Merging the data with Vector Assembler.
from pyspark.ml.feature import VectorAssembler
va = VectorAssembler(outputCol="features", inputCols=dfpen.columns[0:-1])
penlpoints = va.transform(dfpen).select("features", "label")
```

```
penlpoints.show()
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[47.0,100.0,27.0,...|  8.0|
|[0.0,89.0,27.0,10...|  2.0|
|[0.0,57.0,31.0,68...|  1.0|
```

Spark ML – Example 2

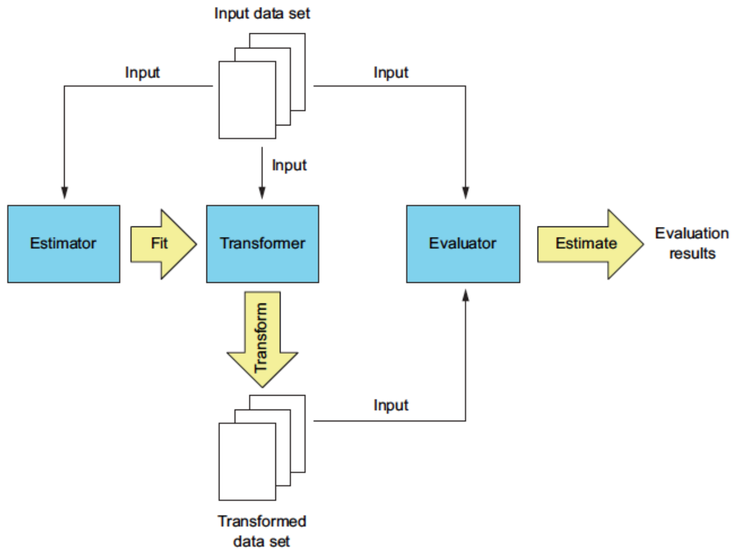


Random Forest Example

4. Train.

```
# Train the model.
from pyspark.ml.classification import RandomForestClassifier
rf = RandomForestClassifier(maxDepth=20)
rfmodel = rf.fit(pendttrain)
```

Spark ML – Example 2



Random Forest Example

4. Train.

```
print(rfmodel.toDebugString)
```

executed in 5.29s, finished 08:37:08 2021-02-13

```
sses=10, numFeatures=16
  Tree 0 (weight 1.0):
    If (feature 13 <= 58.5)
     If (feature 4 <= 41.5)
      If (feature 15 <= 0.5)
       If (feature 5 <= 70.5)
        If (feature 11 <= 33.5)
         If (feature 13 <= 12.5)
          If (feature 14 <= 21.5)
           If (feature 0 <= 7.5)
            Predict: 3.0
           Else (feature 0 > 7.5)
            Predict: 9.0
          Else (feature 14 > 21.5)
           Predict: 2.0
         Else (feature 13 > 12.5)
          If (feature 12 <= 55.5)
           Predict: 6.0
          Else (feature 12 > 55.5)
```

Spark ML – Example 2

Random Forest Example

5. Evaluate the valid dataset.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
# expects two input columns: prediction and label.

metric_name = "f1"
metrics = MulticlassClassificationEvaluator()\
                .setLabelCol("label")\
                .setPredictionCol("prediction")
metrics.setMetricName(metric_name)

metrics.evaluate(rfpredicts)
```

executed in 380ms, finished 08:37:08 2021-02-13

```
0.9830474839122313
```

Worked better than Decision Trees

Contents

Machine Learning With Spark (Spark ML)

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

Spark ML – Example 3

K-mean clustering

- Unsupervised learning
- Dataset should be standardized.
- Example – partition data into groups, anomaly detection, text/topic categorization

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

Spark ML – Example 3

K-mean clustering

- `Kmeans`
  - Parameters (See more on the documentation)
    - `k` : Number of clusters to find (default – 2).
    - `maxIter` : Maximum number of iterations (default – 20).
    - `tol` : Convergence tolerance (default – 0.0001).
    - `seed` : Random seed value for cluster initialization.

https://spark.apache.org/docs/latest/ml-clustering.html
https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.clustering.KMeans

Spark ML – Example 3

K-mean clustering

- `Kmeans`
  - Input
    - features – feature vector
  - Output
    - prediction – predicted cluster center

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

When poll is active, respond at **pollev.com/msds**

Text **MSDS** to **37607** once to join

Kmeans prediction value is same as the label.

True

False

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

When poll is active, respond at **pollev.com/msds**

Text **MSDS** to **37607** once to join

Kmeans prediction value is same as the label.

True

False

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Poll locked. Responses not accepted.

Kmeans prediction value is same as the label.

True

False

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Spark ML – Example 3

K-mean clustering

1. Create an RDD.
2. Convert the RDD to DataFrame (This time **only features**!)
3. Train the data.
4. Evaluate.

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

Spark ML – Example 3

K-mean clustering

1. Create an RDD.
2. Convert the RDD to DataFrame (This time only features!)

```
# Merging the data with Vector Assembler.
from pyspark.ml.feature import VectorAssembler
va = VectorAssembler(outputCol="features", inputCols=dfpen.columns[0:-1]) #except the last col.
penlpoints = va.transform(dfpen).select("features")
```

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

Spark ML – Example 3

K-mean clustering

3. Train the data.

```
from pyspark.ml.clustering import KMeans
kmeans =  KMeans(k = 10, maxIter = 200, tol = 0.1)
# k = 10 as there are 10 different handwritten numbers.
model = kmeans.fit(penlpoints)
predictions = model.transform(penlpoints)
```

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

Spark ML – Example 3

K-mean clustering

4. Evaluate.

```
# Shows the result.
centers = model.clusterCenters()
print("Cluster Centers: ")
for center in centers:
    print(center)
```

```
Cluster Centers:
[ 88.00580833  97.78993224  52.64762827  87.28848015  21.28944821
  59.95062924   7.01548887  28.31945789  32.33881897   4.47918683
  79.51016457  11.47821878  62.06582769  30.81219748  13.39303001
  24.91674734]
[ 27.44980443  83.71968709  63.03259452  94.55997392  85.55280313
  87.22946545  55.13233377  65.58148631  69.64602347  45.38787484
  87.32920469  22.85397653  52.22946545   7.26597132   4.30247718
   9.58083442]
[ 44.53996448  98.30195382  13.6660746   77.04795737   5.36234458
  49.47424512  69.21669627  47.98401421  96.60923623  65.72824156
  77.07513221  67.89698046  62.92717584  34.38543517  50.60035524
[ 87.46744186  87.90813953  58.16046512  92.40232558  35.72325581
  79.76046512  56.75813953  74.44883721  80.47674419  63.75232558
```

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

Spark ML – Example 3

K-mean clustering

4. Evaluate.

```
from pyspark.ml.evaluation import ClusteringEvaluator
evaluator = ClusteringEvaluator()
silhouette = evaluator.evaluate(predictions)
print("Silhouette with squared euclidean distance = " + str(silhouette))
```

```
Silhouette with squared euclidean distance = 0.4267285839299759
```

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

Spark ML – Example 3

K-mean clustering

4. Evaluate.

prediction is a group (like cluster ID), not an actual label.

```
predictions.select('label', 'prediction')\
           .groupBy('label', 'prediction')\
           .count()\
           .show(100)
```

executed in 673ms, finished 09:12:31 202

```
+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|  4.0|         8|  961|
|  1.0|         6|   28|
|  3.0|         3|   40|
|  2.0|         1|    1|
|  0.0|         4|  505|
|  0.0|         5|  483|
|  1.0|         8|    1|
```

This could be used for evaluating a ratio of correctly classified letters or letters often confused within a same group.

Changing hyper parameters including k-values may improve the results

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

Contents

Machine Learning With Spark (Spark ML)

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

Course Summary

Relational Databases (MSDS 691)

Relational Database Concept
ER Model
SQL Operations and Functions
Performance Enhancement

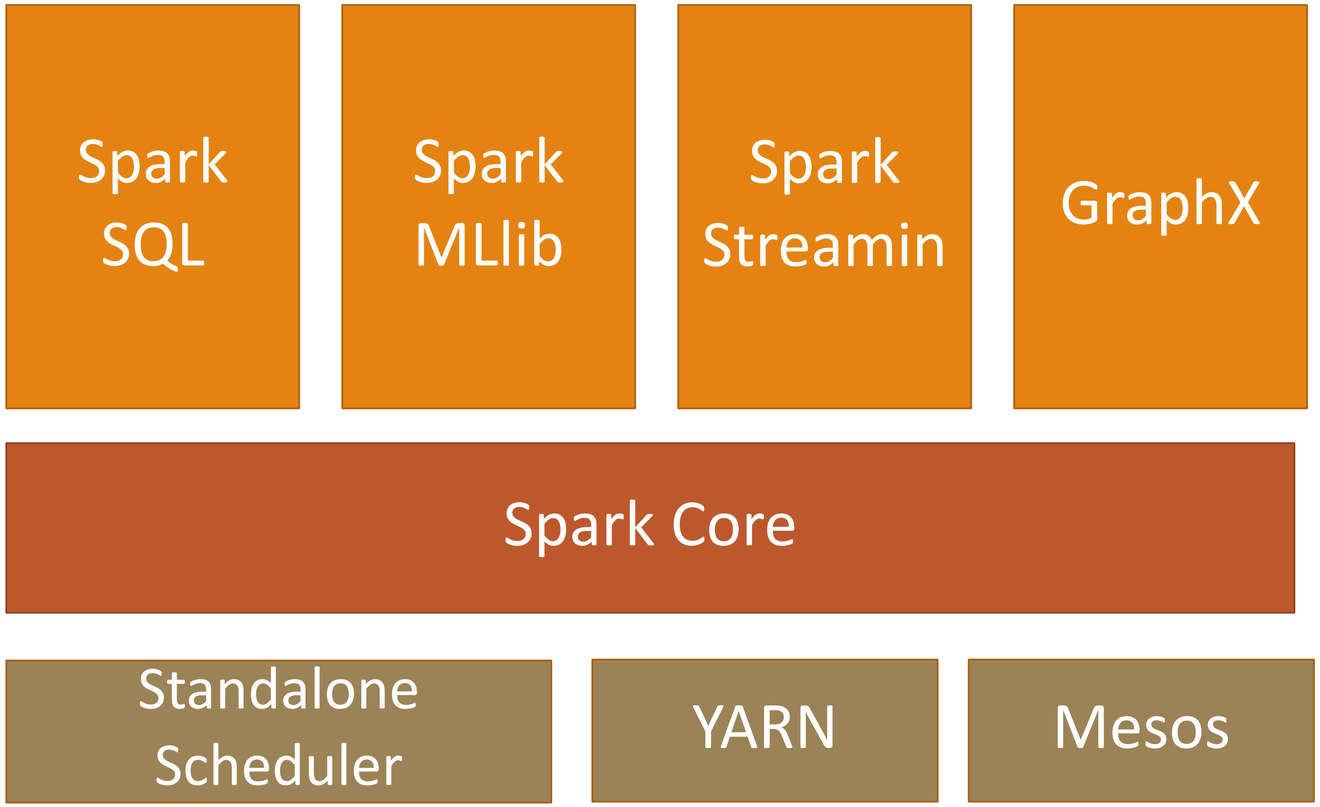
Distributed Computing (MSDS 694)

Distributed Computing Concept
Basic Spark Operations
Running Spark on Clusters

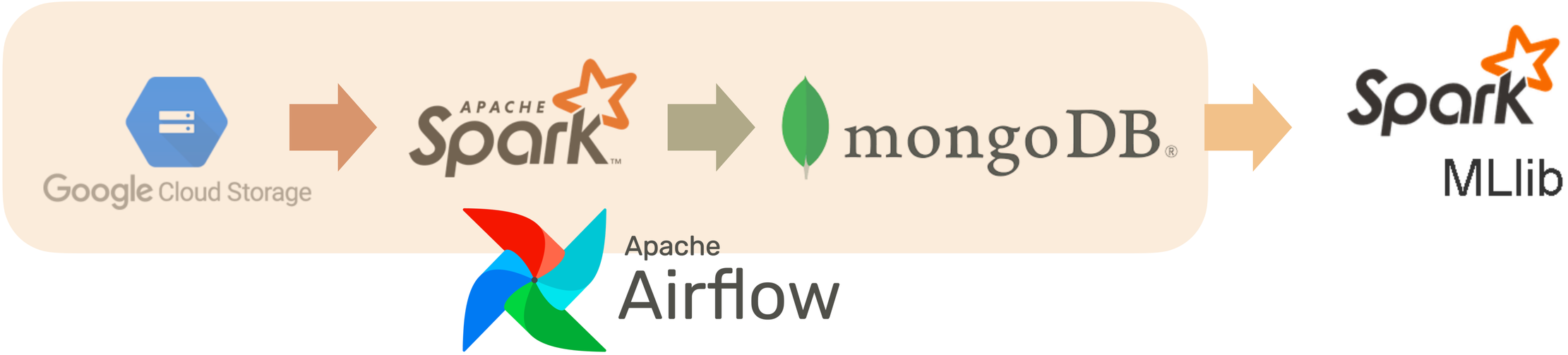
Distributed Data Systems (MSDS 697)

Workflow Management(Airflow)
Distributed Database (NoSQL) Concept
MongoDB and Operations
Running MongoDB on Clusters
Spark Dataframe/SQL
Spark ML

Course Summary



Course Summary



PROJECT EXPERIENCE

MovieLens Recommendation System

- Designed an ensemble recommendation system with matrix factorization, XGBoost, and a deep neural network.
- Experimented with the different negative sampling methods to optimize MSE, RMSE, and accuracy of the system.
- Built a NoSQL database using MongoDB to store 28 million records of explicit user rating and movie metadata.

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.