Date :

Introduction to PySpark.

SparkContext class.   SparkConf() constructor.

Spark - cluster computing.
- computation over clusters.
- data processing & computation → ||l.
- master node ⟸ } worker nodes

Spark's core data structure → RDD - Resilient Distributed Dataset
- Spark DataFrame abstraction built on RDD
easy   optimized.

Create SparkSession object from SparkContext
interface w/ connectn.   connⁿ to cluster

from pyspark.sql import SparkSession.

my_spark = SparkSession.builder.getOrCreate()

.catalog ⇒ data in cluster

.listTables() ⇒ names of table.

x = spark.sql(query)

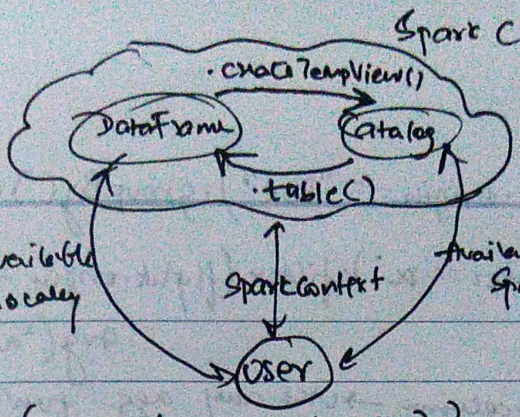
x.show()

x.toPandas() ⇒ spark to pandas.

.createDataFrame() → pandas to spark. → store locally.

.createTempView()

.createOrReplaceTempView()




```
pd = pd.DF (np.random.random(10))
spark_tmp = spark.createDF (pd)
spark_tmp.createOrReplaceTempView ("temp")
```

```
file_path = "          .csv"
airports = spark.read.csv (file_path, header = True)
airports.show()
```

Column Operations

.withColumn (,)
col_name [col itself]

extract ⇒ df.colName

update ⇒ df = df.withColumn ("newCol", df.oldCol + 1)

filtering

.filter()

flights.filter ("air_time > 120").show()

Select

.select()

df.colName. ("colName")

.alias → rename column
.selectExpr

Grouped Data

.min
.max
.count

.groupBy()

df.groupBy().min ("col").show

Date :

flights.filter(flights.origin == "PDX").groupBy().min("distance").show()

flights.filter(flights.carrier == 'DL').filter(flights.origin == 'SEA').groupBy().avg("air_time").show()

.agg() → column → uses any agg. fun^n.

.join() → 2nd DF on 1st DF. → on key → how

flights.join(airports, on = "dest", how = "leftouter")

Machine Learning Pipelines

pyspark.ml → Transformer → .transform() → df.
→ Estimator → .fit() → model obj

planes.withColumnRenamed(..)

Spark - handles only numeric datatypes.

.cast() ← withColumn()

model_data = model_data.withColumn("arr_delay", model_data.arr_delay.cast("integer"))

pyspark.ml.features - one-hot vectors ⟹ categorical feature → Vector

① StringIndexer → Estimator → Transformer
str to o/p Df → Df.

② OneHotEncoder

StringIndexer(inputCol = "carrier", outputCol = "carrier_index")
OneHotEncoder(" ", " carrier_feat)

Last step: → combine all columns into one column.
VectorAssembler

Date : ___/___/_____

VectorAssembler (inputCols = , outputCol =)

Pipeline → Combine all Estimators + Transformers.

Pipeline (stages = [. — — — .])

piped_data = flights_pipe . fit (model_data) . transform (model_data)

Logistic Regression

Hyperparameters — ↑ performance.

pyspark.ml.classification → LogisticRegression()

k-fold Cross Validation.

elasticNetParam regParam.

Evaluation.

pyspark.ml.evaluation.

BinaryClassificationEvaluator (metricName = "areaUnderROC")

Grid.

Look for optimal hyperparam.

pyspark.ml.tuning

ParamGridBuilder

- addGrid()
- build()

```
cv = tune.CrossValidator (estimator = lr,
    estimatorParamMaps = grid,
    evaluator = evaluator)

best_lr = lr.fit (training)
```

```
grid = tune.ParamGridBuilder()
grid = grid.addGrid (lr.regParam,
        np.arange (0, .1, .01))
grid = grid.build()
```

Test.

```
test_res = best_lr.transform (test)
print (evaluator.evaluate (test_res))
```