Machine Learning for the Spark Developer

Paige Liu, Microsoft

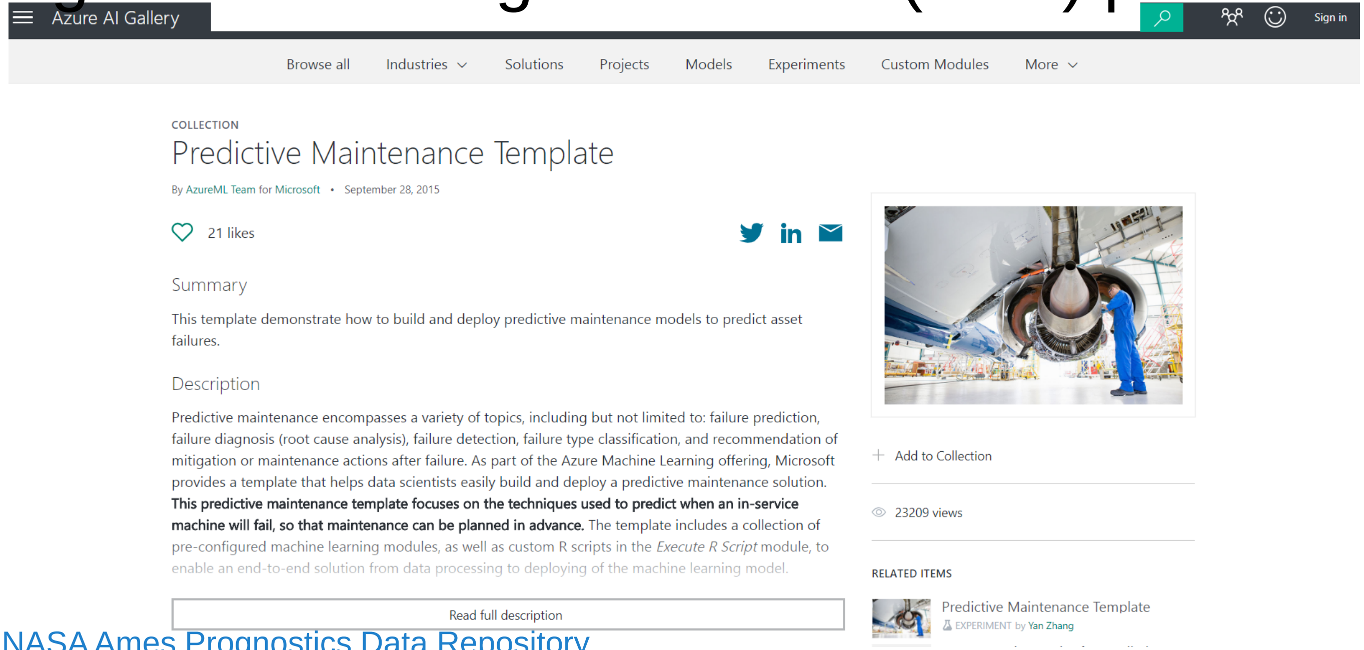
#Dev2SAIS

Agenda

- A simple ML example in R
- Spark MLlib overview
- Implement the same example in Spark MLlib

An example in R

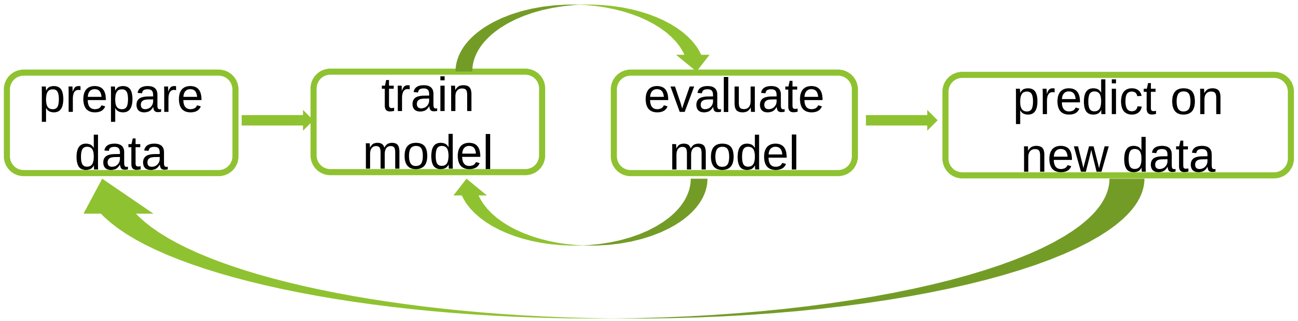
- Engine Remaining Useful Life (RUL) prediction



Data: [NASA Ames Prognostics Data Repository]

R sample

Machine learning workflow



Algorithms in MLlib

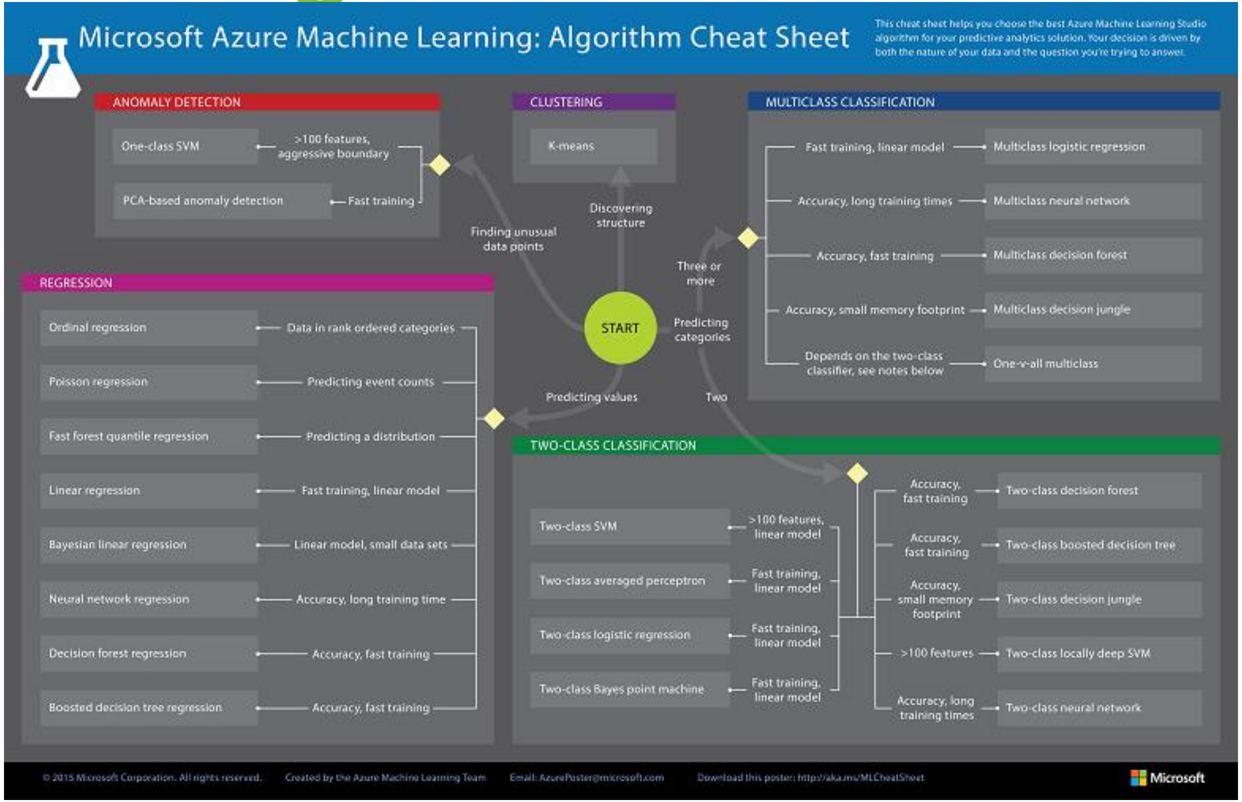
- Classification – predict categories
- Regression – predict numeric values

Supervised learning

- Clustering – group similar items
- Collaborative filtering – recommendation engine
- Frequent pattern mining - anomaly detection

Unsupervised learning

Which algorithm to use?



https://docs.microsoft.com/en-us/azure/machine-learning/studio/algorithm-cheat-sheet

MLlib Concepts

- **Transformers**: data frame -> .transform -> data frame

  ex: VectorAssembler

sensor1	sensor2
1	10
2	20
3	30

VectorAssembler .transform

features
[1,2,[],[1,10]]
[1,2,[],[2,20]]
[1,2,[],[3,30]]

- **Estimators**: data frame -> .fit -> .transform -> data frame

  ex: GBTRegressor

features	label
[1,2,[],[1,10]]	5
[1,2,[],[2,20]]	10
[1,2,[],[3,30]]	15

GBTRegressor .fit

features
[1,2,[],[4,40]]
[1,2,[],[5,50]]
[1,2,[],[6,60]]

GBTRegressionModel .transform

prediction
20
25
30

- **Pipeline**: ML workflow consists of a series of transformers and estimators

Prepare training data

Engine id	Cycle	Setting1	Setting2	Setting3	Sensor1	…	Sensor21
1	1	-0.0007	-0.0004	100.0	518.67	…	23.4190
1	2	0.0019	-0.0003	100.0	518.67	…	23.4236
…							

Engine id	Cycle	Sensor9	Sensor11	Sensor14	Sensor15
1	1	9046.19	47.47	8138.62	8.4195
1	2	9044.07	47.49	8131.49	8.4318
…					

RUL=max – cur cycle
191
190
…

Train the model

features	RUL
[1.0, 9046.19, 47.47, 8138.62, 8.4195]	191
[2.0, 9044.07, 47.49, 8131.49, 8.4318]	190
…	…

```
"GBTRegressionModel (uid=gbtr_2d5316e37e51) with 50 trees
  Tree 0 (weight 1.0):
    If (feature 0 <= 96.5)
     If (feature 0 <= 51.5)
      If (feature 0 <= 25.5)
       If (feature 2 <= 47.35499954223633)
        If (feature 4 <= 8.418050289154053)
         Predict: 205.94230769230768
        Else (feature 4 > 8.418050289154053)
         Predict: 191.46132596685084
       Else (feature 2 > 47.35499954223633)
        If (feature 0 <= 13.5)
         Predict: 190.72403100775193
        Else (feature 0 > 13.5)
         Predict: 178.81116584564862
```

```scala
val vectorAssembler = new VectorAssembler()
  .setInputCols(Array(
    "cycle", "s9", "s11", "s14", "s15"))
  .setOutputCol("features")
```

```scala
val gbtRegressor = new GBTRegressor()
  .setLabelCol("RUL")
  .setFeaturesCol("features")
```

```scala
val pipeline = new Pipeline()
  .setStages(Array(
    vectorAssembler, gbtRegressor))
val gbtRegressionModel = pipeline
  .fit(train_df)
```

Prepare test data

Engine id	Cycle	Setting1	Setting2	Setting3	Sensor1	…	Sensor21
1	1	0.0023	0.0003	100.0	518.67	…	23.3735
1	2	-0.0027	-0.0003	100.0	518.67	…	23.3916
…	…	…	…	…	…	…	…

+

RUL
112
98
…

Engine id	Last cycle	Sensor9	Sensor11	Sensor14	Sensor15	RUL
1	31	9056.4	47.23	8130.11	8.4024	112
2	49	9044.07	47.49	8131.49	8.4318	98
…	…	…	…	…	…	…

Evaluate the model

features	RUL
[31, 9056.4, 47.23, 8130.11, 8.4024]	112
[49, 9044.07, 47.49, 8131.49, 8.4318]	98
…	…

prediction
196.76
152.64
…

root mean squared error

sqrt(mean((labeled-predicted)^2)) = 29.55

```
val predictions = gbtRegressionModel
  .transform(test_df)
```

```
val evaluator = new RegressionEvaluator()
  .setLabelCol("RUL")
  .setPredictionCol("prediction")
  .setMetricName("rmse")
val rootMeanSquaredError = evaluator
  .evaluate(predictions)
```

Tuning the model

- Hyper-parameter tuning

maxIter	maxDepth	stepSize
10	5	0.1
10	5	0.2
10	10	0.1
…	…	…

```
val paramGrid = new ParamGridBuilder()
  .addGrid(gbtRegressor.maxIter, Array(10, 50, 100))
  .addGrid(gbtRegressor.maxDepth, Array(5, 10))
  .addGrid(gbtRegressor.stepSize, Array(0.1, 0.2))
  .build
```

- Cross validation

maxIter	maxDepth	stepSize	train_df		
10	5	0.1	train	train	test
10	5	0.1	train	test	train
10	5	0.1	test	train	train
10	5	0.2	train	train	test
…	…	…	…	…	…

```
val crossValidator = new CrossValidator()
  .setEstimator(pipeline)
  .setEvaluator(evaluator)
  .setEstimatorParamMaps(paramGrid)
  .setNumFolds(3)
  .setParallelism(3)
val crossValidatorModel = crossValidator
  .fit(train_df)
```

Predict on new data in application

- Save the model

```
crossValidatorModel.bestModel
  .write
  .overwrite()
  .save("/path/to/model")
```

- Load the model to make predictions

```
val new_df = …
val model = PipelineModel
  .load ("/path/to/model")
val predictions = model
  .transform(new_df)
```

- Load the model to non-Spark applications
  - MLeap: serialization format + execution engine
  - Databricks ML Model Export

Resources

- GitHub repo containing the sample: https://github.com/liupeirong/sparksummit2018ml
  - Customer churn: binary classification
  - Iris: multi-class classification
  - Predictive maintenance: regression
- Documentation on the sample: https://gallery.azure.ai/Collection/Predictive-Maintenance-Template-3
- NASA data repository hosting the sample data: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

Thank You