Experimental Design for Distributed Machine Learning

Myles Baker (@mydpy)

26 October, 2017

databricks

Common Modeling Problems

PROBLEM 1

- Business problem is defined, machine learning is a solution candidate
- A machine learning algorithm has been selected and implemented
- What is the expected generalization error? Can be the business be confident in the model results?

PROBLEM 2

- Business problem is defined, machine learning is a solution candidate
- Multiple programming / machine learning models apply
- Which model has the least error due to generalization for a given application?

Common Modeling Problems

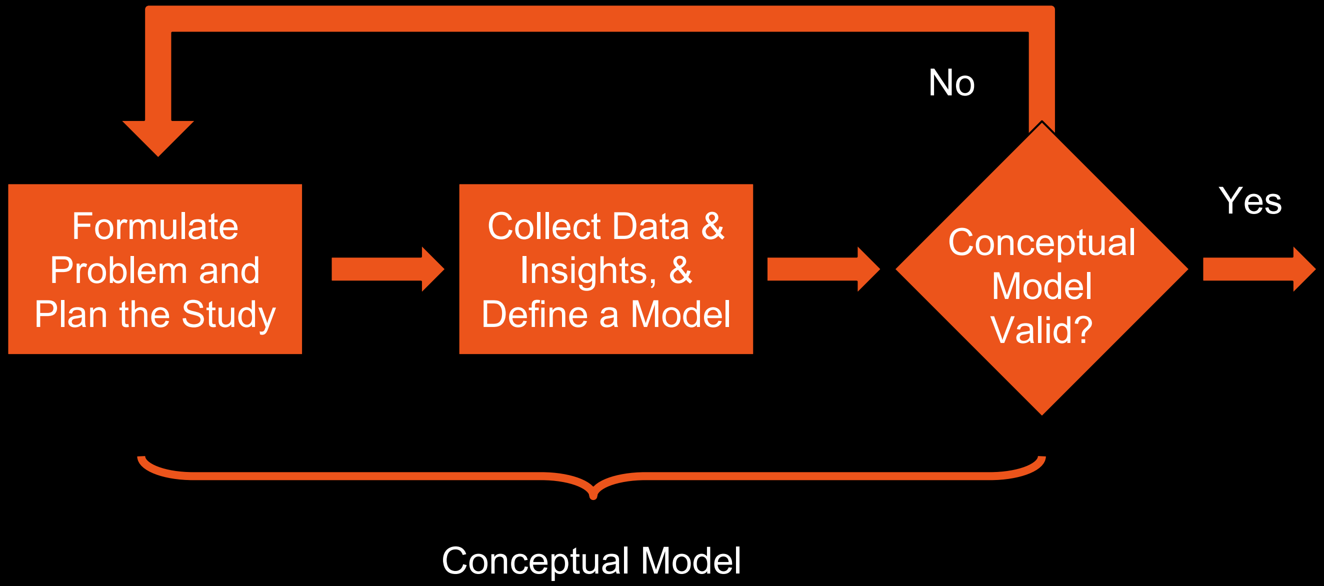
PROBLEM 3

- An existing business problem has been solved with a machine learning model
- The model was designed for a serial-execution system and does not scale with input set
- A parallel model must be developed and tested

PROBLEM 4

- An existing business problem has been solved with a machine learning model
- How do I know how variations in the input will affect the output and predicted outcomes?
- How will results change after the model is retrained?

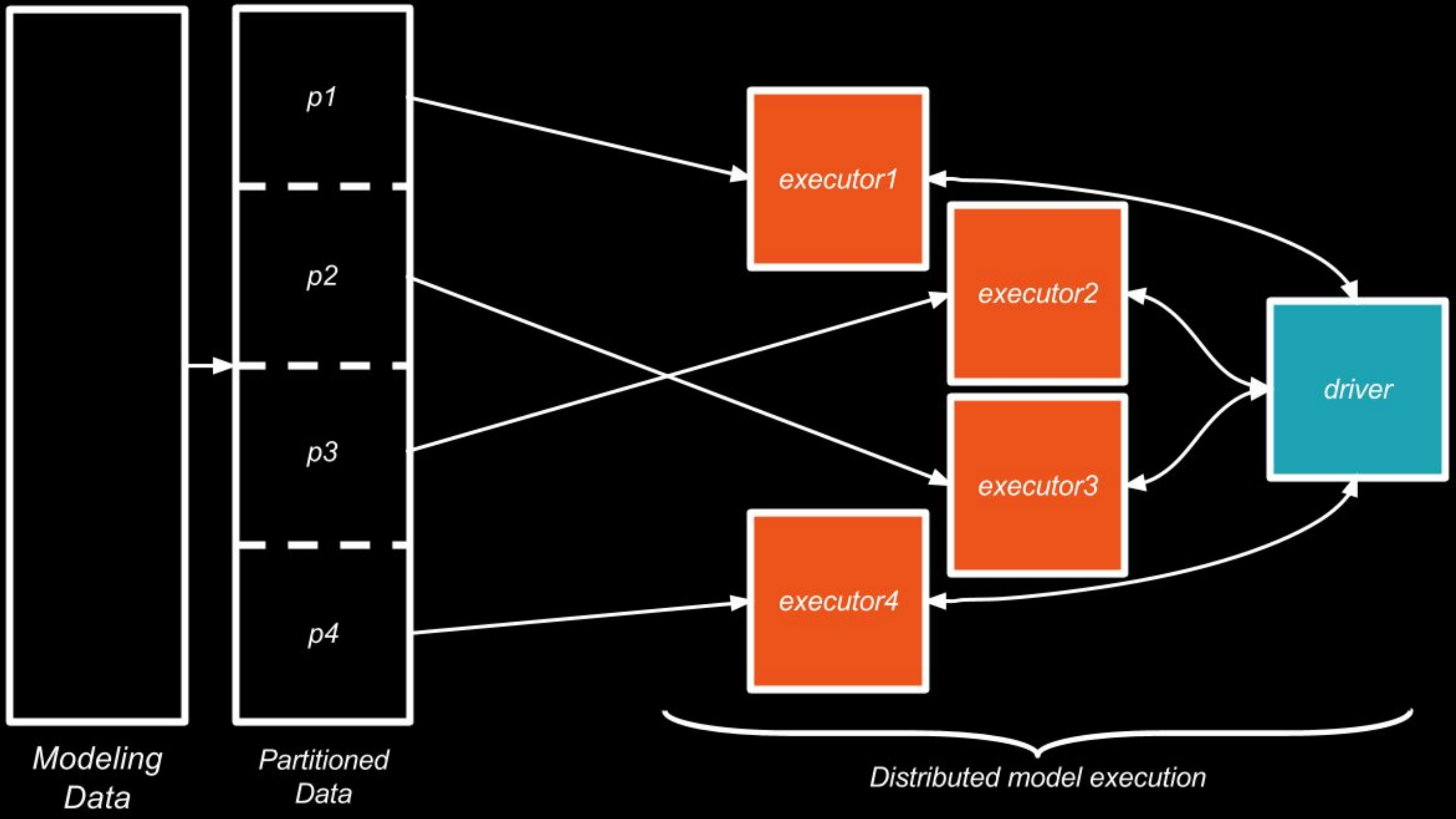
Steps in a Machine Learning Study



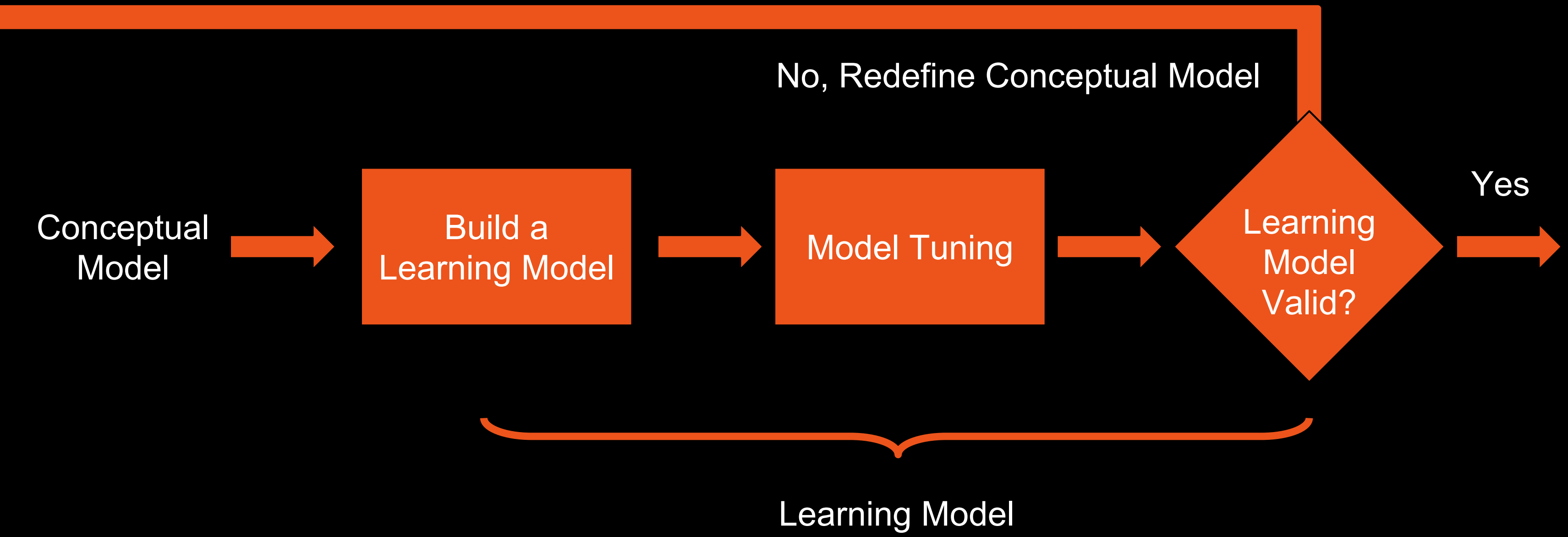
Conceptualizing a Distributed Model

- Model capabilities are different between serial and distributed applications
  - Some algorithms do not have a distributed implementation
  - Understanding computational complexity becomes an increasingly present limitation
  - Solver and optimizer implementations for existing algorithms may be different or not supported
  - Model assumptions may change when migrating a serial model to a distributed model
- Data characteristics are more challenging to reveal
  - Outliers are prevalent but may be incorrectly or poorly modeled
  - Missing value compensation can significantly skew results
  - Synthetic data can poorly represent actual system

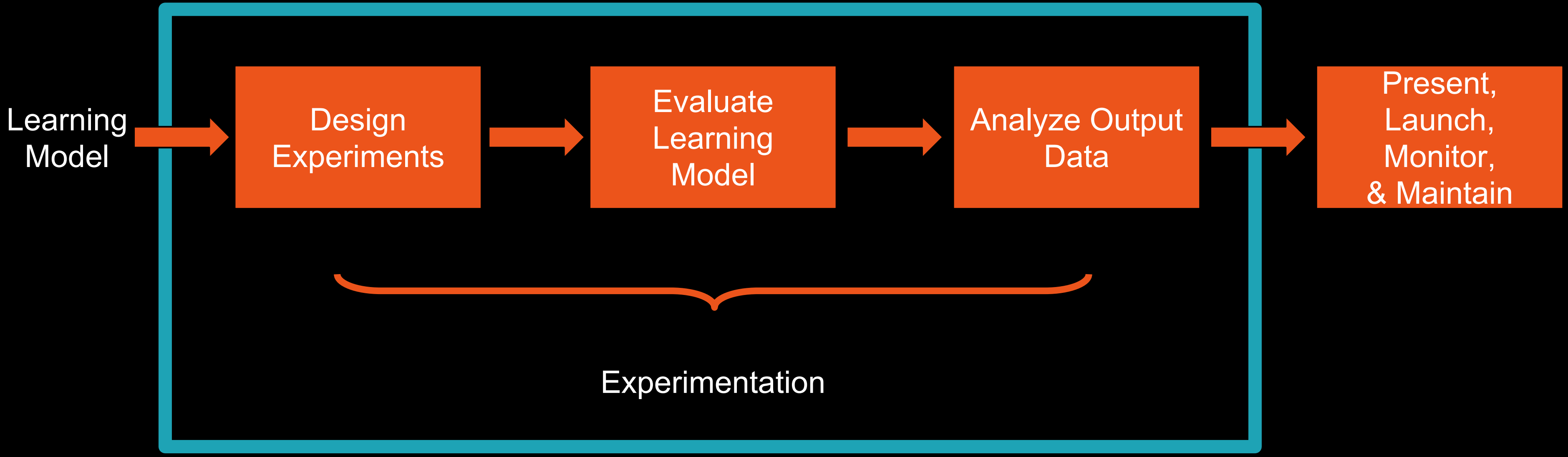
Understanding Scale-out Learning



Steps in a Machine Learning Study



Steps in a Machine Learning Study



Subject of this Talk

The goal of experimentation is to understand the effect of model factors and obtain conclusions which we can consider statistically significant

This is challenging for distributed learning!

How do you design an experiment?

- An **Experiment** is an evaluation of a model using a combination of controllable factors that affect the response
- Experiments must be designed correctly using statistical methodology
- An Experiment should be:
  - Independent of other responses
  - Controlled for variance and uncontrollable error
  - Reproducible, especially between model candidates
- Techniques include:
  - Measuring Classifier Performance
  - Hypothesis Testing

Factors that affect model outcomes

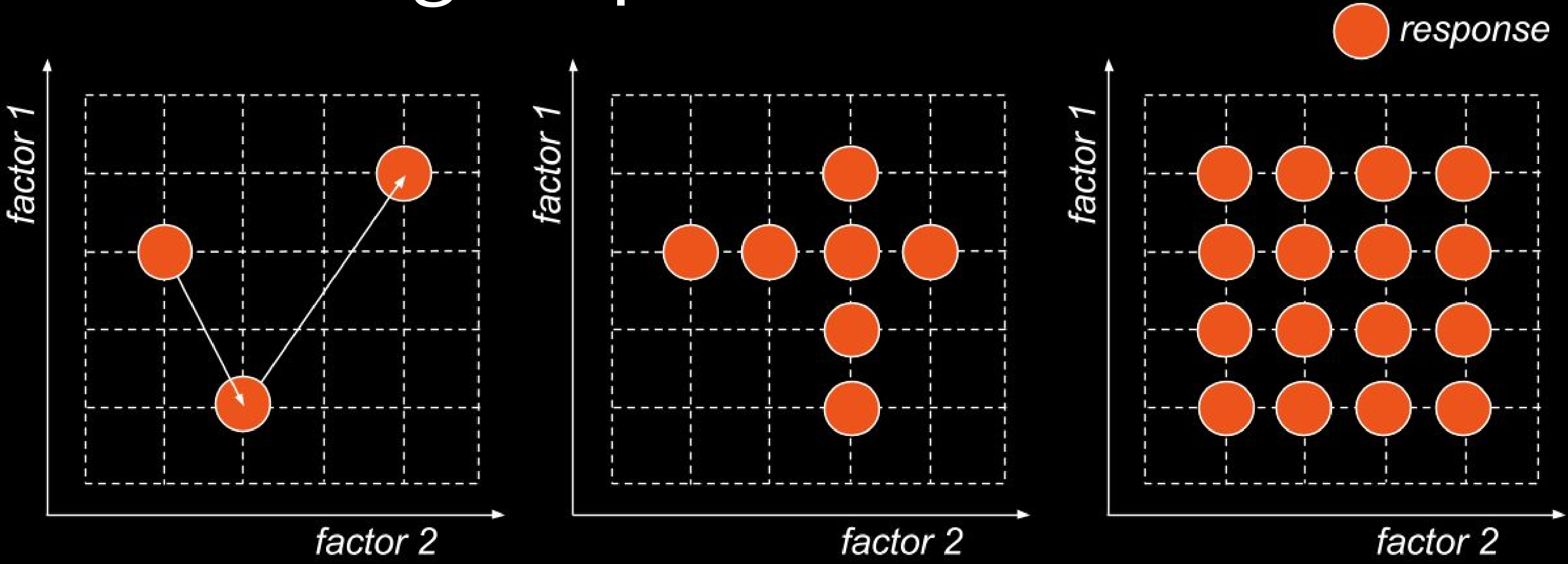
CONTROLLABLE

- Learning algorithm
- Input data
- Model parameters
- Model hyperparameters

UNCONTROLLABLE

- Noise in the data
- Optimization randomness
- Outcomes not observed during training but part of the system being modeled (I.e., a rare disease outcome)

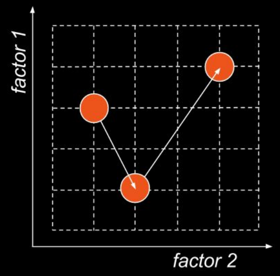
Generating Responses



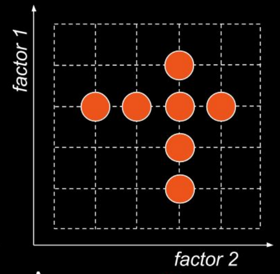
$O(LF)$ $O(LF)$ $O(L^F)$

Generating Responses

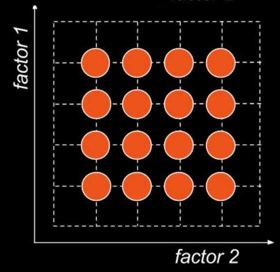
```
val p1 = new ParamMap().put(factor1.w(3), factor2.w(1))
val p2 = new ParamMap().put(factor1.w(1), factor2.w(2))
val p3 = new ParamMap().put(factor1.w(4), factor2.w(4))
```




```
val factorGrid = new ParamGridBuilder()
  .addGrid(factor1, Array(1, 2, 3, 4))
  .addGrid(factor2, Array(3))
  .build()
```




```
val factorGrid = new ParamGridBuilder()
  .addGrid(factor1, Array(1, 2, 3, 4))
  .addGrid(factor2, Array(1, 2, 3, 4))
  .build()
```



Generating Responses

Models support different methods / metrics

Train-Validation Split

```
val tvs =
 new TrainValidationSplit()
  .setEstimatorParamMaps(factorGrid)
  .setEvaluator(new RegressionEvaluator)
  .setTrainRatio(r)
```

```
val model = tvs.fit(data)
model.bestModel
     .extractParamMap
```

- Creates an estimator based on the parameter map or grid
- Randomly splits the input dataset into train and validation sets based on the training ratio r
- Uses evaluation metric on the validation set to select the best model

Generating Responses

Models support different methods / metrics

Cross Validator

```
val cv = new CrossValidator()
  .setEstimatorParamMaps(factorGrid)
  .setEvaluator(new
      BinaryClassificationEvaluator)
  .setNumFolds(k)
```

```
val model = cv.fit(data)
model.bestModel
    .extractParamMap
```

- Creates k non-overlapping randomly partitioned folds which are used as separate training and test datasets
- Controls for uncontrollable factors and variance
- The ‘bestModel’ contains the model with the highest average cross-validation
- Tracks the metrics for each param map evaluated

Reducing the Number of Responses

Computational Complexity Limits Practicality of Factorial Search

- Use a well-formulated conceptual model. This informs factor choice and reduces unnecessary model iterations
- Normalize factors where possible (i.e., factor is determined by input as-opposed to arbitrarily chosen by the modeler)
- If your data is large enough, you can split your dataset into multiple parts for use during cross-validation

How do you analyze model output?

CLASSIFICATION

- Precision / recall relationships for binary classification problems
  - Receiver Operating Characteristic Curve
- For multi-classification problems:
  - Most packages only support 0/1 error functions
  - Confusion matrix
- For multilabel classification:
  - Again, 0/1 indicator function is only supported
  - Measures by label are most appropriate

REGRESSION

- Linear: RSME = Easy
- Non-linear: … *runs*
  - SoftMax
  - Cross Entropy

$$\boxed{\hat{\delta}(x) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{otherwise.} \end{cases}}$$

Analyzing Model Output

Dataframe of (prediction, label)

```
val metrics = new
BinaryClassificationMetrics(predictionAndLabels.rdd.map( r =>
(r.getAs[Double]("prediction"), r.getAs[Double]("label")))
```

Binary Classification

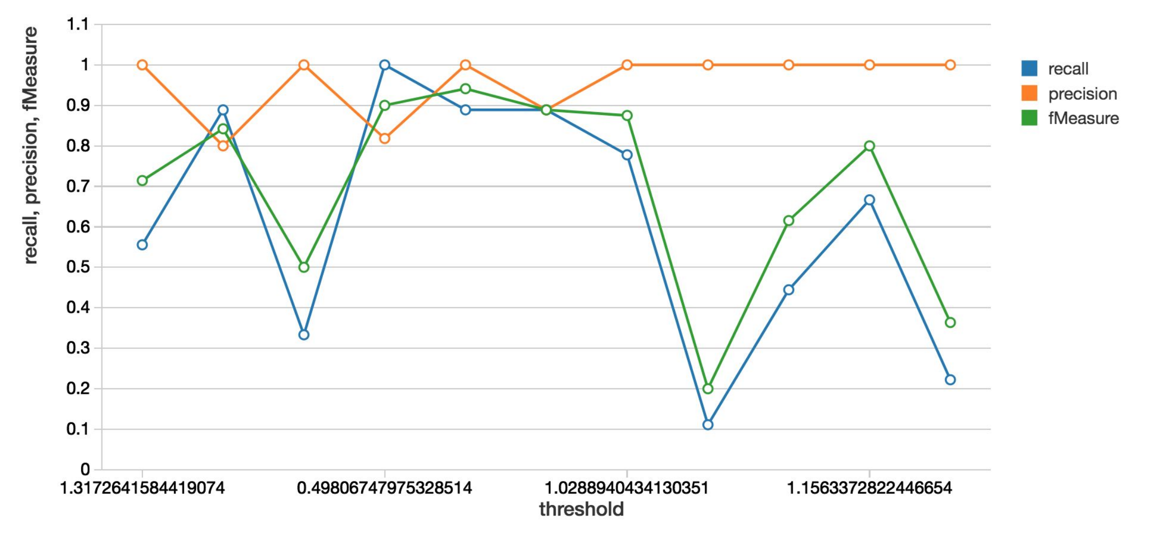
Metric	Spark Implementation
Receiver Operating Characteristic	roc
Area Under Receiver Operating Characteristic Curve	areaUnderROC
Area Under Precision-Recall Curve	areaUnderPR
Measures by Threshold	{measure}ByThreshold

Threshold Curves

Optional beta parameter for F1 measure (default = 1)

```
val threshold = metrics.{measure}ByThreshold
        .toDF('threshold', 'measure')

display(precision.join({measure}, 'threshold')
  .join(recall, 'threshold'))
```



Analyzing Model Output

Dataframe of (prediction, label)

```
val metrics = new
MulticlassMetrics(predictionAndLabels.rdd.map( r =>
(r.getAs[Double]("prediction"), r.getAs[Double]("label")))
```

Multiclass Classification

Metric	Spark Implementation
Confusion Matrix	confusionMatrix
Accuracy	accuracy
Measures by Label	{measure}ByLabel
Weighted Measures	weighted{Measure}

Usually not a robust metric by itself

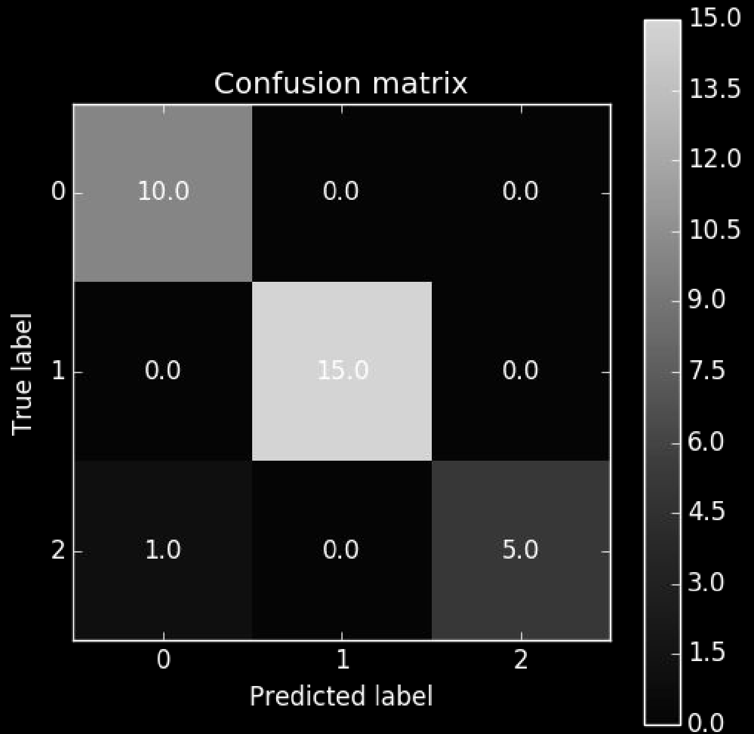
Confusion Matrix

```
metrics.confusionMatrix
       .toArray

//Display using non-Scala tool
%python
confusion = np.array([[...],[...]])
```

$$C_{ij} = \sum_{k=0}^{N-1} \hat{\delta}(\mathbf{y}_k - \ell_i) \cdot \hat{\delta}(\hat{\mathbf{y}}_k - \ell_j)$$

$$\begin{pmatrix} \sum_{k=0}^{N-1} \hat{\delta}(\mathbf{y}_k - \ell_1) \cdot \hat{\delta}(\hat{\mathbf{y}}_k - \ell_1) & \dots & \sum_{k=0}^{N-1} \hat{\delta}(\mathbf{y}_k - \ell_1) \cdot \hat{\delta}(\hat{\mathbf{y}}_k - \ell_N) \\ \vdots & \ddots & \vdots \\ \sum_{k=0}^{N-1} \hat{\delta}(\mathbf{y}_k - \ell_N) \cdot \hat{\delta}(\hat{\mathbf{y}}_k - \ell_1) & \dots & \sum_{k=0}^{N-1} \hat{\delta}(\mathbf{y}_k - \ell_N) \cdot \hat{\delta}(\hat{\mathbf{y}}_k - \ell_N) \end{pmatrix}$$



Why conduct and test a hypothesis?

How do I know my model is correct?

Hypothesis testing describes the significance of the result and provides a mechanism for assigning confidence to model selection

1. What is the likelihood that my model will make a misclassification error? This probability is not known!
2. Given two learning algorithms, which has the lower expected error rate?

	Decision	
Truth	Fail to reject	Reject
True	Correct	Type I error
False	Type II error	Correct (power)

- 1 - Binomial Test
- 1 - Approximate Normal Test
- 1 - t Test
- 2 - McNemar's Test
- 2 - K-Fold Cross-Validated Paired t Test

Chi-Squared Test

New in Spark 2.2!

Hypothesis: Outcomes are statistically independent

- Conducts Pearson's independence test for every feature against the label
- Chi-squared statistics is computed from (feature, label) pairs
- All label and feature values must be categorical

```
import org.apache.spark.mllib.stat.Statistics
import org.apache.spark.mllib.stat.test.ChiSqTestResult
val goodnessOfFitTestResult = Statistics.chiSqTest(labels)
val independenceTestResult = Statistics.chiSqTest(contingencyMatrix)

Chi squared test summary:
method: pearson
degrees of freedom = 4
statistic = 0.12499999999999999
pValue = 0.998126379239318
No presumption against null hypothesis: observed follows the same distribution as
expected..
```

Nice, but you still need to do the hard work of constructing the hypothesis and validating!

McNemar's Test

Hypothesis: Model 1 and model 2 have the same rate of generalization error

```
val totalObs = test.count
val conditions = "..."
val p1c =
predictions1.where(conditions).count()
val p1m = totalObs - p1c
val p2c =
predictions2.where(conditions).count()
val p2m = totalObs - p2c

val e00 = p1m + p2m
val e01 = p1m
val e10 = p2m
val e11 = p1c + p2c
```

e_{00}: Number of examples misclassified by both
e_{10}: Number of examples misclassified by 2 but not 1
e_{01}: Number of examples misclassified by 1 but not 2
e_{11}: Number of examples correctly classified by both

$$\frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}} \sim \chi_1^2$$

databricks

Analyzing of Variance (ANOVA)

Analysis of Variance is used to compare multiple models. What is the statistical significance of running model 1 or model 2?

- Currently no techniques for ANOVA directly within MLlib
- Requires calculating statistics manually
- A very useful technique in-practice despite the manual work needed

Thank you

myles.baker@databricks.com

References

- Design and Analysis of Machine Learning, Chapter 19 of Introduction to Machine Learning, by Ethem Alpaydın
- No free lunch in search and optimization
- Moral Machine (MIT)
- Design and Analysis of Classifier Learning Experiments in Bioinformatics: Survey and Case Studies
- Simulation Modeling and Analysis, 3rd Edition, Averill Law and David Kelton
- Spark Scala API documentation, source code