Spark Machine Learning Evaluation Metrics Support

Options available to measure
model and prediction performance

Spark Metrics

- Overview of types of learning and prediction
- Review each in detail
- Demonstrate each in running code
- Discuss cautions and special cases

Overview

- Classification model evaluation
  - Binary classification
    - Threshold tuning
  - Multiclass classification
    - Label based metrics
  - Multilabel classification
  - Ranking systems
- Regression model evaluation
- Dataframe API Classification / Regression Reference: https://spark.apache.org/docs/latest/ml-classification-regression.html
- RDD API Classification / Regression Metrics Reference: https://spark.apache.org/docs/latest/mllib-evaluation-metrics.html

Classification

- Supervised Learning
- Build a model from examples with known outcomes that predicts those outcomes for new data
- Evaluation based on relationship between these counts:
  - **True Positive (TP)** - label is positive and prediction is also positive
  - **True Negative (TN)** - label is negative and prediction is also negative
  - **False Positive (FP)** - label is negative but prediction is positive
  - **False Negative (FN)** - label is positive but prediction is negative
- Frequently shown in a Confusion Matrix:

		Predicted		
		Cat	**Dog**	**Rabbit**
Actual class	**Cat**	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

Review some basic definitions

In statistical hypothesis testing, a type I error is the incorrect rejection of a true null hypothesis (a "false positive"), while a type II error is incorrectly retaining a false null hypothesis (a "false negative").[1]

More simply stated, a
type I error (FP) is **detecting** an **effect** that is **not present**, while a
type II error(FN) is **failing** to **detect** an **effect** that is **present**.

TP,FP,TN,FN visual

Left side is Positive

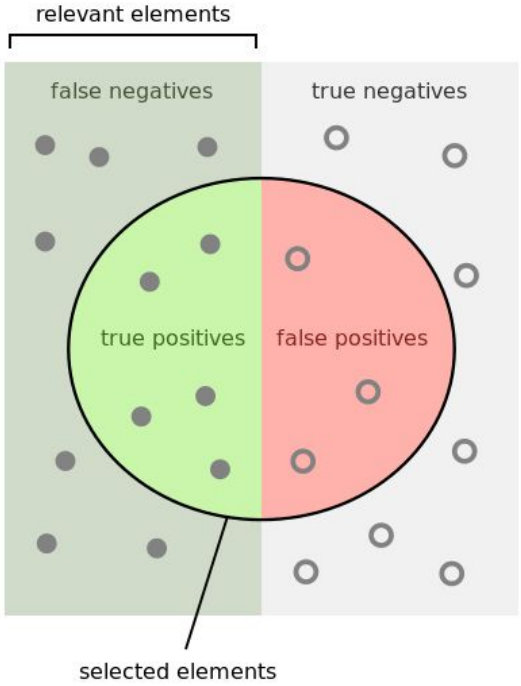
Right side is Negative

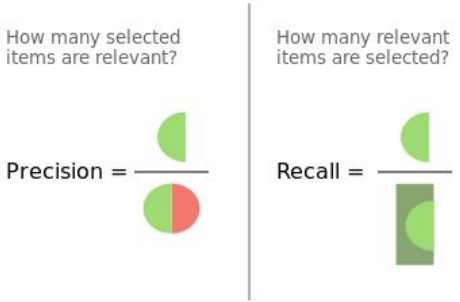
Circle is what we THOUGHT was POSITIVE

Outside the Circle is what we THOUGHT was NEGATIVE

Left Circle half is True Positives

Right Circle half if False Positives





Classification Calculations

- **accuracy** (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N}$$

- **sensitivity**, **recall**, hit rate, or true positive rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- fall-out or false positive rate (FPR)

$$\mathrm{FPR} = \frac{\mathrm{FP}}{N} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - \mathrm{TNR}$$

- **specificity** or true negative rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{N} = \frac{\mathrm{TN}}{\mathrm{FP} + \mathrm{TN}}$$

- **precision** or positive predictive value (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

- **F1** score - *harmonic mean* of Precision and Sensitivity

$$F_1 = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

  - **Harmonic Mean** - can be expressed as the reciprocal of the arithmetic mean of the reciprocalsk

Classification model evaluation

- Binary classification

  Outcome is either True or False
  - Threshold tuning
- Multiclass classification

  Outcome is one of multiple values
  - Label based metrics
- Multilabel classification

  Each sample mapped to one or more labels (e.g. topics for news articles)
- Ranking or Recommender systems

  “Others that bought this also bought ….”

Binary classification

Separate elements of a dataset into one of two possible groups.

Train a model (function) that outcomes True or False

Example: Given a patient's clinical history - are they diabetic

Goal is to determine the features and weights of those features that are common of being in that group or not. Metrics help us evaluate how well the model fits the test data as well as how well it predicts new data. The metrics also help us determine i the model might be over tuned to the test data.

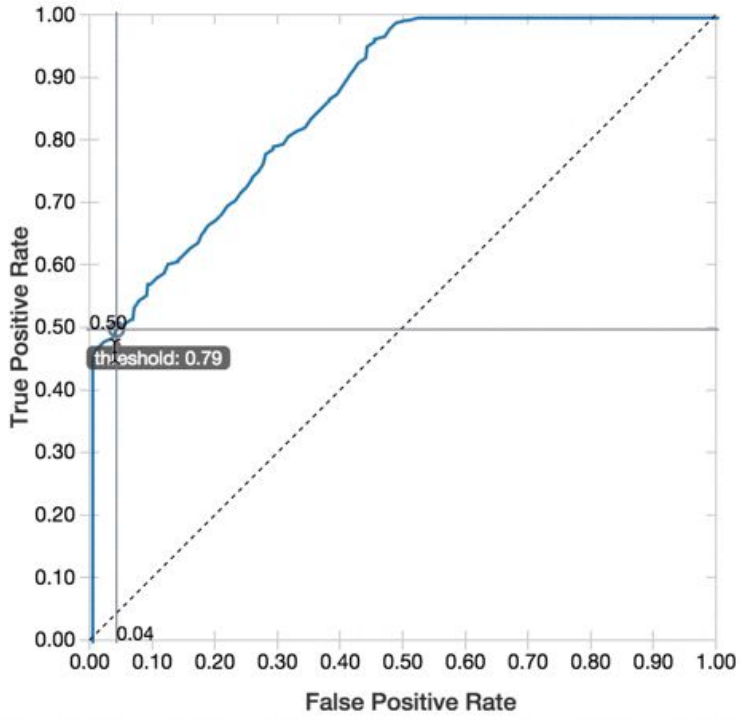
Threshold tuning (Binary classification)

Often determination of being true depends on a Threshold of probability.

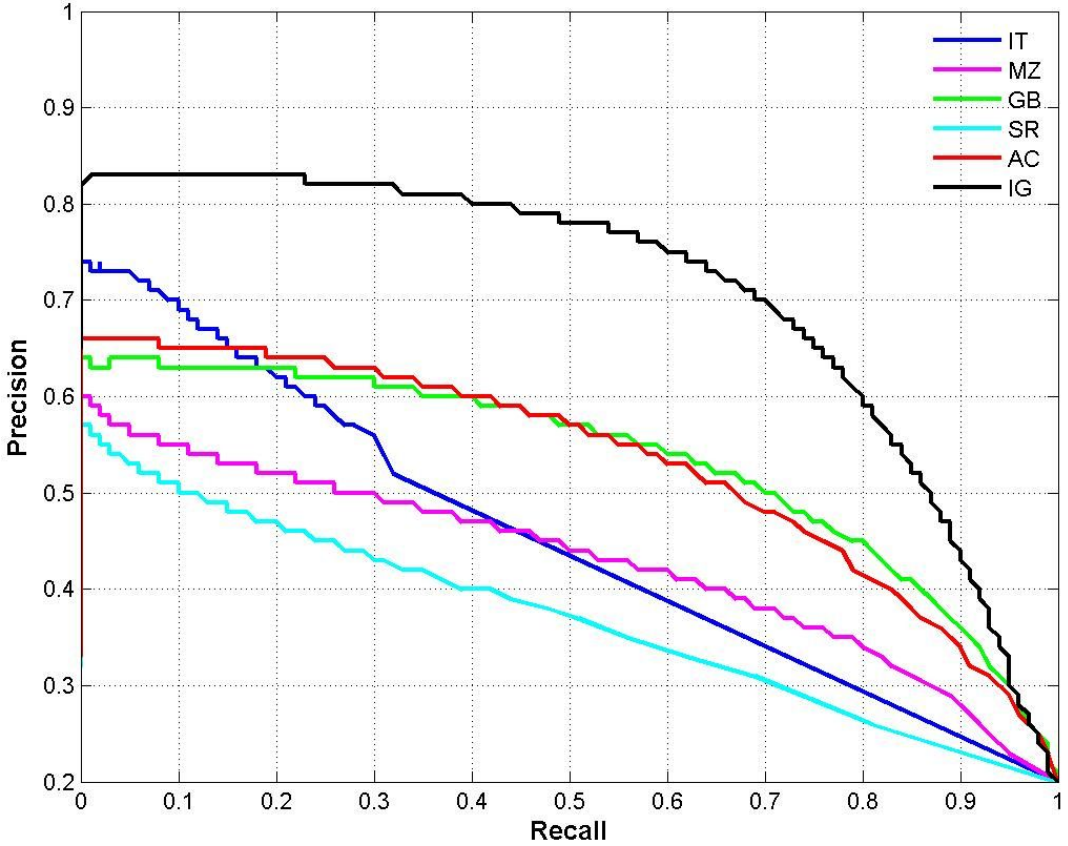
An example would be grant a loan only if probability or confidence is > 80%.

The results will vary as the threshold is changed. This is often presented by graphing one metric against another

- P-R Curve: Precision vs. Recall
- **ROC** (Receiver Operating Characteristic): Recall vs. False Positive Rate



P-R Curve


IT
MZ
GB
SR
AC
IG
Precision
Recall
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Binary Classifier - Available Metrics

Metric	Definition
Precision (Positive Predictive Value)	$PPV = \frac{TP}{TP+FP}$
Recall (True Positive Rate)	$TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$
F-measure	$F(\beta) = (1+\beta^2) \cdot \left(\frac{PPV \cdot TPR}{\beta^2 \cdot PPV + TPR} \right)$
Receiver Operating Characteristic (ROC)	$FPR(T) = \int_T^\infty P_0(T)\,dT$ $TPR(T) = \int_T^\infty P_1(T)\,dT$
Area Under ROC Curve	$AUROC = \int_0^1 \frac{TP}{P} d\left(\frac{FP}{N}\right)$
Area Under Precision-Recall Curve	$AUPRC = \int_0^1 \frac{TP}{TP+FP} d\left(\frac{TP}{P}\right)$

Binary Classifers - MLlib Metric Objects

- RDD based API (classic Spark MLib)

https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.mllib.evaluation.BinaryClassificationMetrics

- Dataframe based API (MLlib)

https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.ml.classification.BinaryLogisticRegressionSummary
https://spark.apache.org/docs/latest/api/scala/org/apache/spark/ml/classification/BinaryLogisticRegressionTrainingSummary.html

Multiclass classification

- Where there are more than 2 classification labels (e.g., digits)
- Example: Handwritten digits (0-9) so 10 possible class labels for “Digit”
- Determination of True Positive when predicted label matches actual label
- Multiple True Negatives when predicted correctly that label is Not a given class label when actual is also not of that class.

Label based metrics (Multiclass classification)

Metrics are modified to reflect and account for the fact that there are multiple possible labels

Accuracy measures precision across all labels - the number of times any class was predicted correctly (true positives) normalized by the number of data points

Precision by label considers only one class, and measures the number of times a specific label was predicted correctly normalized by the number of times that label appears in the output.

Multiclass classification - Variables

Define the class, or label, set as

$$L = \{\ell_0, \ell_1, \ldots, \ell_{M-1}\}$$

The true output vector $\mathbf{y}$ consists of N elements

$$\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{N-1} \in L$$

A multiclass prediction algorithm generates a prediction vector $\hat{\mathbf{y}}$ of N elements

$$\hat{\mathbf{y}}_0, \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{N-1} \in L$$

For this section, a modified delta function $\hat{\delta}(x)$ will prove useful

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

Multiclass Classfication - Metrics

Metric	Definition
Confusion Matrix	$C_{ij} = \sum_{k=0}^{N-1} \hat{\delta}(\mathbf{y}_k - \ell_i) \cdot \hat{\delta}(\hat{\mathbf{y}}_k - \ell_j)$ $\begin{pmatrix} \sum_{k=0}^{N-1} \hat{\delta}(\mathbf{y}_k - \ell_1) \cdot \hat{\delta}(\hat{\mathbf{y}}_k - \ell_1) & \dots & \sum_{k=0}^{N-1} \hat{\delta}(\mathbf{y}_k - \ell_1) \cdot \hat{\delta}(\hat{\mathbf{y}}_k - \ell_N) \\ \vdots & \ddots & \vdots \\ \sum_{k=0}^{N-1} \hat{\delta}(\mathbf{y}_k - \ell_N) \cdot \hat{\delta}(\hat{\mathbf{y}}_k - \ell_1) & \dots & \sum_{k=0}^{N-1} \hat{\delta}(\mathbf{y}_k - \ell_N) \cdot \hat{\delta}(\hat{\mathbf{y}}_k - \ell_N) \end{pmatrix}$
Accuracy	$ACC = \frac{TP}{TP+FP} = \frac{1}{N} \sum_{i=0}^{N-1} \hat{\delta}(\hat{\mathbf{y}}_i - \mathbf{y}_i)$
Precision by label	$PPV(\ell) = \frac{TP}{TP+FP} = \frac{\sum_{i=0}^{N-1} \hat{\delta}(\hat{\mathbf{y}}_i - \ell) \cdot \hat{\delta}(\mathbf{y}_i - \ell)}{\sum_{i=0}^{N-1} \hat{\delta}(\hat{\mathbf{y}}_i - \ell)}$
Recall by label	$TPR(\ell) = \frac{TP}{P} = \frac{\sum_{i=0}^{N-1} \hat{\delta}(\hat{\mathbf{y}}_i - \ell) \cdot \hat{\delta}(\mathbf{y}_i - \ell)}{\sum_{i=0}^{N-1} \hat{\delta}(\mathbf{y}_i - \ell)}$
F-measure by label	$F(\beta, \ell) = (1+\beta^2) \cdot \left(\frac{PPV(\ell) \cdot TPR(\ell)}{\beta^2 \cdot PPV(\ell) + TPR(\ell)} \right)$
Weighted precision	$PPV_w = \frac{1}{N} \sum_{\ell \in L} PPV(\ell) \cdot \sum_{i=0}^{N-1} \hat{\delta}(\mathbf{y}_i - \ell)$
Weighted recall	$TPR_w = \frac{1}{N} \sum_{\ell \in L} TPR(\ell) \cdot \sum_{i=0}^{N-1} \hat{\delta}(\mathbf{y}_i - \ell)$
Weighted F-measure	$F_w(\beta) = \frac{1}{N} \sum_{\ell \in L} F(\beta, \ell) \cdot \sum_{i=0}^{N-1} \hat{\delta}(\mathbf{y}_i - \ell)$

Multiclass Classification - MLlib Objects

- RDD based API (classic MLib) https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.mllib.evaluation.MulticlassMetrics
- Dataframe based API https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator

Multilabel classification

- Matching each dataset to a set of classifications (the labels are no longer exclusive)
- Example: Classifying new articles to a set of topics (more than one per article)
- Because the labels are not mutually exclusive:
  - the predictions and true labels are now vectors of label sets, rather than vectors of labels
  - Multilabel metrics, therefore, extend the fundamental ideas of precision, recall, etc. to operations on sets.
  - E.g., a **true positive** for a given class now occurs when that **class exists** in the **predicted set and** it **exists** in the **true label set**, for a specific data point.

Multilabel Classification - Variables

Here we define a set D of N documents

$$D = \{d_0, d_1, \ldots, d_{N-1}\}$$

Define $L_0, L_1, \ldots, L_{N-1}$ to be a family of label sets and $P_0, P_1, \ldots, P_{N-1}$ to be a family of prediction sets where L_i and P_i are the label set and prediction set, respectively, that correspond to document d_i.

The set of all unique labels is given by

$$L = \bigcup_{k=0}^{N-1} L_k$$

The following definition of indicator function $I_A(x)$ on a set A will be necessary

$$I_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{otherwise.} \end{cases}$$

Multilabel Classification - Metrics

Precision	$\frac{1}{N}\sum_{i=0}^{N-1}\frac{\lvert P_i \cap L_i\rvert}{\lvert P_i\rvert}$
Recall	$\frac{1}{N}\sum_{i=0}^{N-1}\frac{\lvert L_i \cap P_i\rvert}{\lvert L_i\rvert}$
Accuracy	$\frac{1}{N}\sum_{i=0}^{N-1}\frac{\lvert L_i \cap P_i\rvert}{\lvert L_i\rvert+\lvert P_i\rvert-\lvert L_i \cap P_i\rvert}$
Precision by label	$PPV(\ell) = \frac{TP}{TP+FP} = \frac{\sum_{i=0}^{N-1} I_{P_i}(\ell)\cdot I_{L_i}(\ell)}{\sum_{i=0}^{N-1} I_{P_i}(\ell)}$
Recall by label	$TPR(\ell) = \frac{TP}{P} = \frac{\sum_{i=0}^{N-1} I_{P_i}(\ell)\cdot I_{L_i}(\ell)}{\sum_{i=0}^{N-1} I_{L_i}(\ell)}$
F1-measure by label	$F1(\ell) = 2\cdot\left(\frac{PPV(\ell)\cdot TPR(\ell)}{PPV(\ell)+TPR(\ell)}\right)$

Multilabel Classificatin - Metrics (cont)

Hamming Loss	$\frac{1}{N \cdot \lvert L \rvert} \sum_{i=0}^{N-1} \lvert L_i \rvert + \lvert P_i \rvert - 2 \lvert L_i \cap P_i \rvert$
Subset Accuracy	$\frac{1}{N} \sum_{i=0}^{N-1} I_{\{L_i\}}(P_i)$
F1 Measure	$\frac{1}{N} \sum_{i=0}^{N-1} 2 \frac{\lvert P_i \cap L_i \rvert}{\lvert P_i \rvert \cdot \lvert L_i \rvert}$
Micro precision	$\frac{TP}{TP+FP} = \frac{\sum_{i=0}^{N-1} \lvert P_i \cap L_i \rvert}{\sum_{i=0}^{N-1} \lvert P_i \cap L_i \rvert + \sum_{i=0}^{N-1} \lvert P_i - L_i \rvert}$
Micro recall	$\frac{TP}{TP+FN} = \frac{\sum_{i=0}^{N-1} \lvert P_i \cap L_i \rvert}{\sum_{i=0}^{N-1} \lvert P_i \cap L_i \rvert + \sum_{i=0}^{N-1} \lvert L_i - P_i \rvert}$
Micro F1 Measure	$2 \cdot \frac{TP}{2 \cdot TP + FP + FN} = 2 \cdot \frac{\sum_{i=0}^{N-1} \lvert P_i \cap L_i \rvert}{2 \cdot \sum_{i=0}^{N-1} \lvert P_i \cap L_i \rvert + \sum_{i=0}^{N-1} \lvert L_i - P_i \rvert + \sum_{i=0}^{N-1} \lvert P_i - L_i \rvert}$

Multilabel Classifiation - MLlib Objects

- RDD based API (classic MLib) https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.mllib.evaluation.MultilabelMetrics
- Dataframe based API (Couldn’t find one)

Ranking Systems

Return a set of recommendations to a user

Definition of relevance may vary

Metrics aim to quantity the effectiveness of the rankings in various contexts

Ranking Systems - Variable Definitions

- A ranking system usually deals with a set of M users

$$U=\{u_0,u_1,...,u_{M-1}\}$$

- Each user (u_i) having a set of N ground truth relevant documents

$$D_i=\{d_0,d_1,...,d_{N-1}\}$$

- And a list of Q recommended documents, in order of decreasing relevance

$$R_i=[r_0,r_1,...,r_{Q-1}]$$

Ranking System - Metrics

Metric	Definition	Notes
Precision at k	$p(k) = \frac{1}{M} \sum_{i=0}^{M-1} \frac{1}{k} \sum_{j=0}^{\min(\lvert D\rvert,k)-1} rel_{D_i}(R_i(j))$	Precision at k is a measure of how many of the first k recommended documents are in the set of true relevant documents averaged across all users. In this metric, the order of the recommendations is not taken into account.
Mean Average Precision	$MAP = \frac{1}{M} \sum_{i=0}^{M-1} \frac{1}{\lvert D_i\rvert} \sum_{j=0}^{Q-1} \frac{rel_{D_i}(R_i(j))}{j+1}$	MAP is a measure of how many of the recommended documents are in the set of true relevant documents, where the order of the recommendations is taken into account (i.e. penalty for highly relevant documents is higher).
Normalized Discounted Cumulative Gain	$NDCG(k) = \frac{1}{M} \sum_{i=0}^{M-1} \frac{1}{IDCG(D_i,k)} \sum_{j=0}^{n-1} \frac{rel_{D_i}(R_i(j))}{\ln(j+1)}$ Where $n = \min\left(\max\left(\lvert R_i\rvert, \lvert D_i\rvert\right), k\right)$ $IDCG(D,k) = \sum_{j=0}^{\min(\lvert D\rvert,k)-1} \frac{1}{\ln(j+1)}$	NDCG at k is a measure of how many of the first k recommended documents are in the set of true relevant documents averaged across all users. In contrast to precision at k, this metric takes into account the order of the recommendations (documents are assumed to be in order of decreasing relevance).

Regression

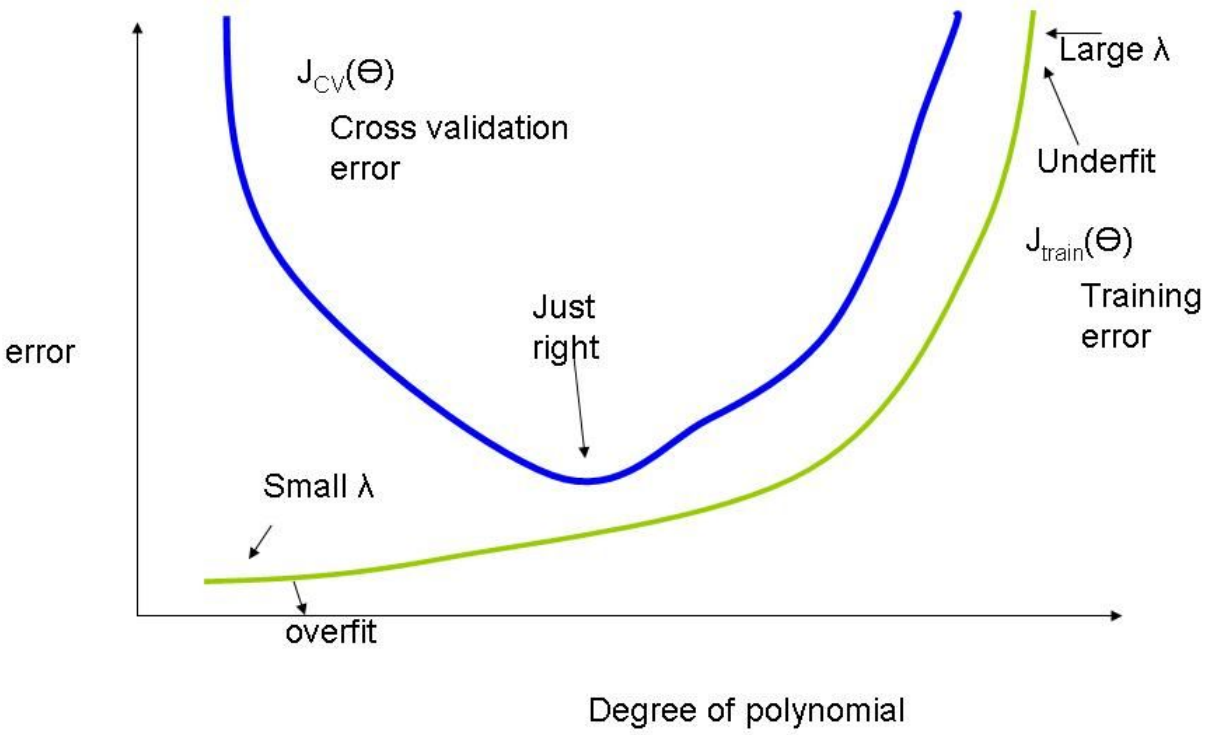
Predicting a value or values based on features. An example would be predicting the **Sale value** of a home and **weeks on the market** before a sale.

Linear Regression - Finds a line through the training data that minimizes sums of the errors. Assumes output follows Gaussian Distributions.
Polynomial Regression (not in Spark?) - Finds a line through the training data but uses Polynomials of features in addition to direct feature weights to provide a better fit to data which isn't purely linear to minimize sums of errors.
Generalized linear regression - Finds linear model where output data is an Exponential distribution.
Decision tree regression -

Classic Regresssion Tuning

Polynomial / Complexity and Adjusting other Tuning Parameters over attempts.

Compares Training error vs.
Cross Validation error



Modern Regression Tuning Techniques

LASSO (least absolute shrinkage and selection operator) - Variable Selection and Regularization to enhance prediction accuracy

Elastic Nets (used in Spark MLib):
http://users.stat.umn.edu/~zouxx019/Papers/elasticnet.pdf

Also supports Variable Selection and Regularization similar to LASSO but better performance when Predictors >> Observations which LASSO performs poorly with.

$$\alpha(\lambda\|\mathbf{w}\|_1) + (1-\alpha)\left(\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right), \alpha \in [0,1], \lambda \geq 0$$

α = 1 similar to LASSO, α = 0 model reduces to ridge regression.

RegressionEvaluator and Tuning

RegressionEvaluator - Evaluator for regression
https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.ml.evaluation.RegressionEvaluator

ParamGridBuilder - Used to specify a grid of parameter values
https://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.ml.tuning.ParamGridBuilder

CrossValidator - k-fold sampling to tune model parameters
https://spark.apache.org/docs/latest/api/java/org/apache/spark/ml/tuning/CrossValidator.html

TrainValidationSplit - Only uses parameter set once not k - times like CrossValdiator
http://spark.apache.org/docs/latest/api/java/org/apache/spark/ml/tuning/TrainValidationSplit.html

Decission Trees

Input Columns

Param name	Type(s)	Default	Description
labelCol	Double	"label"	Label to predict
featuresCol	Vector	"features"	Feature vector

Output Columns

Param name	Type(s)	Default	Description
predictionCol	Double	"prediction"	Predicted label
varianceCol	Double		The biased sample variance of prediction

Random Forests

Are ensembles of Decision Trees - Use multiple decision trees to reduce overfitting.

Tunable Parameters:

numTrees: Number of trees in the forest.

Increasing the number of trees will decrease the variance in predictions, improving the model's test-time accuracy. Training time increases roughly linearly in the number of trees.

maxDepth: Maximum depth of each tree in the forest.

Increasing the depth makes the model more expressive and powerful. However, deep trees take longer to train and are also more prone to overfitting. In general, it is acceptable to train deeper trees when using random forests than when using a single decision tree. One tree is more likely to overfit than a random forest (because of the variance reduction from averaging multiple trees in the forest).

Random Forests (cont)

These last two typically don't need to be tuned but can be to speed up training:

subsamplingRate: This parameter specifies the size of the dataset used for training each tree in the forest, as a fraction of the size of the original dataset. The **default (1.0) is recommended**, but **decreasing this fraction can speed up training**.

featureSubsetStrategy: **Number of features** to use as **candidates for splitting** at each tree node. The number is specified as a fraction or function of the total number of features. **Decreasing this number will speed up training**, but can sometimes impact performance if too low.

Gradient-boosted tree regression

Supports only binary classification and regression. Multiclass not supported.

Iterates through training new ensemble to improve performance of prediction on previous poor predictions.

Re-labeling defined by by a Loss function. With each iteration further reduces this loss function on the training data.

Gradient Boosted Trees (cont)

Losses

The table below lists the losses currently supported by GBTs in `spark.mllib`. Note that each loss is applicable to one of classification or regression, not both.

Notation: N = number of instances. y_i = label of instance i. x_i = features of instance i. $F(x_i)$ = model's predicted label for instance i.

Loss	Task	Formula	Description
Log Loss	Classification	$2\sum_{i=1}^{N} \log(1 + \exp(-2y_i F(x_i)))$	Twice binomial negative log likelihood.
Squared Error	Regression	$\sum_{i=1}^{N} (y_i - F(x_i))^2$	Also called L2 loss. Default loss for regression tasks.
Absolute Error	Regression	$\sum_{i=1}^{N} \lvert y_i - F(x_i) \rvert$	Also called L1 loss. Can be more robust to outliers than Squared Error.

MITx Analytis Edge course

https://courses.edx.org/courses/course-v1:MITx+15.071x_3+1T2016/info

Video: Unit 2: Linear Regression > Moneyball: The Power of Sports Analytics > **Video 3: Predicting Runs**

Discusses how to apply R^2 and other linear regression coefficients and metrics to evaluate a model's performance.

DEMOs

Cautions and Special Cases

- When True Positives are rare or non existent
  (example where you have few samples where a patient has a rare condition)
- Number of Predictors > or >> Observations
- Models with performance too close to 1.0 or too close to 0.5 - 1.0 mean way overfit and 0.5 means model is no better than a flip of a coin. Ideally models should be closer to 0.85 - 0.95.
- Be honest - better no model than one you have poor confidence.
- Everything we do in Data Science is a Tradeoff of Data, Performance and Resources.