Machine Learning with Spark

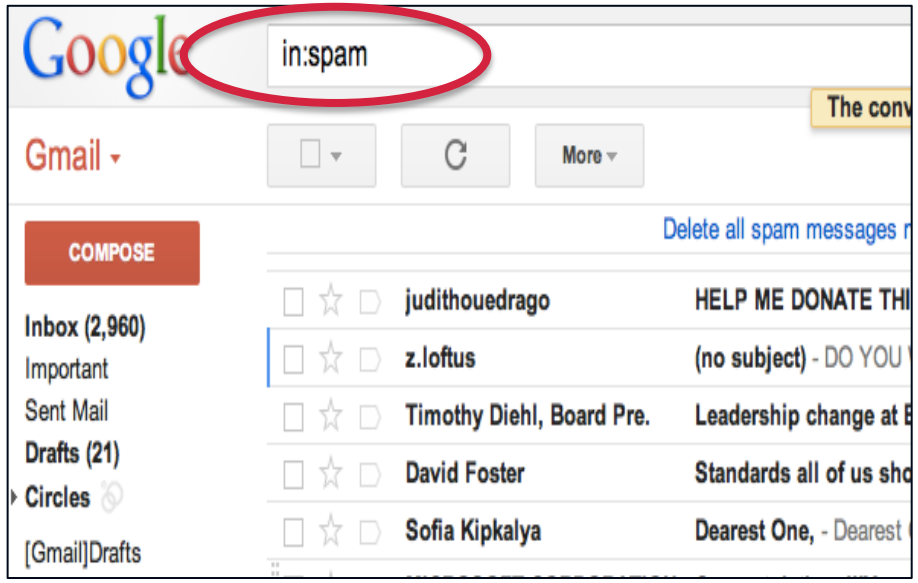
Carol McDonald

Agenda

- Classification
- Clustering
- Collaborative Filtering with Spark
- Model training
- Alternating Least Squares
- The code

Three Categories of Techniques for Machine Learning

classification



identifies category for item

clustering

Collaborative filtering (recommendation)

Recommend items

Groups similar items

What is classification

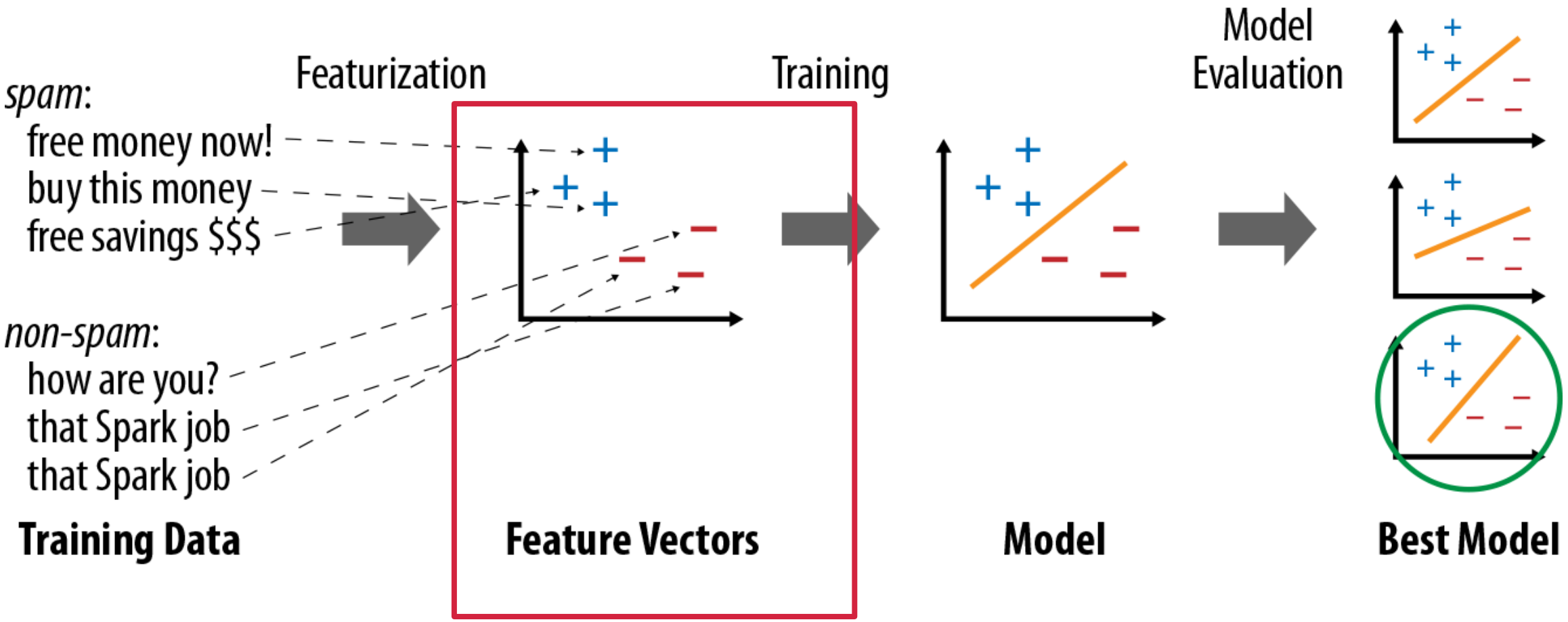
Form of ML that:

- **identifies** which **category** an **item** belongs to
- Uses supervised learning algorithms
  - Data is **labeled**

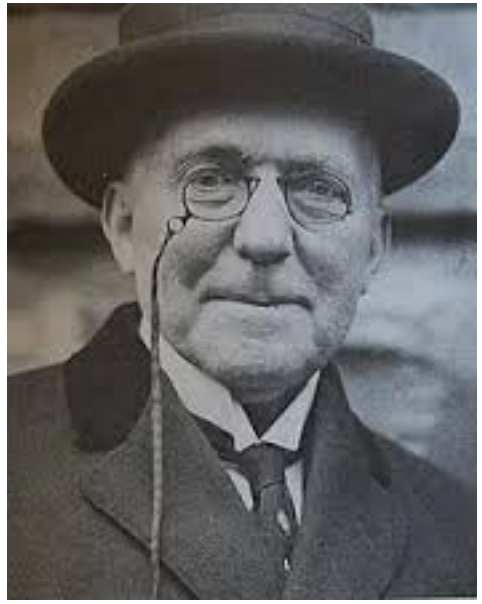
Examples:

- Spam Detection
  - spam/non-spam
- Credit Card Fraud Detection
  - fraud/non-fraud
- Sentiment analysis

Building and deploying a classifier model



If it walks/swims/quacks like a duck …



"When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck."

Attributes, Features:

- If it **walks**
- If it **swims**
- If it **quacks**

Answer, Label:

- Duck
- **Not duck**

classify something based on **"if" conditions.**

… then it must be a duck

ducks

walks

swims

quacks

not ducks

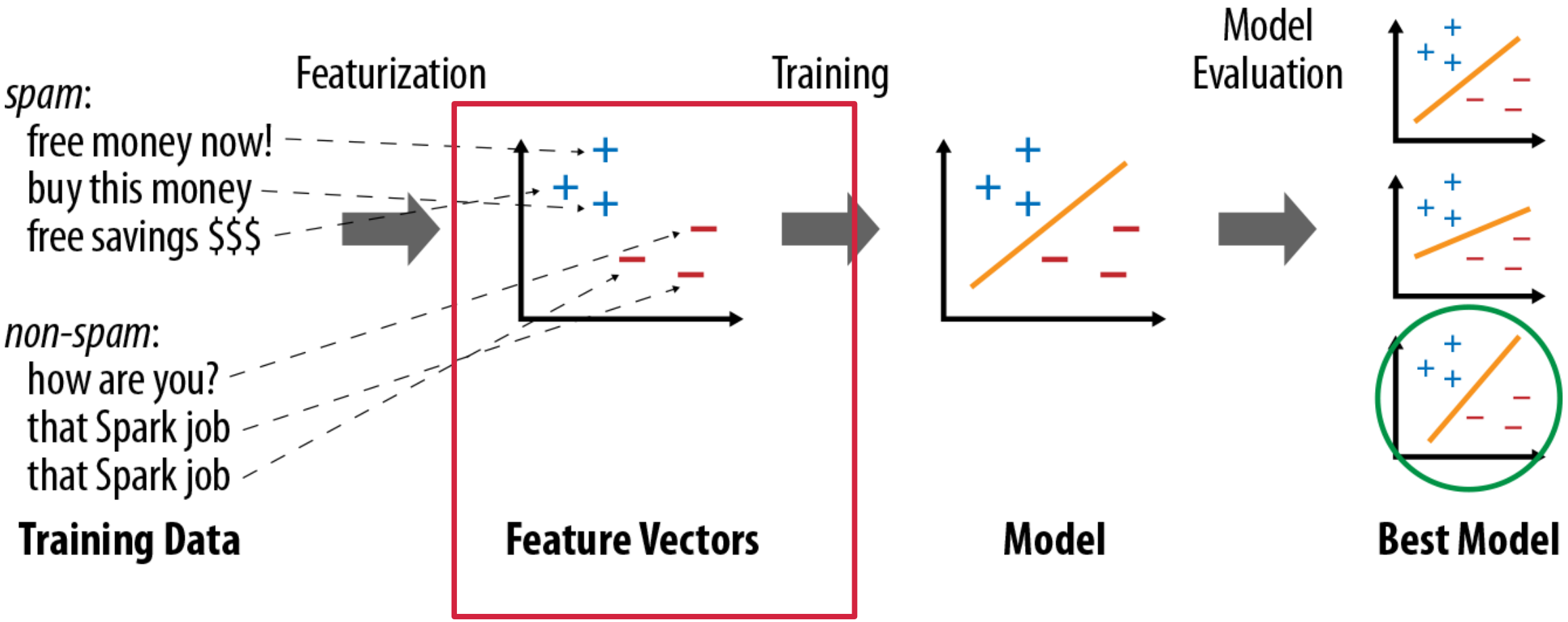
Features:

- **walks**
- **swims**
- **quacks**

Label:

- Duck
- **Not duck**

Building and deploying a classifier model

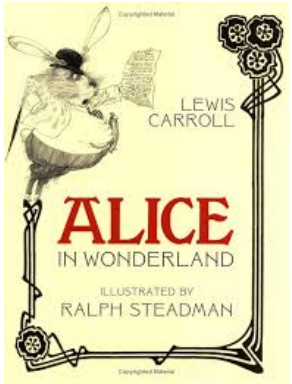


Vectorizing Data

- identify interesting features (those that contribute to the model)
- assign features to dimensions

Example: vectorize a text document
(Term Frequency Inverse Term Frequency)

Dictionary: [a, advance, after, …, you, yourself, youth, zigzag]



[223,1,1,0,…,12,10,6,1]

Example: vectorize an apple

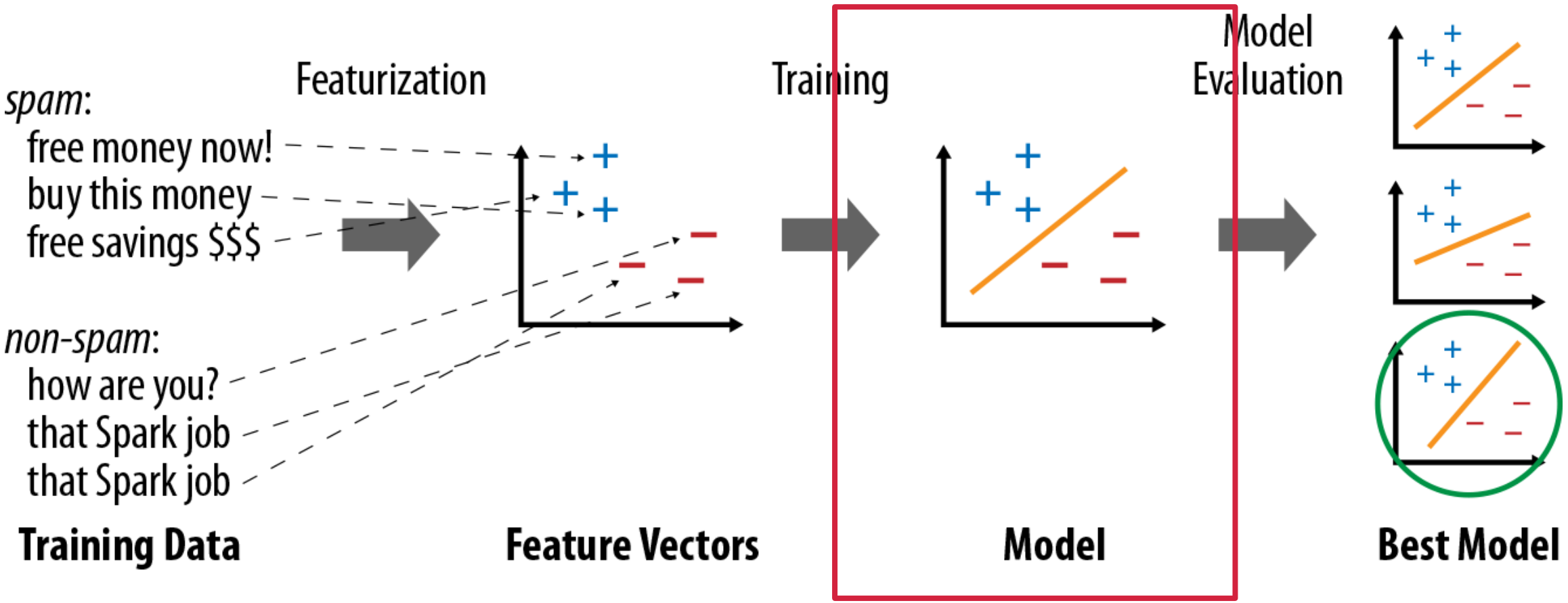
Features: [size, color, weight]

[3.2, 16777184.0, 45.8]

Build Term Frequency Feature vectors

```
// examples of spam
val spam = sc.textFile("spam.txt")
// examples of not spam
val normal = sc.textFile("normal.txt”)
// Create a HashingTF map email text to vectors of features
val tf = new HashingTF(numFeatures = 10000)
// Each email each word is mapped to one feature.
val spamFeatures = spam
    .map(email => tf.transform(email.split(" ")))
val normalFeatures = normal
    .map(email => tf.transform(email.split(" ")))
```

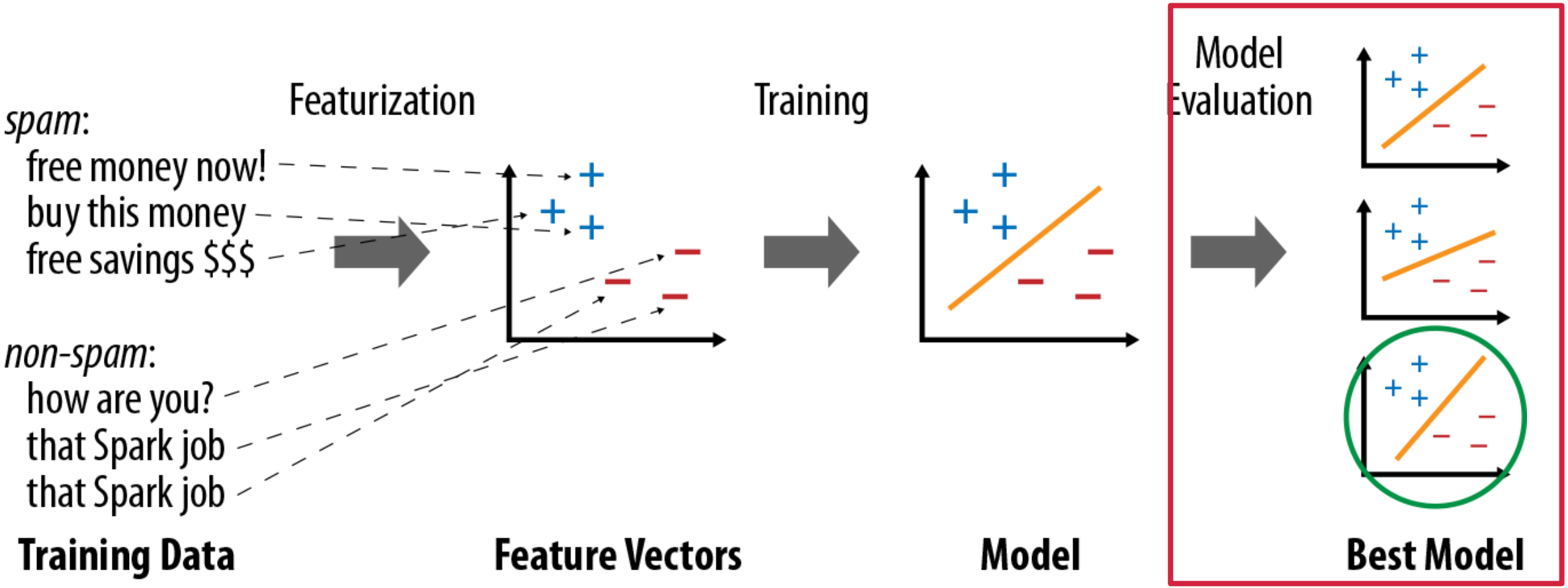
Building and deploying a classifier model



Build Model

```
val trainingData = positiveExamples.union(negativeExamples)
trainingData.cache() // Cache for iterative algorithm.

// Run Logistic Regression using the SGD algorithm.
val model = new LogisticRegressionWithSGD()
    .run(trainingData)
```

Building and deploying a classifier model

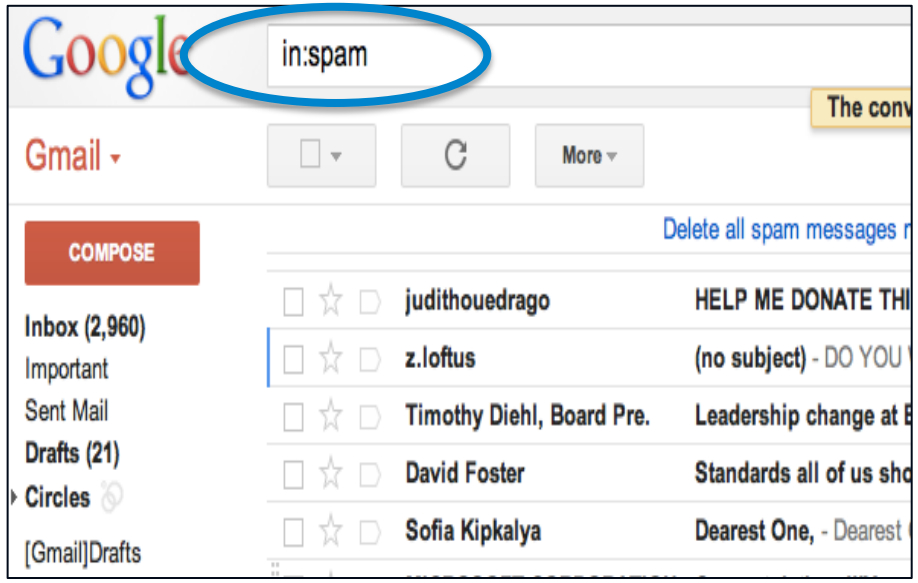


Model Evaluation

```
// Test on a positive example (spam)
Vector posTest = tf.transform(Arrays.asList(
    "O M G GET cheap stuff by sending money to...".split(" ")));
// negative test not spam
Vector negTest = tf.transform(Arrays.asList(
    "Hi Dad, I started studying Spark the other ...".split(" ")));

System.out.println("Prediction for positive: " +
    model.predict(posTest));
System.out.println("Prediction for negative: " +
    model.predict(negTest));
```

Three Categories of Techniques for Machine Learning

classification

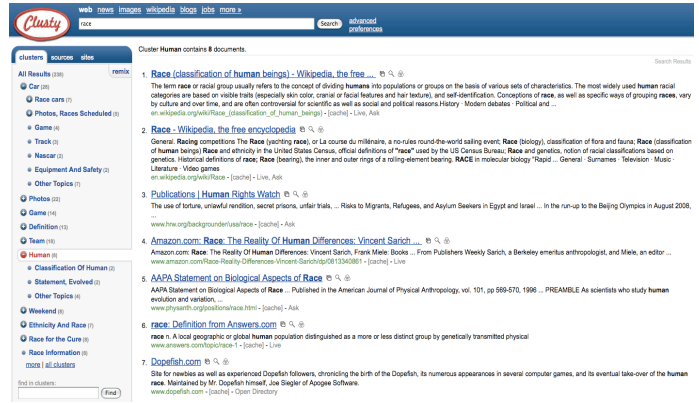


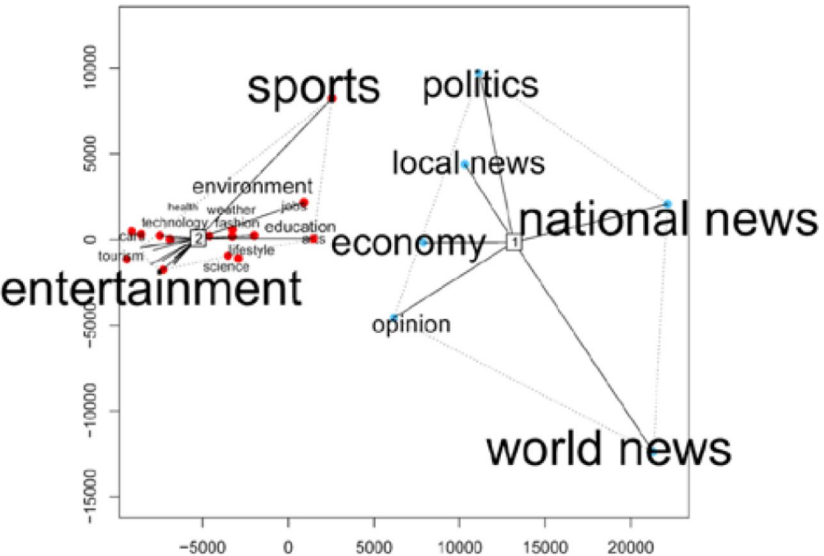
clustering

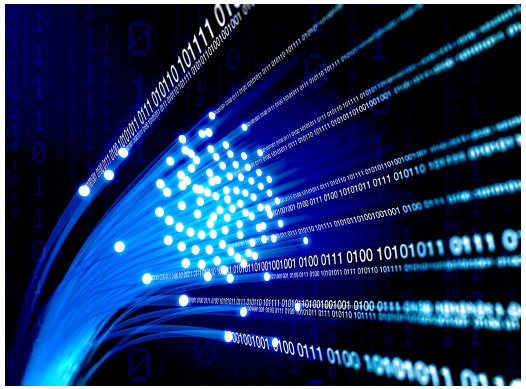
Collaborative filtering (recommendation)

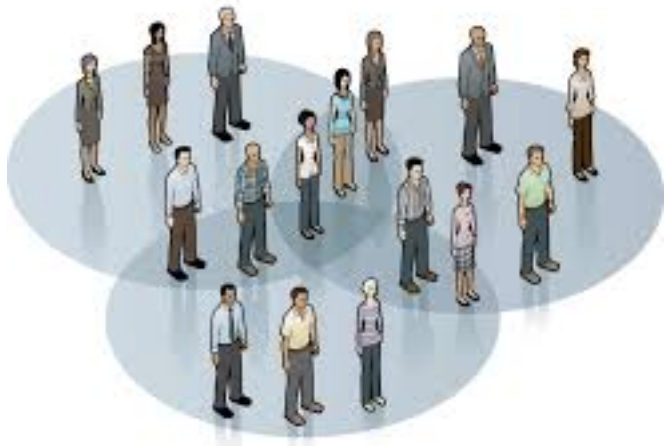
Clustering

- Clustering is the unsupervised learning task that involves grouping objects into ***clusters*** **of high similarity**
  - **Search results** grouping
  - **grouping of customers** by similar habits
  - **Anomaly** detection
    - data traffic
  - Text categorization



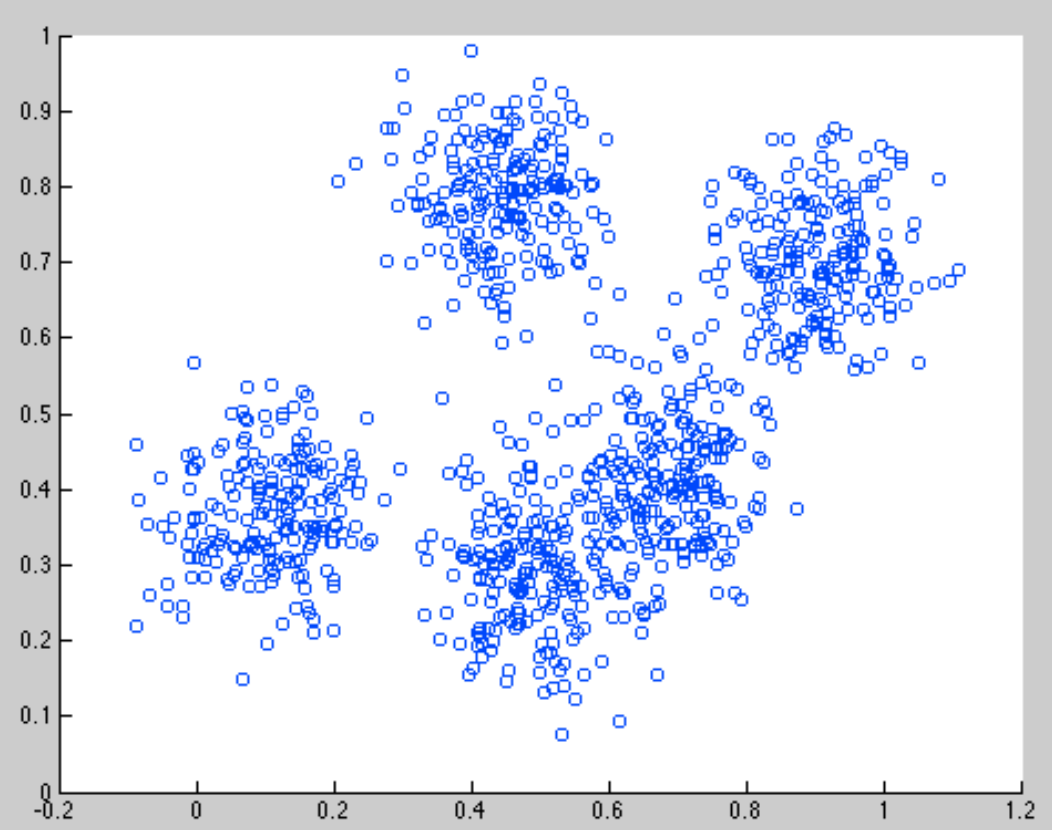






What is Clustering?

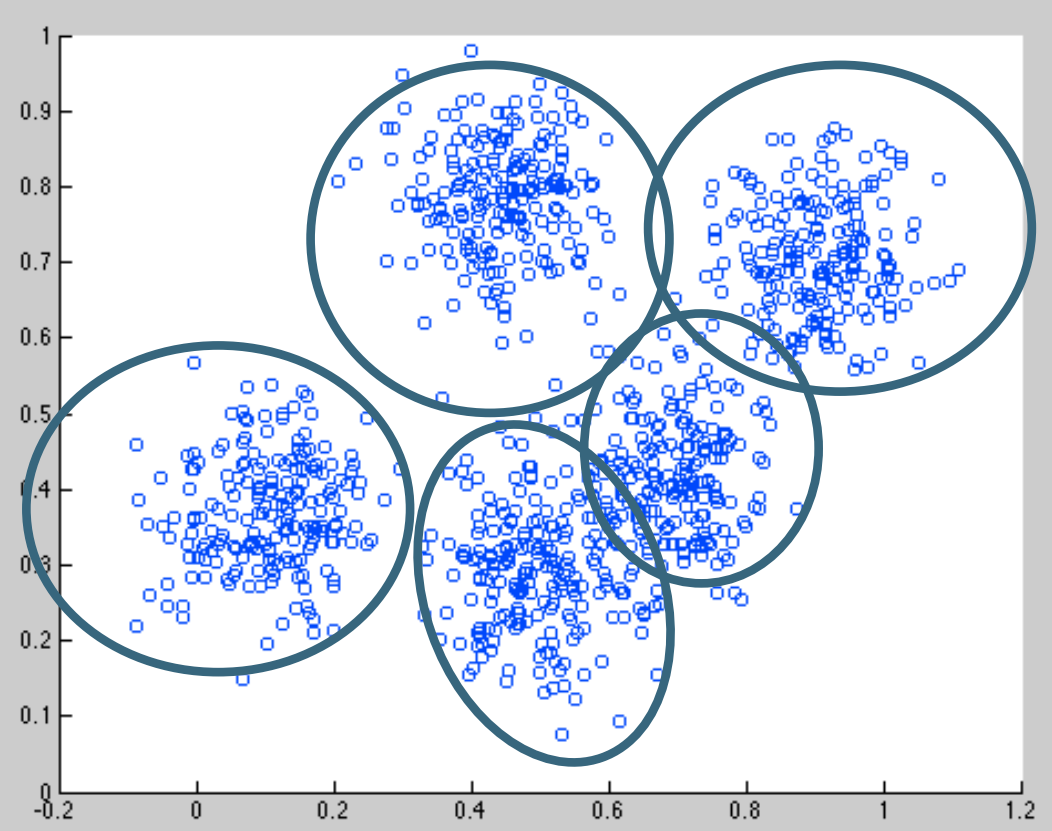
Clustering = (unsupervised) task of grouping similar objects
MLlib K-means algorithm for clustering



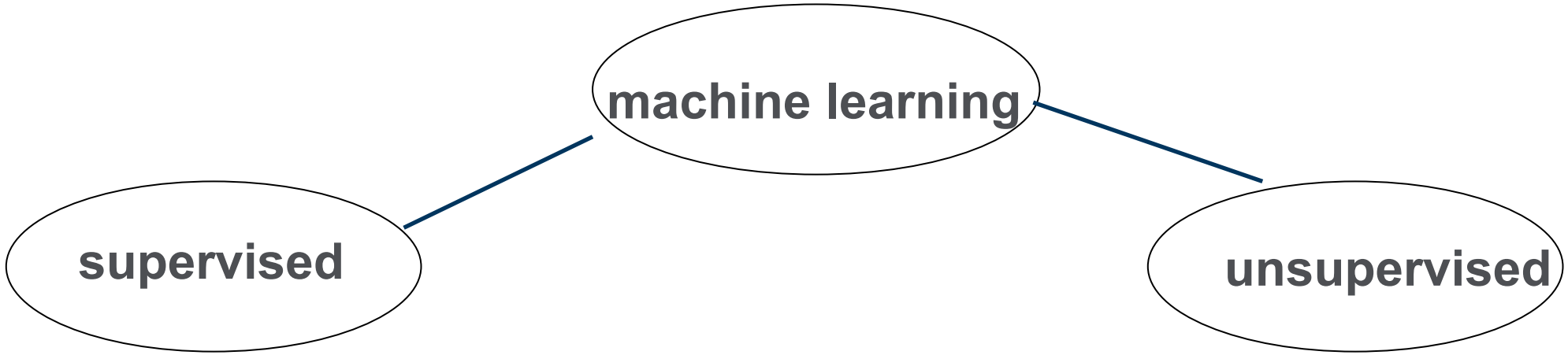
1. **randomly initialize centers of clusters**
2. **Assign all points to the closest cluster center**
3. **Change cluster centers to be in the middle of its points**
4. **Repeat until convergence**

What is Clustering?

Clustering = (unsupervised) task of grouping similar objects



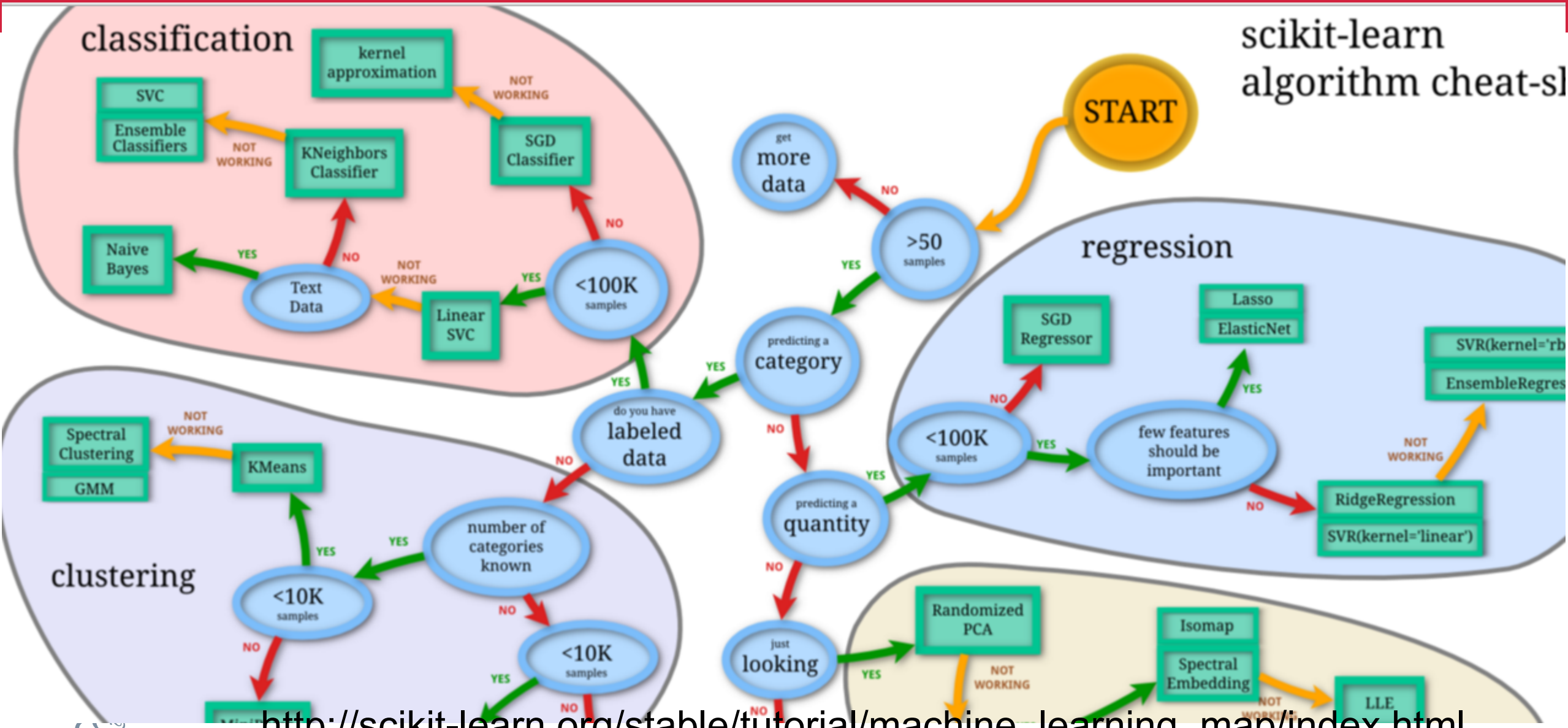
Examples of ML Algorithms



- **Classification**
  - Naïve Bayes
  - SVM
  - Random Decision Forests
- **Regression**
  - Linear
  - logistic

- **Clustering**
  - K-means
- **Dimensionality reduction**
  - Principal Component Analysis
  - SVD

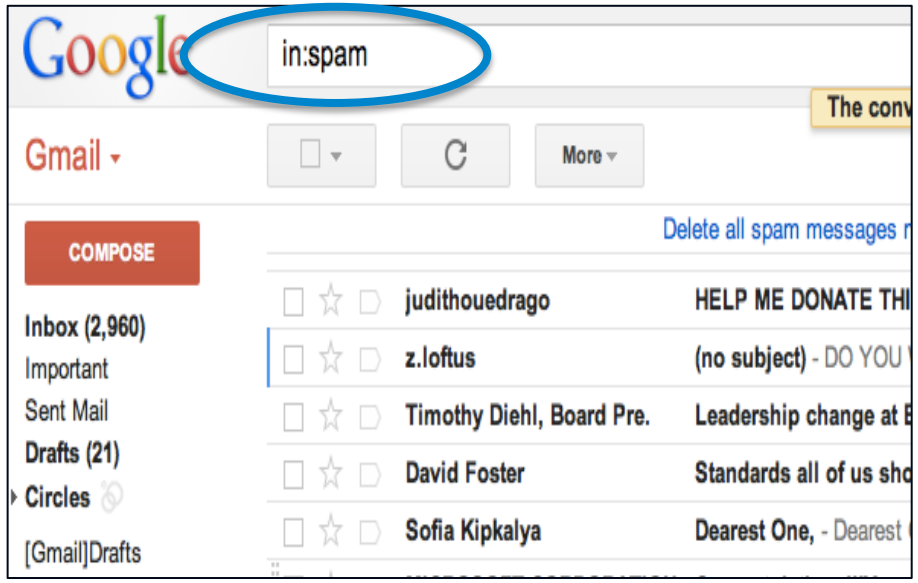
ML Algorithms



http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Three Categories of Techniques for Machine Learning

classification



clustering

Collaborative filtering (recommendation)

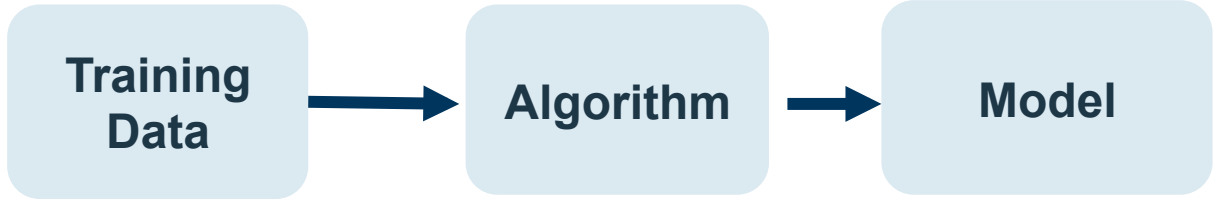
Collaborative Filtering with Spark

- Recommend Items
  - (filtering)
- Based on User preferences data
  - (collaborative)

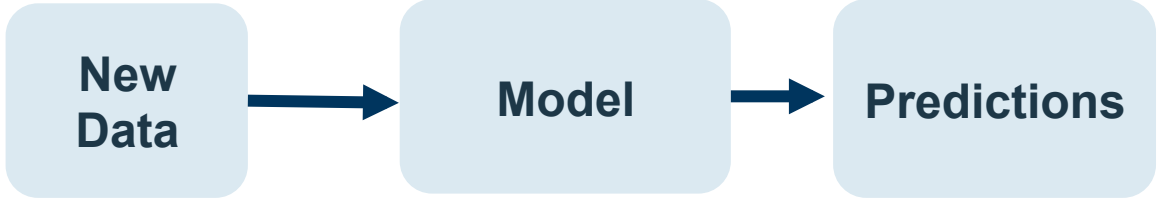
Users Item Rating Matrix

Train a Model to Make Predictions

Ted and Carol like Movie B and C

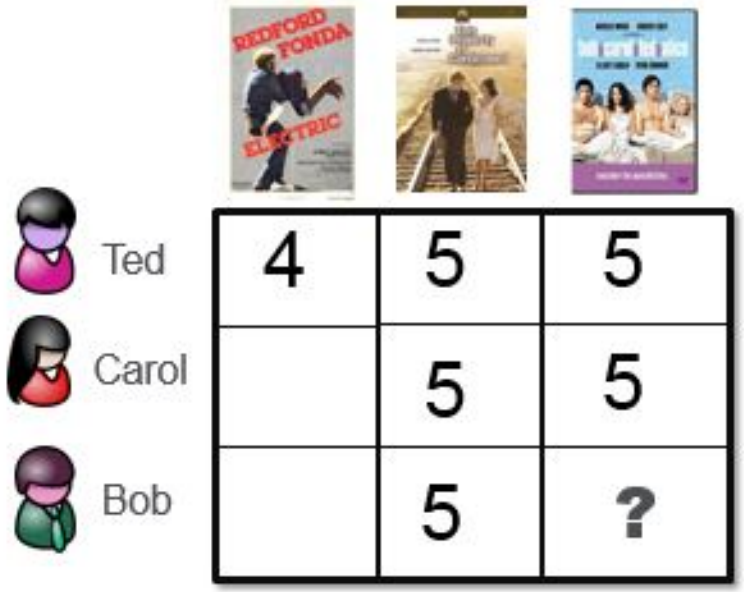


Bob likes Movie B, What might he like ?



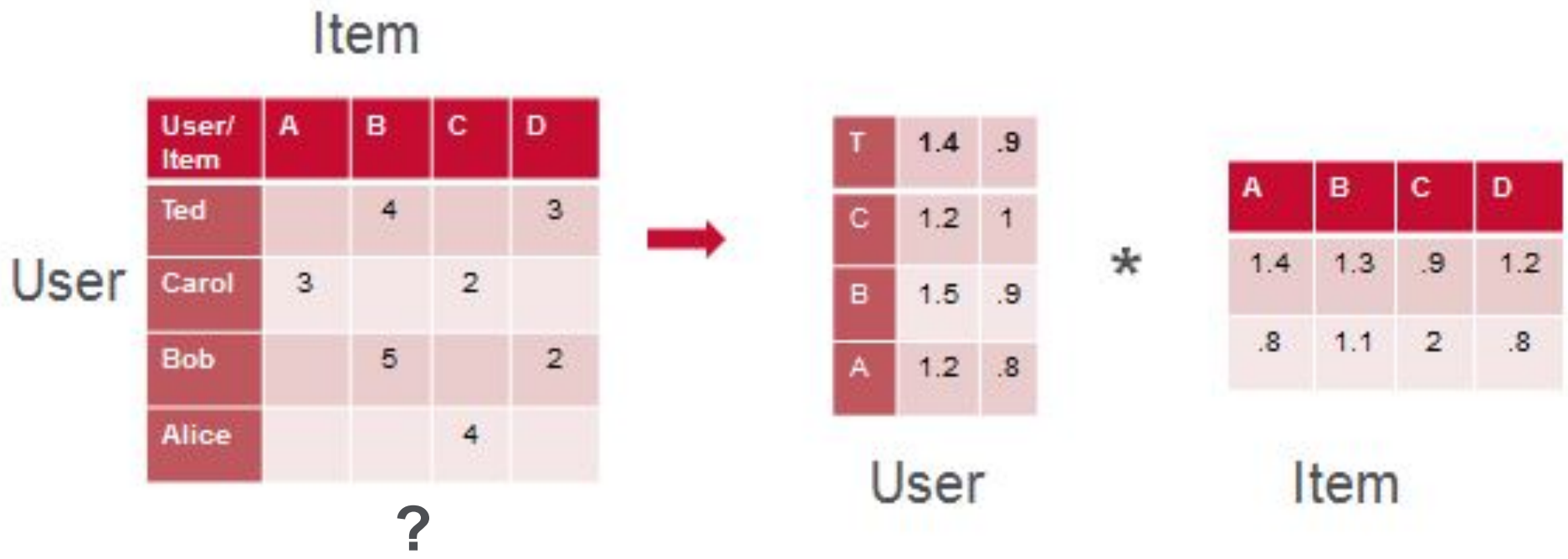
Bob likes Movie B, **Predict C**

Users Item Rating Matrix

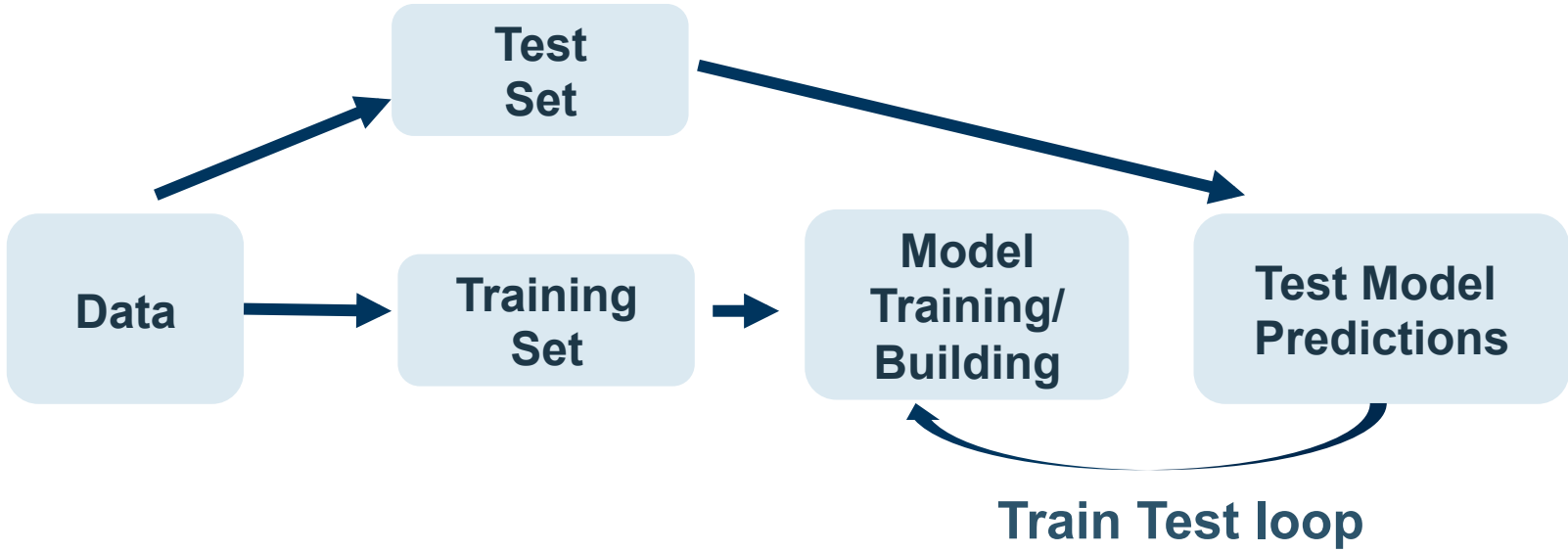


Alternating Least Squares

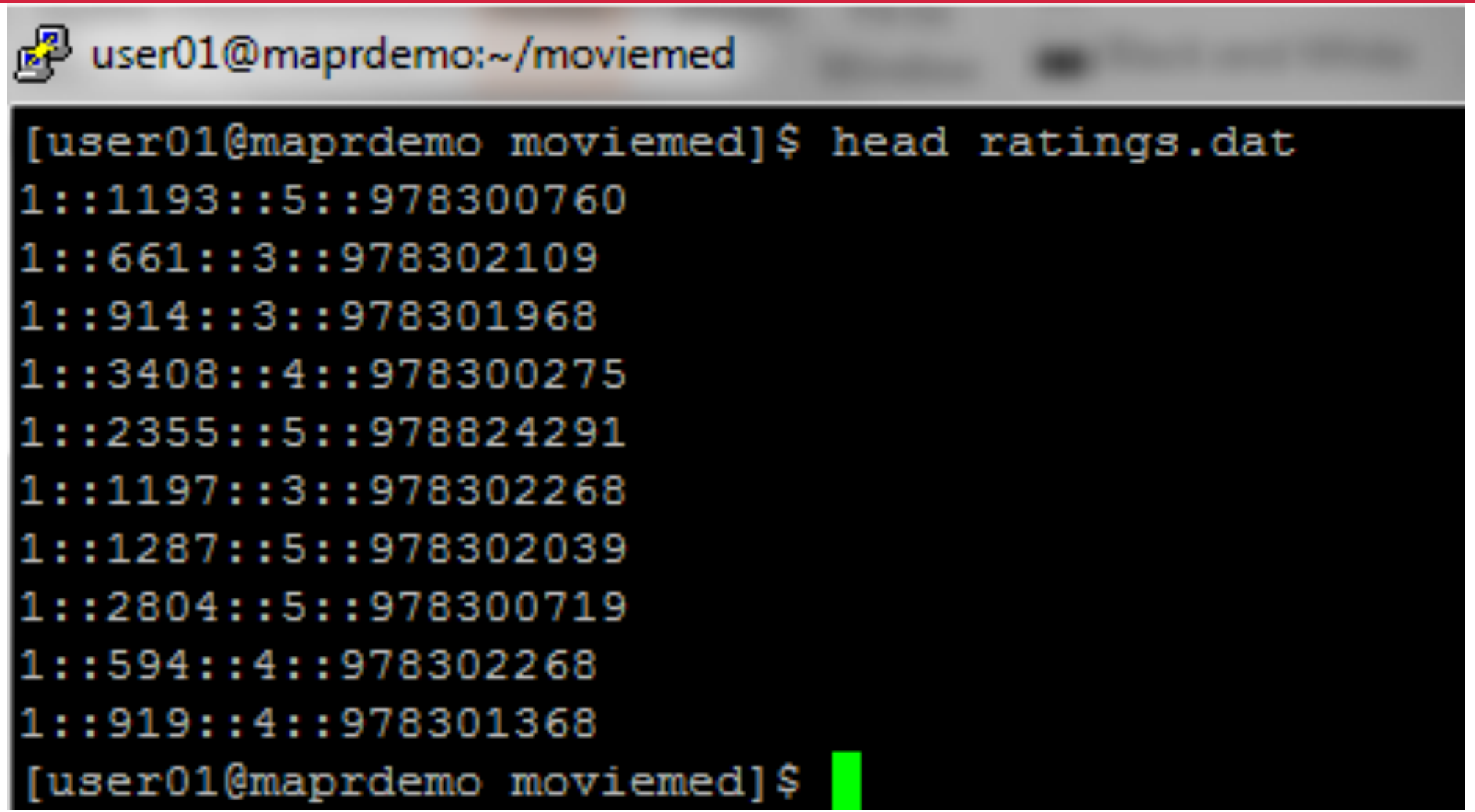
- approximates sparse user item rating matrix
  - as product of two dense matrices, User and Item factor matrices
  - tries to learn the hidden features of each user and item
  - algorithm alternatively fixes one factor matrix and solves for the other



ML Cross Validation Process



Ratings Data

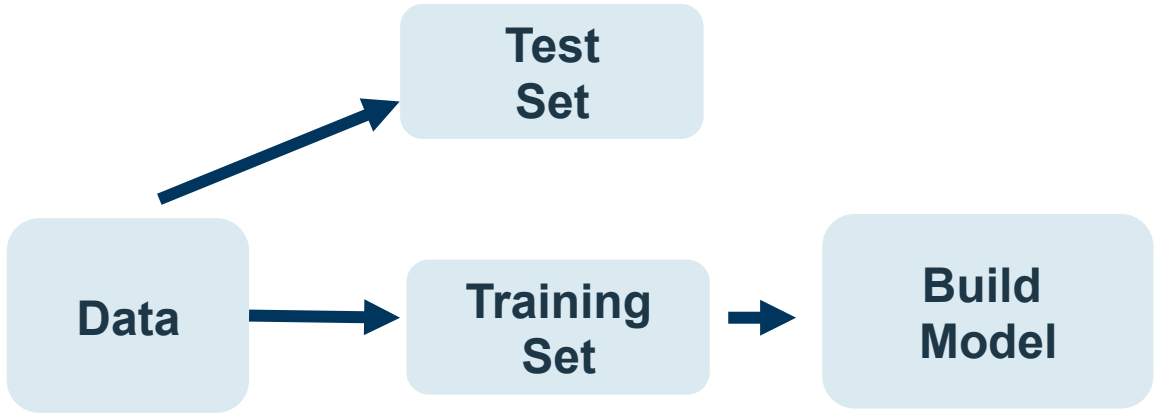


Parse Input

```
// parse input UserID::MovieID::Rating
def parseRating(str: String): Rating= {
    val fields = str.split("::")
    Rating(fields(0).toInt, fields(1).toInt,
      fields(2).toDouble)
}

// create an RDD of Ratings objects
val ratingsRDD = ratingText.map(parseRating).cache()
```

Build Model

```
split ratings RDD into training data RDD (80%)
and test data RDD (20%)
```



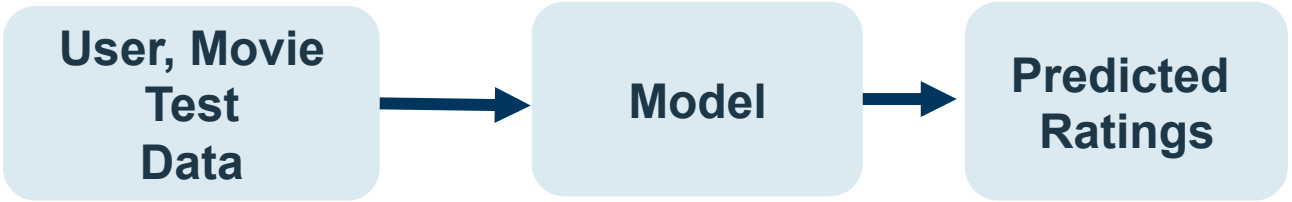

```
build a user product matrix model
```

Create Model

```
// Randomly split ratings RDD into training data RDD (80%)
and test data RDD (20%)
val splits = ratingsRDD.randomSplit(Array(0.8, 0.2), 0L)

val trainingRatingsRDD = splits(0).cache()
val testRatingsRDD = splits(1).cache()

 // build a ALS user product matrix model with rank=20,
iterations=10
val model = (new ALS().setRank(20).setIterations(10)
    .run(trainingRatingsRDD))
```

Get predictions

```
// get predicted ratings to compare to test ratings
val testUserProductRDD = testRatingsRDD.map {
  case Rating(user, product, rating) => (user, product)
}
// call  model.predict with test Userid, Movield  input data
val predictionsForTestRDD = model.predict(testUserProductRDD)
```



Compare predictions to Tests

Join predicted ratings to test ratings in order to compare

Key, Value

```
((user, product),test rating)
```

Key, Value

```
((user, product), predicted rating)
```

Key, Value

```
((user, product),(test rating, predicted rating))
```

Test Model

```
// prepare  predictions for comparison
val predictionsKeyedByUserProductRDD = predictionsForTestRDD.map{
  case Rating(user, product, rating) => ((user, product), rating)
}
// prepare  test for comparison
val testKeyedByUserProductRDD = testRatingsRDD.map{
  case Rating(user, product, rating) => ((user, product), rating)
}
//Join the  test with  predictions
val testAndPredictionsJoinedRDD = testKeyedByUserProductRDD
  .join(predictionsKeyedByUserProductRDD)
```

Compare predictions to Tests

```
Find False positives: Where
    test rating <= 1  and   predicted rating >= 4
```

Key, Value

```
((user, product),(test rating, predicted rating))
```

Test Model

```
val falsePositives =(testAndPredictionsJoinedRDD.filter{
  case ((user, product), (ratingT, ratingP)) =>
       (ratingT <= 1 && ratingP >=4)
  })
falsePositives.take(2)
```

```
Array[((Int, Int), (Double, Double))] =
((3842,2858),(1.0,4.106488210964762)),
((6031,3194),(1.0,4.790778049100913))
```

Test Model Mean Absolute Error

```
//Evaluate the model using Mean Absolute Error (MAE) between
test and predictions
val meanAbsoluteError = testAndPredictionsJoinedRDD.map {
  case ((user, product), (testRating, predRating)) =>
    val err = (testRating - predRating)
    Math.abs(err)
}.mean()
meanAbsoluteError: Double = 0.7244940545944053
```

Get Predictions for new user

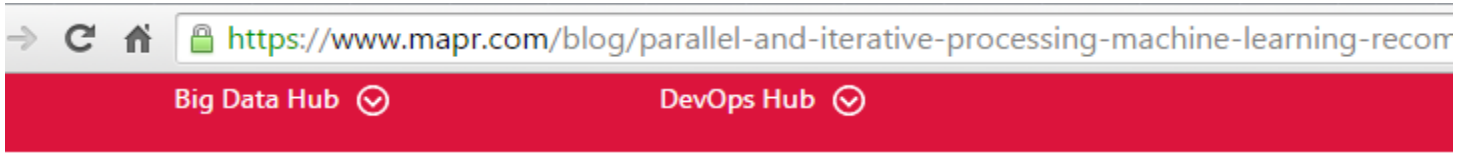
```
val newRatingsRDD=sc.parallelize(Array(Rating(0,260,4),Rating(0,1,3))
// union
val unionRatingsRDD = ratingsRDD.union(newRatingsRDD)
 // build a ALS user product matrix model
val model = (new ALS().setRank(20).setIterations(10)
    .run(unionRatingsRDD))

//  get 5 recs for userid 0
val topRecsForUser = model.recommendProducts(0, 5)
```

Soon to Come

- Spark On Demand Training
  - https://www.mapr.com/services/mapr-academy/
- Blogs and Tutorials:
  - Movie Recommendations with Collaborative Filtering
  - Spark Streaming

Machine Learning Blog

- https://www.mapr.com/blog/parallel-and-iterative-processing-machine-learning-recommendations-spark



Parallel and Iterative Processing for Machine Learning Recommendations with Spark

Spark on MapR

- Certified Spark Distribution
- Fully supported and packaged by MapR in partnership with Databricks
- YARN integration
  - Spark can then allocate resources from cluster when needed

References

- Spark Online course: learn.mapr.com
- Spark web site: http://spark.apache.org/
- https://databricks.com/
- Spark on MapR:
  - http://www.mapr.com/products/apache-spark
- Spark SQL and DataFrame Guide
- Apache Spark vs. MapReduce – Whiteboard Walkthrough
- Learning Spark - O'Reilly Book
- Apache Spark

Q&A

Engage with us!

@mapr   maprtech

mapr-technologies   MapR

  maprtech