CS 686: Special Topics in Big Data

Machine Learning With Spark

Lecture 30

Today's Schedule

- Info & Updates
- Features
- Building Machine Learning Models
- Evaluating the Models
- Feature Engineering

Today's Schedule

- **Info & Updates**
- Features
- Building Machine Learning Models
- Evaluating the Models
- Feature Engineering

Grades

- All grades (including P2) should be posted now
- Let me know if you notice any discrepancies
- We didn't have time for grading interviews this time around, **but** you are welcome to stop by to discuss your implementation or grade!
  - Like before, if you want to fix anything you definitely can.

Project 3

- Turn in collaboration plans by this evening!
- We'll have a lab on Wednesday to work on P3
- We may also have a half-lab, half-discussion day on Friday
  - Time is running out!

Today's Schedule

- Info & Updates
- **Features**
- Building Machine Learning Models
- Evaluating the Models
- Feature Engineering

Features (1/2)

- We've already talked quite a bit about features in the context of our datasets
- But we didn't really discuss: *what are features?*
- In Machine Learning, a **feature** is a measurable property or characteristic
- Choosing the right features is the most important part of machine learning
  - **Feature Engineering**

Features (2/2)

- A feature by any other name…
  - Dimension
  - Explanatory variable
  - Independent variable
- Collections of features are called **feature vectors**
  - Or in our dataset: observations
  - These come in different types

Some Feature Types

- Numeric
- Categorical
- Boolean
- String
- Graph
- Pixels

Choosing Feature Types

- Some models may expect inputs in a particular format
  - For instance, numeric
- Some are more robust to different types of data than others
  - Read the documentation! ☺
- In many cases, you can convert one feature type to another
  - Pixels: x, y, r, g, b

Numeric Features

- The most common feature type
- Think about some properties of the features…
- Units
  - May or may not be important
- Ranges
  - Are there mins/maxes or unbounded?
  - Could we truncate outliers?
    - Would this be a good or bad thing…?
- Distributions
  - Normal distribution?

Boolean Features

- A simple T/F
  - Or in our dataset Yes/No
- Can be coverted to numeric: 1/0
- While simple, can be quite powerful
- In some cases, maybe we are just interested in predicting a boolean feature

Categorical Features

- Used to describe a few specific states
- For instance: on an online store, you don't want RGB sliders to specify the color of the product you're looking for
- Can be particularly important in classification problems

Splitting Categorical Features

- It is often advantageous to convert categorical features to boolean features
- For instance, maybe you a “color” feature:
  - Red, blue, green, brown, yellow
- We can split this to be:
  - Red_YN, Blue_YN, Green_YN, Brown_YN, Yellow_YN
  - And our feature vector becomes
  - 0 1 0 0 0 (for blue)… or 0 1 1 0 0 (blue green), etc.

Today's Schedule

- Info & Updates
- Features
- **Building Machine Learning Models**
- Evaluating the Models
- Feature Engineering

Building Models

- For today's discussion, we will consider two types of models
- **Regression**
  - (Prediction, Forecasting)
- **Classification**
- Many problems can be broken down into these two categories

Regression

- Statistical process for estimating the relationships between features
  - Dependent variables – to be predicted
  - Independent variables – used to make the estimation
    - Often called predictors
- Predicting income based on job history
- Estimating how much it will rain today
- Determining how long a disease outbreak will last

Classification

- On the other hand, maybe we want to **label** something based on the data
- In some cases, we may not know what the labels are
- Related: clustering
  - Taking features and splitting them into groups
- Or for example, labeling an action in a video "sitting"
- …or whether something is a hotdog or not

Hotdog

Not Hotdog

Building a Model (1/2)

1. First, choose what you want to do: predict or classify
2. What feature (or set of features) do you want to predict?
   - These will be your dependent variables
3. Next, choose your model

Building a Model (2/2)

4. Choosing:
   - https://spark.apache.org/docs/latest/ml-classification-regression.html
   - Often best to start with something simple
   - Linear regression?
     - Not "cool" but surprisingly powerful
5. **Train** your model
6. And finally, **evaluate** your model

Training the Models

- Let's assume we've already decided on a model
- Now we need to feed it with some data so it can learn
  - **Training**
- We want to create a model that will generalize to new, unseen data points
  - But usually these new data points don't exist yet (or we don't have them)
- So instead, we **partition** our existing dataset…

Dataset Partitioning

- First step: **shuffle it!**
- Split our dataset into two parts:
  - Training dataset
  - Test dataset
- A 70/30% split is good, 90/10% is also common
  - We want to train our model using the most data possible, but we also want to be able to evaluate it well
- **Never ever** use your entire dataset to train and then test on a subset!!

Partitioning With Spark

- Lucky for us, Spark DataFrames have us covered:
- ```
  (trainingData, testData)
      = dframe.randomSplit([0.9, 0.1])
  ```
- This returns two DataFrames, partitioned and ready to go
  ```

Creating the Model: Rand. Forest

```
rf =
    RandomForestRegressor(
        numTrees=100,
        featureSubsetStrategy="auto",
        impurity='variance',
        maxDepth=10,
        maxBins=32)
```

Training the Model

```
# Chain indexer and forest in a Pipeline
pipeline = Pipeline(stages=[featureIndexer, rf])
# Train model.  This also runs the indexer.
model = pipeline.fit(trainingData)
# Make predictions.
predictions = model.transform(testData)
# Select example rows to display.
predictions.select("prediction", "label", "features").show(5)
# Select (prediction, true label) and compute test error
evaluator =
    RegressionEvaluator(labelCol="label", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

Random Forests?

- Random forests are based on **decision trees**
- At their core, just simple binary relationships:
  - Is it sunny out? (Y/N)
    - Ok, then go down this/that path…
      - Is it raining? (Y/N)
        - Ok, then go down this/that path…
- Builds many decision trees as an **ensemble**
  - Uses the knowledge of the group to come to a final answer

Today's Schedule

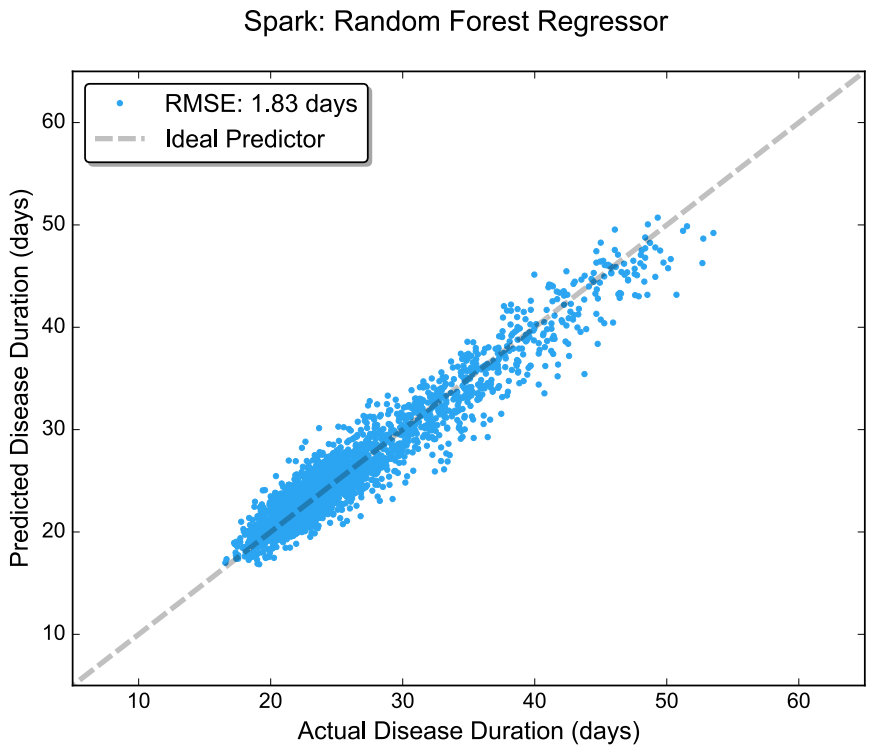
- Info & Updates
- Features
- Building Machine Learning Models
- **Evaluating the Models**
- Feature Engineering

Basic Model Evaluation

- We can compare **how far** predicted values are from the actual values
- MSE = mean squared error
- RMSE = root mean squared error
  - Describes the error in the same units
  - "We're off by 2.3 days"
- Warning: this can hide issues with the model
  - Maybe things look good **on average**…

Evaluation - Regression

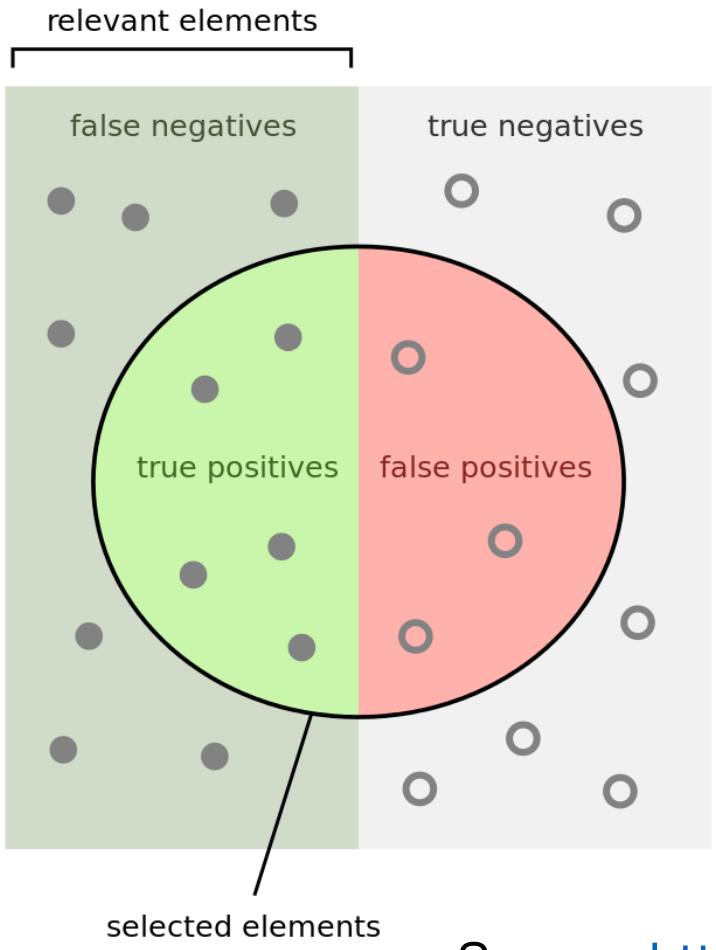
- Most common way to visualize accuracy: lag plot
- Plot actual vs predicted values
  - Same axes
- The closer to the line, the better



Evaluation - Classification

- For pattern recognition and binary classification, we can use **precision** and **recall**
- Precision – how useful the search results are
- Recall – how complete the results are
- So we look at true/false positives and true/false negatives

Precision & Recall



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

Source: https://en.wikipedia.org/wiki/Precision_and_recall

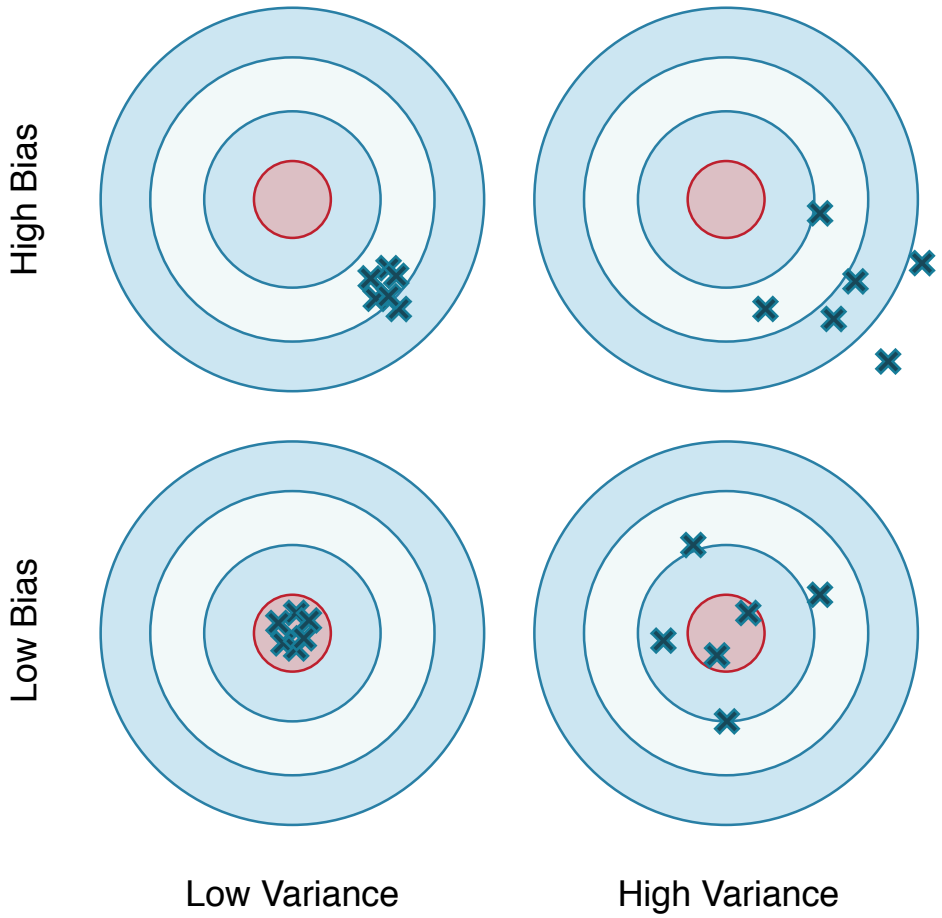
Bias-Variance Decomposition

- Another way to evaluate models is by their **bias** and **variance**

$$MSE(\hat{\theta}) = Bias(\hat{\theta})^2 + Var(\hat{\theta})$$

- Bias: overlooking some relationships in the data
  - High bias leads to **underfitting**
- Variance: how susceptible our model is to noise
  - High variance: **overfitting**

Bias and Variance



Today's Schedule

- Info & Updates
- Features
- Building Machine Learning Models
- Evaluating the Models
- **Feature Engineering**

Feature Engineering (1/2)

- Many of the recent advancements in machine learning are based on novel **feature engineering**
- In some cases, you'll use the dataset to build metafeatures that generalize better
- Starting from the ground up, we can just throw every feature into the model
  - This is a good start
  - Some features will have a bigger impact than others
  - The big problem: **computational complexity!**

Feature Engineering (2/2)

- If we don't care how long it takes to train our models, throwing the kitchen sink at them is reasonable
- But by this point, we all know that "big data" means slow I/O, huge numbers of features, and lots of waiting on spinning rust
  - Definitely not great if Elon Musk wants you to ship an autonomous Tesla truck ***yesterday…***
- So we can guide our quest to choose the right features…

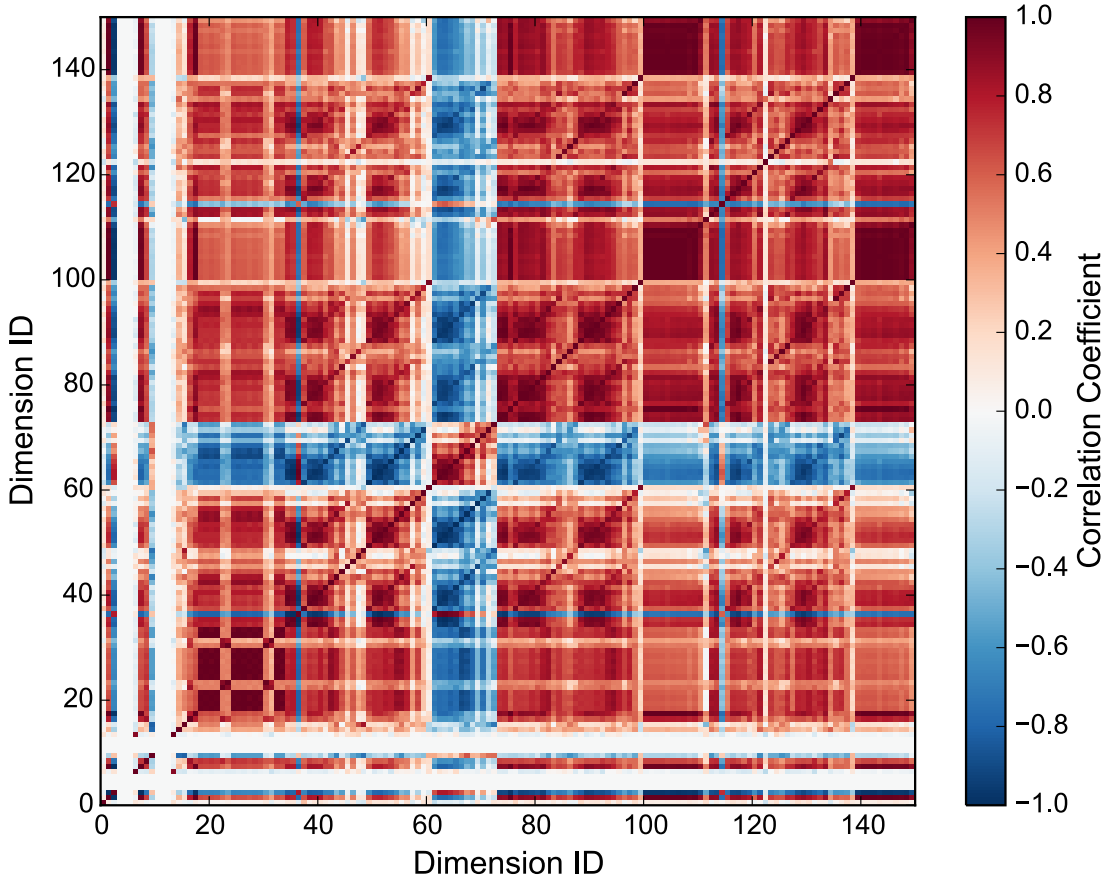
Dimensionality Reduction

- The best way to deal with big data is…
  - …not dealing with big data!
- Instead, reduce the number of dimensions to just the most important/influential
- Correlation
  - While being aware of collinearity
- Principle component analysis (PCA)
  - Scree plot

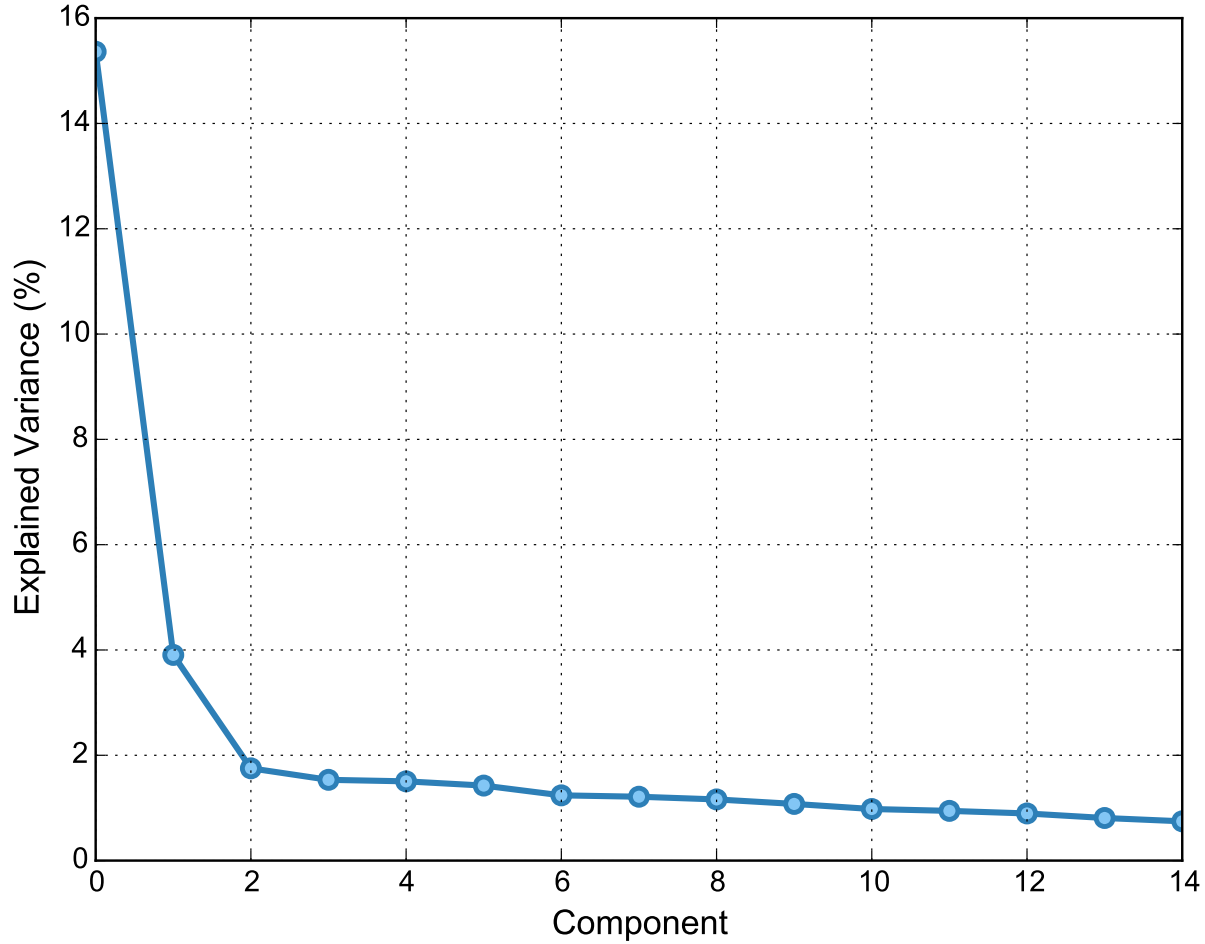
Correlation

- If one feature can predict another to some degree, then they may be correlated
- We use Pearson's Correlation Coefficient (PCC) in Project 3
- In some cases, features will exhibit **collinearity**

Correlation Heatmap



Scree Plot: The "Elbow"



Wrapping Up

- Be ready to experiment when building ML models
- In fact, some folks use machine learning models to parameterize their machine learning models
  - Yo dawg
- Another common theme: let's try training with this matrix of parameter combinations
  - Have each machine in a cluster build a model
  - Exhaustively try combinations of parameters to find the best ones