Lecture 1

IDS575: Statistical Models and Methods
Theja Tulabandhula

We are drowning in information and starving for knowledge. - Rutherford D. Roger

Notes derived from the book titled "Elements of Statistical Learning [2nd edition] (Chapters 1 and 2.1-2.3)

1 Motivation

The impact of data science on a variety of fields, businesses and areas of works needs no introduction. So, here is a list of problems that involve some sort of statistical modeling or other. I am sure you can all relate statistical modeling to your past professional experiences.

- Sports analytics
- Online retail and pricing
- Advertising and personalization
- Social networks
- Education and MOOCs
- Transportation systems
- Automatic speech recognition
- Robotics and computer vision, image processing
- Fraud detection and finance
- ICU monitoring and clinical decision support
- ...

What are some common components among these applications?

1.1 Application in Sports Analytics

Here is an example of how a decision support tool was built using statistical modeling. The context is that a team makes pit stop calls during a race. When to make a call and what to do in it constitutes a pit strategy. for example, replacing four tires (Figure 1) takes more time but the car is much better with respect to lap times.

Figure 1: Car racing: a crew member inspecting a tire.

- The problem to be addressed is the following: use live race measurements to forecast future and decide/update a pit strategy in real time.
- Is there even predictability in this setting?
- Data: text strings are beamed by the organizers over the radio, so there could be several noisy values.

Initial exploration to build features from 17 races ($\sim$ 100000 laps) resulted in the following plots (see Figure 2).

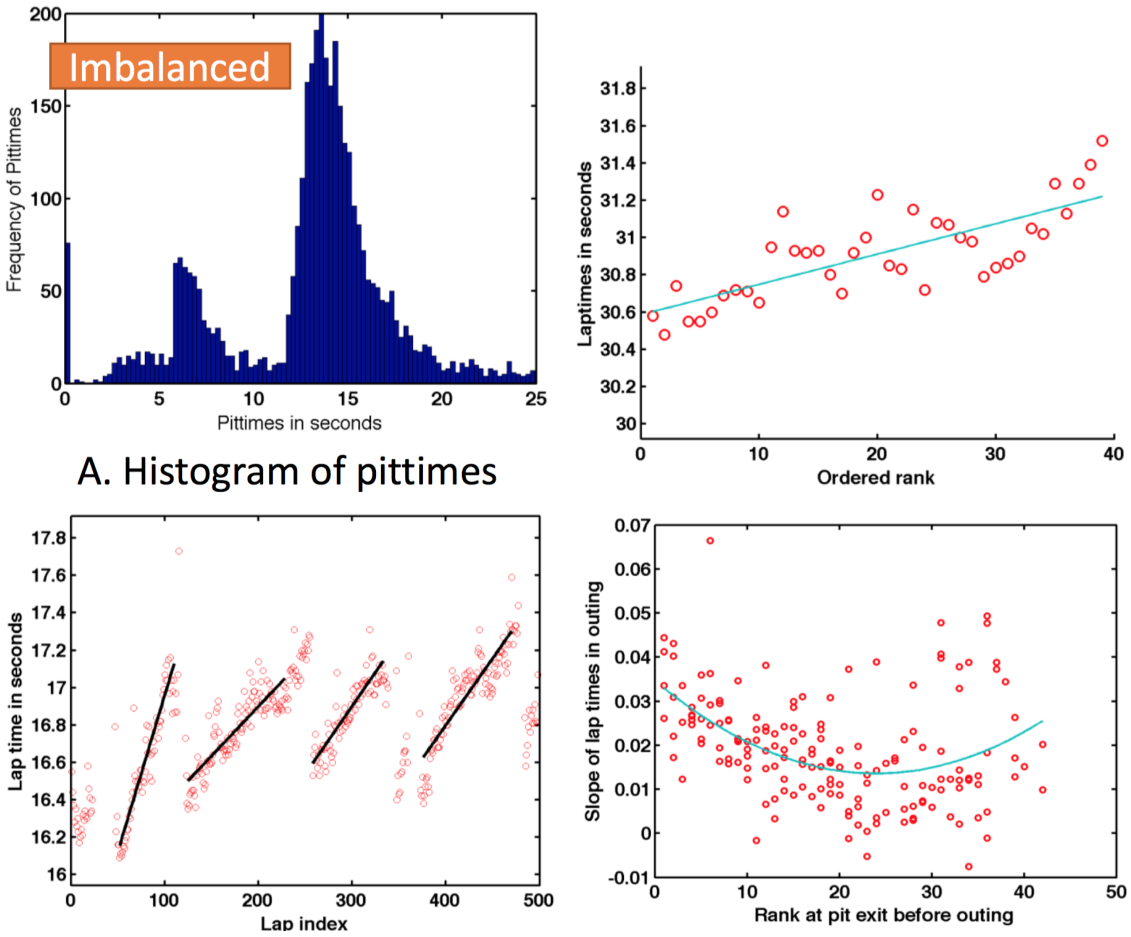

Figure 2: Some histogram and correlation plots to design features.

So after that, one can jump into using off the shelf predictive models to predict a quantity of interest. In this particular application, it was the change in rank position in the laps following a pit stop given measurements of what happened in the pits and what happened before in the race. The performance of various models is shown in Figure 3.

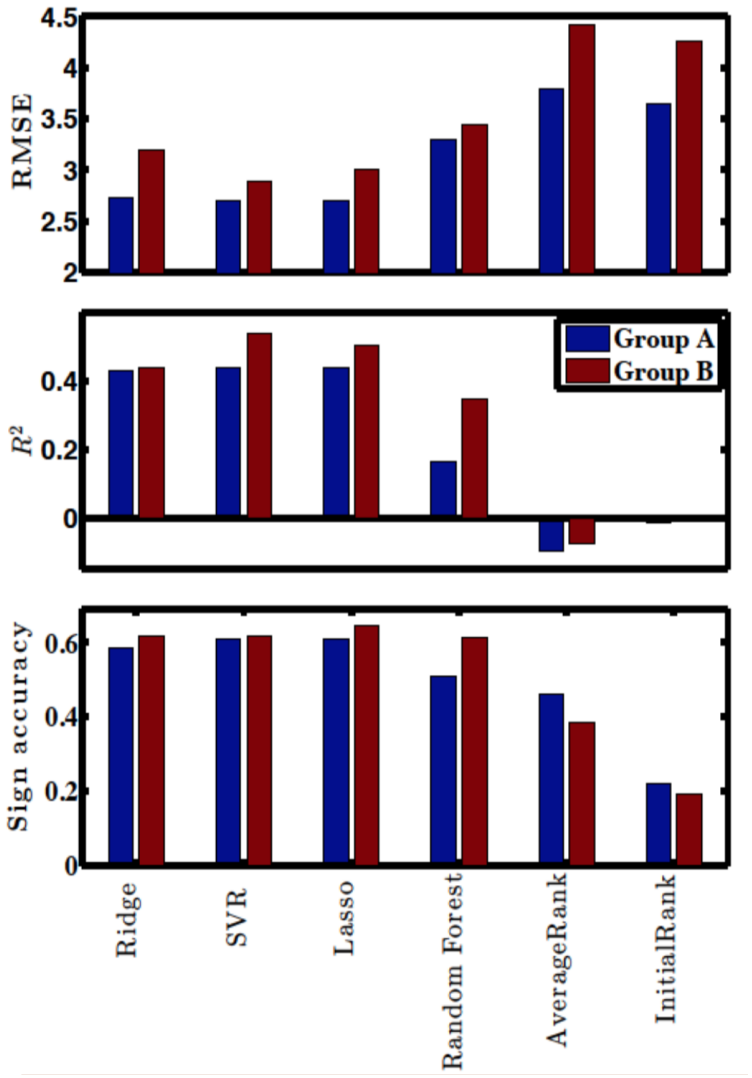

Figure 3: Different predictive models.

All models look similar in performance. What to do next? Can this be improved? Is this accuracy enough to actually build a strategy? By the end of this course, you will be in a better position to answer these questions!

2 Building a Language for Statistical Modeling

In the applications described above, we use some keywords. These are:

- Inputs: these *variables* are measured in some way: e.g., can be physical measurements, can be answers to a survey, etc.
- Outputs: we hypothesize that inputs influence the values these *variables* take.

Informally, we can define supervised learning as the problem of finding a relation between inputs and outputs. You can think of it as a function or a mapping. The method that does the search/finding is typically called a *learner*.

Below is a table of equivalent keywords that you may have come across:

Inputs	Predictors	Independent variables	Features
Outputs	Responses	Dependent variables	Labels

Lets focus on outputs for a bit. These can be quantitative, i.e., taking numerical values. Or they can be qualitative, with no order relationship between the values the outputs take.

When the learner learns a mapping for the quantitative setting, we will call the mapping a regression function and the process *regression*. On the other hand, when the learner learns a mapping for the qualitative setting, we will call the mapping a classifier, and the process *classification*.

2.0.1 Examples

For instance, in the setting where the learner learns a cat classifier, we can specify 1-dimensional representation for the output that takes a value 0 when there is no cat and 1 otherwise. Of course, you could have chosen value 0 to indicate the presence of a cat instead. This is what we mean by no order relationship. These values are sometimes called *targets*.

Another example is that of data drive decisions. Think of a stylized example where an autonomous car's controller is designed using statistical modeling. Then all the features that are measured (location on the road, location of other people and cars, traffic signals) can be mapped to descriptive actions or decisions such as *accelerate, brake, turn left, steer right by 10 degrees* etc., using supervised learning. Outputs such as these are also called *discrete* or *categorical*.

2.1 Describing Statistical Modeling

We will use statistical modeling as a way to describe data mining and machine learning solutions where statistics and probabilistic models play a key role. I will enumerate the different types of (application independent) problems that comprise this area:

- Pattern mining: processing databases to find *rules* (e.g., bread $\rightarrow$ oatmeal).
- Clustering: group objects that *belong* together.
- Classification: map inputs to outputs, where the values that outputs take have little/no relation with each other.
- Regression: same as above, but with values of outputs having some relationship. Typically these values are numerical.

- Ranking: order new data using historical data. This is related to classification and regression.
- Density estimation: finding the probability distribution that describes how data realized.
- ...

Both regression and classification can be viewed as a function approximation problem: given inputs and outputs, fit a *nice* function.

A language to describe these various problems is as follows. We will use X to represent an input variable or a vector of input variables (in the latter case, the jth variable is X_j). Quantitative outputs are denoted Y and qualitative ones G.

Lets observe N realizations of input X. The ith observed input is denoted using x_i. Say this is p-dimensional. Then if you stack them together as rows, you get a matrix $\mathbf{X}$ which is $N \times p$ dimensional.

Why did we introduce this notation? We did it to describe the learning task more formally: given X, find a function that predicts $\widehat{\mathrm{Y}}$ that is *close to* Y. For the qualitative setting, replace Y with G.

3 Two Learners

We will look at linear regression and k-nearest neighbor, especially because they are very different from each other in various ways and will give you an idea about the solution space (you could come up with your own learner if you understand these two!)

We will use the word *model* to describe the function or mapping output by a learner.

3.1 Linear Model

A linear model is given by: $\widehat{\mathrm{Y}} = X^T \widehat{\beta}$ (note: vectors are always column vectors unless specified explicitly). $\widehat{\beta}$ is the parameter of the model. That is, $\widehat{\beta}_1$ and $\widehat{\beta}_2$ specify two different models. If $\widehat{\mathrm{Y}}$ is K dimensional, then $\widehat{\beta}$ is $K \times p$ dimensional.

If we want to find a function approximator that maps Xs to Ys and we are allowed to only search in the space of linear models, we can do the following: we can pick an objective that measures how well we are doing.

For example, let the objective be $\sum_{i=1}^{N}(y_i - x_i^T\beta)^2$. This is called the least squared objective[1]. If we minimize this, we get the best approximator $\widehat{\beta}$, where best is defined precisely in terms of the objective function.

[1] sometimes also called the residual sum of squares

If we have enough observations such that $\mathbf{X}^T\,\mathbf{X}$ is not singular, then there is a single $\widehat{\beta}$ that minimizes the least squares objective, and it is given by $\widehat{\beta} = (\mathbf{X}^T\,\mathbf{X})^{-1}\,\mathbf{X}^T\,\mathbf{y}$.

Think of it this way: we reduced the $N \times p$ matrix and the N observed outputs (vectorized as $\mathbf{y}$) to a p-dimensional vector. This vector allows us to make predictions. In some sense, we have done *compression*, We do not need the original data (observations of inputs and outputs) anymore.

3.1.1 Example

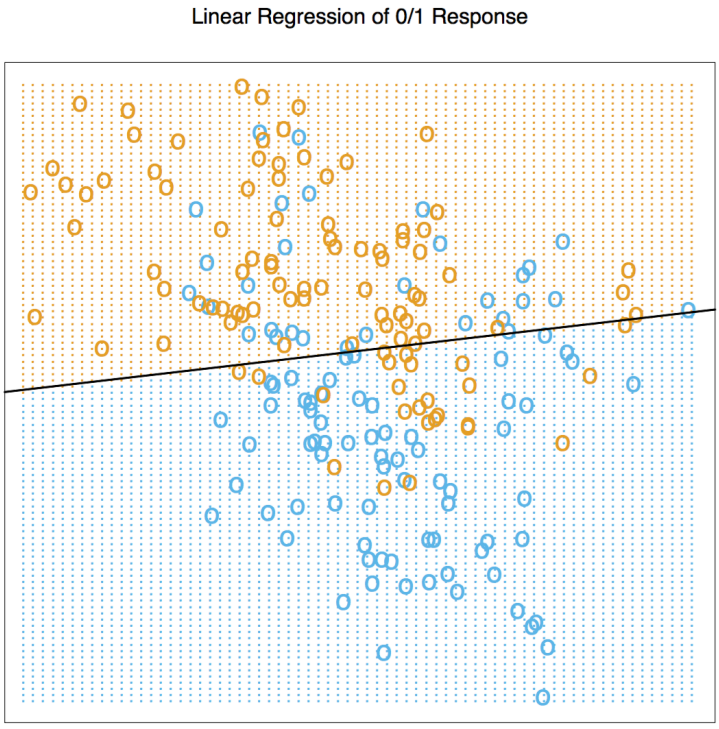

Figure 4: Linear model with 2-dimensional data.

Figure 4 shows a scatterplot of the observations of inputs which are 2-dimensional. The colors of each point represent the value that output variable G takes. We will use a linear model to fit this data (say blue is 1 and orange is 0) and convert $\widehat{\mathrm{Y}}$ to $\widehat{\mathrm{G}}$ via a non-linear transformation:

$$\widehat{\mathrm{G}} = \mathbf{1}[\widehat{\mathrm{Y}} > 0.5].$$

The black line corresponds to the decision boundary. Here $x^T\,\widehat{\beta} = 0.5$. The decision boundary is linear.

As you can see, the boundary does not separate blue points from red points well. We can eyeball and see this for 2-dimensional data. What if we have 100 dimensions? We will

necessarily have to rely on projections or other summary diagnostics to assess model fit. We will discuss this much more detail in a subsequent lecture.

Lets take a tangent and hypothesize how the 2-dimensional data was generated. What if it is just a mixture of two 2-dimensional Gaussians? One can reason that if these two Gaussians overlap too much, a linear decision boundary will not be adequate. Would any other boundary be better?

What if it was a mixture of mixtures of Gaussians, whose means were sampled from Gaussians? Sounds pretty complex, isn't it?

Note that a mixture of Gaussians can be described *generatively*. Think of a Bernoulli random variable. If it takes value 1, you sample a point from one Gaussian, and if it takes value 0, you sample from a different Gaussian. This is a generative description of a mixture of two Gaussians. We will see later that a linear boundary is the best in the least squares sense when data is realized this way.

If it is a much more complex model (mixture of mixture of ...) then a linear boundary may be very bad.

3.2 Nearest-neighbor Methods

These are *lazy* methods. Given a input, you can predict its output $\widehat{\mathrm{Y}}$ as:

$$\widehat{\mathrm{Y}} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i.$$

Here $N_k(x)$ is the *neighborhood* of value x. It is a set of k points closest to x in terms of some *metric*.

What is a metric?

3.2.1 Example

Figure 5 shows the same data as Figure 4 and uses $k = 15$. There is no fitting here (except for choosing k).

A point is orange if at least 8 of the 15 neighbors of that point are orange, and vice-versa.

As can be inferred from the two figures, we get a non-linear decision boundary with this method. Also, arguably a better fit[2]

What if we choose $k = 1$? We get a Voronoi tesselation. There are many decision boundaries! See Figure 6.

What would change if we were doing regression instead?

[2]We have only been looking at *in-sample* fit!

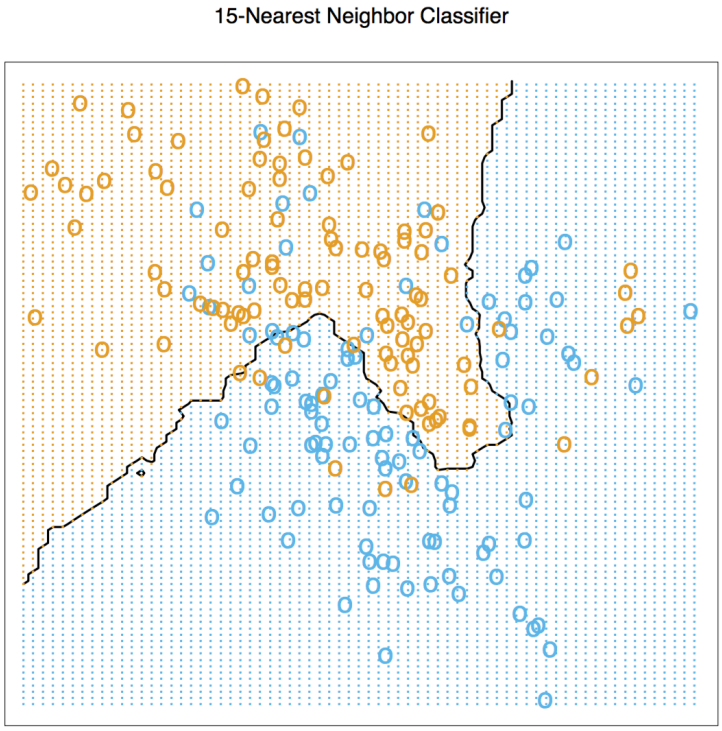

Figure 5: 15-Nearest neighbor method with 2-dimensional data.

3.3 Comparing Nearest-neighbor with Linear Modeling

If you thought nearest-neighbor methods are superior, think again. In Figure 6, all the points are classified correctly. Hence, we can reason that as we increase k, our errors will increase, starting from value 0 when $k = 1$. These errors are somewhat meaningless though. What we should care about is performance on an independent *test set*.

How do we pick k? Does its choice matter? If you used the least squares objective[3] to pick k, what would you get?

If the data in Figures 5 and 6 were from two overlapping Gaussians, k-nearest-neighbor methods would probably capture 'noisy patterns'.

3.3.1 Comparing Parameters

We know that the linear model had p parameters. How many parameters does the k-nearest-neighbor method have? 1. But there is more to this. It turns out that the *effective* number of parameters is N/k, which grows with N and shrinks with k.

The intuition is as follows: lets say the neighborhoods were non-overlapping. Then there would be N/k neighborhoods and we would fit one parameter (the mean) in each neighborhood.

[3]That is, for each k, we find the residual sum of squares.

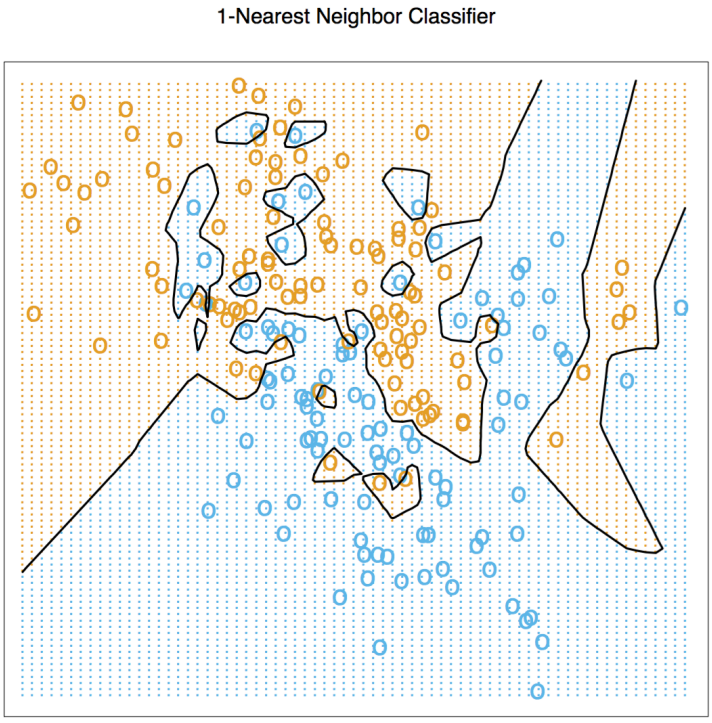

Figure 6: 1-Nearest neighbor method with 2-dimensional data. More-irregular than the 15-Nearest neighbor method.

3.3.2 Comparing Decision Boundaries

The decision boundary from a linear model is *stable* and *smooth*[4] This is a manifestation of *low variance* and *high bias*.

For the k-nearest-neighbor, since any classification decision only depends on a few nearby points, it is unstable and wiggly. But it can adapt to complex data. This is a manifestation of *high variance* and *low bias*.

As you will see throughout the course, there is no single best method for making the data talk. Any method can shine if its properties are aligned with the unknown true characteristics of the data. This is true of deep neural nets as well.

3.3.3 Comparing Testing Performance

In the examples, we have 200 *training* observations. The performance of nearest neighbor methods is shown in Figure 7 as a function of N/k over an independent test data (10000 in number, so fairly accurate representation of how these methods do in reality). The

[4]We are using both these terms loosely here.

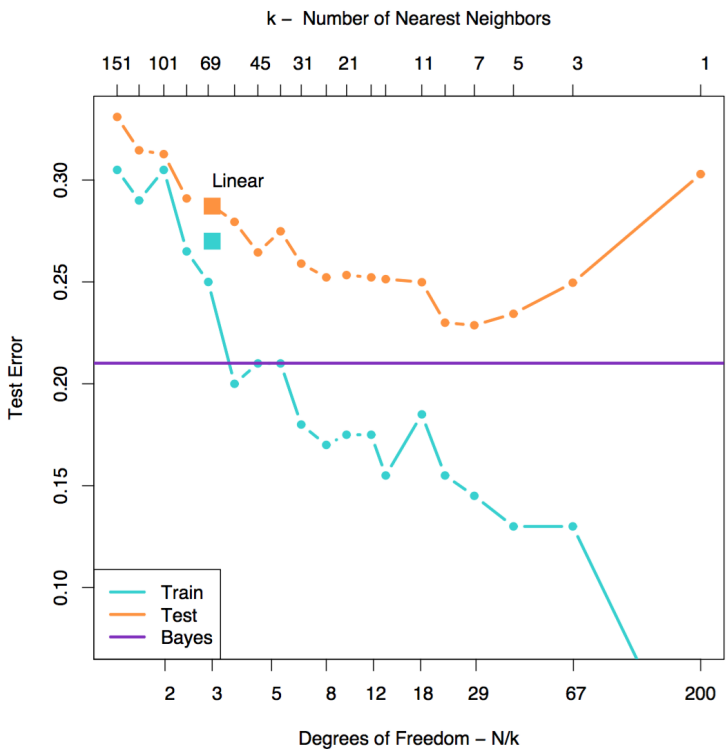

Figure 7: Error for different models.

performance is just the number of points on the wrong side of the decision boundary. The figure also includes the performance of a linear model with $p = 3$.

Many methods discussed in the course can be thought about in similar terms, and will work intuitively the same way as either of the two methods we just saw.

4 Summary

We learned the following two things:

- Introduced vocabulary to describe statistical modeling.
- Looked at two intuitive prediction models: the linear model and the k-nearest-neighbor method.

In the next lecture, we will discuss the bias-variance tradeoff and jump into the nuances of regression modeling.

A Sample Exam Questions

- What are the different types of statistical modeling tasks? What are their differences/distinctions?
- Describe the dependence of k-nearest neighbor method's performance on k? How is the performance in-sample different from out-of-sample?