STOR 320 Modeling VI

Lecture 20

Yao Li
Department of Statistics and Operations Research
UNC Chapel Hill

Final Presentation Time

- See Schedule via Group Assignment
- Submit slides via Sakai before Presentation Day.
- 5-7 minutes presentation.

Introduction

- Non-Parametric Classification

- K-Nearest Neighbors (k-NN)
  - Machine Learning Technique
  - Intuitive
  - Non-Parametric
  - Used for Predicting Classes of an Output Variable

K-NN Algorithm

- Step 1: Choose a *k*

- Step 2: Select the *k* Most Similar Observations in a Database Which are the “Closest” According to the Input Variables

- Step 3: Find the Most Common Classification Among These

- Step 4: Classify the New Observation Based on What is Category is Known to Occur Most

Tutorial 14

- Instructions
  - Data `> library(titanic)`
  - Required Packages
    - `library(tidyverse)`
    - `library(ISLR)`
    - `library(class)`
  - Download Tutorial 14 and Open .Rmd File

Part 1: Feature Engineering and Visualization

- Titanic Survival Data

```
> library(titanic)
```

  - Response Variable

$$Y = \begin{cases} 1 & if \;\; Survived \\ 0 & if \;\; Did\ Not\ Survive \end{cases}$$

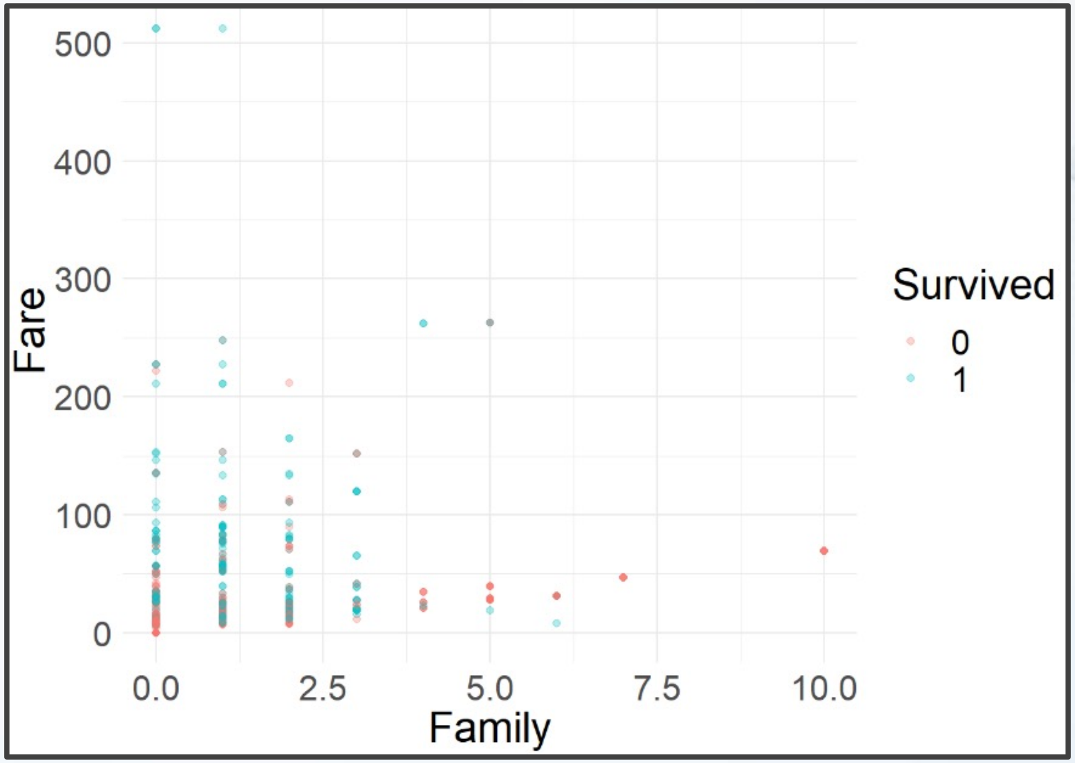
  - Explanatory Variables
    - Siblings/Spouses Aboard
    - Parents/Children Aboard
    - Passenger Fare

- Goal: Use k-NN to Predict a Passenger to Survive or to Die a Miserable, Cold Death

Part 1: Feature Engineering and Visualization

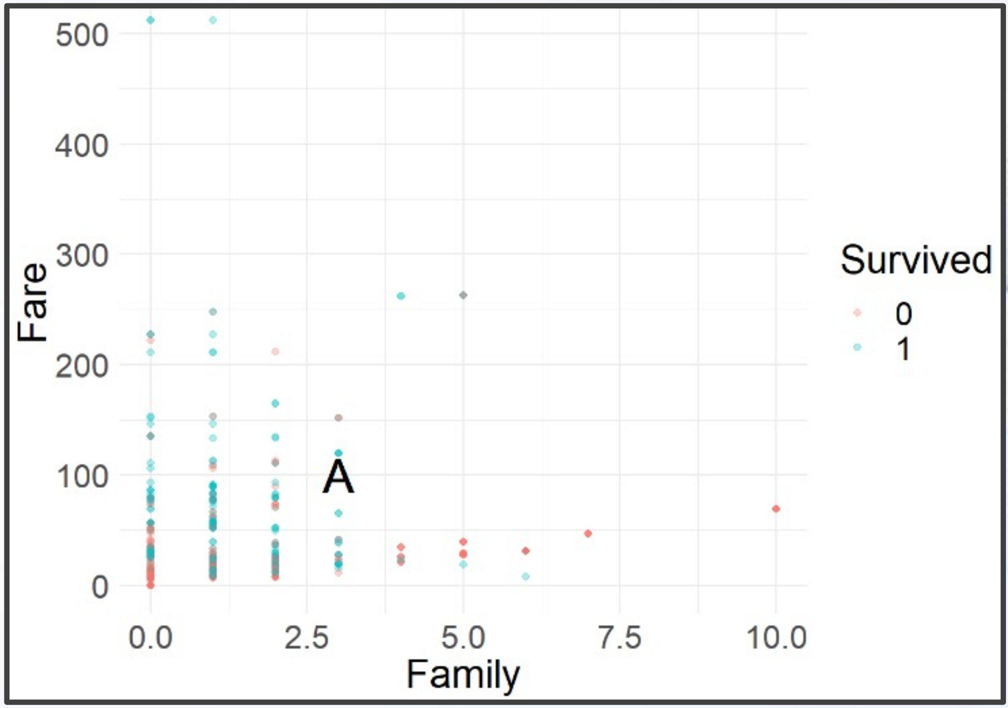
- Run Chunk 1
  - Creating a New Variable
  - What Does This Variable Represent?

- Run Chunk 2



Part 2: Obtaining Predictions Using K-NN

- New Individual: Alice
  - Had 3 Family Members on Ship
  - Spent $100 on Ticket
  - Survived or Died?
- Run Chunk 1



Part 2: Obtaining Predictions Using K-NN

- Finding Similar Passenger
  - Out-of-Sample Passenger
    - $X_{11} = Family\ Onboard$
    - $X_{12} = Fare$
  - Passenger in Training Data
    - $X_{21} = Family\ Onboard$
    - $X_{22} = Fare$
  - Geometric Distance Formula

$$d = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2}$$

  - Two Scenarios
    - Distance is Small
    - Distance is Large

Part 2: Obtaining Predictions Using K-NN

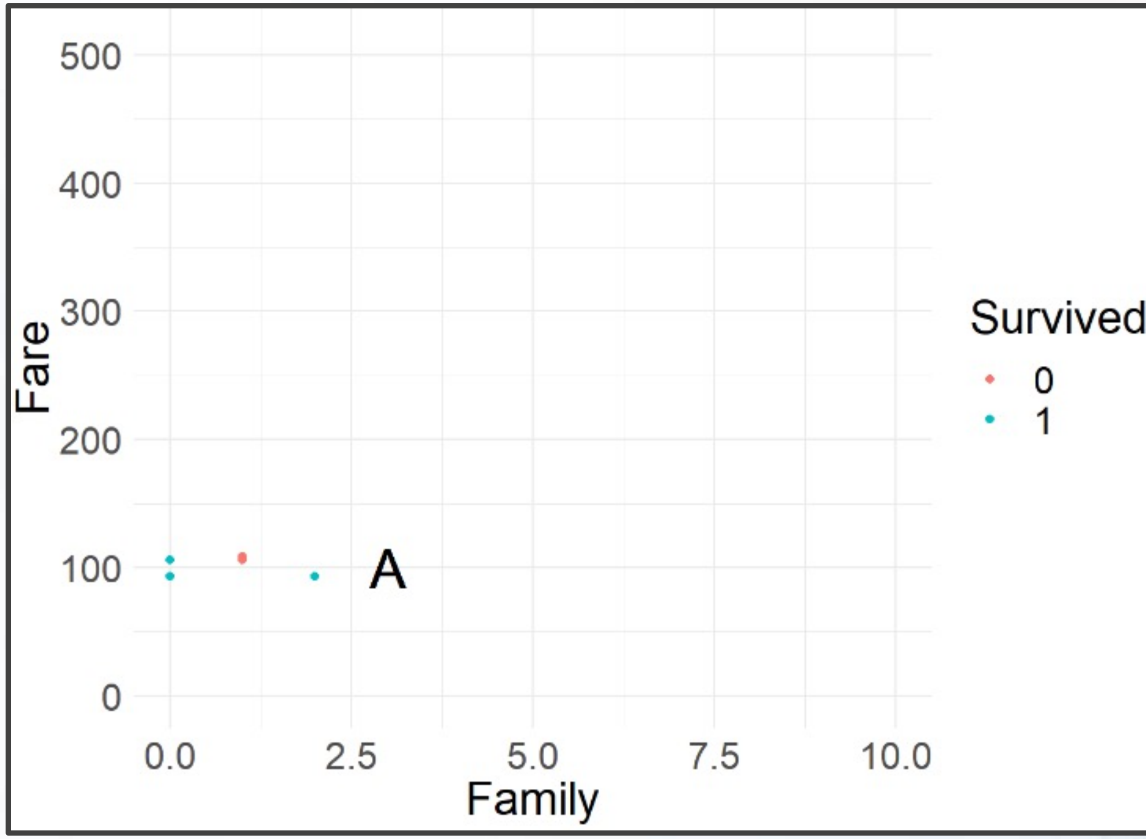
- Run Chunk 2
  - Suppose k=5
  - Five Most Similar Passengers

Survived	Fare	Family	d
1	93.500	2	6.576473
0	106.425	1	6.729088
1	106.425	0	7.090883
1	93.500	0	7.158911
1	108.900	1	9.121952
0	108.900	1	9.121952

  - Why are There Six?
  - Did Alice Survive or Die?

Part 2: Obtaining Predictions Using K-NN

- Run Chunk 3
  - Output Figure



- What Did You Expect to See?
- Are You Surprised?

Part 3: Transform and Revisit K-NN

- Consider Standardization
  - Multiple Methods
  - Classic Formula

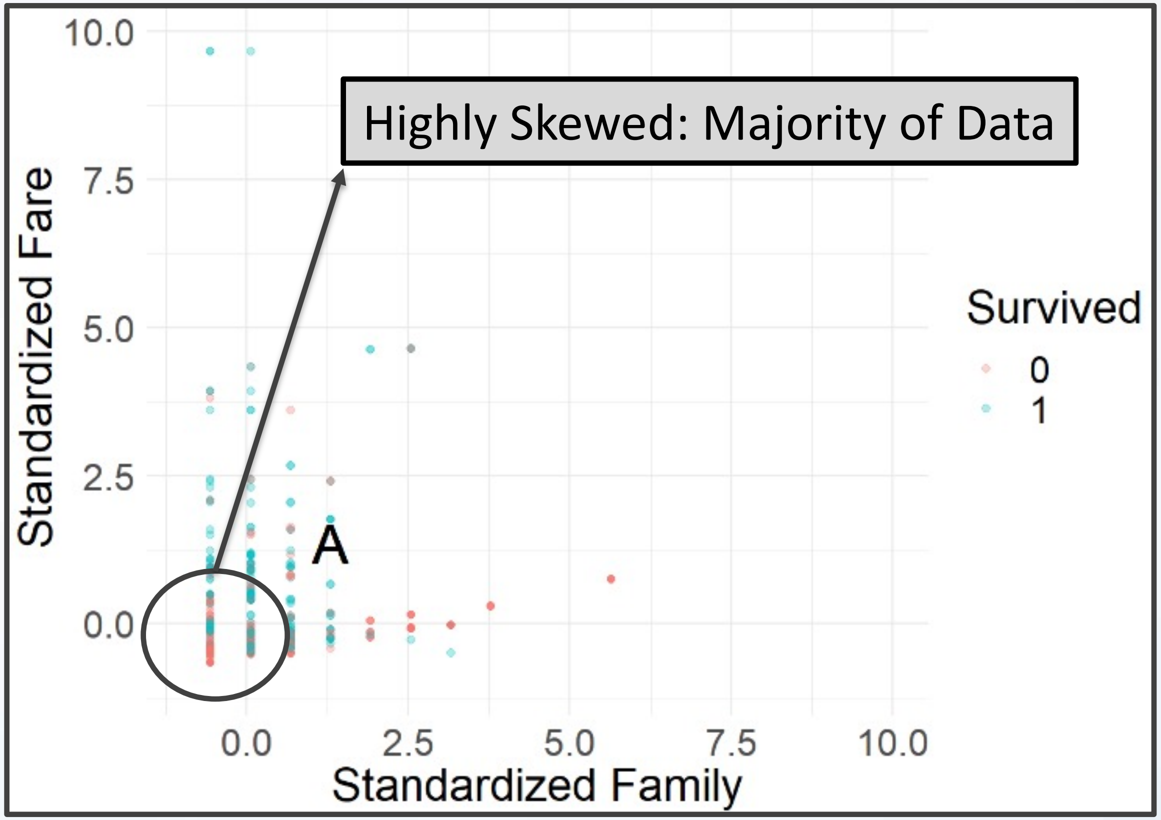
$$Z = \frac{X - \mu}{\sigma_x}$$

  - Use $\bar{x}$ and s_x
  - What We are Doing
    - Centering Data
    - Scaling Data

```
> scale(x,center=T,scale=T)
```

Part 3: Transform and Revisit K-NN

- Run Chunk 1
  - Units: Standard Deviations
  - Alice: Above Average Family Size and Fare



Part 3: Transform and Revisit K-NN

- Run Chunk 2
  - Recall: Alice
    - Family Size of 3
    - $100 Ticket
  - Before & After Standardization

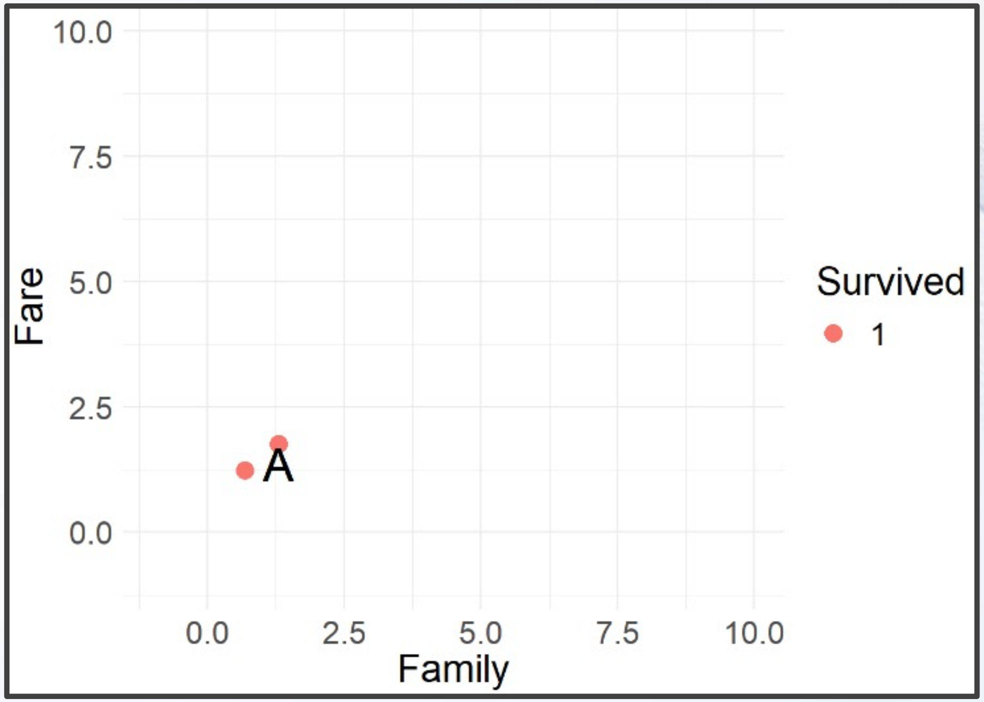
Survived	Fare	Family	d
1	93.500	2	6.576473
0	106.425	1	6.729088
1	106.425	0	7.090883
1	93.500	0	7.158911
1	108.900	1	9.121952
0	108.900	1	9.121952

Survived	Fare	Family	d
1	120.0	3	0.4024677
1	120.0	3	0.4024677
1	120.0	3	0.4024677
1	120.0	3	0.4024677
1	93.5	2	0.6334387

Part 3: Transform and Revisit K-NN

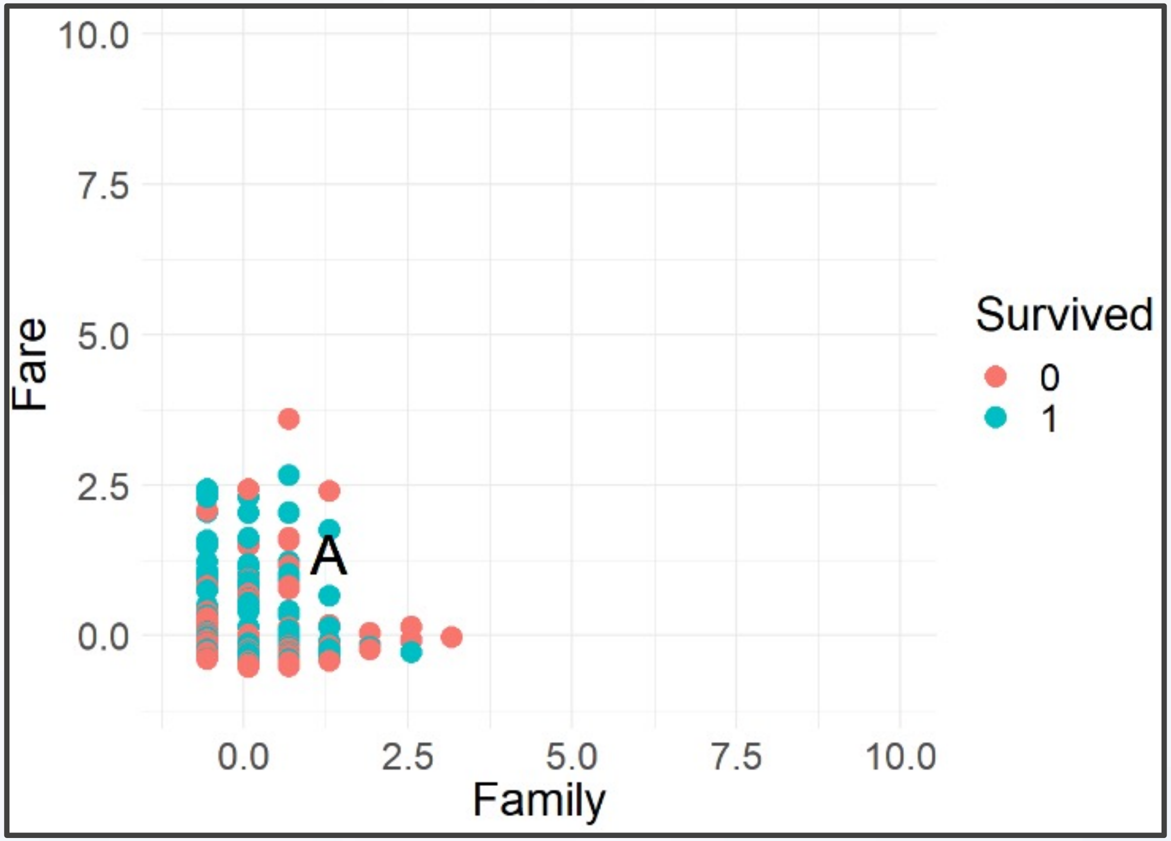
- Chunk 2 Continued
  - Both Before and After Standardization We Would Predict Alice to Survive

  - Updated Figure



Part 4: Tuning K for K-NN

- Run Chunk 1
  - Suppose k is Large (k=500)



Part 4: Tuning K for K-NN

- Chunk 1 Continued
  - Votes From Neighbors

```
KNN.PREDICT=table(ST5$Survived)
print(KNN.PREDICT)
```

```
##
##   0   1
## 258 251
```

- Based on k-NN When k=500
  - 258 Neighbors Died
  - 251 Neighbors Survived

- Predict Alice is Food for Fish

Part 4: Tuning K for K-NN

- Leave-on-Out Cross Validation
  - Helpful Package for k-NN `> library(class)`

  - Install the R Package

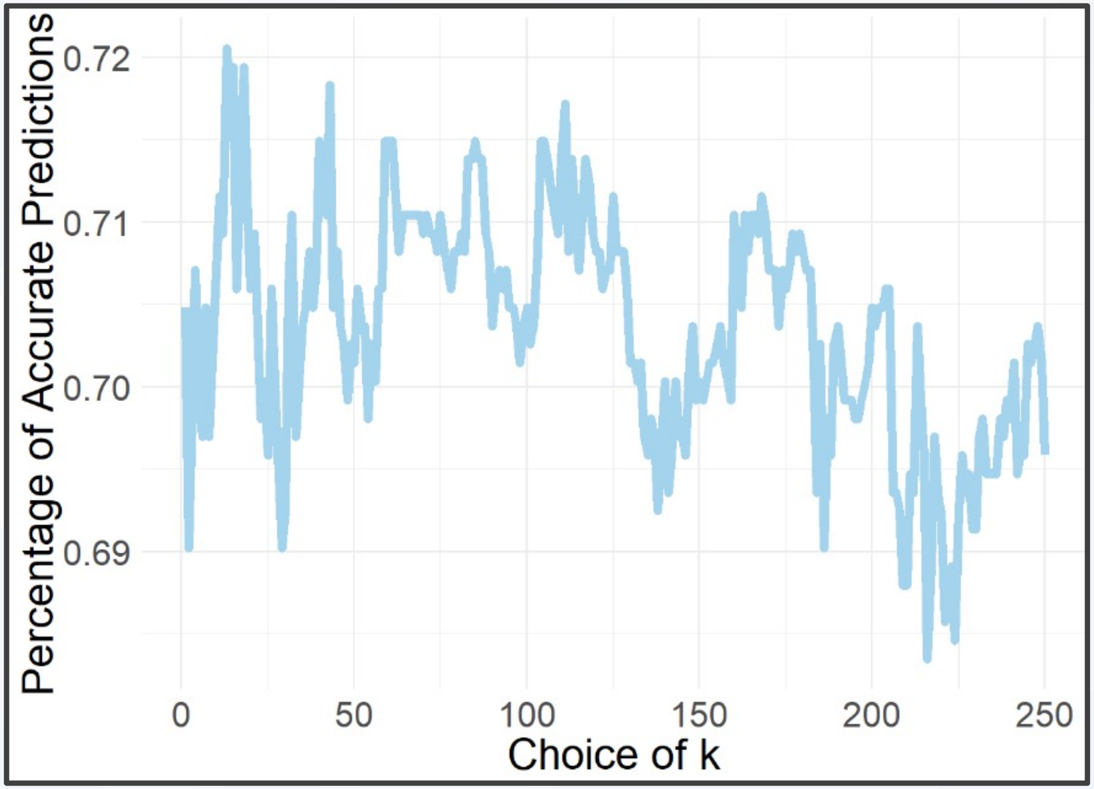
  - Helpful Functions
    - Peforming k-NN `> knn(train, test, cl, k = 1)`

    - LOOCV `> knn.cv(train, cl, k = 1)`

  - For Other Important Arguments, See Documentation

Part 4: Tuning K for K-NN

- Run Chunk 2
  - Consider k=1,2,3,…,250
  - Use CV, to Generate Out-of-Sample Predictions for Each k
  - Calculate Overall Accuracy Percentage



Part 4: Tuning K for K-NN

- Run Chunk 3
  - Identify Best Choice for k (k=18)
  - Use k to Generate Predictions on Future Data With Unknown Survival

```
> titanic_test
```

  - Figure Illustrating Predictions on Test Set for Competition