STA 235H - Prediction: K-Nearest Neighbors

Fall 2021

McCombs School of Business, UT Austin

Announcements

No classes on Thanksgiving week

Last week

- Talked about shrinkage methods:
  - Why do they usually work better than OLS
  - Ridge vs. Lasso;
  - Estimating hyper-parameters (λ)

Today

- We will be discussing K-Nearest neighbors:
  - How we can use a non-parametric method for prediction?
  - Regression vs classification tasks

Won't you be my neighbor?

Prediction tasks

- We have seen the main issue with bias vs variance trade-off
- Beyond regression, what methods can we use for prediction?

K-nearest neighbor

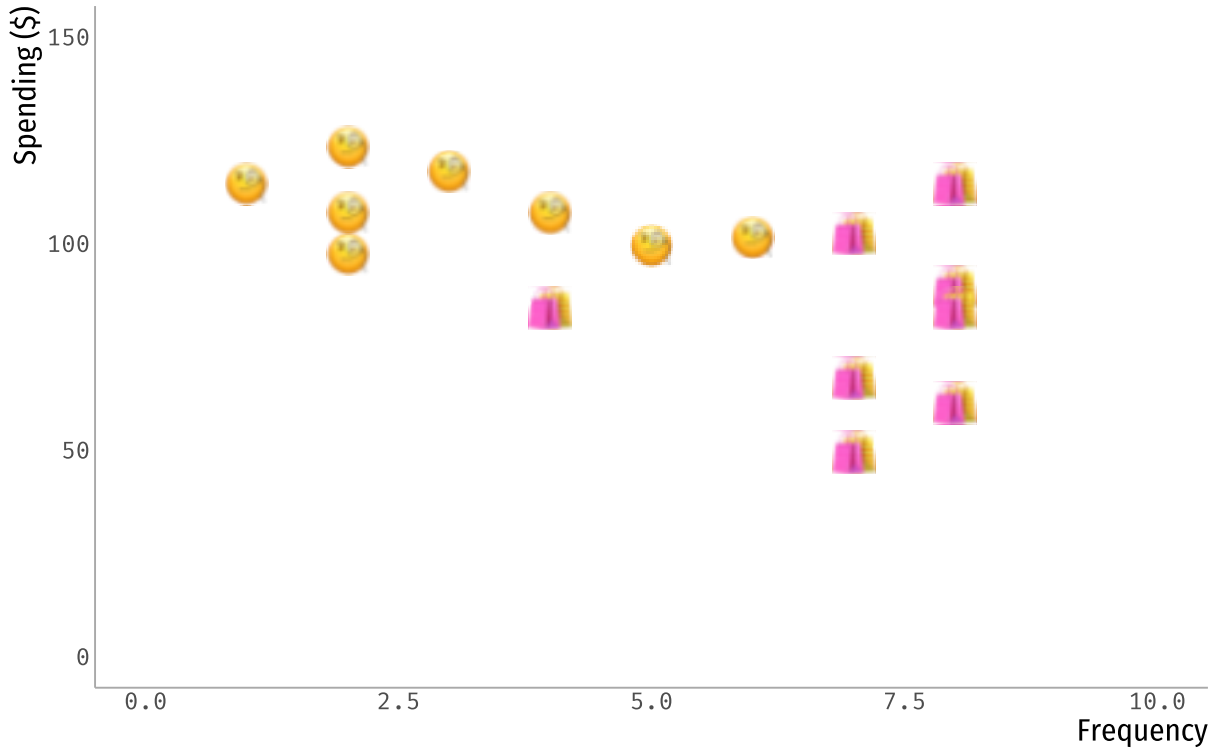
Continuous outcome

Binary outcome

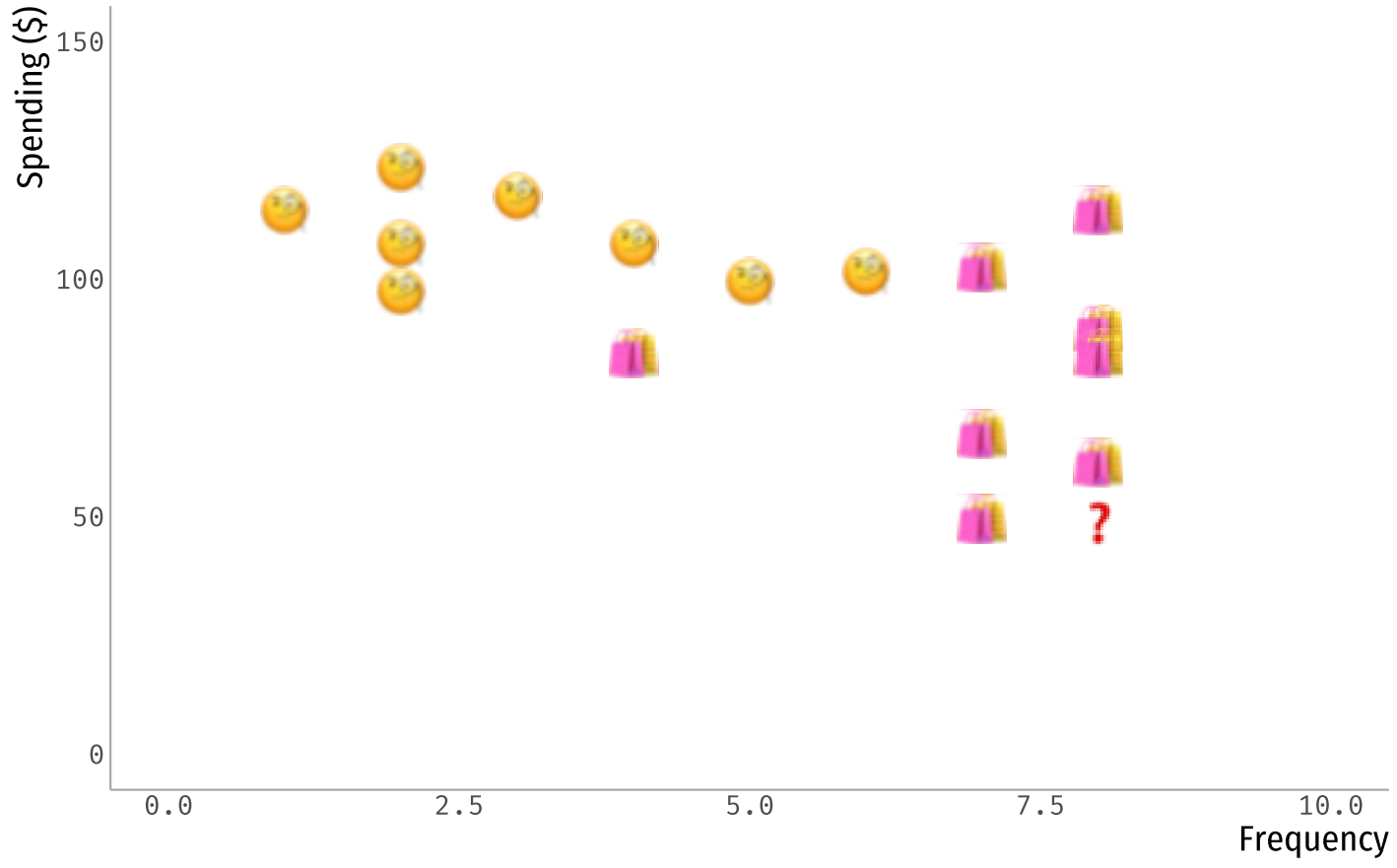
Let's make it classy

KNN as a classification problem

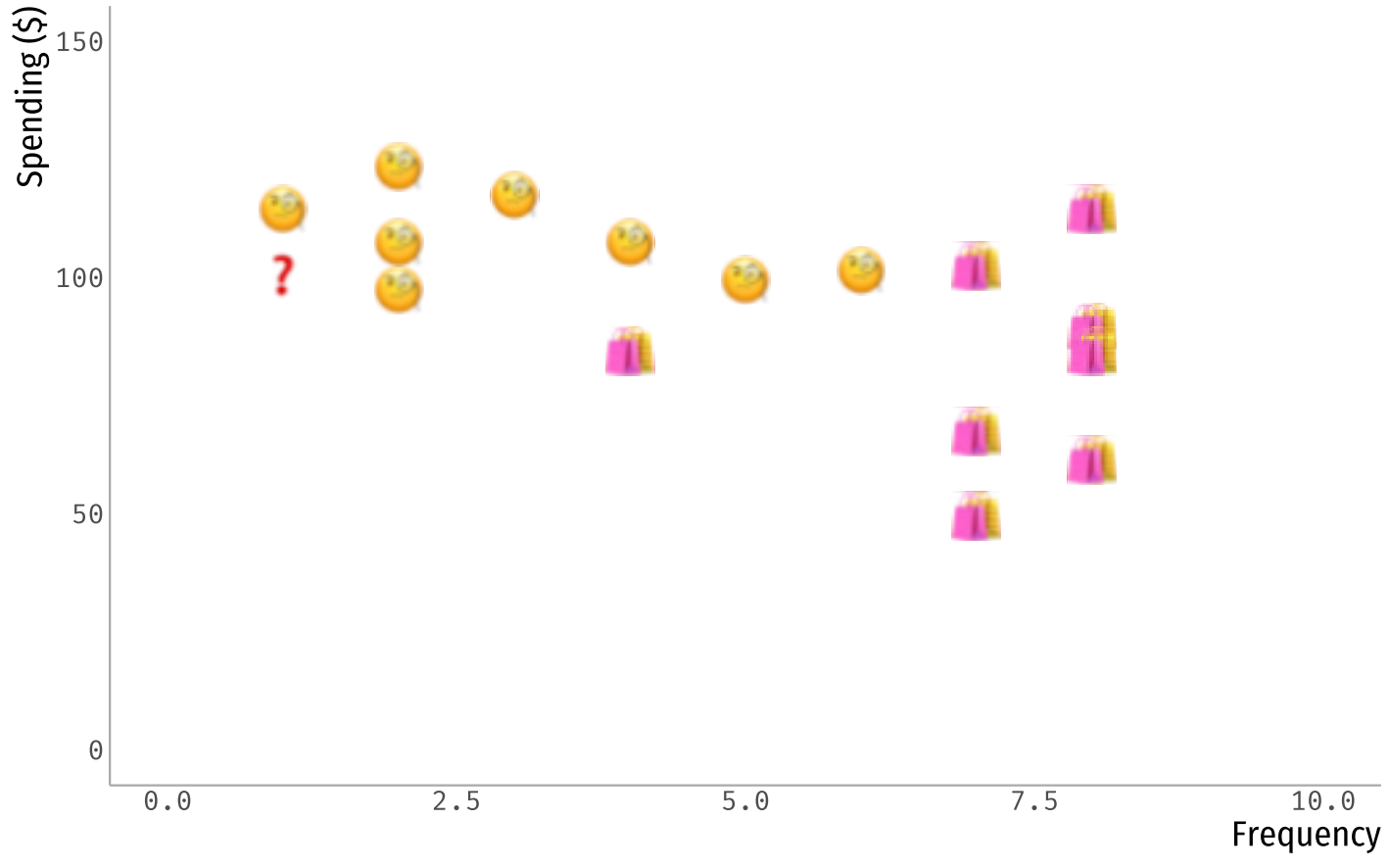
- Again: Window shoppers vs high rollers



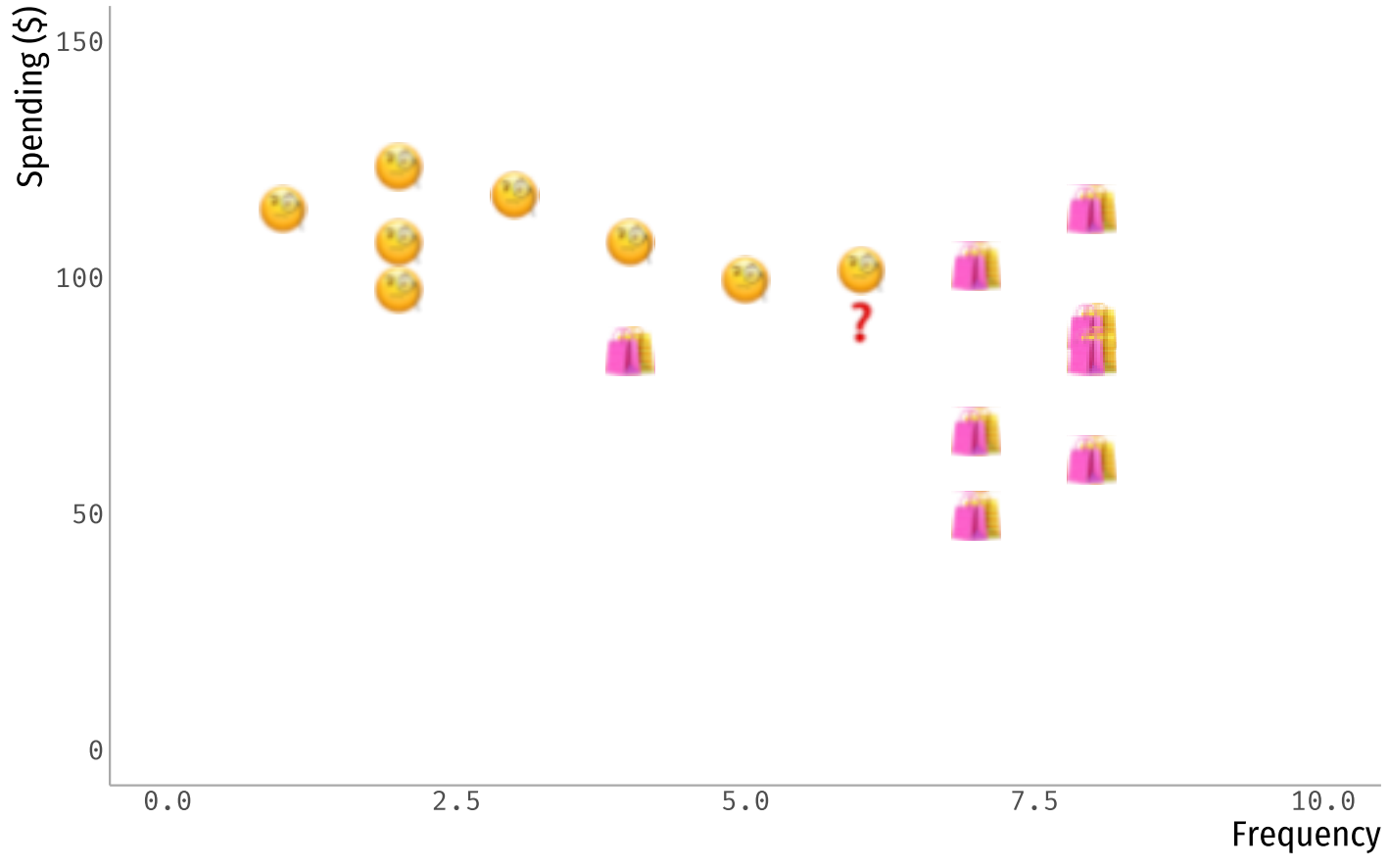
How would you classify this unit?



How would you classify this unit?



But what about this one?



K-nearest neighbor classifier

One of the simplest classifications methods

Algorithm:

1. Choose a distance measure (e.g. eucledian).
2. Choose a number of neighbors, K (*Note: Choose an odd number!*).
3. Calculate the distance between data and other points.
4. Calculate the rate for each class according to K: $Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in N_0} \mathrm{I}(y_i = j)$.
5. Assign the majority class.

KNN with K=1

Classifier: High-roller

KNN with K=3

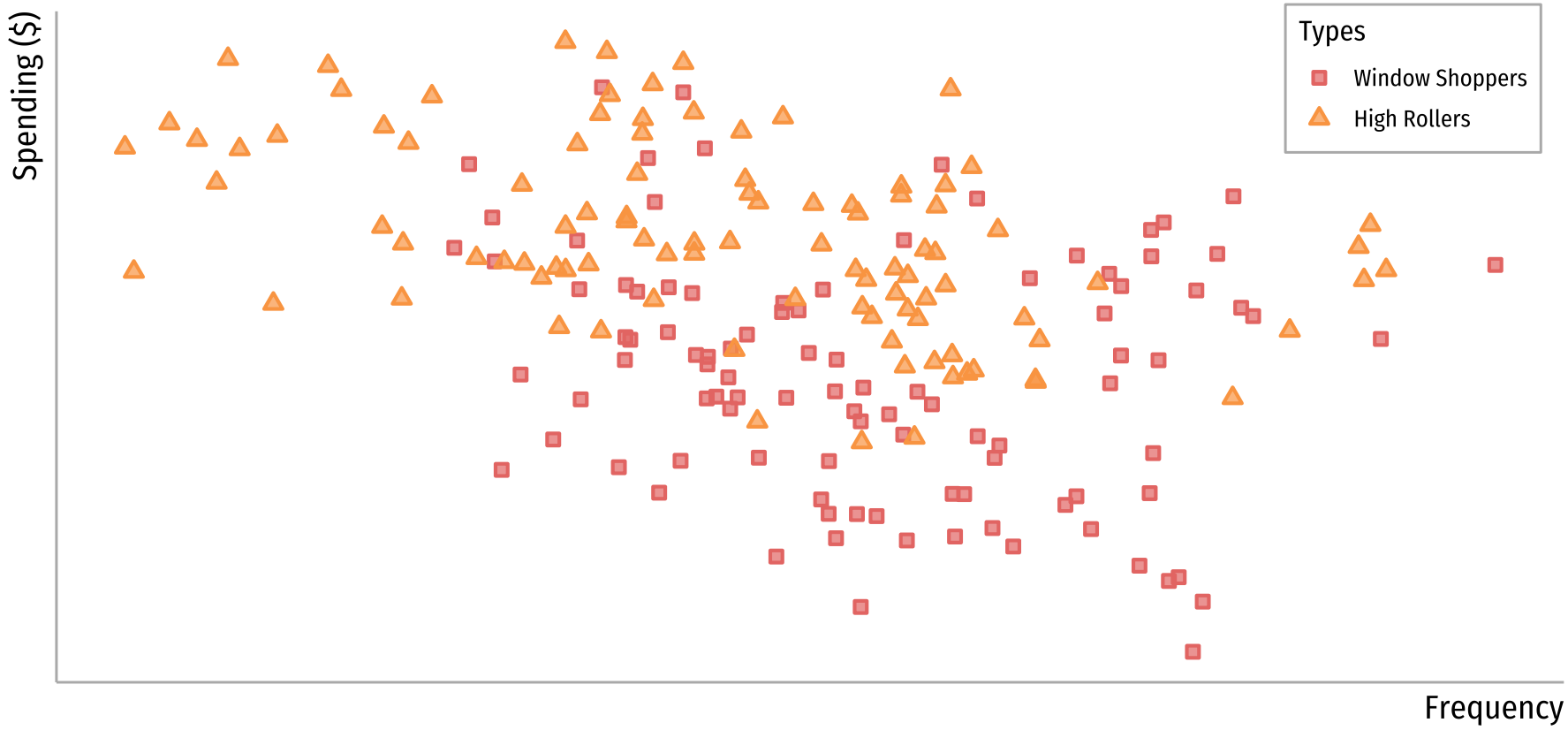
Classifier: High-roller

KNN with K=9

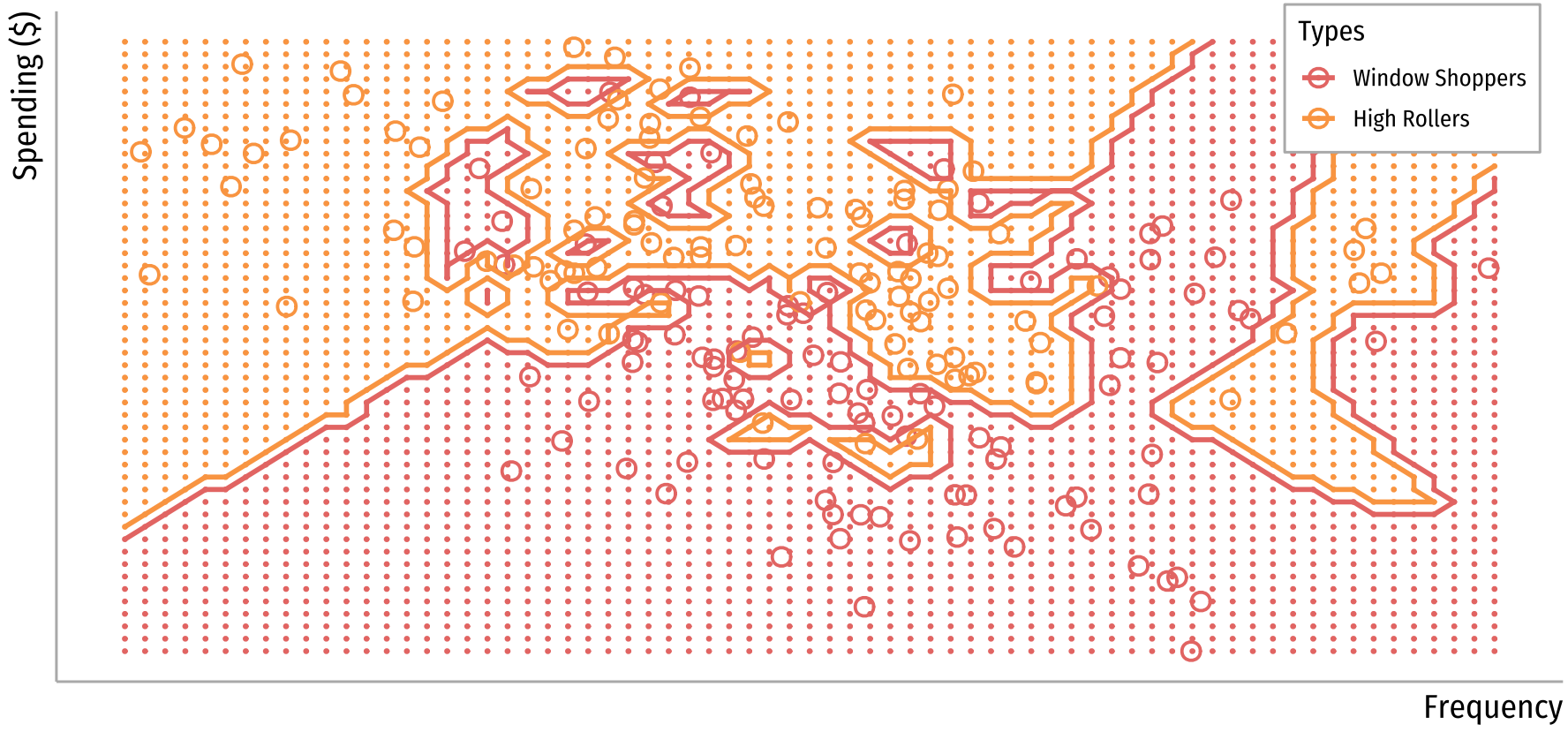
Classifier: Window-shopper

A lower number of neighbors K is associated to a (...) variance model. Why?

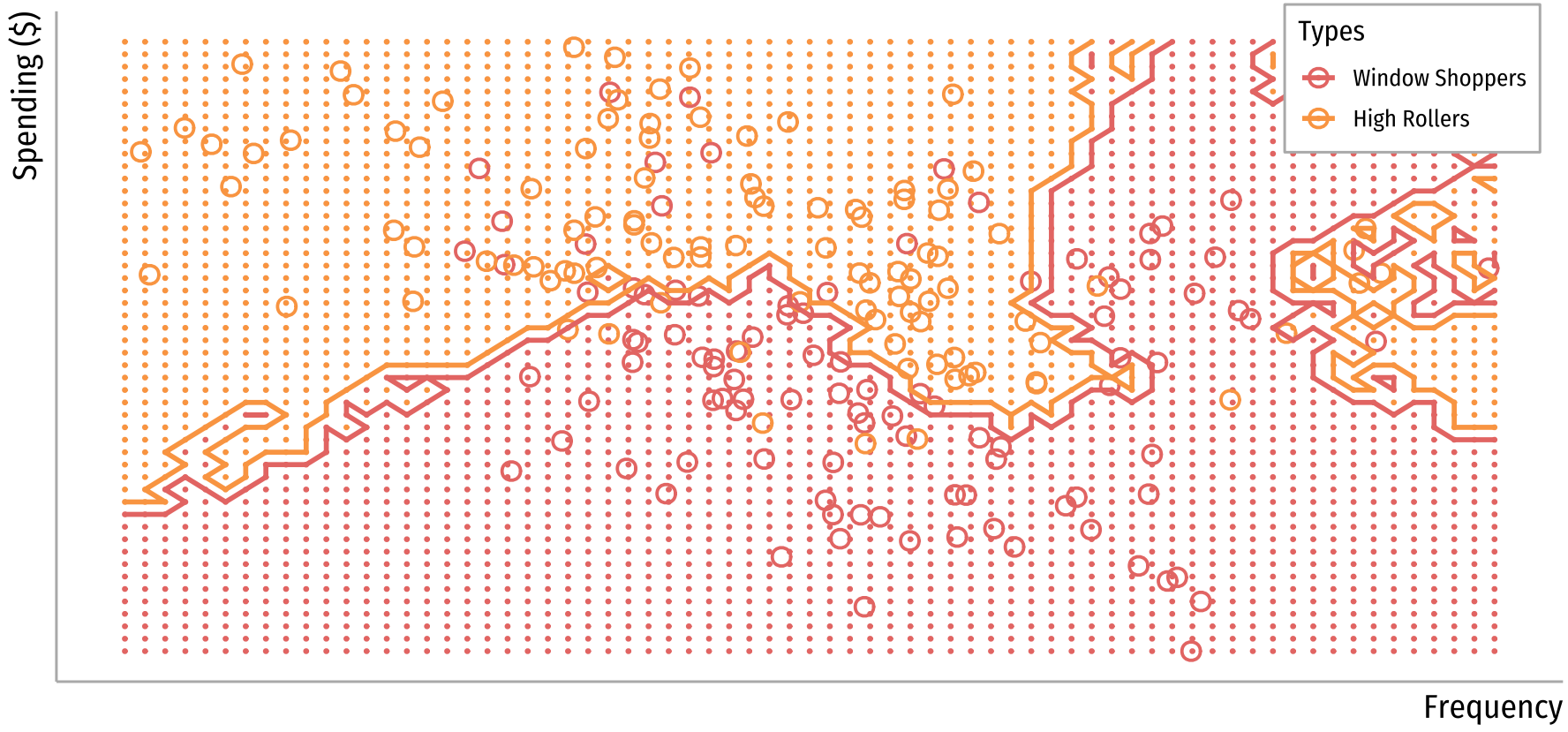
So what is happening when we change K?



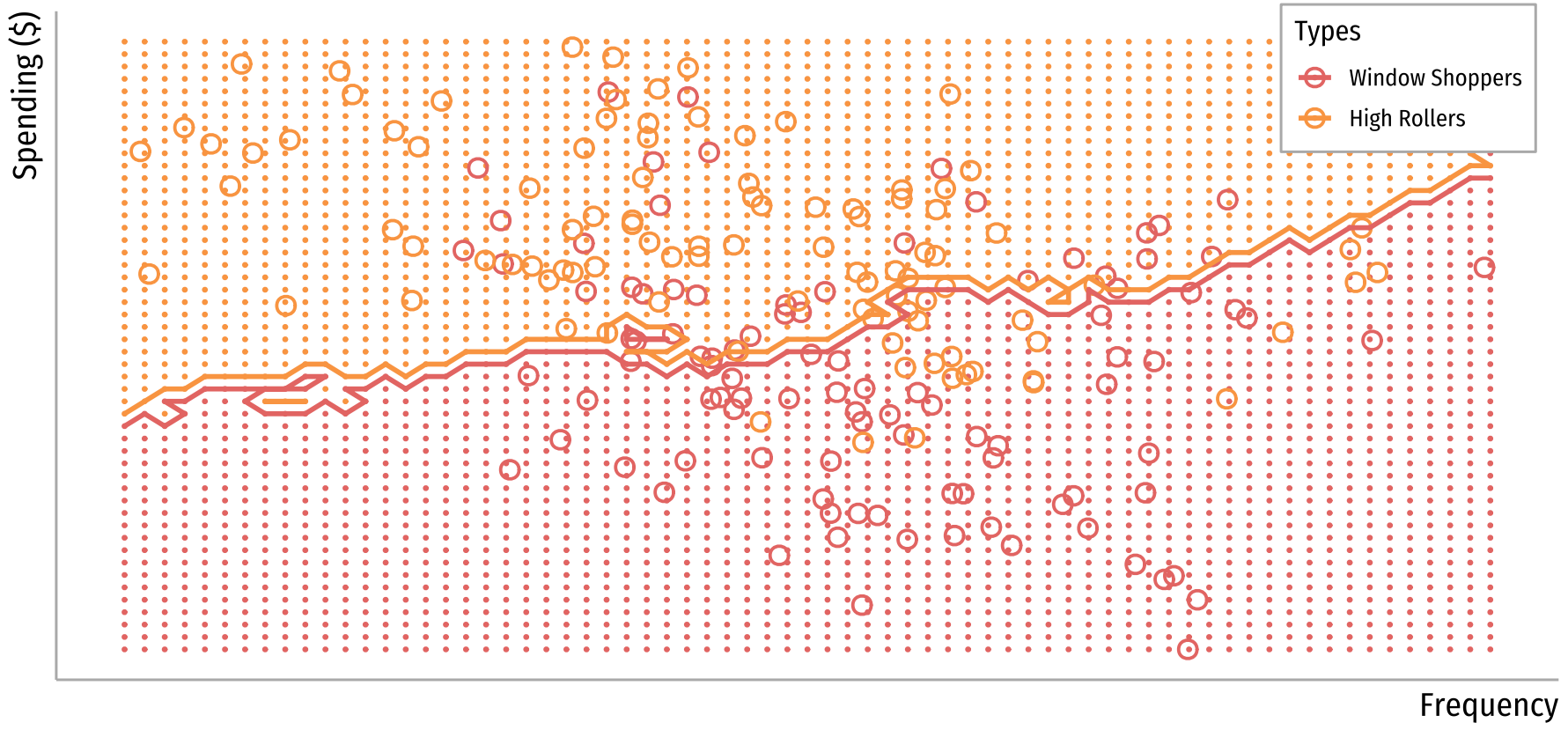
Classification boundaries for K=1



Classification boundaries for K=10



Classification boundaries for K=100



KNN Classifier in R?

```
d <- read.csv("https://raw.githubusercontent.com/maibennett/sta235/main/exampleSite/content/Classes/

head(d)
```

```
##   freq female spend type
## 1   10      1    59   WS
## 2    7      1    71   WS
## 3    6      1    79   WS
## 4    3      0    97   HR
## 5    9      1    52   WS
## 6   10      1    56   WS
```

KNN Classifier in R?

```
library(caret)

set.seed(100)

n <- nrow(d)

train.row <- sample(1:n, 0.8*n)

test.data <- d %>% slice(-train.row)
train.data <- d %>% slice(train.row)

knn <- train(
  type ~., data = train.data,
  method = "knn",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 15
  )
```

- Again, we'll be using the `caret` package.

KNN Classifier in R?

```
library(caret)

set.seed(100)

n <- nrow(d)

train.row <- sample(1:n, 0.8*n)

test.data <- d %>% slice(-train.row)
train.data <- d %>% slice(train.row)

knn <- train(
  type ~., data = train.data,
  method = "knn",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 15
  )
```

- Again, we'll be using the `caret` package.
- Create a training and testing dataset.

KNN Classifier in R?

```
library(caret)

set.seed(100)

n <- nrow(d)

train.row <- sample(1:n, 0.8*n)

test.data <- d %>% slice(-train.row)
train.data <- d %>% slice(train.row)

knn <- train(
  type ~., data = train.data,
  method = "knn",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 15
  )
```

- Again, we'll be using the `caret` package.
- Create a **training** and **testing** dataset.
- Use the method `knn` on a factor variable (i.e. classification)

KNN Classifier in R?

```
library(caret)

set.seed(100)

n <- nrow(d)

train.row <- sample(1:n, 0.8*n)

test.data <- d %>% slice(-train.row)
train.data <- d %>% slice(train.row)

knn <- train(
  type ~., data = train.data,
  method = "knn",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 15
  )
```

- Again, we'll be using the `caret` package.
- Create a **training** and **testing** dataset.
- Use the method `knn` on a factor variable (i.e. classification)
- We also **pre-process** the data. Why?

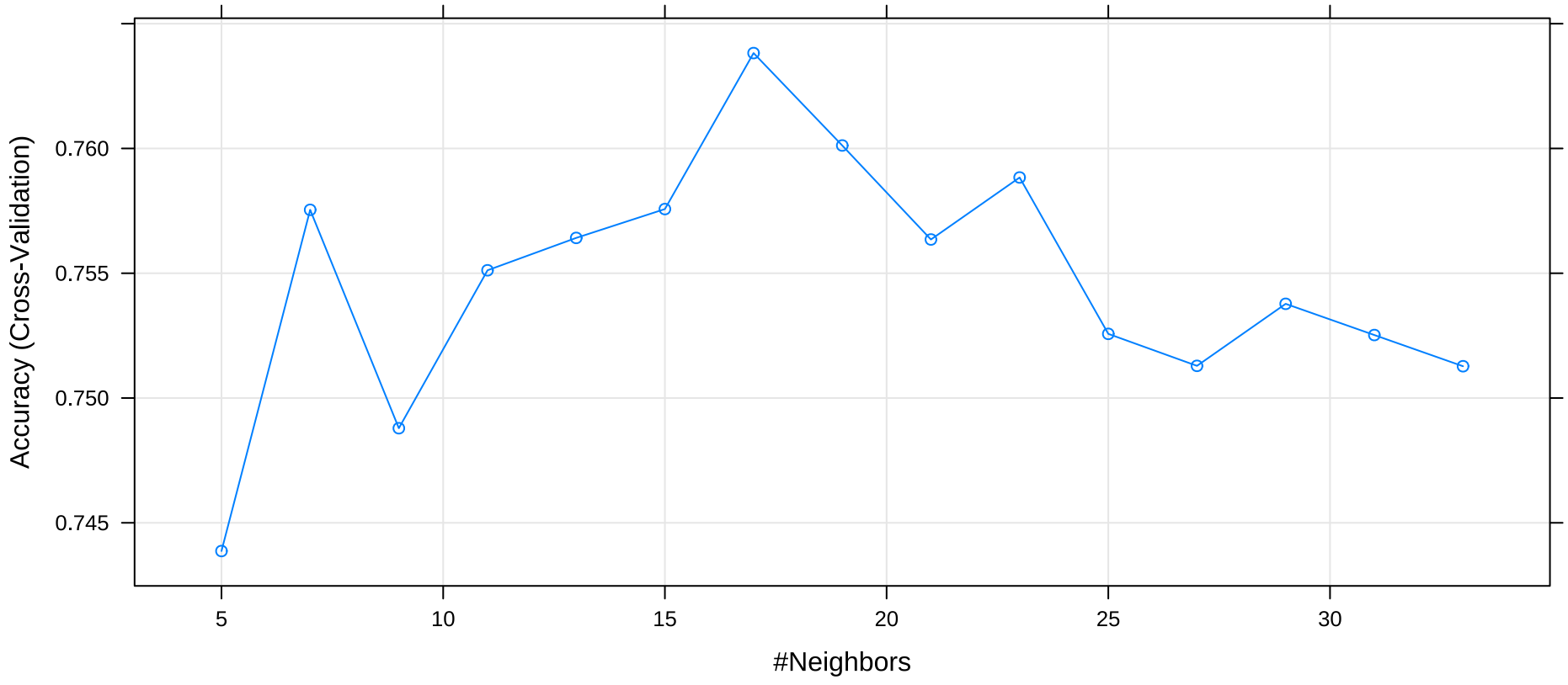
KNN Classifier in R?

```
library(caret)

set.seed(100)

n <- nrow(d)

train.row <- sample(1:n, 0.8*n)

test.data <- d %>% slice(-train.row)
train.data <- d %>% slice(train.row)

knn <- train(
  type ~., data = train.data,
  method = "knn",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 15
  )
```

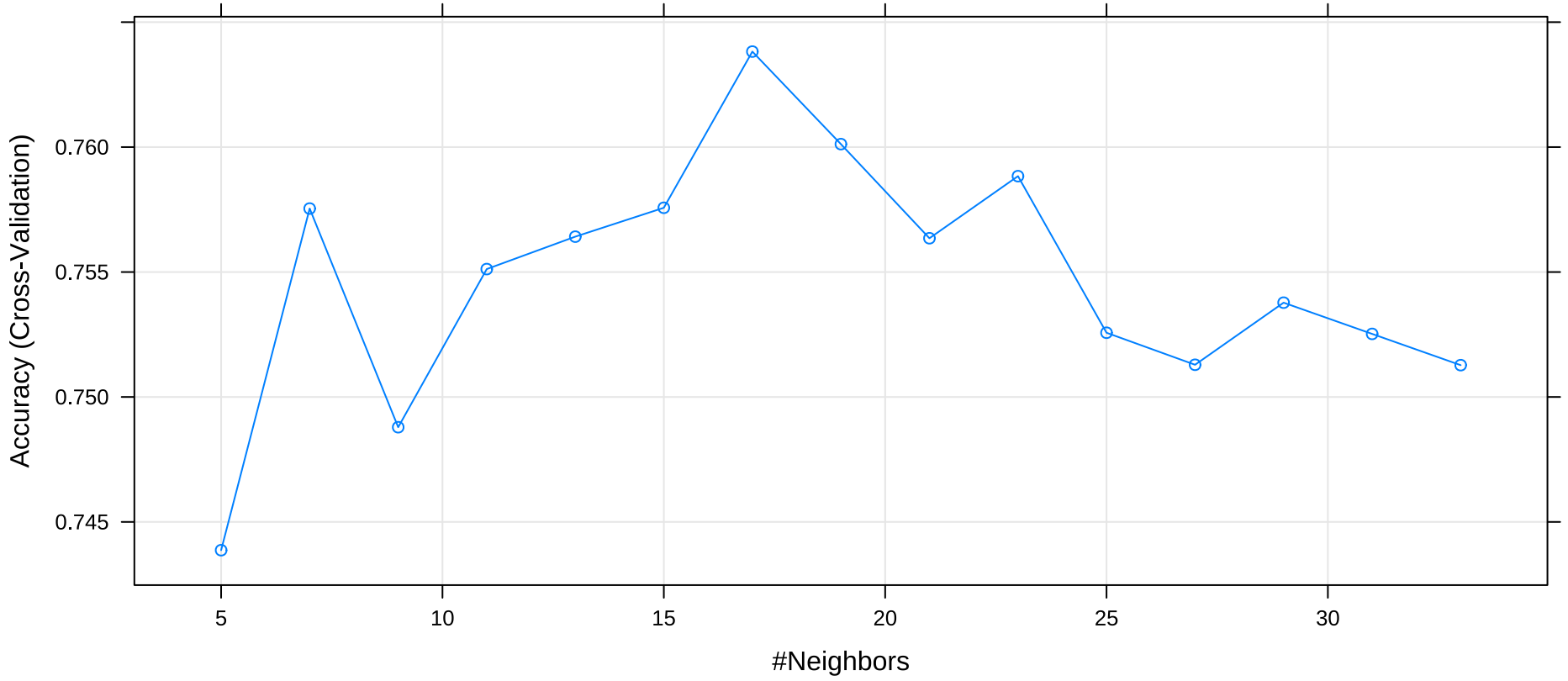
- Again, we'll be using the `caret` package.
- Create a **training** and **testing** dataset.
- Use the method `knn` on a factor variable (i.e. classification)
- We also **pre-process** the data. Why?
- `tuneLength` is the level of granularity for searching K.

How many neighbors?

We can see the optimal K using `bestTune` parameter.



Which K would you choose?



Let's go to R

How accurate is this?

- For classification problems, we care about *false positive* and *false negative*.
    - Sometimes you will care more about being wrong on one side than the other.

```
pred.type <- knn %>% predict(test.data)

test.data <- test.data %>% mutate(prediction =


test.data %>% select(type, prediction) %>% tab
```

```
##     prediction
## type HR WS
##   HR 72 17
##   WS 28 83
```

```
test.data %>% select(type, prediction) %>% tab
  round(., 3)
```

```
##     prediction
## type    HR    WS
##   HR 0.809 0.191
##   WS 0.252 0.748
```

In a table like this, where would you like to see most of the observations?

```
test.data %>% select(type, prediction) %>% table
```

```
##     prediction
## type HR WS
##   HR 72 17
##   WS 28 83
```

```
test.data %>% select(type, prediction) %>% table %>% proportions(., margin = 1) %>%
  round(., 3)
```

```
##     prediction
## type    HR    WS
##   HR 0.809 0.191
##   WS 0.252 0.748
```

What about continuous outcomes?

K-Nearest Neighbors Regression

- We can also use KNN for **continuous outcomes**
- **Similar** to the KNN classifier, but now we will take the *average of the K-neighbors* for prediction:

$$\hat{f}(x_0) = \frac{1}{K} \sum_{i \in N_0} y_i$$

KNN Regression in R?

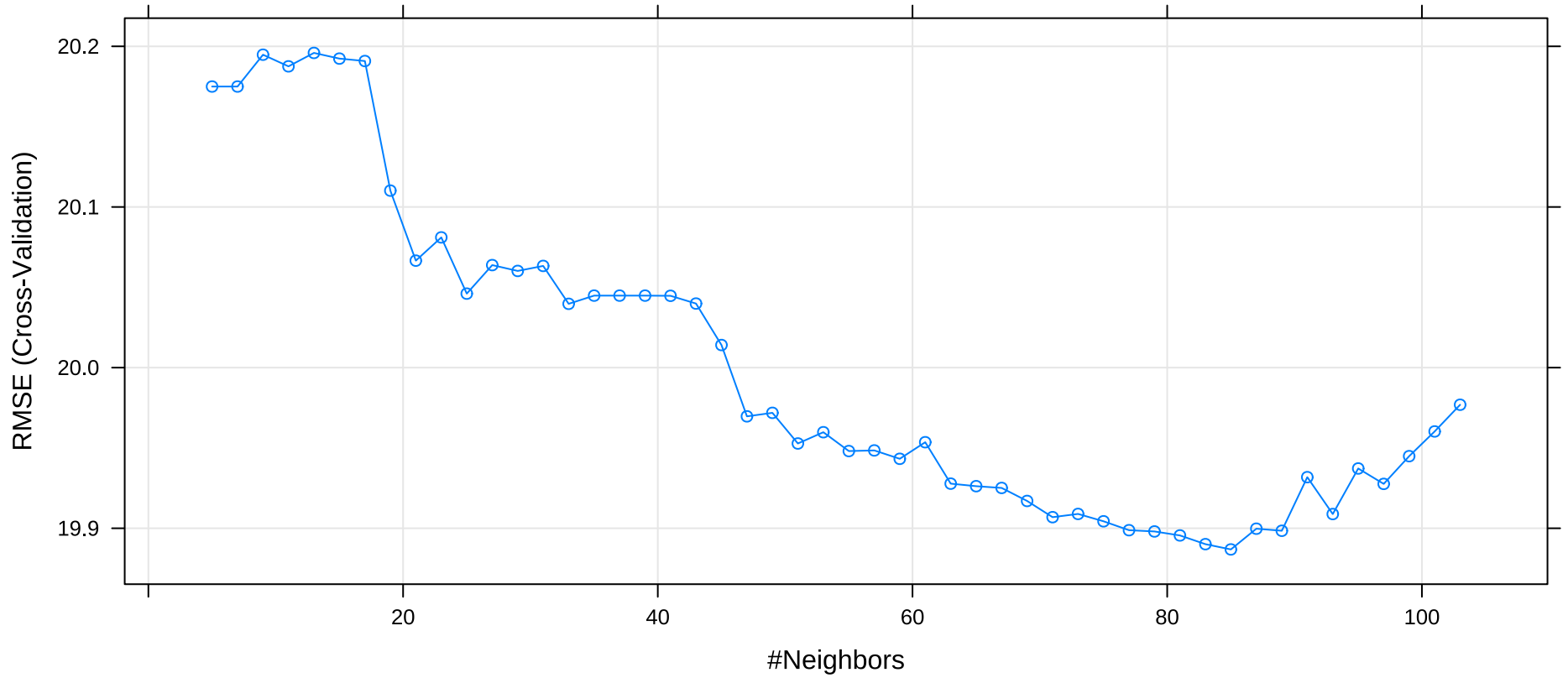
```
library(caret)

d <- read.csv("https://raw.githubusercontent.co

set.seed(100)

n <- nrow(d)

train.row <- sample(1:n, 0.8*n)

test.data <- d %>% slice(-train.row)
train.data <- d %>% slice(train.row)

knnr <- train(
  spend ~. - type, data = train.data,
  method = "knn",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 50
  )
```

Same as before!

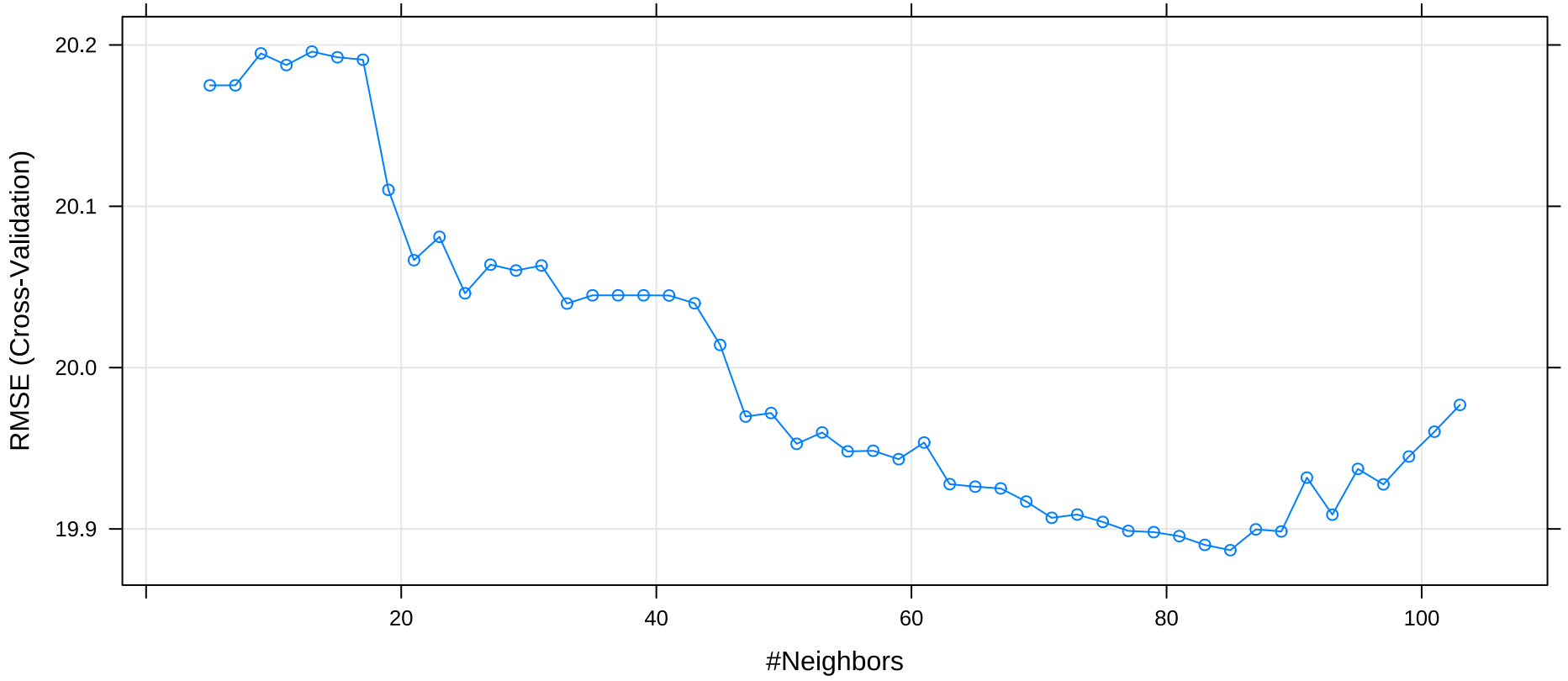
... but with a continuous variable

Choose optimal K

We get the optimal K the same way, using `knnr$bestTune`



Which K would you choose?



Takeaway points

- KNN is a simple, nonparametric way to do prediction for both categorical and continuous outcomes.
- Be sure to check your accuracy/error metric depending on your outcome.
- Pre-processing can play an important role!

Plot your data and results

Next class

- Dive into new prediction methods: **Decision Trees!**
  - How to choose order of the variables
  - How to choose splits
  - How deep should we go?

References

- James, G. et al. (2021). "Introduction to Statistical Learning with Applications in R". *Springer. Chapter 2, Chapter 3.*

- STDHA. (2018). "KNN: K-Nearest Neighbors Essentials"