K-Nearest Neighbors

Brooke Anderson

January 20, 2016

Load required libraries.

```
library(dplyr) ## Data wrangling
library(class) ## Includes `knn` function
library(ggplot2)
```

Read in the data.

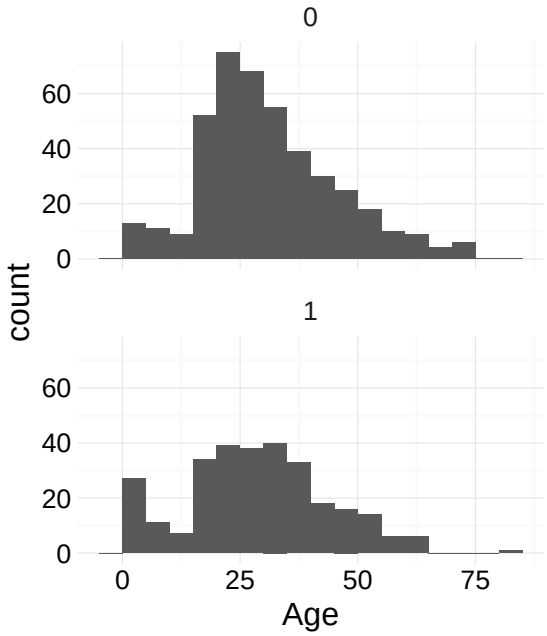
```
train <- read.csv("../data/train.csv") %>%
  mutate(Survived = factor(Survived),
         Pclass = factor(Pclass),
         Name = as.character(Name),
         Sex = factor(Sex))
test <- read.csv("../data/test.csv") %>%
  mutate(Pclass = factor(Pclass),
         Name = as.character(Name),
         Sex = factor(Sex))
```

Model with k = 1 and single continuous predictor

I started by fitting a model with a k of one, so just looking at the single nearest neighbor, and using only `Age` as a predictor. Here's the relationship between `Age` and `Survived` in the training data:

```
ggplot(train, aes(x = Age)) +
  geom_histogram(binwidth = 5) +
  facet_wrap(~ Survived, ncol = 1) +
  theme_minimal()
```

```
## Warning: Removed 177 rows containing non-finite values (stat_bin).
```



Next I fit the model. The model seemed to have some problems when the predictive variable had missing values, so I removed those from both testing and training data sets before fitting.

```
train_knn <- filter(train, !is.na(Age))
train_x <- select(train_knn, Age)
train_y <- train_knn$Survived
test_x <- filter(test, !is.na(Age)) %>%
  select(Age)
```

Next, I fit the model, with k = 1. I fit it first for the training data, to get an estimate of that accuracy. (That means I repeat `train_x` in the model statement.)

```
set.seed(1201)
train_pred <- knn(train_x, train_x, train_y, k = 1)
head(train_pred)
```

```
## [1] 0 0 0 1 1 0
## Levels: 0 1
```

The `knn` function gives the predictions directly, unlike the function for Naive Bayes, which works more like a typical modeling structure for R.

Now, to assess accuracy, I'll need to get these back into the original dataframe and pick which values I want to use for predictions when `Age` is missing. I'll use "0", since the majority of people in the training dataset did not survive.

This part is a bit tricky, because you have to merge back in with the missing values and align things correctly with the passenger IDs. Normally, I would have kept `PassengerId` in to do this, but `knn` was finicky about letting me do that without modeling it.

```
x_accuracy <- data.frame(Survived = train$Survived,
                         pred = factor("0", levels = c("0", "1")))
x_accuracy$pred[!is.na(train$Age)] <- train_pred
head(cbind(x_accuracy, train$Age))
```

```
##   Survived pred train$Age
## 1        0    0        22
## 2        1    0        38
## 3        1    0        26
## 4        1    1        35
## 5        0    1        35
## 6        0    0        NA
```

```
mean(x_accuracy$Survived == x_accuracy$pred)
```

```
## [1] 0.6767677
```

Accuracy for this model in the training set is 0.67677.

Next, I fit the same model to the testing data and check out the accuracy against the Kaggle Leaderboard.

```
set.seed(1201)
test_pred <- knn(train_x, test_x, train_y, k = 1)
test_accuracy <- data.frame(PassengerId = test$PassengerId,
                            Survived = factor("0", levels = c("0", "1")))
test_accuracy$Survived[!is.na(test$Age)] <-
  as.numeric(as.character(test_pred))
write.csv(test_accuracy, file = "../predictions/knn_age_k1.csv",
          row.names = FALSE)
```

My score on Kaggle was 0.57416. This was my worst model to date.