HarvardX: PH125.9x Data Science: Capstone Course Choose Your Own Project: Analysising my Perfomance with Strava Data

Byron Wicks

16 June 2019

Table of Contents

Introduction

The Choose You Own Project forms part of the HarvardX: PH125.9x Data Science: Capstone course. The aim of the project is to analyse and investigate data of your choosing and predict outcomes using machine learning.

The author has chosen to analyse his personal fitness data from Strava recorded over almost 3 years containing bike rides, hikes, runs, and walks and to investigate any correlations in fitness, performance, and types of activities employed.

This report will present an overview of the data, analysis, results and a conclusion.

Dataset

An introductory review of the dataset is performed in order to familiarise ourselves. With some summarisation of the data for easier manipulation. Data is downloaded as per instruction from the Strava settings page.

```
# Libraries
#####################################################

library(tidyverse)
library(gridExtra)
library(strava)
library(lubridate)
library(gtools)
library(sugrrants)
library(DescTools)
library(class)
library(caret)

load(file = "StravaData.Rdata")

# Summarise data
data_summary <- data %>%
  mutate(time = lubridate::date(data$time),
         year = strftime(data$time, format = "%Y"),
         date_without_month = strftime(data$time, format = "%j"),
         month = strftime(data$time, format = "%m"),
         day_of_month = strftime(data$time, format = "%d"),
         year_month = strftime(data$time, format = "%Y-%m"),
         type = type) %>%
  group_by(time, year, date_without_month, month, day_of_month, year_month)
%>%
  summarise(total_dist = sum(dist_to_prev), total_time =
sum(time_diff_to_prev), type = type[1]) %>%
  mutate(speed = (total_dist) / (total_time /60^2)) %>%
  mutate(pace = (total_time / 60) / (total_dist)) %>%
```

```r
  ungroup %>%
  mutate(id = as.numeric(row.names(.)))
```

Data Summary

The data_summary data set containing the formatted data. Each row is an activity and consists of the variables; "time", "year", "date_without_month", "month", "day_of_month", "year_month", "total_dist", "total_time", "type", "speed", "pace", and "id".

```r
# Summarise Data
head(data_summary, 5)
```

```
## # A tibble: 5 x 12
##   time       year  date_without_mo~ month day_of_month year_month
##   <date>     <chr> <chr>            <chr> <chr>        <chr>
## 1 2016-06-16 2016  168              06    16           2016-06
## 2 2016-06-21 2016  174              06    22           2016-06
## 3 2016-06-22 2016  174              06    22           2016-06
## 4 2016-06-27 2016  179              06    27           2016-06
## 5 2016-06-28 2016  180              06    28           2016-06
## # ... with 6 more variables: total_dist <dbl>, total_time <dbl>,
## #   type <chr>, speed <dbl>, pace <dbl>, id <dbl>
```

Summarising the dataset reveals a well formatted set with no missing values. Dates are between 2016-06-16 and 2019-06-08 with 274 activities.

```r
summary(data_summary)
```

```
##       time                year           date_without_month
##  Min.   :2016-06-16   Length:274         Length:274
##  1st Qu.:2017-04-04   Class :character   Class :character
##  Median :2018-05-01   Mode  :character   Mode  :character
##  Mean   :2018-01-22
##  3rd Qu.:2018-11-28
##  Max.   :2019-06-08
##     month           day_of_month        year_month
##  Length:274         Length:274         Length:274
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##    total_dist         total_time        type               speed
##  Min.   : 0.05198   Min.   :   48   Length:274         Min.   : 0.06016
##  1st Qu.: 5.02472   1st Qu.: 1976   Class :character   1st Qu.: 8.37228
##  Median : 7.52681   Median : 3466   Mode  :character   Median :10.09737
##  Mean   :14.08330   Mean   : 4584                      Mean   :11.44235
##  3rd Qu.:23.37276   3rd Qu.: 5808                      3rd Qu.:15.16998
##  Max.   :81.93648   Max.   :27369                      Max.   :28.84448
##       pace              id
##  Min.   :  2.080   Min.   :  1.00
```

```
##  1st Qu.:  3.955   1st Qu.: 69.25
##  Median :  5.942   Median :137.50
##  Mean   : 13.333   Mean   :137.50
##  3rd Qu.:  7.167   3rd Qu.:205.75
##  Max.   :997.325   Max.   :274.00
```

Calendar

A calendar of activities shows a rather irregular pattern of activities, though on average 1-2 activites each week. As the location of activity occured in both northern and southern hemispheres, estimation of the impact of sunlight hours, season, temperature and weather on the freqency of activity will further time to analyse. This will be investigated post assignment.

```
# Plot Calendar
#####################################################

# Generate plot data
time_max <- today()

daily_data <- data_summary %>%
  group_by(time) %>%
  mutate(dist = sum(total_dist), type = type[1]) %>%
  ungroup() %>%
  mutate(time = lubridate::date(time))

daily_data <- daily_data %>%
  group_by(type) %>%
  mutate(max_distance = max(dist))

daily_data <- daily_data %>%
  group_by(time) %>%
  mutate(scaled_distance = dist / max_distance)

daily_data_cal <- daily_data %>%
  frame_calendar(x = time, y = 1, date = time, calendar = "monthly") %>%
  transform(week = as.POSIXlt(time)$yday %/% 7 + 1,
            wday = weekdays(as.POSIXlt(time)),
            year = as.POSIXlt(time)$year + 1900)

# Reverse days for graphing
daily_data_cal$wday <- factor(daily_data_cal$wday, day.name[7:1])

# Graph data
p <- daily_data_cal %>%
  ggplot(aes(x = week, y = wday, fill = type, alpha=scaled_distance)) +
  geom_tile(size=1) +
  scale_fill_manual(values=c(Ride="red", Run="blue", Walk="green",
Hike="yellow")) +
```

```
  ylab("Weekday") +
  xlab("Month") +
  theme(legend.position="right") +
  scale_x_continuous(breaks = (c(0:11)*52/12), labels = month.abb) +
  facet_wrap(~year, ncol = 1)
p
```

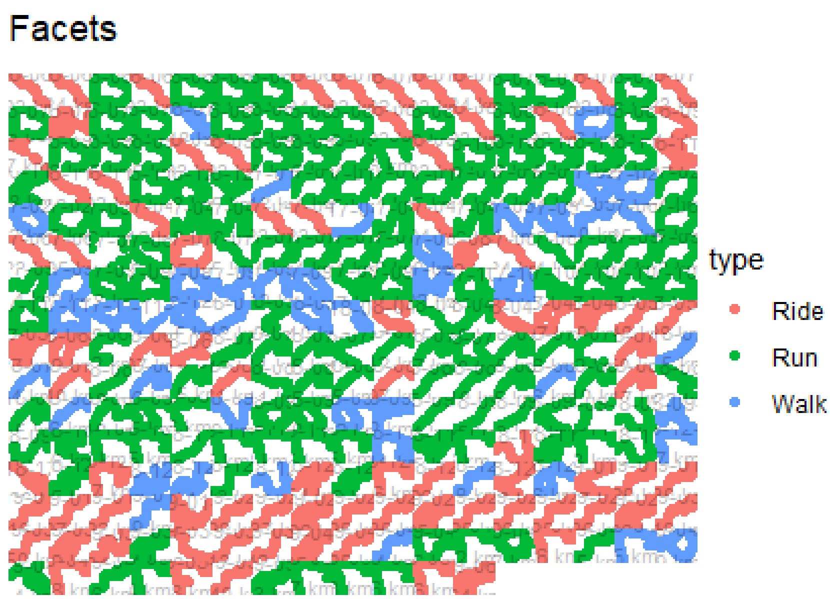

```
# Save plot
ggsave(paste("calendar",Sys.Date(),".png"), p, width = 30, height = 30, units
= "cm", dpi = 300)
```

Plot Facets

Maps were plotted for each activity, with distance, speed and time, allowing identification of routes that were attempted multiple times and the consistancy of the performance.

```
data_facets <- data %>%
  group_by(id) %>%
  summarise(lon_avg = mean(lon),
            lat_avg = mean(lat),
            time = max(elapsed_time),
            distanceKm = max(distance/1000),
            timePerKm = round((time) / (distanceKm)),
            description =
paste(type[1],"\n",as.Date(date[1]),"\n",round(distanceKm),"km\n",
tolower(seconds_to_period(timePerKm)), "/ km"))
```

```r
p <- data %>% ggplot(aes(x = lon, y = lat, group = id, col = type)) +
  geom_point() +
  facet_wrap(~id, scales = "free") +
  theme_void() +
  ggtitle("Facets") +
  theme(panel.spacing = unit(0, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank(),
        plot.margin = unit(rep(1, 4), "cm"),
        legend.position="right") +
  geom_text(aes(lon_avg, lat_avg, label = description), data = data_facets,
alpha = 0.25, size = 3, color = "black")
p
```




```r
# Save plot
ggsave(paste("facets",Sys.Date(),".png"), p, width = 50, height = 50, units =
"cm")
```

Analysis and Results

Metrics for fitness, fatigue and form were developed using the information published by Chris Spada and Will Meyer. An impulse for each type of activity was created, using the available speed and distance information. Fitness, form and fatigue effects were also modelled. Note that HR was not incorporated in this metric and the intensity of the activity and the fitness of the author was assumed a constant.

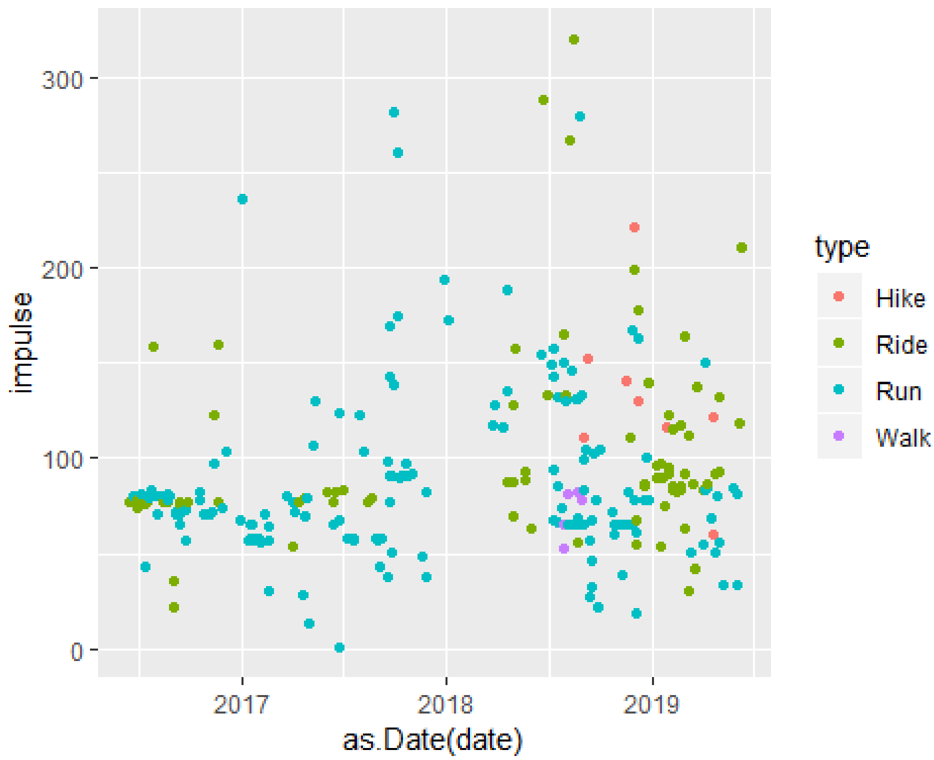
Create Fitness Metrics

Impulse

The following graph shows the estimated impulse for each activity. Results correlate strongly with the percieved effort.

```
#####################################################

# Fitness, Freshness and Form Graphs
# Analyse and plot graph of fitness and freshness using Strava running and
cycling data

# Assumptions
impulse_runningPerKm <- 13 # between ~10 - 15
impulse_ridingPerKm <-  3.3 # between ~3 - 5

# Fitness constants
#fitness = 1*impulse, -20% per week, ~2% per day, use ~3 months data
fitness_constant <- 0.98
impulse_to_fitness <- 0.02

# Fatigue constants
# fatigue = 5*impulse, -80% per week, ~15% per day, use ~3 months data
fatigue_constant <- 0.85
impulse_to_fatigue <- 0.125

# Form
# form = difference between fitness and fatigue

# Calculate form, fitness and fatigue for each day
df_raw <- read.csv(file = "./data/activities.csv") # read data

#Specify Start and End dates
start_date <- min(as.numeric(as.Date(df_raw$date)))
end_date <- max(as.numeric(as.Date(df_raw$date)))

df_raw <- df_raw %>%
  filter(as.numeric(as.Date(date)) >= start_date)
```

```
# Impulse
df <- df_raw %>%
  mutate(impulse = ifelse(type=="Ride", impulse_ridingPerKm * distance /
1000, impulse_runningPerKm * distance / 1000))

plot <- df %>%
  ggplot(aes(x = as.Date(date), y = impulse, color = type)) +
  geom_point()
plot
```




```
# Save plot
ggsave(paste("Impulse_Date",Sys.Date(),".png"), plot, width = 50, height =
50, units = "cm")
```

Form, Fitness and Fatigue

The form, fitness and fatigue metrics were created by scaled exponential decay functions. An approximation of the function is shown below;

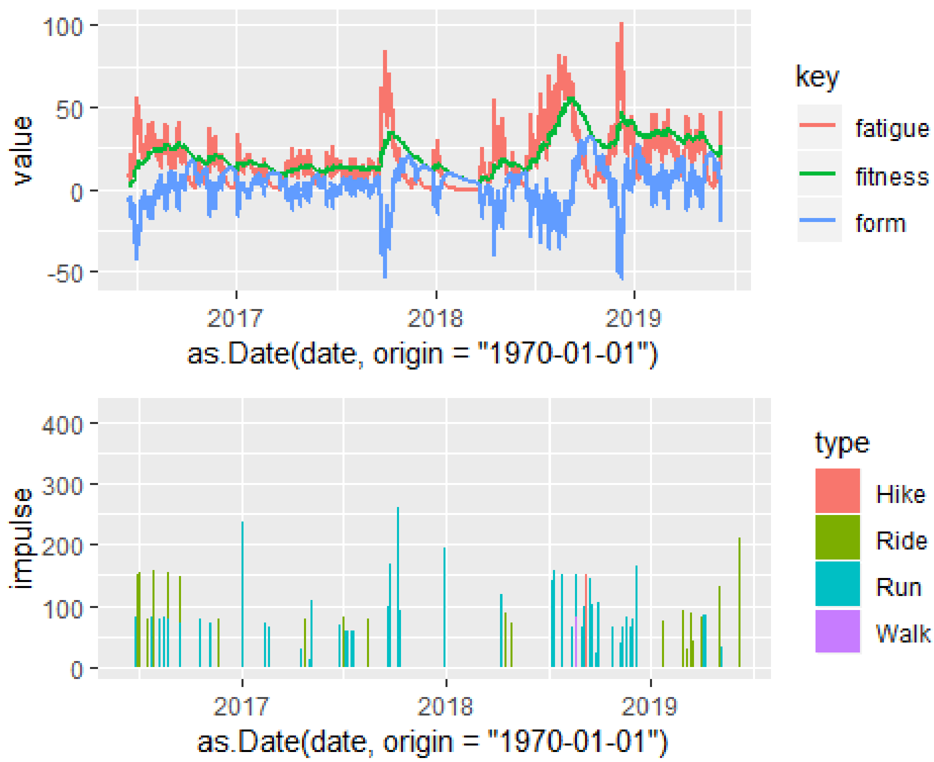
$$Y_t = \sum(a * e^{-\alpha t} + b * e^{-\beta t})$$

where Y_t is the predicted fitness, form and freshness, a and b are the scaling constants and α and β the exponential scaling constants.

A historical timeline of form, fitness and fatigue metrics is plotted over the entire timeline, with the daily impulse that impact the metric plotted below.

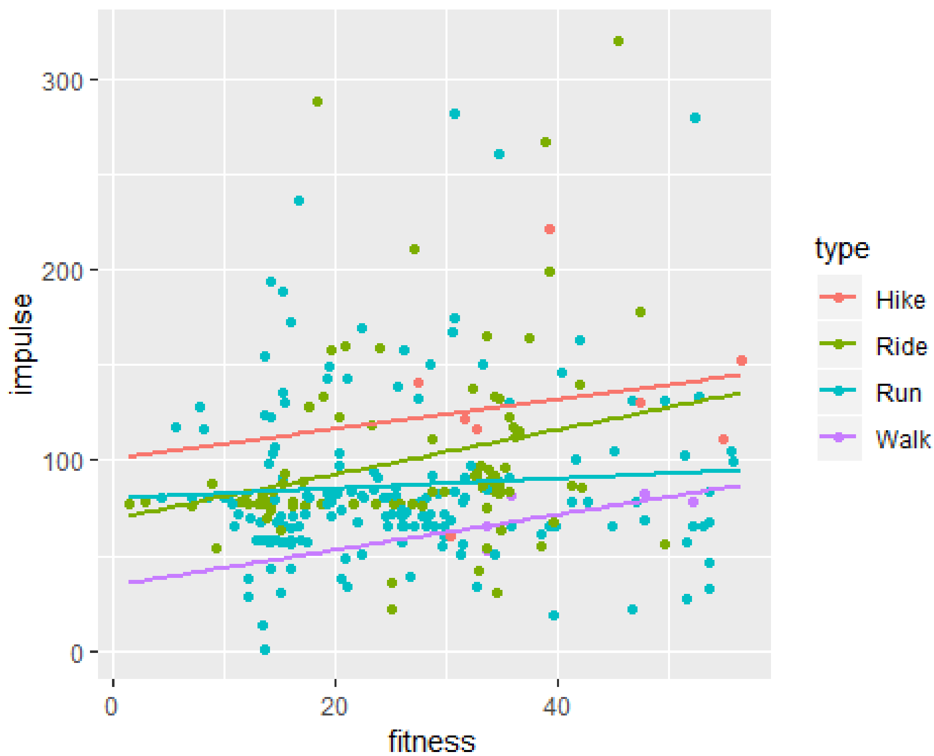
```
#Function - Fitness
f_fitness <- function(i_date) {
  df_temp <- df %>%
    mutate(date = as.Date(date)) %>%
    filter(as.numeric(date) - as.numeric(as.Date(i_date, origin = "1970-01-
01")) <= 0) %>%
    mutate(relative_impulse = 0) %>%
    mutate(relative_impulse = impulse * impulse_to_fitness *
fitness_constant^(as.numeric(as.Date(i_date, origin = "1970-01-01") -
as.numeric(date))))
  return(sum(df_temp$relative_impulse))
}

# Function - Fatigue
f_fatigue <- function(i_date) {
  df_temp <- df %>%
    mutate(date = as.Date(date)) %>%
    filter(as.numeric(date) - as.numeric(as.Date(i_date, origin = "1970-01-
01")) <= 0) %>%
    mutate(relative_impulse = 0) %>%
    mutate(relative_impulse = impulse * impulse_to_fatigue *
fatigue_constant^(as.numeric(as.Date(i_date, origin = "1970-01-01") -
as.numeric(date))))
  return(sum(df_temp$relative_impulse))
}

# Update df with Fitness, Fatigue, Form Metrics
df <- df %>%
  mutate(fitness = sapply(date, f_fitness)) %>%
  mutate(fatigue = sapply(date, f_fatigue)) %>%
  mutate(form = fitness - fatigue)

# Function - Form, Fitness and Fatigue
#####################################################

df_fff <- data_frame("date" = start_date:end_date)

## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.

df_fff <- df_fff %>%
  mutate(fitness = sapply(date, f_fitness)) %>%
  mutate(fatigue = sapply(date, f_fatigue)) %>%
  mutate(form = fitness - fatigue)

# Plot for fitness and fatigue data
df_tidy <- df_fff %>% gather(type = fitness:form)

plot_1 <- df_tidy %>% ggplot(aes(x = as.Date(date, origin = "1970-01-01"), y
= value, col = key)) +
```

```
  geom_line(size=1)

plot_2 <- df %>% ggplot(aes(x = as.Date(date, origin = "1970-01-01"), y =
impulse, fill=type)) +
  geom_bar(stat = "identity")

grid.arrange(plot_1, plot_2, ncol = 1)
```



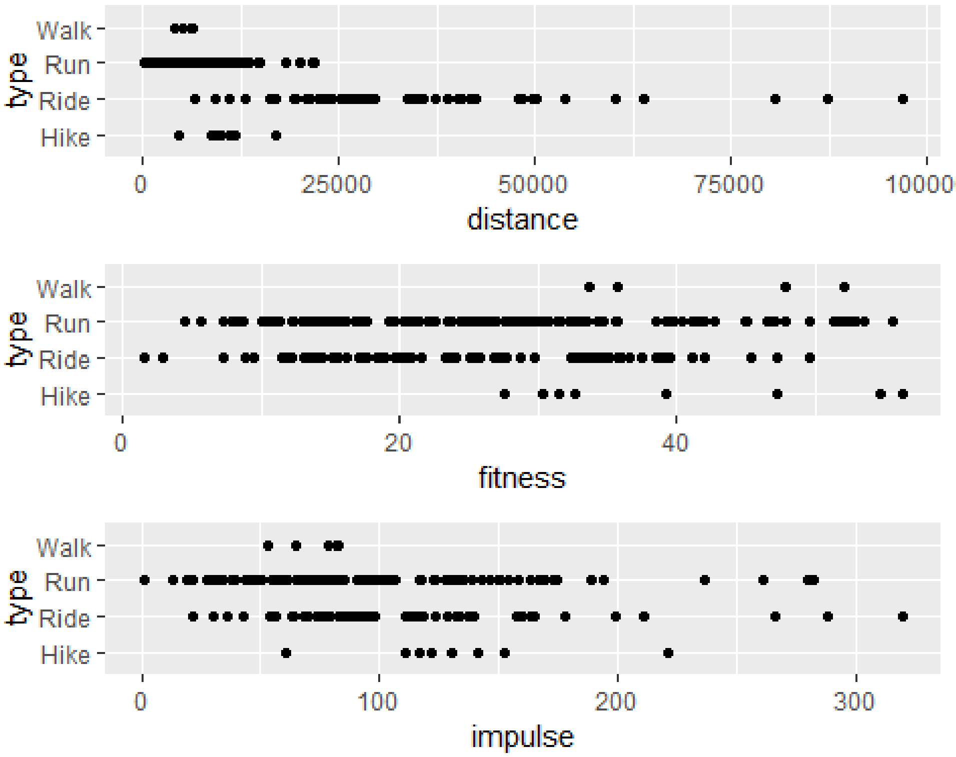
A linear model is fitted to the data to assertain any broad correlations. Hike, Ride, Run and Walk, all trend slightly upward with fitness. Indicating that fitness and impact are weakly correlated.

```
# Plot Type, Fitness Impulse, Linear approximation
plot_2 <- df %>% ggplot(aes(x = fitness, y = impulse, color = type)) +
  geom_point() +
  geom_smooth(se = FALSE, method = "gam", formula = y ~ x, fullrange=TRUE)
plot_2
```



For each type of activity, the distance, fitness,and impulse is plotted. An obvious correlation exists between distance and activity type, and no obvious correlation between impulse and fitness.

```
# Plot Attributes against type
plot_3 <- df %>% ggplot(aes(x = distance, y = type)) +
  geom_point()

plot_4 <- df %>% ggplot(aes(x = fitness, y = type)) +
  geom_point()

plot_5 <- df %>% ggplot(aes(x = impulse, y = type)) +
  geom_point()

grid.arrange(plot_3, plot_4, plot_5, ncol = 1)
```



Walk
Run
Ride
Hike
type
0
25000
50000
75000
10000
distance
Walk
Run
Ride
Hike
type
0
20
40
fitness
Walk
Run
Ride
Hike
type
0
100
200
300
impulse

Create Fitness Machine Learning Models

Both K-Nearest-Neighbours (KNN) and Quadrature Discriminant Analysis (QDA) were performed on the dataset. Distance was the obvious paramter to train the machine learning algorithm, though fitness and impulse was also implemented.

Training and Test Sets

```
#####################################################
## 75% of the sample size
smp_size <- floor(0.75 * nrow(df))
smp_size

## [1] 201

## set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(df)), size = smp_size)

df_train <- df[train_ind, ]
df_test <- df[-train_ind, ]
```

KNN

Distance

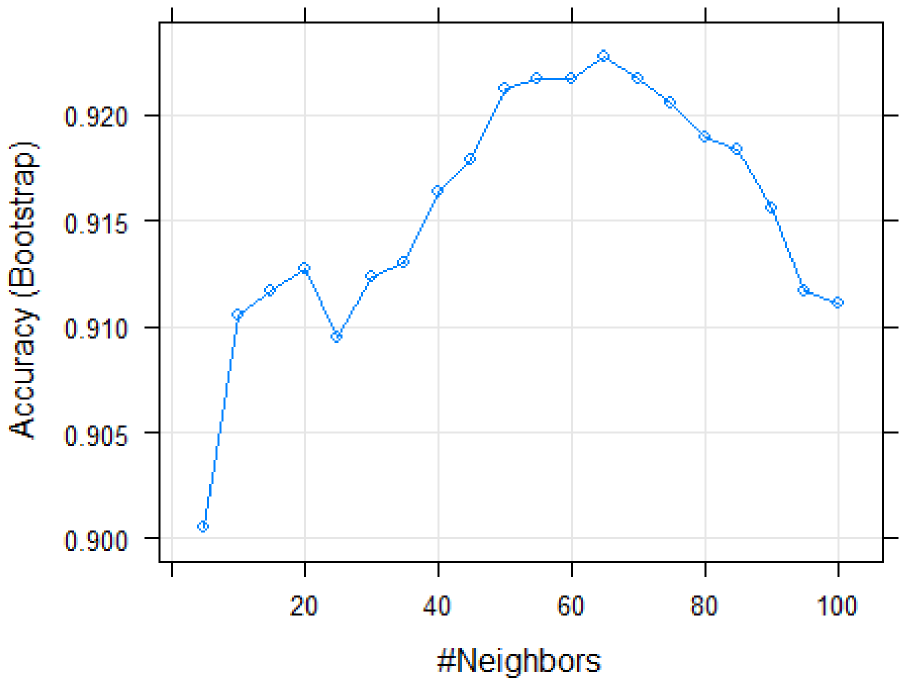
KNN with distance was initally chosen to model the type of activity and achieved the highest accuracy as expected.

```
#####################################################

train_knn <- train(type ~ distance, method = "knn",
                   data = df_train,
                   tuneGrid = data.frame(k = c(1:20)*5))
y_hat_knn <- predict(train_knn, df_test)
y_hat_knn

##  [1] Ride Run  Ride Ride Run  Ride Ride Run  Ride Run  Run  Run  Run  Run
## [15] Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Run
## [29] Run  Ride Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Run
## [43] Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Ride
## [57] Ride Ride Ride Ride Ride Ride Run  Run  Ride Ride Ride
## Levels: Hike Ride Run Walk

cm_knn <- confusionMatrix(data = y_hat_knn, reference = df_test$type)
cm_knn$overall

##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.402985e-01   8.576739e-01   8.541368e-01   9.834956e-01   6.865672e-01
## AccuracyPValue  McnemarPValue
##   4.379726e-07            NaN
```

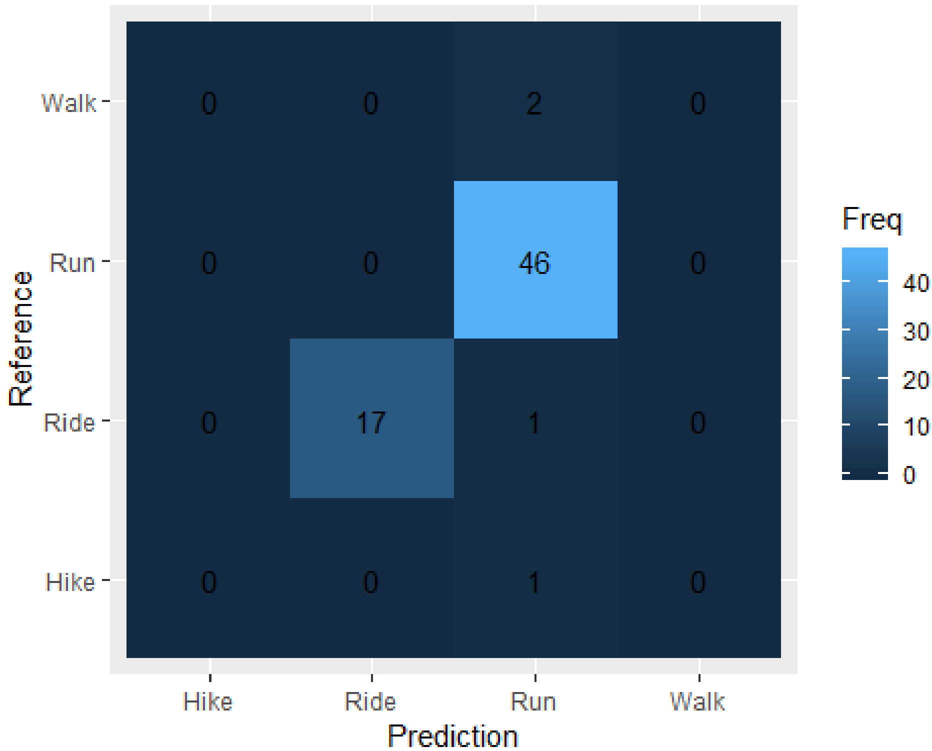
```r
train_knn
```

```
## k-Nearest Neighbors
##
## 201 samples
##   1 predictor
##   4 classes: 'Hike', 'Ride', 'Run', 'Walk'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 201, 201, 201, 201, 201, 201, ...
## Resampling results across tuning parameters:
##
##   k    Accuracy   Kappa
##     5  0.9004707  0.7976204
##    10  0.9105352  0.8142824
##    15  0.9116719  0.8159602
##    20  0.9127453  0.8181163
##    25  0.9095113  0.8110574
##    30  0.9123538  0.8170444
##    35  0.9129962  0.8183659
##    40  0.9163216  0.8259355
##    45  0.9179362  0.8296770
##    50  0.9211895  0.8366632
##    55  0.9217023  0.8376830
##    60  0.9216949  0.8377343
##    65  0.9228021  0.8401236
##    70  0.9216753  0.8377090
##    75  0.9205538  0.8352928
##    80  0.9189649  0.8319444
##    85  0.9183788  0.8305620
##    90  0.9156279  0.8245246
##    95  0.9116966  0.8158541
##   100  0.9110676  0.8139766
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 65.
```

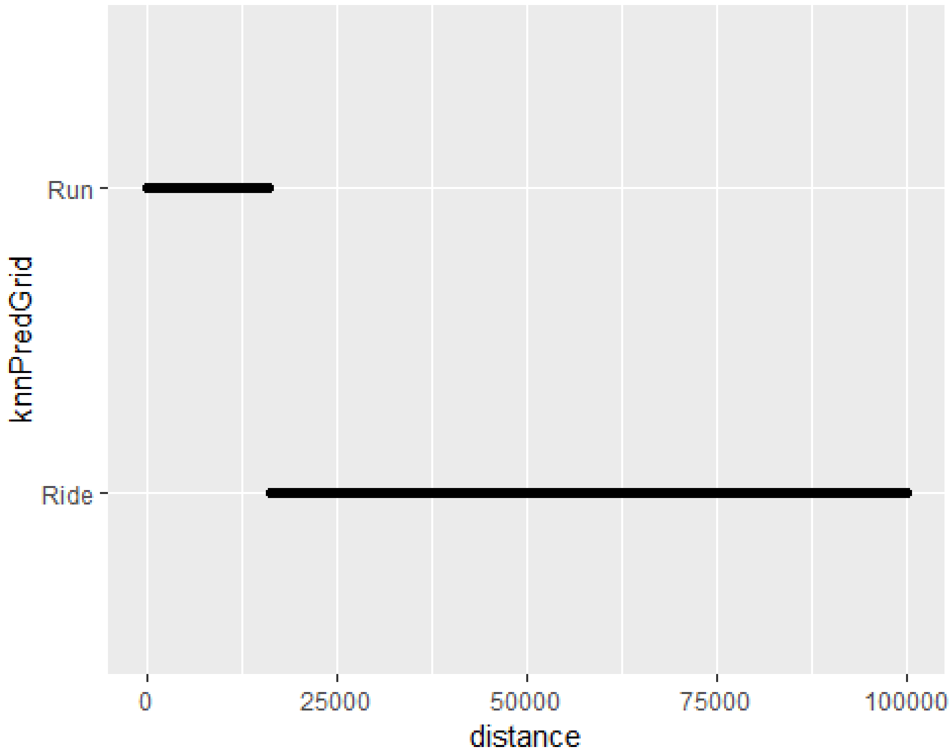
```r
plot(train_knn)
```




```r
df_cm <- as.data.frame(cm_knn$table)

df_cm %>% ggplot(aes(Prediction, Reference)) +
  geom_tile(aes(fill = Freq)) +
  geom_text(aes(label = Freq))
```




```
# Plot descision boundary
lgrid <- expand.grid(distance=seq(0, 100000, by=100))
knnPredGrid <- predict(train_knn, newdata=lgrid)

knnPred <- cbind(lgrid, knnPredGrid)

knnPred %>% ggplot(aes(x = distance, y = knnPredGrid)) +
  geom_point()
```



Fitness and Impulse

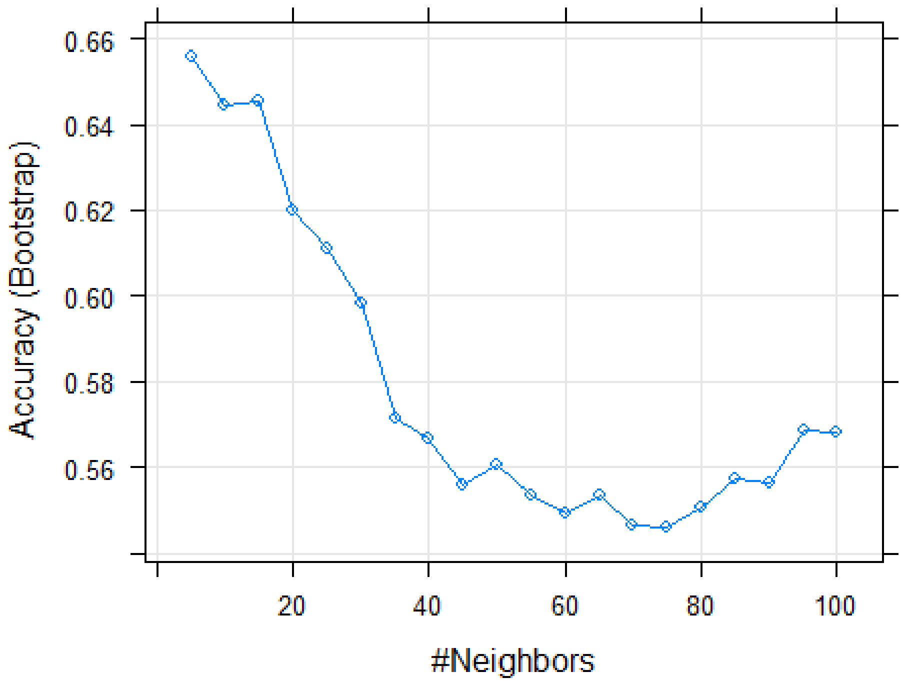
KNN with fintness and impulse was chosen to model the type of activity and achieved the limited accuracy.

```
#######################################################

train_knn <- train(type ~ fitness + impulse, method = "knn",
                   data = df_train,
                   tuneGrid = data.frame(k = c(1:20)*5))
y_hat_knn <- predict(train_knn, df_test)
y_hat_knn

##  [1] Run  Run  Ride Ride Run  Run  Ride Run  Ride Run  Ride Run  Run  Run
## [15] Run  Run  Ride Ride Run  Run  Run  Run  Run  Ride Ride Ride Ride Ride
## [29] Run  Ride Ride Run  Run  Ride Run  Run  Run  Ride Ride Run  Run  Run
## [43] Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Run  Ride Run  Run
## [57] Run  Ride Ride Ride Run  Run  Run  Ride Ride Ride Ride
## Levels: Hike Ride Run Walk

cm_knn <- confusionMatrix(data = y_hat_knn, reference = df_test$type)
cm_knn$overall

##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.6417910      0.2368296      0.5153035      0.7553051      0.6865672
## AccuracyPValue  McnemarPValue
##      0.8223528            NaN
```

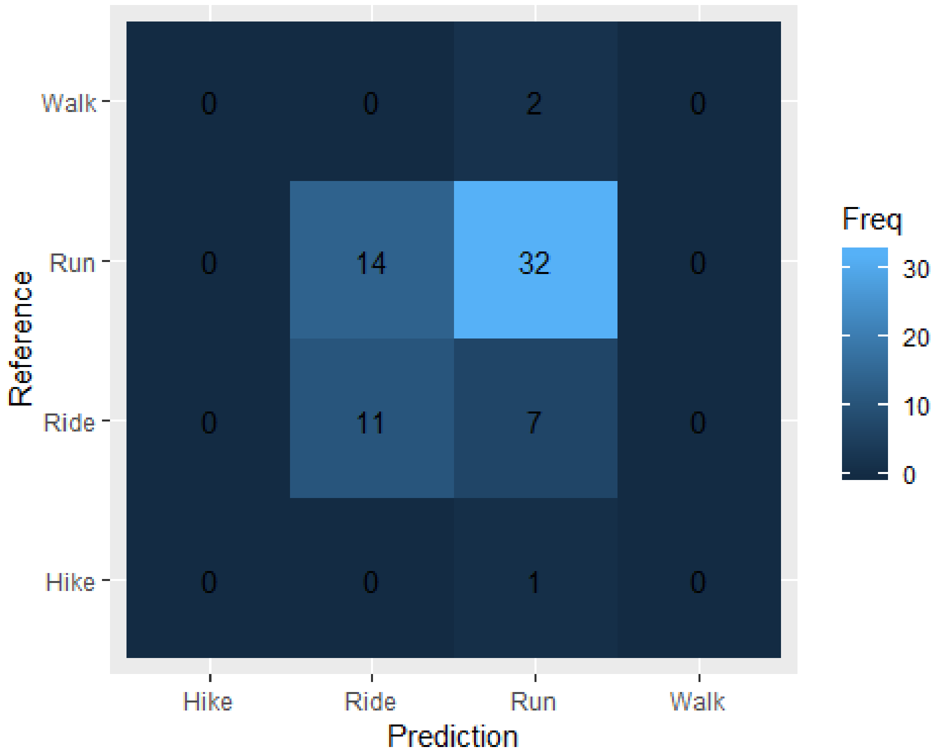
```r
train_knn
```

```
## k-Nearest Neighbors
##
## 201 samples
##   2 predictor
##   4 classes: 'Hike', 'Ride', 'Run', 'Walk'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 201, 201, 201, 201, 201, 201, ...
## Resampling results across tuning parameters:
##
##   k    Accuracy   Kappa
##     5  0.6560740   0.3213886780
##    10  0.6447358   0.2855328055
##    15  0.6452596   0.2904320173
##    20  0.6200935   0.2347893166
##    25  0.6111775   0.2080185768
##    30  0.5981116   0.1700144604
##    35  0.5715389   0.0981805841
##    40  0.5665343   0.0698183507
##    45  0.5559202   0.0246279934
##    50  0.5606594   0.0268207445
##    55  0.5534453   0.0116656113
##    60  0.5493913   0.0039412035
##    65  0.5532466   0.0075667687
##    70  0.5462079  -0.0049485815
##    75  0.5456928  -0.0051562248
##    80  0.5506593  -0.0058935376
##    85  0.5570759  -0.0039340172
##    90  0.5560613  -0.0103205766
##    95  0.5684422   0.0009328715
##   100  0.5679941  -0.0074279880
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

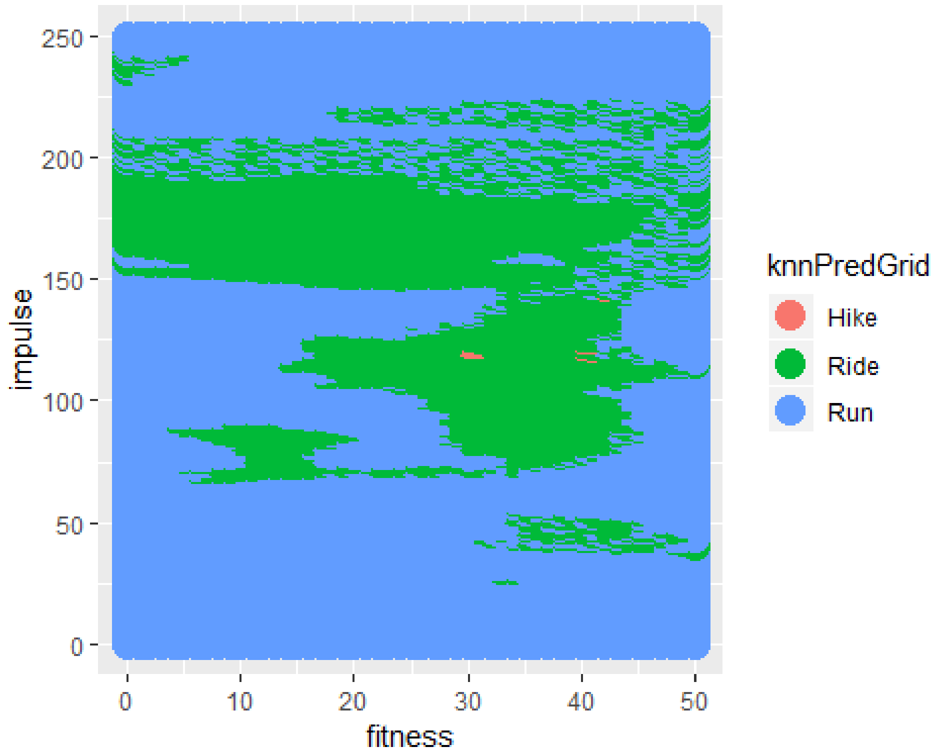
```r
plot(train_knn)
```




```
df_cm <- as.data.frame(cm_knn$table)

df_cm %>% ggplot(aes(Prediction, Reference)) +
  geom_tile(aes(fill = Freq)) +
  geom_text(aes(label = Freq))
```




```
# Plot descision boundary
lgrid <- expand.grid(fitness=seq(0, 50, by=1),
                     impulse=seq(0, 250,by=1))
knnPredGrid <- predict(train_knn, newdata=lgrid)

knnPred <- cbind(lgrid, knnPredGrid)

plot <- knnPred %>% ggplot(aes(x = fitness, y = impulse, color=knnPredGrid))
+
  geom_point(size=5)
plot
```



QDA

Distance

A QDA machine learning model was used to estimate the type of activity with distance. The QDA algorithm performed similar to the KNN algorithm on the dataset.

```
#####################################################

train_qda <- train(type ~ distance, data = df_train)
train_qda

## Random Forest
##
## 201 samples
##   1 predictor
##   4 classes: 'Hike', 'Ride', 'Run', 'Walk'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 201, 201, 201, 201, 201, 201, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.864068  0.7309603
```

```
##
## Tuning parameter 'mtry' was held constant at a value of 2
```

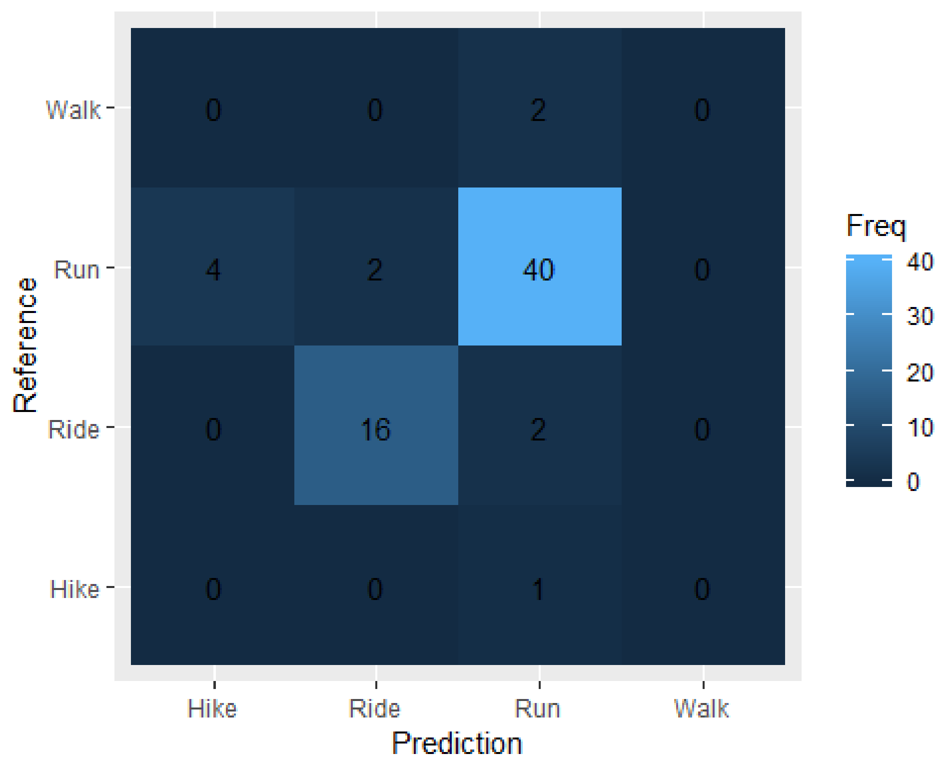
```r
y_hat <- predict(train_qda, df_test)
y_hat
```

```
##  [1] Ride Run  Ride Ride Run  Ride Ride Run  Ride Run  Run  Run  Run  Run
## [15] Run  Run  Run  Run  Run  Hike Run  Run  Run  Run  Run  Run  Ride Run
## [29] Hike Ride Run  Run  Run  Run  Run  Run  Run  Hike Run  Run  Run  Run
## [43] Run  Run  Run  Run  Run  Run  Run  Hike Run  Run  Run  Ride Run  Ride
## [57] Run  Ride Ride Ride Ride Ride Run  Run  Ride Ride Ride
## Levels: Hike Ride Run Walk
```

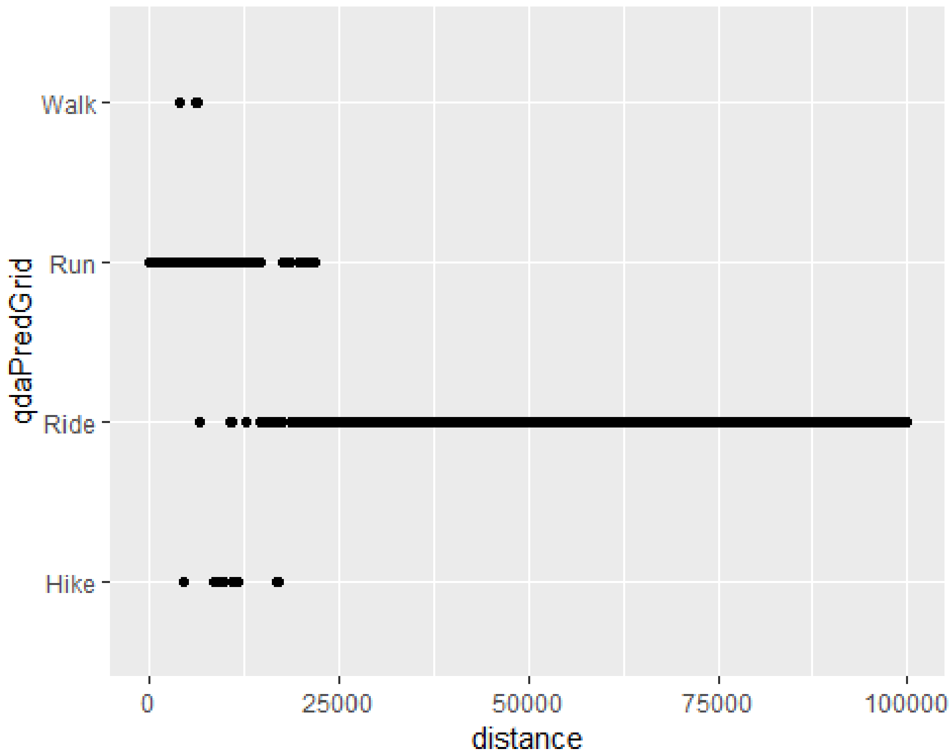
```r
cm <- confusionMatrix(data = y_hat, reference = df_test$type)
cm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##    0.835820896    0.647537064    0.725199184    0.915087454    0.686567164
## AccuracyPValue  McnemarPValue
##    0.004363945            NaN
```

```r
df_cm <- as.data.frame(cm$table)

df_cm %>% ggplot(aes(Prediction, Reference)) +
  geom_tile(aes(fill = Freq)) +
  geom_text(aes(label = Freq))
```



```r
# Plot descision boundary
lgrid <- expand.grid(distance=seq(0, 100000, by=10))
qdaPredGrid <- predict(train_qda, newdata=lgrid)

qdaPred <- cbind(lgrid, qdaPredGrid)

p <- qdaPred %>% ggplot(aes(x = distance, y = qdaPredGrid)) +
  geom_point()
p
```



Conclusion

Strava data was investigated, metrics for fitness and freshness implemented and models created to predict my performance and type of activity. Distance was showns as the most reliable predictor for the type of activity, with limited collelation between fitness, fatigue and activity.

Overall the author has learned a significant amout for the Harvard Data Science course and a lot of fun was had. Thank you!