SVM over EcoBici 2017 Data

MD01 CIC IPN

Emanuel Becerra Soto

December 2018

Introduction

The dataset analyzed was a subset of the EcoBici 2017 trips data. All the trips made on 2017 are approximately 10,000,000 but after some cleaning a 20% of the data was taken for exploratory analysis.

Here we further divide this 20% percent into only the trips containing the two most used stations (starting trips): 27 (Reforma) and 271 (Buenavista), for a total of ~37,000 trips

The goal of the analysis is to classify the trips that ended up on either Reforma (27) or Buenavista(271), possibility uncovering some mobility patterns on Mexico City.

Part of the the reason of only classifying only two stations was due by time and computational constrains but a future objective is to extend the reach of this analysis.

For the classification a Support Vector Machine was used.

```
# Setting seed for reproducibility
set.seed(8745)

####### Libraries #######
library(e1071)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ggthemes)
library(wesanderson)
```

```
partition_data <- function(data_all, train = 0.6, test = 0.2){
  validation <- 1 - (train + test)
  n <- nrow(data_all)
```

```
  n_train <- floor(n * train)
  n_test <- ceiling(n * test)
  n_validation <- n - (n_train + n_test)

  random_idx <- sample(1:n)
  idx_train <- random_idx[1:n_train]
  idx_test <- random_idx[(n_train+1) : (x <- (n_train+1 + n_test-1))]
  idx_validation <- random_idx[ (x+1) : n]

  idx_train <- sort(idx_train)
  idx_test <- sort(idx_test)
  idx_validation <- sort(idx_validation)

  data_train <- data_all[idx_train,]
  data_test <- data_all[idx_test,]
  data_val <- data_all[idx_validation,]

  return(list(data_train, data_test, data_val))
}
```

Wrangling the data for the classification task

The data was loaded.

```
####### Loading the data #######

file <- '2017_bikes_val_3.csv'
bikes <- read_csv(file)
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer(),
##   sex = col_character(),
##   trip_time_seconds = col_double(),
##   leave_month = col_character(),
##   leave_day = col_character(),
##   arrive_month = col_character(),
##   arrive_day = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
head(bikes, n = 10)
```

```
## # A tibble: 10 x 20
##    sex     age station_start station_end  bike trip_time_secon~ leave_month
##    <chr> <int>         <int>       <int> <int>            <dbl> <chr>
##  1 M        42            33           1  1570              703 ene
##  2 F        24           152           5  4056             1480 ene
##  3 M        32            30          74  7114              734 ene
##  4 M        24           170         288  3718              259 ene
##  5 M        27            22          28  3625              481 ene
##  6 M        27            30         256  1849             1020 ene
```

```
##  7 F        24            30         254  4266             951 ene
##  8 M        55           386         182  9482            1436 ene
##  9 M        26            27         293  3983            1023 ene
## 10 M        40           270           1  9502            1339 ene
## # ... with 13 more variables: leave_day <chr>, leave_hour <int>,
## #   leave_minute <int>, leave_second <int>, leave_epoch <int>,
## #   leave_epoch_day <int>, arrive_month <chr>, arrive_day <chr>,
## #   arrive_hour <int>, arrive_minute <int>, arrive_second <int>,
## #   arrive_epoch <int>, arrive_epoch_day <int>
```

```
dim(bikes)
```

```
## [1] 1893573      20
```

Any trip with a duration of more than a day was considered as an outlier and removed.

```
####### Removing trips that lasted more than 24hrs. #######

# The trips lasting more than 24 hours were consideered as outliers
# Removing trips that were more than a day
threshold <- 24 * 3600
bikes <- filter(bikes, bikes$trip_time_seconds <= threshold)
```

A three new columns were derived from the data and then added. The new columns are: The trips that exceeded the 45 min and 1 hour mark and column combining both exceeding times.

The reason for adding the new columns is that the EcoBici service charges and extra fee for exceeding certain times, providing more information for the classification.

Exceeding time trips

```
####### Adding extra columns with exceding times #######

# Setting the exceding times
# EcoBici system set prices (MXN) for exceeding time tips as follows:
#
# From 0min-45min No extra cost.
# From 45min-60min $12.00.
# From Each extra hour $39.00.
# From More than a day 24 hrs. $5485.00.

exceeding_time <- 45 * 60
exceeding_time_hour <- 1 * 3600

# Adding a vector of all the exeding time trips
bikes$exceeding <- bikes$trip_time_seconds > exceeding_time

# Adding the 1 hour exeding trips
bikes$exceeding_hour <- bikes$trip_time_seconds > exceeding_time_hour

# Adding a colunm with the type of exceeding
exceeding_type <- paste(bikes$exceeding, bikes$exceeding_hour)
exceeding_type <- sub( x = exceeding_type, pattern =  'FALSE FALSE', replacement = 'no_exceeding' )
exceeding_type <- sub( x = exceeding_type, pattern =  'TRUE FALSE', replacement = '45_min' )
```

```
exceeding_type <- sub( x = exceeding_type, pattern =  'TRUE TRUE', replacement = '1_hour' )
bikes$exceeding_type <- exceeding_type
```

```
ggplot(bikes, aes(x = exceeding, fill = exceeding_type))+
  geom_bar()+
  ggtitle('Exceeding Time Trips Eco Bici 2017')+
  scale_x_discrete(labels=c('On time','Exceeding'))+
  ylab('Trips')+
  scale_fill_manual(values=wes_palette("Royal1"),
                    name='Trip Duration',
                    labels=c("1 hour or more", "Between 45 min - 1 hour",
                             "Below 45 min"))+
  theme_fivethirtyeight()
```

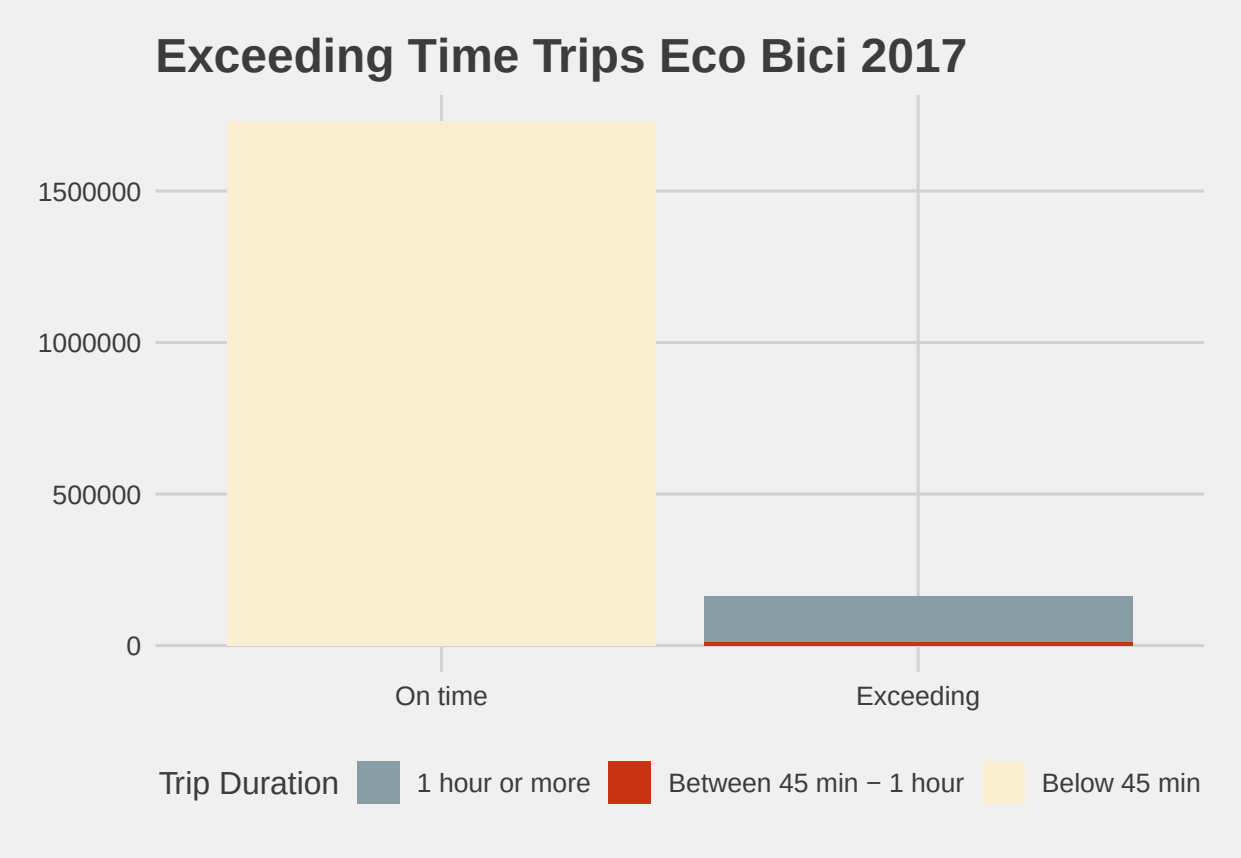

Figure 1.- Bar plot for showing the number of exceeding trips of the 20% of all the Eco Bici Trips on 2017.

Feature Selection

From previous analyzes of the data, probably uninformative and redundant variables, some of them directly redundant as were derived from the originally 9 ones.

On future analyses more variables could be added.

From the 23 variables: sex, age, station start number, trip duration in seconds, month, hour, minute and if the trip exceeded the 45 min mark were selected. first approach for the classification

```r
# Feature Selection by hand
names(bikes)
```

```
##  [1] "sex"               "age"               "station_start"
##  [4] "station_end"       "bike"              "trip_time_seconds"
##  [7] "leave_month"       "leave_day"         "leave_hour"
## [10] "leave_minute"      "leave_second"      "leave_epoch"
## [13] "leave_epoch_day"   "arrive_month"      "arrive_day"
## [16] "arrive_hour"       "arrive_minute"     "arrive_second"
## [19] "arrive_epoch"      "arrive_epoch_day"  "exceeding"
## [22] "exceeding_hour"    "exceeding_type"
```

```r
hand_picked_features <- c('sex', 'age', 'station_start',
                          'trip_time_seconds', 'leave_month',
                          'leave_day', 'leave_hour', 'leave_minute',
                          'exceeding', 'station_end')
bikes2 <- bikes[ hand_picked_features ]
```

Station Usage Statistics

As explained in the introduction only two stations were classified due to time and computational constrains. The classification task was as follows: from the trip data, including initial station, predict the destination. The two most used initial stations were used: 27 (Reforma) and 271 (Buenavista).

So the two most used stations were found:

```r
# Counting the most used stations
station_usage_start <- arrange( as.tibble( table(bikes$station_start) ), desc(n) )
station_usage_start$perct_start <- ( station_usage_start$n / sum(station_usage_start$n) )
station_usage_end <- arrange( as.tibble( table(bikes$station_end) ), desc(n) )
station_usage_end$perct_end <- ( station_usage_end$n / sum(station_usage_end$n) )

# Joining start and end stations usage percent onto one table
station_usage <- inner_join(station_usage_start, station_usage_end, by = 'Var1')
station_usage <- rename(station_usage, st_id = Var1 , n_start = n.x, n_end = n.y)
station_usage <- arrange(station_usage, desc(n_end))

# Printing the top 10 most used stations
station_usage[1:10,]
```

```
## # A tibble: 10 x 5
##    st_id n_start perct_start n_end perct_end
##    <chr>   <int>       <dbl> <int>     <dbl>
##  1 27      22570     0.0119  22029   0.0116
##  2 266     10722     0.00566 16427   0.00868
##  3 1       15375     0.00812 15970   0.00844
##  4 18      15183     0.00802 15613   0.00825
##  5 271     17921     0.00947 14708   0.00777
##  6 43      11361     0.00600 13532   0.00715
##  7 21      13943     0.00736 12887   0.00681
##  8 217     10593     0.00560 12116   0.00640
##  9 64      11312     0.00598 12057   0.00637
## 10 25      11693     0.00618 12044   0.00636
```

Table of the top 10 most used stations ordered by start.

```r
# Cumulative percents over station usage
plot(cumsum(station_usage$perct_start))
```

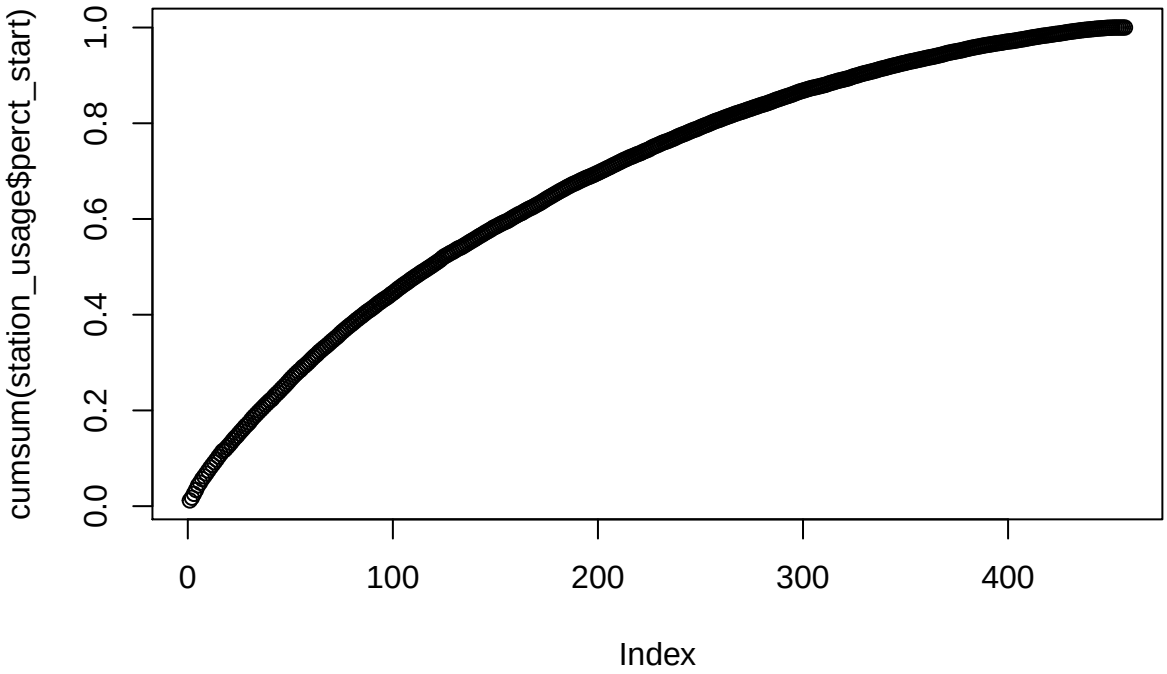

Figure 2.- Cumulative frequency of usage of the starting stations.

```r
plot(cumsum( sort(station_usage$perct_end, decreasing = TRUE) ))
```

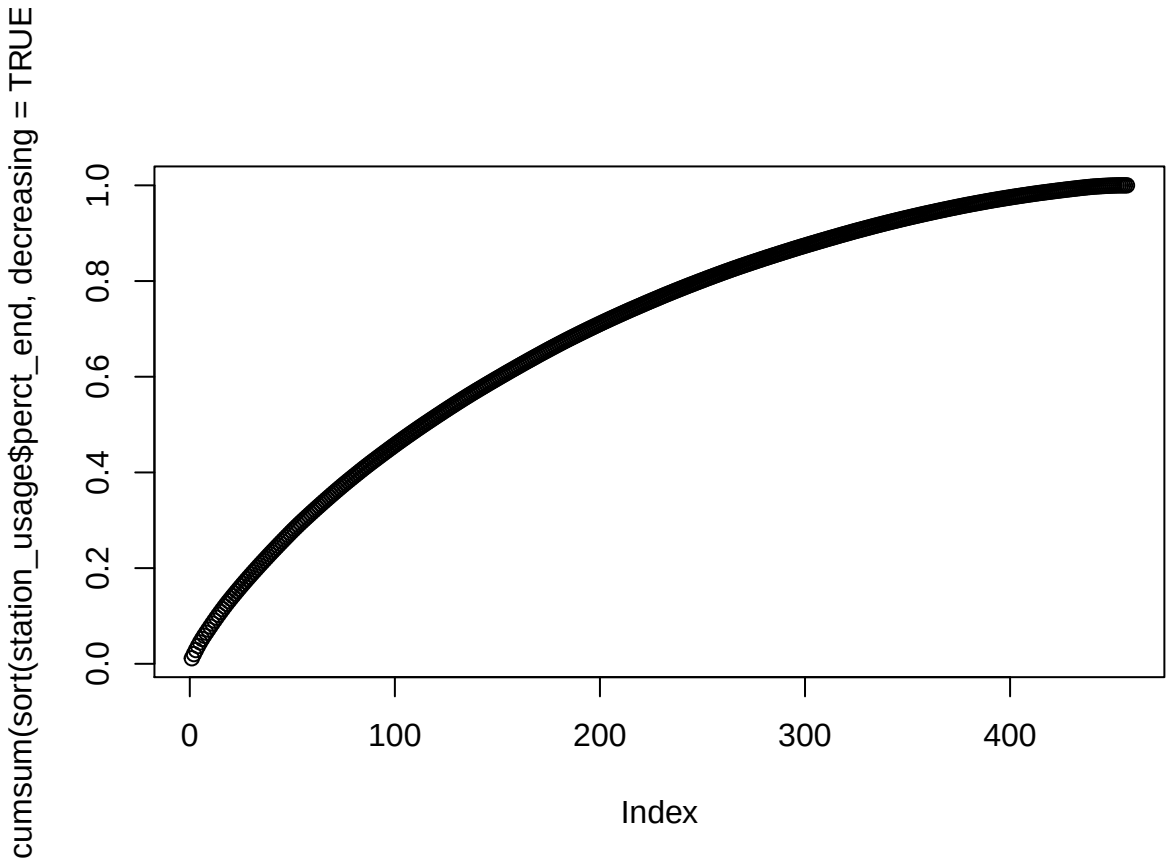

Figure 3.-Cumulative frequency of usage of the ending stations.

The most used stations are:

```
# Printing the top 10 most used arriving stations
( top_10 <- station_usage$st_id[1:10] )
```

```
##  [1] "27"  "266" "1"   "18"  "271" "43"  "21"  "217" "64"  "25"
```

One Hot encoding and Normalization.

One hot encoding over the categorical variables were used and min-max normalization was used due to the Support Vector Machine have troubles with unscaled data and only work with numeric data.

```
# Get the trips that ended on a top 10
bikes3 <- bikes2[ as.character(bikes2$station_end) %in% top_10, ]

# Changing data types
bikes3$station_start <- as.character( bikes3$station_start )
bikes3$station_end <- as.character( bikes3$station_end )
bikes3$exceeding <- as.numeric( bikes3$exceeding)

# Removing target variable
station_end_y <- bikes3$station_end
bikes3 <- select( bikes3, -station_end )

# One hot encoding of categorical variables
var_types <- sapply( bikes3, class)
```

```
bikes3_cat <- bikes3[ var_types == 'character' ]
bikes3_num <- bikes3[ var_types == 'numeric' | var_types == 'integer' ]
# One Hot
dummies <- dummyVars( ~ . , bikes3_cat )
bikes3_1_hot <- predict( dummies, bikes3_cat )

# Normalizing the numeric variable to fall into the 0-1 inteval
normalization_max_min <- function(x){
  ( x - min(x) ) / ( max(x) - min (x) )
}
bikes3_num_norm <- apply( bikes3_num, 2 , function(x) { ( x - min(x) ) / ( max(x) - min (x) ) } )

# Binding everything together
# Y class target X Numeric Normalized and X Categorical 1-hot encoded
bikes_4 <- bind_cols( tibble(station_end_y), as_tibble(bikes3_num_norm), as_tibble(bikes3_1_hot) )
bikes_4$station_end_y <- factor(bikes_4$station_end_y)

# Writting the normalized table
# write_csv(x = bikes_4, path = 'bikes_svm.csv')

bikes_4[1:10,1:10]
```

```
## # A tibble: 10 x 10
##    station_end_y   age trip_time_secon~ leave_hour leave_minute exceeding
##    <fct>         <dbl>            <dbl>      <dbl>        <dbl>     <dbl>
##  1 1              0.26          0.00816      0           0.0847         0
##  2 1              0.24          0.0156       0.217       0.797          0
##  3 266            0.14          0.00809      0.304       0.0508         0
##  4 18             0.13          0.00200      0.304       0.610          0
##  5 43             0.13          0.00538      0.348       0              0
##  6 18             0.14          0.0428       0.348       0.881          1
##  7 64             0.26          0.00387      0.348       0.949          0
##  8 27             0.09          0.0242       0.348       1              0
##  9 18             0.41          0.00316      0.391       0.847          0
## 10 1              0.1           0.0175       0.391       0.881          0
## # ... with 4 more variables: sexF <dbl>, sexM <dbl>, station_start1 <dbl>,
## #   station_start10 <dbl>
```

SVM training

```
# Loading the data
bikes <- read_csv('bikes_svm.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer(),
##   age = col_double(),
##   trip_time_seconds = col_double(),
##   leave_hour = col_double(),
##   leave_minute = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```r
# Selecting the most used starting stations 27 (Reforma) and 271 (Buenavista)
bikes_svm <- filter(bikes, station_end_y == '27' | station_end_y == '271')

# Removing stations withzero sum
bikes_sums <- sapply(bikes_svm[-1], sum)
sum_0_cols <- names( bikes_sums[bikes_sums == 0] )
# Removing
bikes_svm <- bikes_svm[ -match(sum_0_cols, names(bikes_svm)) ]

bikes_svm$station_end_y <- factor(bikes_svm$station_end_y)
# Partitioning the data into Train, Test and Validations Sets
partition_bikes <- partition_data(data_all = bikes_svm, train = 0.60, test = 0.20)
train_bikes <- partition_bikes[[1]]
test_bikes <- partition_bikes[[2]]
val_bikes <- partition_bikes[[3]]

# Checking for 0 sum vectors after the splitting
x <- sapply(train_bikes[-1], sum)
sum_0_cols <- names( x[x == 0] )
colSums(test_bikes[sum_0_cols])
```

```
## station_start1002  station_start313  station_start342  station_start372
##                 0                 2                 0                 0
##  station_start374  station_start375  station_start378  station_start388
##                 1                 0                 0                 1
##  station_start391  station_start401  station_start402  station_start403
##                 1                 5                 1                 0
##  station_start415  station_start422  station_start424  station_start430
##                 0                 0                 1                 0
##  station_start434  station_start436  station_start442
##                 1                 1                 1
```

```r
colSums(val_bikes[sum_0_cols])
```

```
## station_start1002  station_start313  station_start342  station_start372
##                 1                 0                 1                 1
##  station_start374  station_start375  station_start378  station_start388
##                 0                 1                 1                 0
##  station_start391  station_start401  station_start402  station_start403
##                 1                 1                 0                 2
##  station_start415  station_start422  station_start424  station_start430
##                 1                 1                 1                 1
##  station_start434  station_start436  station_start442
##                 1                 0                 0
```

```r
# Removing 0 suming vector in training in all sets
train_bikes <- train_bikes[ -match(sum_0_cols, names(train_bikes)) ]
test_bikes <- test_bikes[ -match(sum_0_cols, names(test_bikes)) ]
val_bikes <- val_bikes[ -match(sum_0_cols, names(val_bikes)) ]
```

Training the model.

```r
# Training the svm
# svm_model <- svm( station_end_y ~ . , data = train_bikes )
# summary(svm_model)
```

```r
# Save an object to a file
# saveRDS(svm_model, file = "svm_model_271.rds")
# Restore the object
svm_model <- readRDS(file = "svm_model_271.rds")
```

Results

Printing the parameters of the model.

```r
svm_model
```

```
##
## Call:
## svm(formula = station_end_y ~ ., data = train_bikes)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.002304147
##
## Number of Support Vectors:  14423
```

Printing the confusion matrix.

```r
predictions <- predict(svm_model, val_bikes)
# Confusion Matrix
confusion <- as.matrix( table(predictions, val_bikes$station_end_y) )
confusion
```

```
##
## predictions   27  271
##         27  3750 1270
##         271  671 1656
```

```r
tp <- confusion[1,1]
tn <- confusion[2,2]
fp <- confusion[1,2]
fn <- confusion[2,1]
```

```r
accuracy <- (tp + tn) / (tp + tn + fp + fn)
# Accuracy of the svm in the validation set
acc_h <- round(accuracy * 100, 2)
```

The accuracy was 73.58.

The accuracy is not great and can probably be further improved by tuning the parameters.

The next following plots are the decision boundaries see through 2 selected dimensions.

```r
plot(svm_model, data = train_bikes, age ~ trip_time_seconds)
```

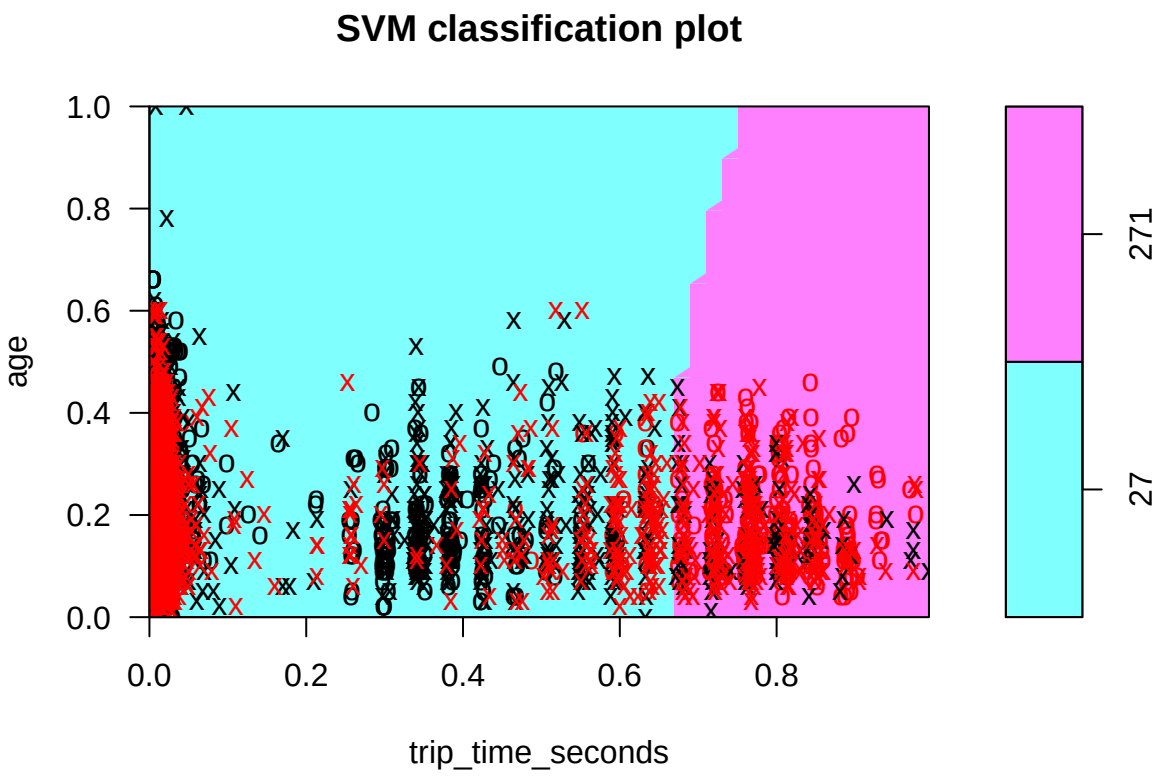

Figure 4.- Decision boundary of age and trip duration.

```
plot(svm_model, data = train_bikes, leave_hour ~ trip_time_seconds)
```

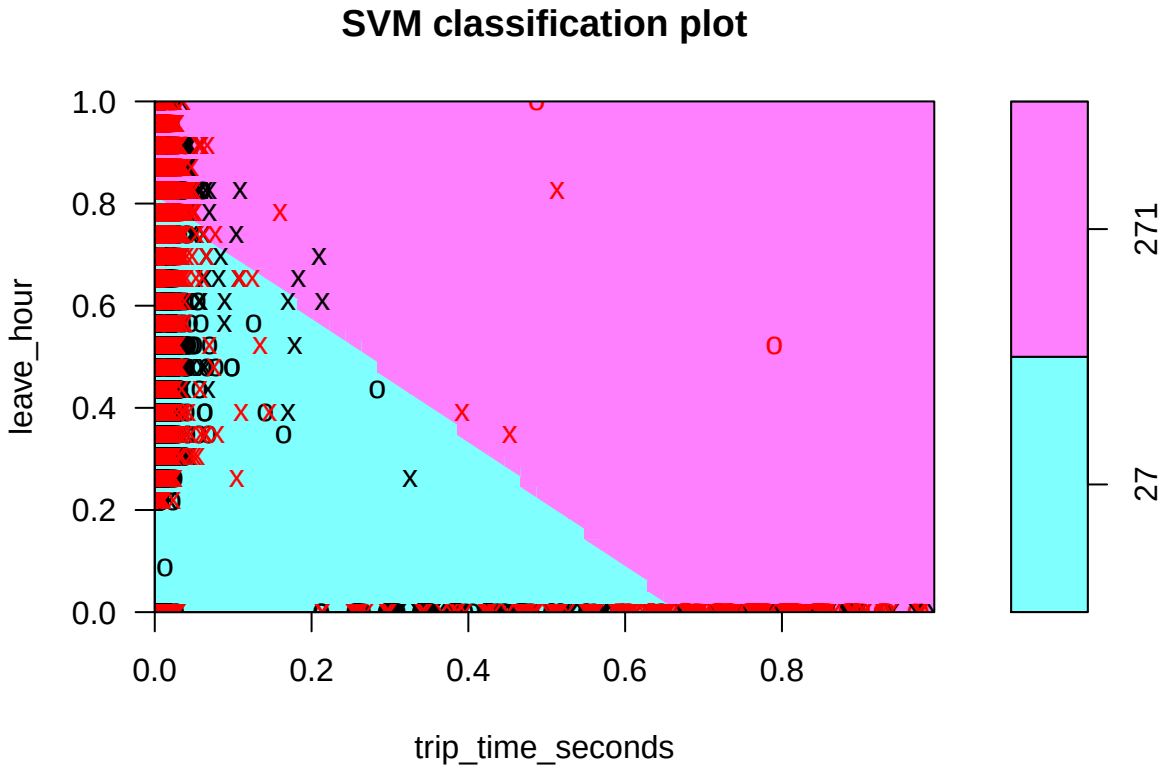

Figure 5.- Decision boundary of hour and trip duration.

```
plot(svm_model, data = train_bikes, leave_hour ~ station_start1)
```

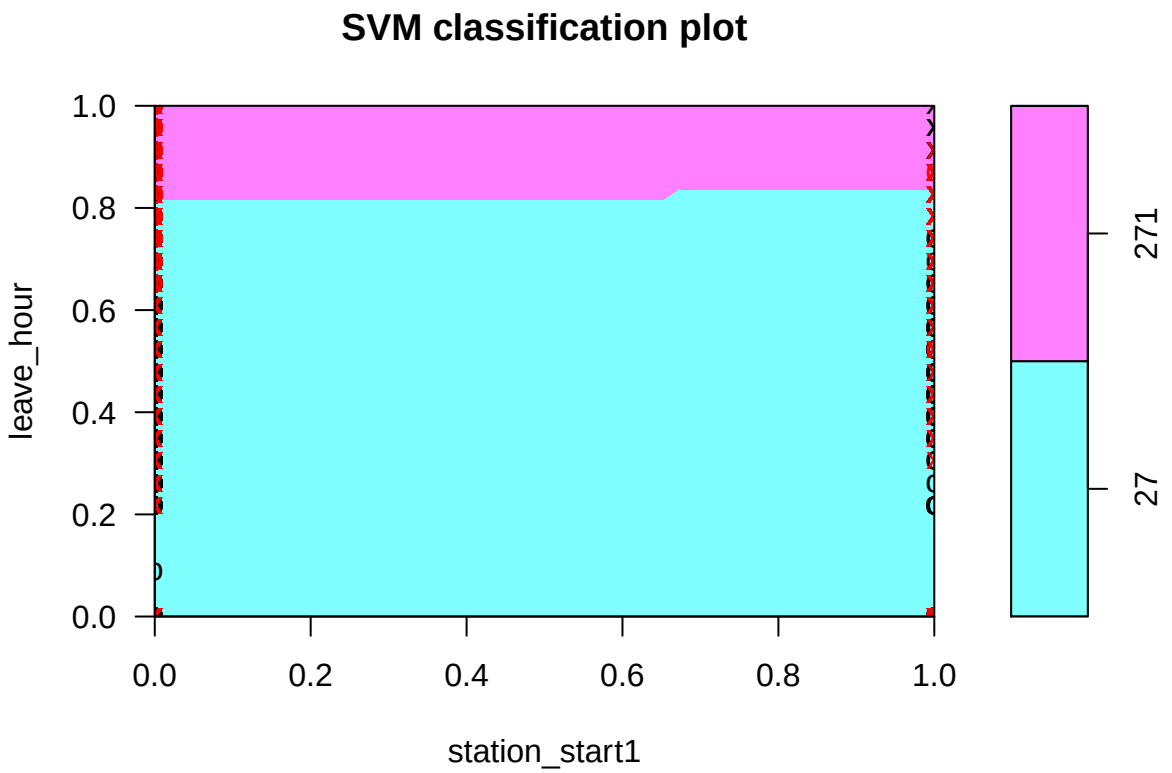

Figure 6.- Decision boundary of hour and starting station 1.

```
plot(svm_model, data = train_bikes, leave_hour ~ station_start1)
```

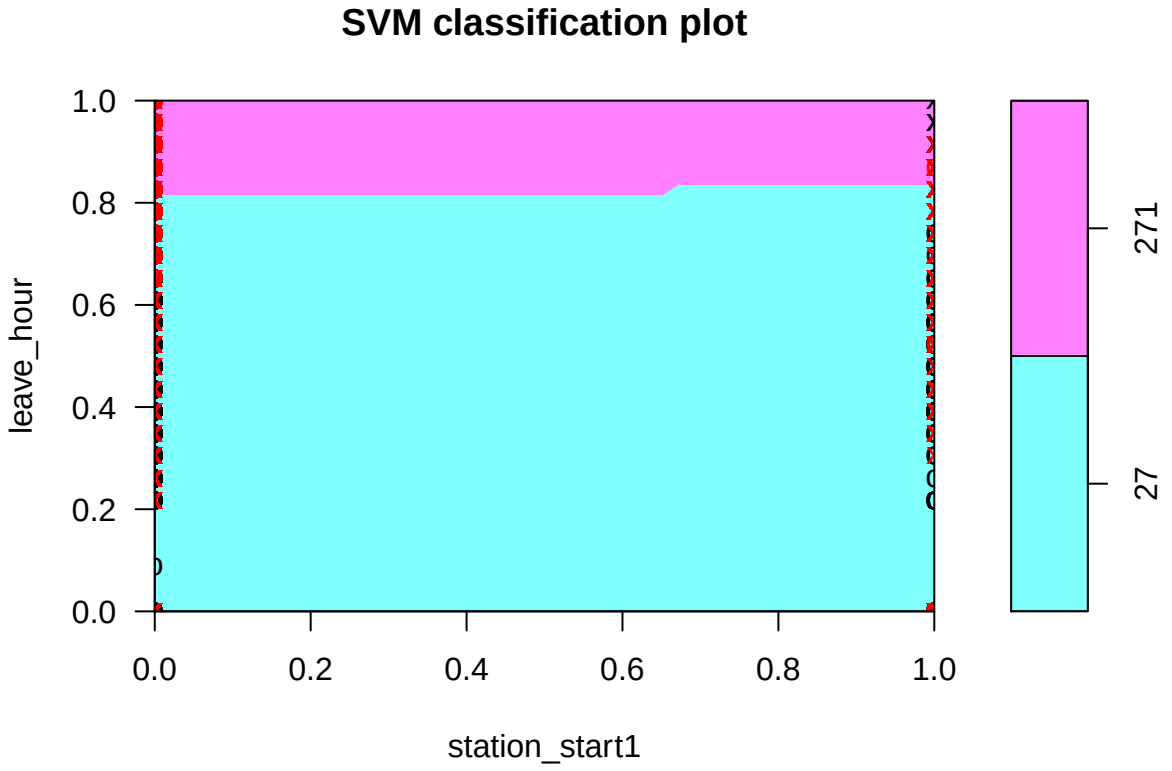

Figure 7.- Decision boundary of hour and starting station 1.

```
plot(svm_model, data = train_bikes, station_start1 ~ station_start2)
```

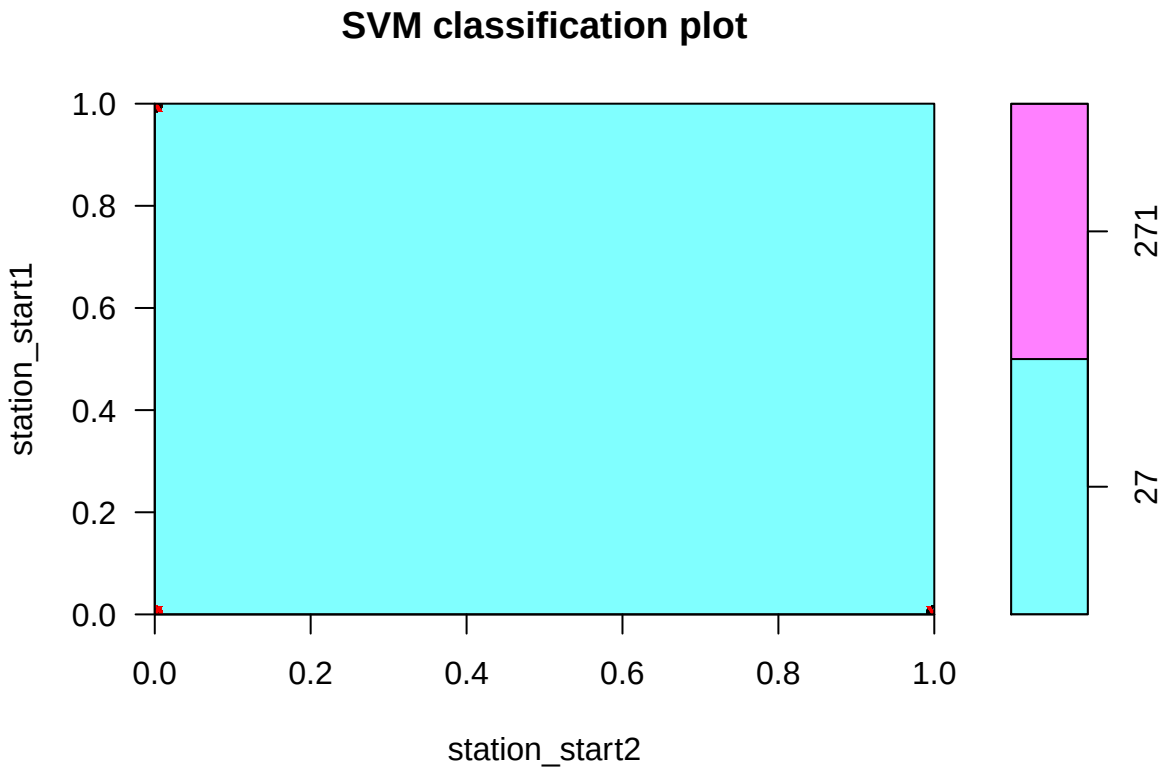

Figure 8.- Decision boundary of starting station 1 and starting station 2.

```
plot(svm_model, data = train_bikes, station_start1 ~ station_start3)
```

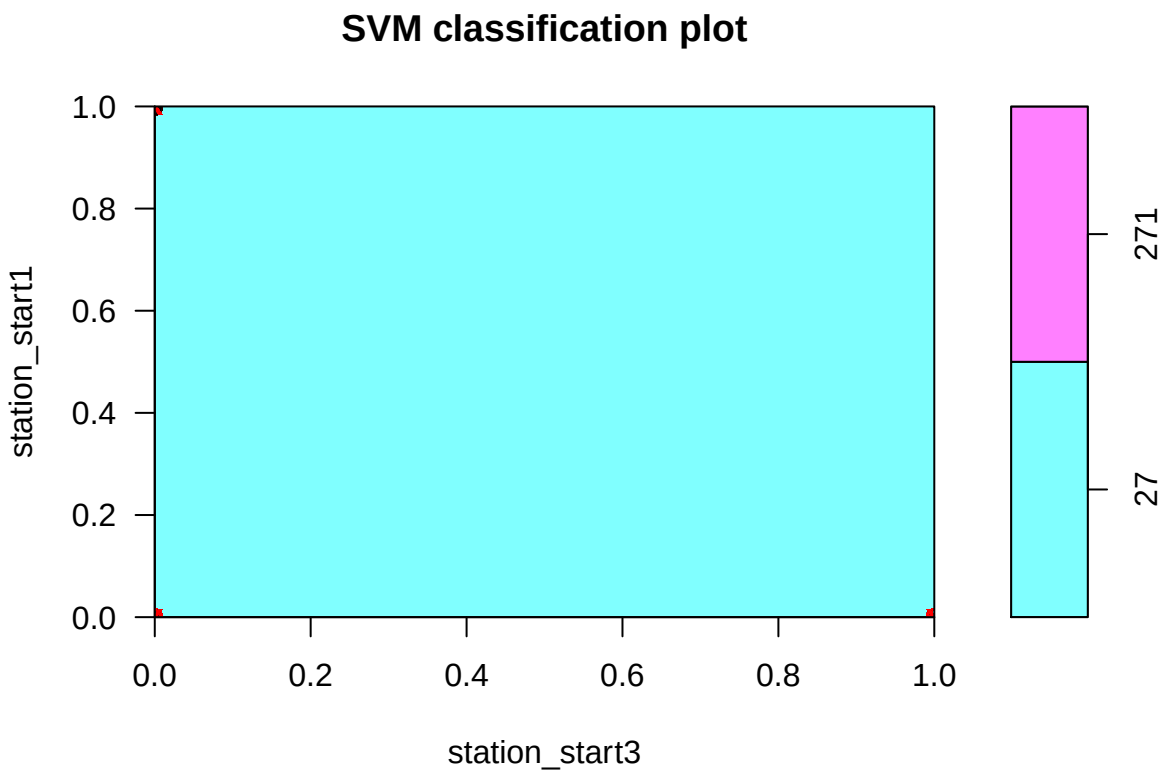

Figure 9.- Decision boundary of starting station 1 and starting station 3.