Scott Alexander – PH125.9x Final Project

Formula One Results Predictions

Overview and Summary

This project is submitted as the final "Create Your Own" assignment for PH125.9x. In this project we look at historical Formula One racing data. The primary hypothesis that will be tested in simply put; is it possible to predict the finishing position of a racer based on the starting position. We look at 345 different races and over 7500 drivers who started a race with dates spanning 1994 to 2008. Data is sourced from the Kaggle Formula One Race Archive (https://www.kaggle.com/cjgdev/formula-1-race-data-19502017/download/5571_8322_bundle_archive.zip).

In addition to answering the question of predicting finishing result based on starting position question, we also spend other analysis of interest regarding why drivers did not finish the race (DNF).

The findings are somewhat conclusive in the fact that there is not much variability in where a racer starts the race and where they finish. This in part could lead to an explanation of why many racing fans complain that modern Formula One races are boring to watch. The race outcomes are all independent from one another. How a driver finishes in one race has no impact where they finish in the next race, etc.

Methods and Analysis

This is a classic prediction problem; Can we predict the value of Y based on the value of X while Y is independent. The method chosen was Support Vector Regression (SVR) over a simple linear regression. SVR is based on the Support Vector Machine (SVM) algorithm. It is a supervised learning model with associated learning algorithms that analyze data used for classification and regression analysis. Because this is not a classification problem we can utilize the SVR functions of SVM by providing 2 discrete axis. SVM will detect this and utilize a regression function instead.

Advantages of SVM include High-Dimensionality, memory-efficiency, and versatility over other methods. Unlike traditional linear regression methods that depend on Gauss-Markov assumptions, SVR does not depend on dependent and independent variables but instead relies on kernel functions. Also, instead of performing detailed analysis on RMSE of a linear regression model to determine the optimum errors permissible, SVR includes a tune function to iterate various levels of errors and costs to determine the optimum model. We then use the best fit model for our prediction.

Instead of covering all of the code that is used to complete the model in this report, I have provided detailed comments in the markdown and R source files. The intent of this report is to focus on the data science methods, analysis, and outcomes.

Our data is somewhat simple – where did a driver start in the race and where did they finish. There are a maximum of 26 drivers per race. The main data source, `F1_Results`, is described as follows:

First we look at the distribution of our training and testing sets to ensure both sets contain a representative view of numbers of racers that finished in each position

TESTING DATASET – 1367 Rows

```
> prop.table(table(testing$Qualifying_position)) * 100

        1         2         3         4         5         6         7         8
6.2179956 5.2670080 4.1697147 5.3401609 4.9012436 4.3891734 4.9012436 4.9012436
        9        10        11        12        13        14        15        16
5.3401609 5.1207023 4.0234089 4.7549378 4.5354792 4.3891734 4.0234089 4.2428676
       17        18        19        20        21        22        23        24
4.9743965 4.3891734 4.5354792 3.3650329 2.5603511 1.2435991 1.0972933 1.0972933
       25
0.2194587
```

TRAINING DATASET – 4108 Rows

```
> prop.table(table(training$Qualifying_position)) * 100

         1          2          3          4          5          6          7
5.06329114 5.28237585 5.52580331 5.18500487 4.81986368 5.18500487 4.79552093
         8          9         10         11         12         13         14
4.86854917 4.38169426 4.69814995 4.72249270 4.40603700 4.43037975 4.72249270
        15         16         17         18         19         20         21
4.06523856 4.35735151 3.89483934 4.33300876 4.11392405 4.18695229 2.14216164
        22         23         24         25         26
2.82375852 0.97370983 0.77896787 0.09737098 0.14605648
```

We can tell that each set is somewhat identical with a random variation in the overall percentages for each position.

We are now ready to perform our model fitting and run our prediction. The basic linear regression is follows the simple model of

$$y = wx + b$$

The SVM attempts to minimize error, individualizing the hyperplane for maximize margin, while tolerating error. Thus, the full model becomes;

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle x_i, x \rangle + b$$

This model is implemented in our code with tuning characteristics. This iterative tuning approach uses 10 epsilon values, 0 to 1 in .1 increments and attempts 10 levels of kernel costs, thus applying 100 combinations to determine the optimum RMSE. Kernel cost can be increased, however, the processing time can become long.

```
model <- tune(svm, Results_position ~ Qualifying_position, data = training,
                   ranges = list(epsilon = seq(0,1,0.1), cost = (1:10)))
```

The model executed for approximately 30 minutes and concluded with the following results. For brevity, I have removed 90 of the 100 permutations. The total values can be observed by running the markdown file included in the submission.

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 epsilon cost
     0.5    1

- best performance: 12.11213

- Detailed performance results:
    epsilon cost    error dispersion
1       0.0    1 12.34669   1.295212
2       0.1    1 12.37414   1.297242
3       0.2    1 12.21991   1.233831
4       0.3    1 12.15960   1.208834
5       0.4    1 12.12323   1.156164
6       0.5    1 12.11213   1.119318
7       0.6    1 12.26596   1.088038
8       0.7    1 12.48627   1.062647
9       0.8    1 12.78104   1.062344
10      0.9    1 13.28471   1.040922
```

Based on tuning the model has determined the best fit for this analysis is

```
Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  1
    epsilon:  0.5


Number of Support Vectors:  1686
```

This produces the lowest error of 12.11 with a dispersion of 1.119 as indicated in the highlighted portion of the grid above. Final predictions are calculated based on an epsilon of .5 with a kernel cost of 1 utilizing the following code.

As required by course instructions we also ran the same data using the basic linear regression model (`lm`) and predictions. This code for this is:

```
lm_model <- lm(Results_position ~ Qualifying_position, data = training)
summary(lm_model)
pred <- round(predict(lm_model, data.frame(Qualifying_position=1:26)), digits =0)
pred <- as.data.frame(pred)
pred$Qualifying_position <- 1:nrow(pred)
```

The results of this compute the following:

```
Residuals:
    Min      1Q  Median      3Q     Max
-12.538  -2.128  -0.329   2.067  16.074

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        2.321893   0.107245   21.65   <2e-16 ***
Qualifying_position 0.600714   0.008513   70.57   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.476 on 4106 degrees of freedom
Multiple R-squared:  0.5481,   Adjusted R-squared:  0.548
F-statistic:  4980 on 1 and 4106 DF,  p-value: < 2.2e-16
```

NOTE: The code to execute the above model is comments out in the submitted R file. One will have to uncomment them to verify the results. The code is not setup to execute both models in one pass.

A comparison of the lm vs. svm methods will be covered in the results section. For the remainder of our analysis we will be using svm exclusively. With the svm fit model we will execute the prediction functions and create a result set.

```
pred <- round(predict(model$best.model, data.frame(Qualifying_position=1:26)), digits =0)
pred <- as.data.frame(pred)
pred$Qualifying_position <- 1:nrow(pred)

TestingOutput <- inner_join(testing, pred, by="Qualifying_position")
FinalResults <- TestingOutput %>% mutate(pred_diff = pred-Results_position) %>%
select(Results_position,pred, pred_diff)
```

Predictions are then joined to the testing test to create the final results vector. The detailed results are included in the project submission in file `F1Output.csv`.

Results

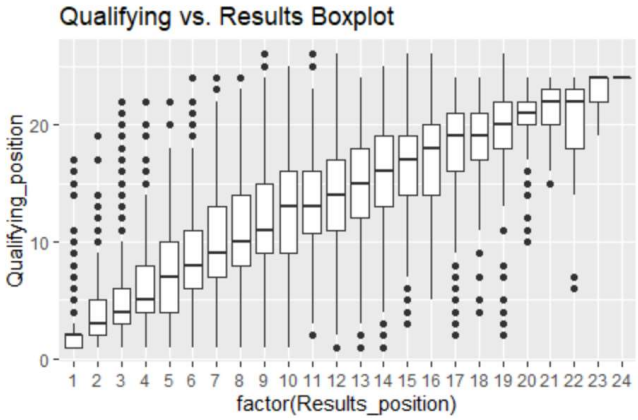

Table 1

The above boxplot shows that there is a somewhat consistent difference between the 25% and 75% quartiles with the mean usually falling somewhere close to the middle, with the exception of the first through fourth finishers. The dispersion between min and max is also lower for the first four finishers. This tends to indicate that there is not much variability between where a driver starts and where they finish if they are in the top four starters.

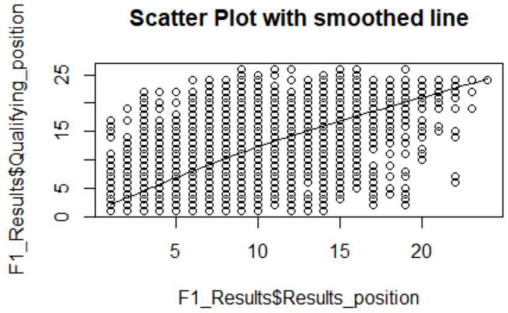

Table 2

The relationship between starting (qualifying) and finishing (results) position is mostly linear with a bend somewhere around 10th finishing position.

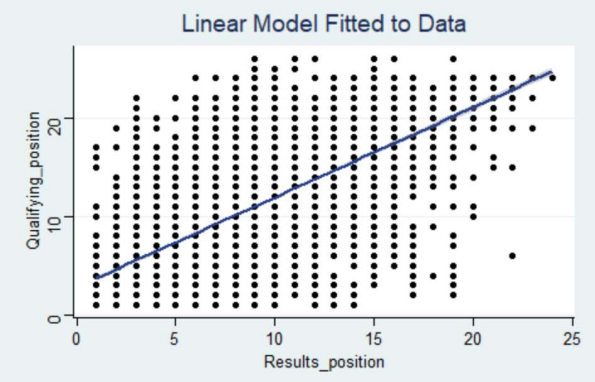

Table 3

The above model has been fitted closer to a more linear relationship using a Loess smoothing method. The gray area surrounding the blue line above is very narrow indicating that the smoothed line is very close to the original regression line.

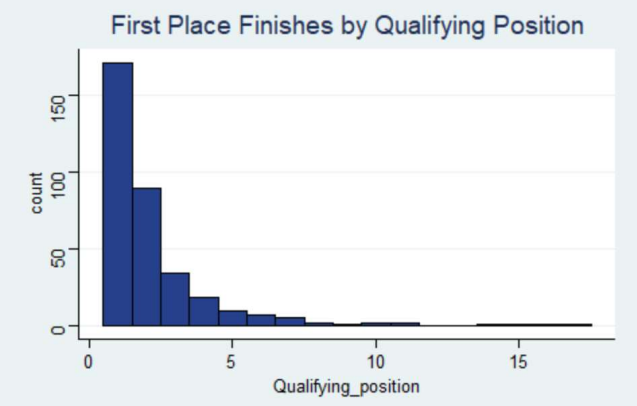

Table 4

This histogram is descriptive in nature and does not use predicted values. Its intent is to create a visual that partially supports my initial hypothesis and help visualize a key factor in my hypothesis. This graph shows what position all race winners (first place) started in. It clearly shows that a majority of racers who win a race started in pole position (the first car to the starting line). In fact, more than twice that of a driver who started in second place. It goes on to show that the chances of a driver who starts lower than fifth is very unlikely to win the race. Furthermore, no driver has ever won a race who started in ninth, twelfth, and thirteenth has never won the race from 1994 to 2008. This supports a common opinion about Formula One racing…that it is very predictable.

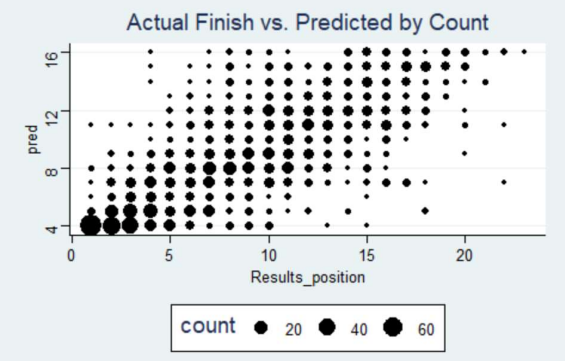

Table 5

We now focus on our predicted finishing places based on our test qualifying data. The above scatterplot that we are able to predict the first three finishing positions somewhat accurately. The points are sized based on the number of the combinations of a prediction and a finish. It is possible to tell that there is a visual pattern that reflects a linear model.

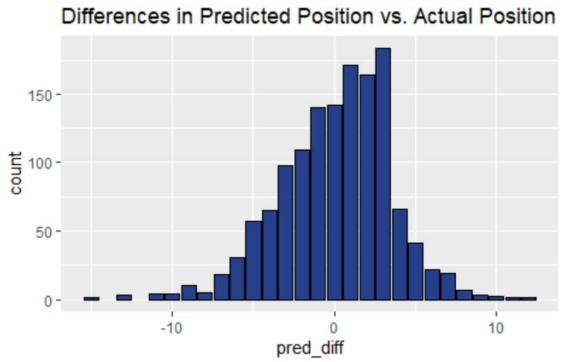

Table 6A - svm

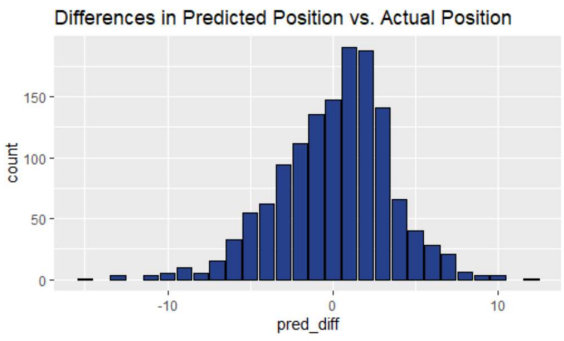

Table 6B - lm

Table 6A was produced utilizing the `svm` model and Table 6B was created with the basic `lm`. This histogram shows the difference in what was predicted versus that actual result. 0 is a perfect prediction – we predicted that if a driver started in X position that they finished in that position. A positive value means they finished X number of places higher than predicted.

Therefore, our model is skewed and tends to underestimate drivers performance. Many drivers finished 1 to 3 places higher than we predicted.

While the `svn` model made fewer prefect predictions than `lm` did (145 vs. 148 respectively), the overall model performance in the accuracy of other top 4 places of the `svm` was preferred over the `lm` model. Therefore, it will serve as the bases for our findings.

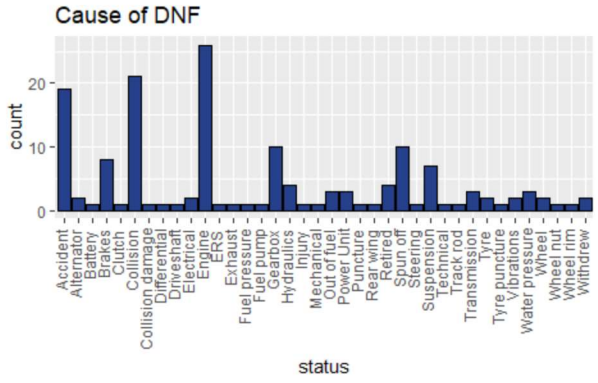

Table 7

This was a secondary question I wanted to answer in my analysis. It does not have a direct impact on the original hypothesis, but is included here as more of a curiosity. This shows the reasons why drivers did not finish a race. There are two categories I can see – those pertaining to driver and the other being mechanical. Top driver errors include collision, accident, and running of the track. Mechanical failures seems to centered around engine, transmission, and brake related issues.

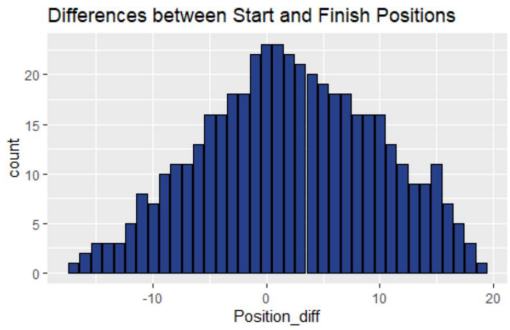

Table 8

This histogram uses actual data and no predictions. This standard distributed data clearly shows that most drivers in Formula One racing finish in the race in the same position they started in. When we compare it to our predicted values in `Table 6` we can see a very similar pattern, although our predicted model is slightly skewed.

Conclusion

To summarize my findings, I determined that predicting finishing positions can greatly depend on starting position and there is a strong correlation. There may be other models that are more accurate and other approaches an experienced data scientist would have taken. I feel that based on my experience in this course I applied the knowledge in the best way I know how to complete this project.

This study was based on my desire to focus on something that interests me (auto racing) and to meet the requirements of the assignment to be original and use publicly available data. While I think I achieved both of my objectives, I think this dataset was a little too linear and made for some less-than-exciting analysis. Again, this supports the popular opinion that Formula One racing is boring. I could extend this model and data for some other studies that may yield more interesting and fruitful results. Some ideas I have are;

- Compare Formula One predictability with other racing series such as NASCAR, Indycar, and some of the grand touring series.
- Add a timeseries analysis to determine if, over time, racing has become more predictable
- Add other variables such as driver and circuit to determine if some drivers or tracks lend them self to more exciting racing