F1 RACE PREDICTION SYSTEM

Term Project

Presented to

Dr. Mark Stamp

Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements of the Class

CS 271

By

AKSHAR PANCHAL AND KALPNIL ANJAN

December 3, 2019

INTRODUCTION

Formula One also called as F1 in short is an international racing sport. It's the highest level of single seat, open wheel and open cockpit professional motor racing. The main objective of Formula One is to determine which driver crosses the finish line first after specified number of laps and pit stops.

Also, F1 being a popular sport there are many bidding activities going on for winners and other qualifying positions in race championships. Hence there are quite few race prediction systems out there.

Our project focuses on designing such a F1 Race Prediction System. We make this prediction based on certain features and parameters which are inferred from previous historical data of different championships. The data makes use of various parameters and different machine learning algorithms for predicting qualifying positions in an upcoming race.

The project aims at predicting driver positions, first 5 positions, first 10 positions as well as podium finish (top 3).

BACKGROUND

Since, the main goal of this project is to predict winning positions of race drivers, we would need substantial information regarding drivers, their past statistical data along with information regarding geographical locations and respective tracks.

As per our project requirement we have extracted data from Kaggle (https://www.kaggle.com/cjgdev/formula-1-race-data-19502017). The data is in the form of .csv files. There are different files like circuits, drivers, pitstops, constructors, races, results, etc. The data which we obtain in these files is unstructured and out of order. Hence, we need to make sure that before we try and run our algorithms, these data should be appropriately pre-processed and converted into one single file which would be useful for having different features during machine learning training and evaluation phases.

- Steps for structuring data:

1. For this unstructured data, we first identify which tables would be beneficial for our model to predict the future outcomes of races and import those.

```
# Importing all Datasets
races_df = pd.read_csv("races.csv", header=[0], encoding="ISO-8859-1")
qualifying_df = pd.read_csv("qualifying.csv", header=[0], encoding="ISO-8859-1")
driver_standings_df = pd.read_csv("driver_standings.csv", header=[0], encoding="ISO-8859-1")
constructor_standings_df = pd.read_csv("constructor_standings.csv", header=[0], encoding="ISO-8859-1")
driver_results_df = pd.read_csv("results.csv", header=[0], encoding="ISO-8859-1")
constructor_results_df = pd.read_csv("constructor_results.csv", header=[0], encoding="ISO-8859-1")
```

2. After we identify key tables, we need to pre-process columns in a way which will help different columns and rows from different tables connect in logical way.

```
races_df.sort_values(by=['Year', 'Round'], inplace=True)
races_df.reset_index(drop=True, inplace=True)
races_df['RaceIndexId'] = races_df.index
races_df = races_df.iloc[:, [8, 0, 1, 2, 3]]
races_df = races_df[races_df['Year'] < 2020]
```

3. We use pandas library in python to read different csv files and then extract columns which we desire.
4. These columns are then checked for discrepancies in data values if any.
5. If discrepancies are found, then those values are either discarded or replaced with default values.
6. We then merge all the columns into one single .csv file which will be used for both training and evaluations. E.g:

```
final_data_set = pd.merge(races_df, qualifying_df_temp, on=['RaceId'], how='left')
beforePreprocessing = final_data_set.shape
final_data_set = final_data_set.dropna(subset = ['DriverId', 'ConstructorId'])
afterPreprocessing = final_data_set.shape
print("Before:", beforePreprocessing,"and After:", afterPreprocessing)
final_data_set.head(3)
```

7. We finally merge all the columns into single file.

```
results = pd.merge(final_data_set, filtered_driver_results, on=['RaceId', 'DriverId', 'ConstructorId'], how='left')
results.head(3)
```

We have generated data file, ‘Data.csv’ which has columns as shown in below snippet.

RaceIndexId	RaceId	Year	Round	CircuitId	DriverId	ConstructorId	QualifyingPosition	GridNumber	DriverPosition	TotalDriverPoints	ConstructorPoints	ConstructorPosition
548	257	1994	1	18	30	22	2	2	1	10	10	1
548	257	1994	1	18	55	6	3	3	3	4	4	3
548	257	1994	1	18	71	3	4	4	2	6	6	2
548	257	1994	1	18	91	15	7	7	6	1	1	6
548	257	1994	1	18	79	25	10	10	5	2	2	5
548	257	1994	1	18	100	33	13	13	9	0	0	9
548	257	1994	1	18	22	17	14	14	4	3	3	4
548	257	1994	1	18	94	18	15	15	8	0	0	8
548	257	1994	1	18	44	27	19	19	11	0	0	10
548	257	1994	1	18	65	32	21	21	7	0	0	7
548	257	1994	1	18	83	32	24	24	10	0	0	7
548	257	1994	1	18	101	31	26	26	12	0	0	11
549	258	1994	2	28	30	22	2	2	1	20	20	1
549	258	1994	2	28	77	6	5	5	2	6	10	2
549	258	1994	2	28	22	17	8	8	3	7	7	3
549	258	1994	2	28	104	29	9	9	4	3	3	5
549	258	1994	2	28	49	15	11	11	5	2	3	6
549	258	1994	2	28	100	33	16	16	6	1	1	8
549	258	1994	2	28	103	27	18	18	10	0	0	11
549	258	1994	2	28	44	27	22	22	9	0	0	11
549	258	1994	2	28	65	32	23	23	7	0	0	9
549	258	1994	2	28	83	32	24	24	8	0	0	9
549	258	1994	2	28	107	31	26	26	11	0	0	12
550	259	1994	3	21	30	22	2	2	1	30	30	1
550	259	1994	3	21	71	3	4	4	6	7	7	3
550	259	1994	3	21	78	6	6	6	2	6	16	2
550	259	1994	3	21	49	15	7	7	7	2	6	5
550	259	1994	3	21	57	1	8	8	3	4	4	6
550	259	1994	3	21	79	25	9	9	5	4	4	7
550	259	1994	3	21	91	15	10	10	4	4	6	5
550	259	1994	3	21	87	25	12	12	9	0	4	7

Different Machine Learning techniques have been applied on this data set to generate accurate predictions as far as possible. Some of the methods which we have implemented are KNN, MLP and Logistic Regression. We will analyze performance of all the models and compare their results in the end.

Due to high variation of data available, we have trained above models using our pre-processed data and then derived results from the same.

IMPLEMENTATION

The implementation part will focus on three different techniques to predict the outcomes of race.

- **KNN (K Nearest Neighbor)**

  KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are nearer to each other. KNN captures the idea of similarity with mathematics – calculating the distance between two points.

  ## The KNN Algorithm for F1 Race Prediction:

1. The first step here is to load the data i.e. Data.csv file which we prepared after pre-processing.
2. We initialize value of K at the start as 1. We need to try and experiment with different values of K as we keep on running the model. Different values of K might fetch us different results. In our case we go from K = 2 to K = 30.
3. We perform 5 fold cross validation on our data for below steps to determine the best value of K along with accuracy.
4. For each example in the data file we need to do following:
   - Calculate distance between current example and query example from the data.
   - Add the distance and the index of example to an ordered collection.

5. Sort the ordered collection of indices from smallest to largest by distances.

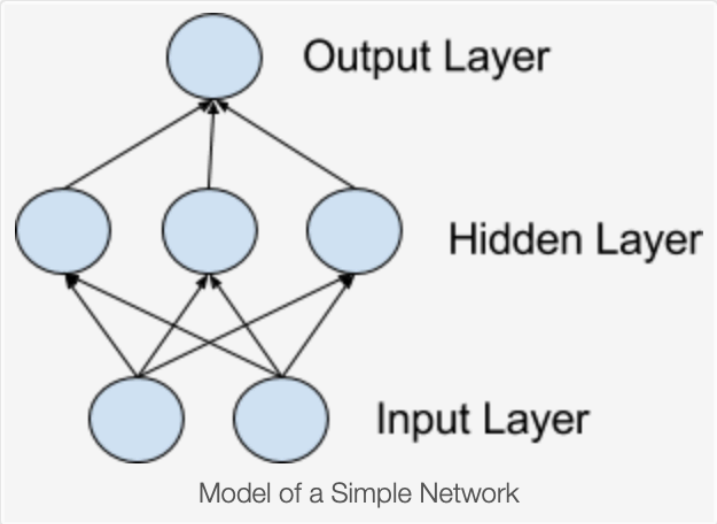
6. Pick the first K entries from sorted collection.
7. Get the labels of selected K entries.
8. Return the mode of K labels.
   For our F1 Race Data we have experimented with different values of K and found substantial increase in accuracy for predicting the qualifying positions.

   How to choose right value for K:

1. As we decrease our value to K = 1, our predictions become less stable. Imagine driver position point being close to one podium finish point and surrounded by Top 10 finishes. It might get incorrectly classified as Podium Finish just because value of K = 1, instead of correctly classifying as Top 10 finishes.
2. Inversely as we increase the value of K, our predictions become more stable due to majority voting and thus we are likely to make more accurate predictions up to a certain point. After that we begin to witness errors. At this point of time we know that we have increased the value of K too far.
3. Thus, we need to make sure that while choosing the right value of K we look for increasing accuracy as well as minimizing the errors.

- **MLP (Multi Level Perceptron)**

  MLPs are kind of classical type of neural networks which are part of Artificial Neural Networks (ANNs). They are comprised of one or more layers of neurons. Data is fed to the input layer, there may be one or more hidden layers providing levels of abstraction and predictions are made on the outer layer also called the visible layer.



Model of a Simple Network

MLPs are suitable for classification prediction problems where inputs are assigned a class or label. They are also suitable for regression prediction problems where a real-valued quantity is predicted given a set of inputs. Data is often provided in a tabular format, such as you would see in a CSV file or a spreadsheet.

Due to above reason, we found MLP to be perfect candidate for implementing F1 Race Predictions.

RESULTS

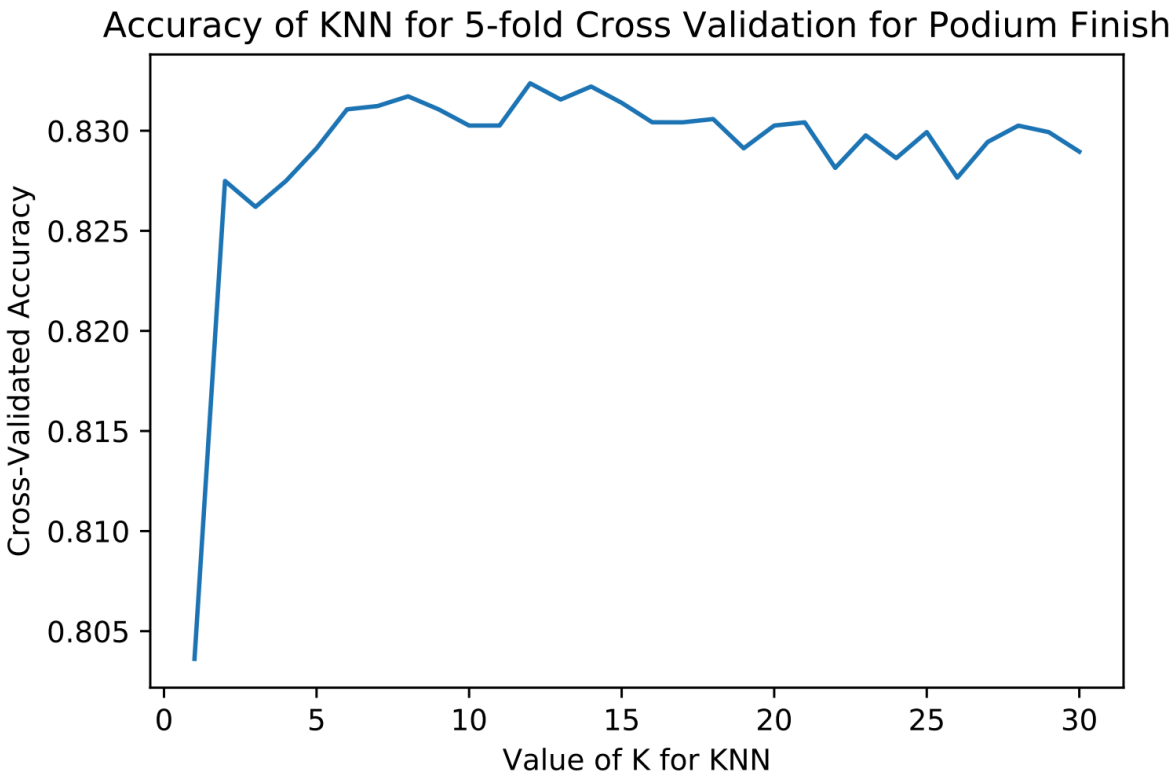
- **KNN**

For KNN we experimented with different values of K starting from K = 1 till 30. We get varied accuracy based on different values of K.

We have predicted 4 different results namely: Driver Positions, Top 10 finishes, Top 5 finishes and Podium Finish. The graph below shows how the accuracy varies with K.

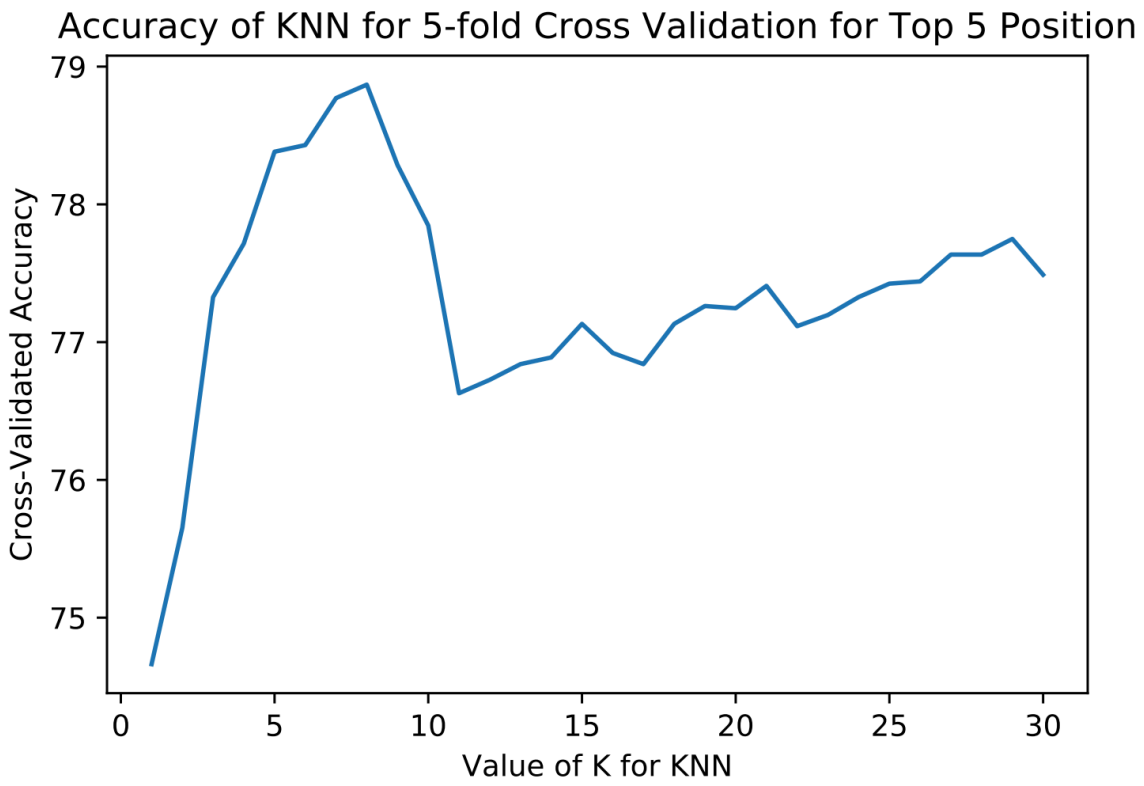
For Podium Finish:

5-fold Cross Validation(mean): 82.903
5-fold Cross Validation(max): 83.237
Best value of K for KNN: 27



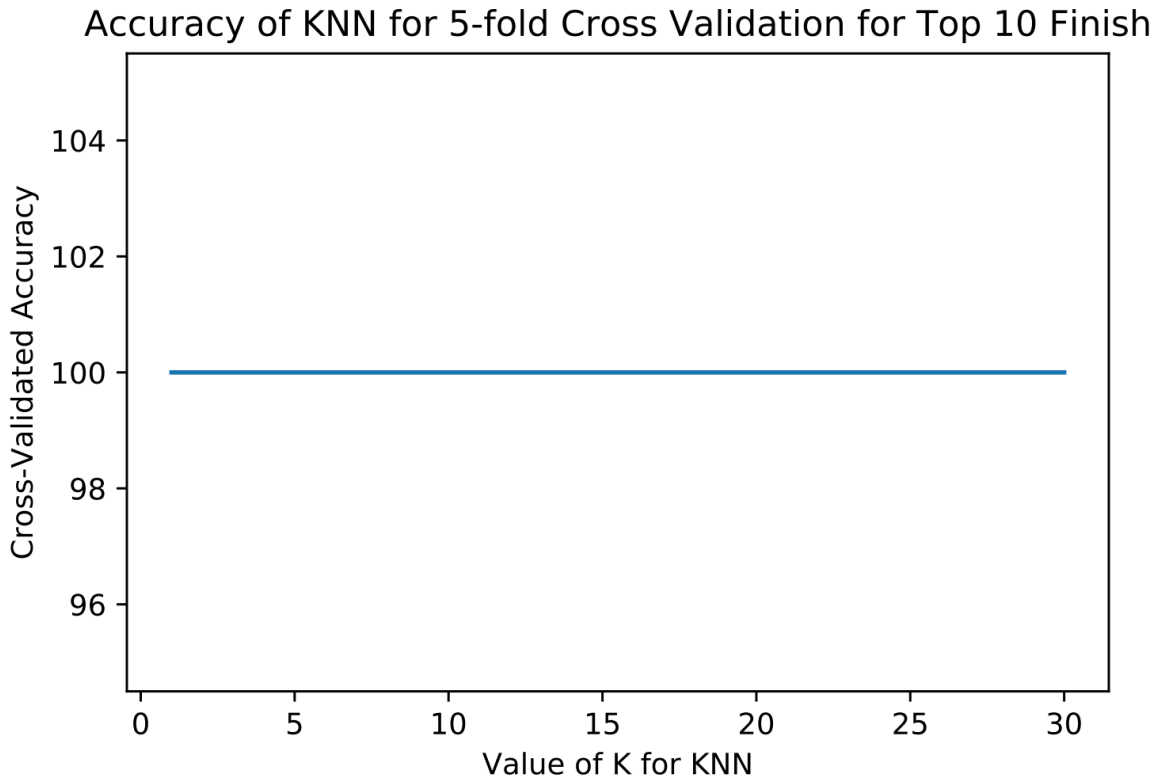
For Top 5 Finish

5-fold Cross Validation(mean): 77.0
5-fold Cross Validation(max): 78.869
Best value of K for KNN: 15



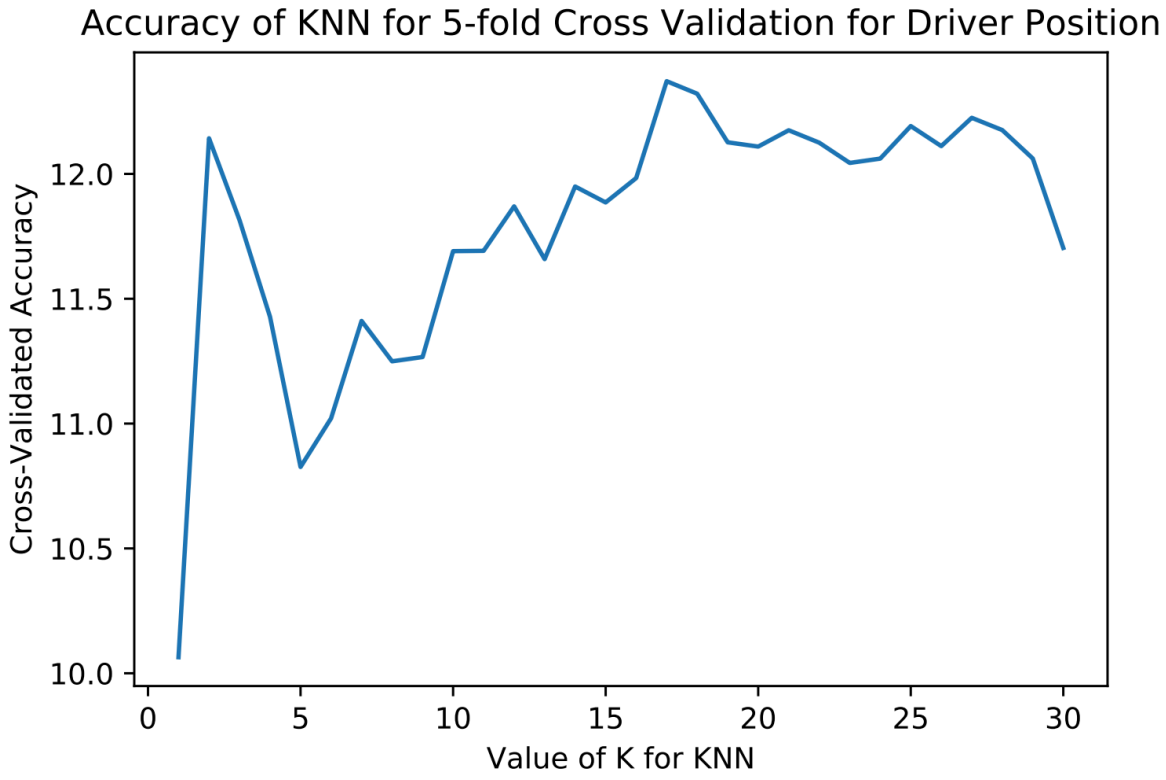
For Top 10 Finish

5-fold Cross Validation(mean): 100.0
5-fold Cross Validation(max): 100.0
Best value of K for KNN: 1



For Driver Position

5-fold Cross Validation(mean): 12.0
5-fold Cross Validation(max): 12.371
Best value of K for KNN: 15



Below is the summary of the results obtained after testing the KNN model:

	Best K value	Max Cross Val Score	Mean Cross Val Score
Driver Position	15	12.371	12
Top 10 Finish	1	100	100
Top 5 Finish	15	78.869	77
Podium Finish	27	83.237	82.903

- **MLP (Multi Level Perceptron)**

  For MLP we trained the model using below parameters:
  Activation Function: Relu
  Solver for back propagation: Adams
  Max Iterations: 400
  Hidden layers size: (100,50,50)

We obtained following results using MLP for our race predictions:

	Train Accuracy	Test Accuracy
Driver Position	18.95	15.02
Top 10 Finish	100	100
Top 5 Finish	86.23	84.9
Podium Finish	89.38	90.1

Comparison of both approaches: KNN and MLP

Both KNN and MLP have fair scoring in predicting race outcomes. They do differ by small amount, but the difference isn't significant enough.

Below graph gives a gist of how both approaches compare to each other