Ranking Prediction for Indianapolis 500 Race

Yashaswini Dhatrika
School of Informatics, Computing and Engineering
ydhatrik@iu.edu

Utkarsh Kumar
School of Informatics, Computing and Engineering
utkumar@iu.edu

ABSTRACT

While being one of the most challenging problems, forecasting and predicting ranking is being explored and has been gaining popularity with availability of more data and improved prediction techniques in real time. We present the ranking prediction algorithms which predicts the leading car in Indianapolis 500 race which is held every year by using 7 years of data from IndyStats website. Our approach is to extract and transform data to the desirable form, then perform exploratory analysis in order to understand the key features for the ranking and then apply different algorithms (Linear, Bagging, Boosting and Deep Learning Models(LSTM, Seq2Seq, Seq2Seq with Attention) models for predicting the rank. As the last step of process the comparative study is performed to analysis the pros /cons of each model deployed

Keywords : Rank Forecast, Rank Prediction, Indianapolis 500, RNN

Figure 1: 103rd Indianapolis 500 at Indianapolis Motor Speedway on May 26, 2019 in Indianapolis, *b*y Clive Rose/Getty Images.

1 INTRODUCTION

Advances in Machine Learning and Deep Learning today have paved way for solving more and more problems. One such challenge is predicting the rank of cars in races. This is particularly a difficult problem because of the uncertainty involved and interplay of a large number of factor involved. However, with the availability of large amount of data from sensors and other sources, we are optimistic about the possibility of finding some patterns that affect the ranking more than others. In this view, we are working on forecasting ranking and studying variables involved of Indianapolis 500 race.The race involves around 33 cars which compete for 200 laps for across 7 years (2013-2019) .Numerous sensors collect data in real time such as vehicle speed,engine speed, throttle, lap times, etc. The size of this data could be between 500 MB to 2 GB.In this project, the main objective is to analyze the data given,clean it and build model that can forecast ranking of cars. We have performed the various steps like Data Extraction, Data Preprocessing ,Exploratory Data Analysis to get more understanding of the data and applying a various models (Linear,Bagging and Boosting, Deep Learning Models) to accurately predict the rankings.

2 RELATED WORK

Unfortunately, till now there has not been any substantial work in the field of ranking prediction for car races but we have managed to find some similar work in other fields such as Harness Racing. There are other resources available too such as the article Forecasting F1 Race Outcomes by Chris Tucker which talks about predicting the outcome for the 2015 season.

The articles on Horse Racing ranking/outcome predictions involve a range of methods. Papers [1] and [2] talk about predicting rank based on probability estimation models. Machine learning methods such as Support Vector Machines have been explored in the paper [3]. More recently, neural networks and deep learning have also been utilized in predicting Horse Race outcomes and have shows very promising results. Papers [4] and [8], using DNN have been able to achieve accuracy of 74% and 77%, respectively.

All these studies and results are exciting and useful as they provide direction and basis for our research. Figure 1.

3 METHODOLOGY

The steps followed for the lead rank prediction are data extraction, data pre-processing, exploratory data analysis and application of various models like Linear Models(ARIMA), Random Forest, Gradient Boosting Regression, Deep Learning Models(LSTM, Encoder-Decoder with attention).And in the final step the models are compared based on various factors like accuracy, stability, time complexity etc.. The following sections discuss each of these in detail.

3.1 Data and Prepossessing

For our initial analysis and Base line Model we have used the Completed lap result records from the log data which contains all the Multi-Loop Protocol and Results Protocol data records. We filtered the completed lap result data records based on "$C" identifier. As can be seen from the Figure 2, many fields in the data are in Hexadecimal form including date/time, ranks and counts. We had to convert each of these hexadecimal values into respective formats using python. Finally, we also removed duplicate rows.

$C – Completed lap results

Fieldname	Data description	Comments
Rank	1 – FFFF	
Car number	characters	4 characters maximum.
Unique identification	0 – FFFFFFFF	This will be the transponder number.
Completed laps	0 – FFFF	Number of completed laps.
Elapsed time	0 – FFFFFFFF	Elapsed Time in ten thousandths sec (hex chars)
Last laptime	0 – FFFFFFFF	Time in ten thousandths sec (hex chars)
Lap status	T, P	Indicates where the lap was completed T = Track, P = Pit lane.
Fastest laptime	0 – FFFFFFFF	Time in ten thousandths sec (hex chars)
Fastest lap	0 – FFFF	
Time behind leader	0 – FFFFFFFF	Time in ten thousandths sec (hex chars)
Laps behind leader	0 – FFFF	
Time behind prec	0 – FFFFFFFF	Time in ten thousandths sec (hex chars) behind preceding car.
Laps behind prec	0 – FFFF	Laps behind preceding car.
Overall rank	0 – FFFF	Rank of car in all sessions of this type.
Overall best laptime	0 – FFFFFFFF	Time in ten thousandths sec (hex chars)
Current status	characters	"Active" or text from comment field from operator edit competitor option. Typically used for reason out of race. Examples: DNQ, DNS, DNF, Contact, Mechanical, Garage, etc
Track Status	G, Y, R, W, K, U	G = Green Y = Yellow R = Red W = White K = Checkered U = Unflagged (warm up)
Pit stop count	0 – FFFF	Number of pitstops in this session
Last pitted lap	0 – FFFF	Lap number the car last pitted.

Figure 2: Initial Data -Completed Lap Results

But Later we have focused mainly on using the Completed lap results data an pit stop data extracted from the Stats webpage on IndyCar website(https://www.indycar.com/Stats) for 2013-2019 years.The raw data(2013-2019) obtained is shown in the Figure 4.. Some transformation are performed to this data. The following are the steps:

(1) the pdf file(2013-2019) is converted to excel file.
(2) From the excel selected rows and columns are selected and combined with last pit stop data to extract time passed since last pit stop.
(3) Then for each car starting or lap time is calculated using linear interpolation method
(4) Then the lap time data points are converted into lap distance using the Figure 3 interpolation method

Convert **Lap Time** to Time Series **Lap Distance**

Interpolation

Lap Distance[T] = Lap Length * (T-T0)/(Lap Time)

Figure 3: Lap Distance Interpolation

(5) the final data contains columns of time lap distance and last pit stop for all the 33 cars. So in totality there are 33*2 columns.The final data set is showed in Figure 5

Section Data for Car 10 - Rosenqvist, Felix (R)

Lap	T/S	SF to T1	T1 to SS1	SS1 to T2	T2 to BS	BS to T3	T3 to SS2	SS2 to T4	T4 to FS	FS to SF	Lap	PI to PO	PO to SF	SF to PI
1	T	3.2704	5.9560	5.7572	5.3118	5.0240	5.4565	5.4848	3.3339	3.7904	43.3850			
	S	205.980	188.885	195.407	211.793	223.925	206.176	205.112	215.963	225.930	207.445			
2	T	3.0764	5.7505	5.5282	5.1801	4.9504	5.4058	5.4965	3.3600	3.8001	42.5480			
	S	218.969	195.635	203.502	217.177	227.254	208.110	204.676	214.286	225.353	211.526			
3	T	3.0482	5.8369	5.7025	5.1994	5.0383	5.4331	5.3519	3.2862	3.7295	42.6260			
	S	220.995	192.739	197.282	216.371	223.290	207.064	210.206	219.098	229.619	211.139			
4	T	2.9617	5.5958	5.6575	5.2343	4.9806	5.4991	5.6077	3.3694	3.7830	42.6891			
	S	227.449	201.044	198.851	214.928	225.876	204.579	200.617	213.688	226.372	210.827			
5	T	3.0111	5.4726	5.4508	5.1070	4.9541	5.4178	5.5734	3.3955	4.5735	42.9558			
	S	223.718	205.570	206.392	220.286	227.085	207.649	201.852	212.045	187.245	209.518			
6	T	3.9723	7.6944	7.8448	7.6810	7.5965	9.3029	9.6416	6.2027	6.5707	66.5069			
	S	169.583	146.210	143.407	146.465	148.095	120.930	116.682	116.078	130.331	135.324			
7	T	6.2587	12.6713	11.5497	10.7327	11.2945	10.9891	10.8217	8.8266	12.7803	95.9246			
	S	107.632	88.783	97.405	104.820	99.606	102.374	103.958	81.572	67.007	93.824			
8	T	8.5855	13.5646	13.0214	12.5268	12.2996	12.7533	10.6426			115.2353	33.6886		92.9107
	S	78.462	82.936	86.396	89.807	91.466	88.212	105.707			78.101	41.429		89.121
9	T				7.8528	12.6301	10.8570	9.9696	8.9805	11.1333	90.5290		79.1650	
	S				143.261	89.073	103.620	112.843	80.174	76.919	99.416		105.177	
10	T	7.3402	11.6981	9.7207	8.7959	9.3156	9.2201	6.7917	3.7364	4.0363	70.6550			
	S	91.774	96.169	115.732	127.900	120.765	122.016	165.643	192.699	212.166	127.380			
11	T	3.1678	6.1543	6.0648	5.4279	5.1470	5.8231	5.5542	3.3507	3.9094	44.5992			
	S	212.651	182.799	185.497	207.262	218.574	193.196	202.549	214.880	219.052	201.797			
12	T	3.0204	5.4819	5.4558	5.1370	5.1849	5.5183	5.3810	3.2985	3.7656	42.2434			
	S	223.029	205.221	206.203	218.999	216.976	203.867	209.069	218.281	227.418	213.051			
13	T	2.9862	5.3925	5.3640	5.0982	5.0034	5.6074	5.4109	3.3062	3.7480	41.9168			
	S	225.583	208.623	209.732	220.666	224.847	200.628	207.914	217.773	228.485	214.711			

Figure 4: Raw Data obtained from the IndyStats website

Time Lap distance Time passed since last pitstop

3070	68.29461	1056.296	68.79213	1379.237
3071	68.31899	1057.296	68.81112	1380.237
3072	68.34337	1058.296	68.83011	1381.237
3073	68.36774	1059.296	68.8491	1382.237
3074	68.39212	1060.296	68.86809	1383.237
3075	68.4165	1061.296	68.88708	1384.237
3076	68.44088	1062.296	68.90607	1385.237
3077	68.46526	1063.296	68.92506	1386.237
3078	68.48964	1064.296	68.94405	1387.237
3079	68.51401	1065.296	68.96304	1388.237
3080	68.53839	1066.296	68.98203	1389.237
3081	68.56277	1067.296	69.00074	0.0536
3082	68.58715	1068.296	69.01461	1.0536
3083	68.61153	1069.296	69.02848	2.0536

Figure 5: Lap Distance Interpolation

We had to extract the Completed lap result records from the log data which contains all the Multi-Loop Protocol and Results Protocol data records. We filtered the completed lap result data records based on "$C" identifier. As can be seen from the Figure 2, many fields in the data are in Hexadecimal form including date/time, ranks and counts. We had to convert each of these hexadecimal values into respective formats using python. Finally, we also removed duplicate rows.

3.2 Exploratory Data Analysis

We used Python and Tableau for doing the exploratory data analysis on completed lap result data records. We focused mainly on understanding pitstops data, time behind leader and laptimes in reference to final rank outcome. As we considered the final rank, we ignored the players who did not complete the race. Below is the summary of graphs that we plotted:

(1) Percentage of number of pitstops taken to understand the distribution of pitstops among the players who completed the race.

(2) Number of pitstops taken by each player who completed the race and their corresponding rank to learn about their correlation.
(3) Average time behind leader for each player over 200 laps who completed the race and their corresponding rank to learn about their correlation.
(4) Starting position for each player who completed the race and their corresponding rank to learn about their correlation.
(5) Distribution of lap times for each player who completed the race and those who didn't to recognize patterns and strategies.

This shed light on some interesting patterns in the data which is discussed elaborately in the results section.

3.3 Baseline Model

3.3.1 Overview of ARIMA Time Series Analysis. ARIMA processes are a class of stochastic processes used to analyze time series. The application of the ARIMA methodology for the study of time series analysis is due to Box and Jenkins Box and Jenkins in 1970 introduced the ARIMA model. It also referred to as Box-Jenkins methodology composed of set of activities for identifying, estimating and diagnosing ARIMA models with time series data. The model is most prominent methods in financial forecasting [7] [6]. ARIMA models have shown efficient capability to generate short-term forecasts. It constantly outperformed complex structural models in short-term prediction[5]. In ARIMA model, the future value of a variable is a linear combination of past values and past errors, expressed as follows:

$$Y_t = \phi_0+\phi_1 Y_{t-1}+\phi_2 Y_{t-2}+\ldots+\phi_p Y_{t-p}+\varepsilon_t-\theta_1\varepsilon_{t-2}-\ldots-\theta_q\varepsilon_{t-q} \quad (1)$$

Y_t is the actual value and $\mathcal{E}_t$ is the random error at t, ϕ_i and θ_j are the coefficients, p and q are integers that are often referred to as autoregressive and moving average, respectively.

3.3.2 Implementation.

(1) Lap Time is predicted using the ARIMA model for each car.
(2) While predicting lap time for a car, each car is considered independently from another.
(3) As there 200 laps in total, the predictions are done in windows. Windows considered are after 20,50,100,150,199.
(4) Python's autoarima() function is used which optimizes the parameter based on the Akaike information criterion (AIC) and mean squared error(MSE).
(5) After each window prediction using ARIMA for all the cars, based on the predicted lap-time, rank is obtained and compared to the actual rank output
(6) Finally the predicted and actual rank are plotted, to see how closely they are related

3.4 Random Forest Regression

3.4.1 Overview. Random forest is a Supervised Learning algorithm which uses ensemble learning method for classification and regression. Random forest is a bagging technique and not a boosting technique. The trees in random forests are run in parallel. There is no interaction between these trees while building the trees. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

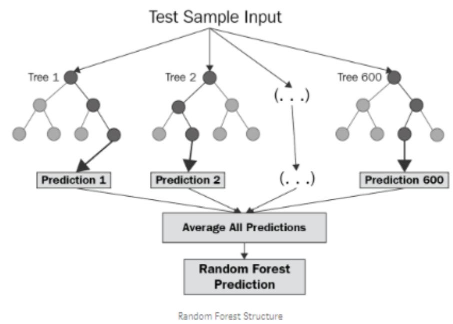

Figure 6: Random Forest

A random forest is a meta-estimator (i.e. it combines the result of multiple predictions) which aggregates many decision trees, with some helpful modifications: The number of features that can be split on at each node is limited to some percentage of the total (which is known as the hyper-parameter). This ensures that the ensemble model does not rely too heavily on any individual feature, and makes fair use of all potentially predictive features. Each tree draws a random sample from the original data set when generating its splits, adding a further element of randomness that prevents over-fitting. These decision tree classifiers can be aggregated into a random forest ensemble which combines their input.

3.4.2 Implementation.

(1) Creating of labels for each data points (1,2,3…33).
(2) Creating lag variable for the lap distance (lag10, lag 20…lag 200)
(3) Normalizing the data
(4) Split of train and test. (70 percent of first-time stamp of each year for training and remaining data points are for validation.)
(5) Using scikit learn library for implementation of Random Forest Regressor.K-fold valid is performed using Grid search to hyper tune the Hyper Parameters like number of decision trees, disorder calculating and depth of the tree.
(6) Prediction is done on the test data using the optimized hyper-parameters

3.5 Gradient Booster Regression

3.5.1 Overview . Gradient boosting involves three elements:

(1) A loss function to be optimized.
(2) A weak learner to make predictions.
(3) An additive model to add weak learners to minimize the loss function

1. Loss Function The loss function used depends on the type of problem being solved. It must be differentiable, but many standard loss functions are supported and you can define your own. For example, regression may use a squared error and classification may

use logarithmic loss. A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

2. Weak Learner: Decision trees are used as the weak learner in gradient boosting. Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and "correct" the residuals in the predictions.Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss.Initially, such as in the case of AdaBoost, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels.It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes.This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.

3. Additive Model: Trees are added one at a time, and existing trees in the model are not changed.A gradient descent procedure is used to minimize the loss when adding trees.Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss. The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model.A fixed number of trees are added or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

3.5.2 Implementation .

(1) Creating of labels for each data points (1,2,3…33).
(2) Creating lag variable for the lap distance (lag10, lag 20…lag 200)
(3) Normalizing the data
(4) Split of train and test. (70 percent of first-time stamp of each year for training and remaining data points are for validation.)
(5) Using scikit learn library for implementation of Random Forest Regression.K-fold valid is performed using Grid search to hyper tune the Hyper Parameters like number of decision trees, Learning rate and depth of the tree
(6) Prediction is done on the test data using the optimized hyper-parameters

3.6 LSTM and Attention based Deep Neural Network

3.6.1 Overview . We also used an LSTM and an attention based deep neural network to forecast ranking. This is a multivariate time series model with single step prediction where we have taken frames of data to predict the probability distribution of the leading car. The LSTM network is a simple RNN with LSTM cell and a dense layer with softmax activation function to output the probability distribution of leading car. In the next experiment we used attention decoder to capture important data in our feature map that affects ranking to give a better prediction. These experiments were based on Tensorflow tutorial for multivariate time series model. Also, since Tensorflow before 2.1.0 doesn't have an inbuilt Attention layer, we used Zafarali Ahmed's "How to Visualize Your Recurrent Neural Network with Attention in Keras" in 2017 and GitHub project called "keras-attention".

3.6.2 Limitations of Encoder-Decoder Models. The main drawback of Encoder – Decoder Architecture is that the decoder receives only the encoder vector i.e., only the last encoder hidden state. This encoder vector is expected to summarize the entire input sequence and for long input sentences, we expect the decoder to create the output just based on one vector sequence. If the given input sentence is too large this will it harder for the decoder to predict the output sentence given the input sentence. For the purpose of overcoming this difficulty, we are using Attention-Based architecture.

3.6.3 Architecture of Attention Based SEQ2SEQ Model . SEQ2SEQ Model with Attention Layer consists of the following components:

(1) Encoder: The encoder is responsible for stepping through the input time steps and encoding the entire sequence into a fixed length vector called a context vector.
(2) Decoder: The decoder is responsible for stepping through the output time steps while reading from the context vector.
(3) The Attention layer is considered as an interface between Encoder and the Decoder that provides each unit in the decoder with the information from the encoder hidden state.

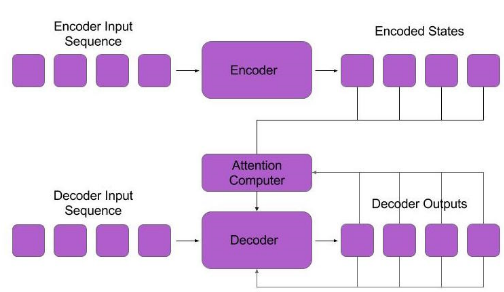

Figure 7: Overview of architecture with attention layer in Seq2Seq model

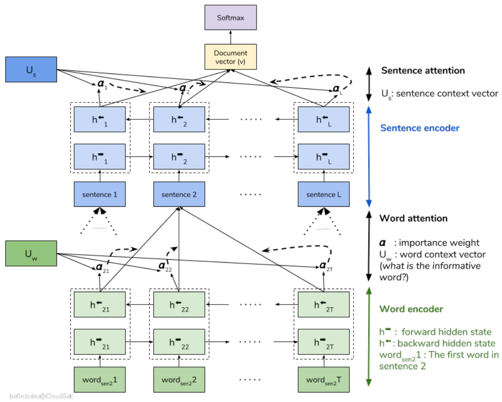

Figure 8: Detailed figure of attention mechanism

3.6.4 Input Data and Processing . The input data consisted of 33 cars having 2 features each: distance and time since last pit stop. So, in total we had 66 features in the input data and around 10,000 records per race representing each second. We used data from years 2013 to 2018. Data from 2013 to 2017 were used for training and 2018 was used for validation. For input, we used frames of data consisting of a certain number of steps for each experiment. The size of each input frame was 20. We also considered top 20 ranked cars as well as all 33 cars in our experiments. We achieved better results with keeping only top 20 cars in our data based on the final rank.

The data was standardized in two phases – first the pit stop time column was standardized for each year data separately. The distance column was standardized for each frame using the min max scaler function. To do this we took the maximum value in the last row of each frame and the minimum value in the first row of each frame. Then we subtracted the minimum value from all frame values for distance column and divided by the range.

3.6.5 Models Architecture . There were two models we used. The architecture of each model was as below:

(1) First a simple LSTM network, with 2 LSTM layers consisting of 128 and 64 units, respectively. Another Dense layer for 64 units and finally a "SoftMax" layer for predicting the output.
(2) The other attention based network consisted of - two layers attention encoder decoder, encoder layer with 150 units, (window size, number of features) dimension and return sequence = True. The attention decoder was of the same shape.

4 RESULTS

4.1 Exploratory Data Analysis

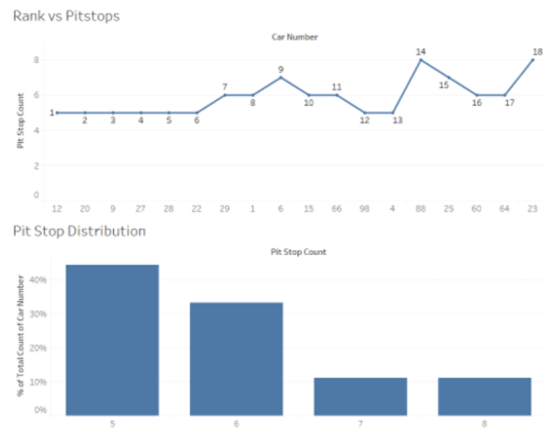

Figure 9: Analysis of pitstops in context with the final rank. Top graph shows Rank vs Pitstops and bottom graph shows the distribution of pitstops among the players who completed the race.

As we can see from Figure 9, most of the player who completed the race tried to take less number of pitstops. Top graph also indicates that players who ranked from rank 1 to 5 took 5 pitstops which was least in the race. Also, players who have ranked lower taken more pitstops. This means that on average, there is some correlation (negative) between the number of pitstops and final rank.

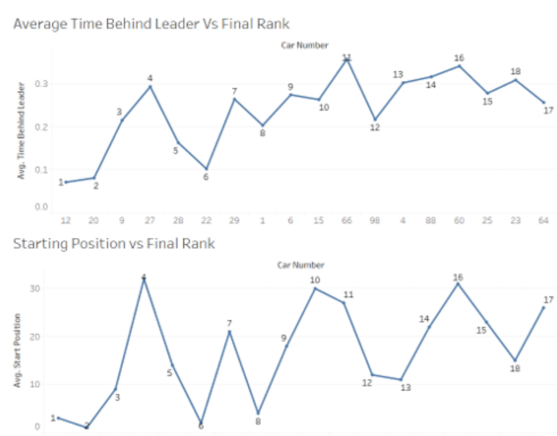

Figure 10: Analysis of average time behind leader and starting position in context with the final rank. Top graph shows Av. Time Behind Leader vs Final Rank and bottom graph shows Starting Position vs Final Rank for the players who completed the race.

From Figure 11, there seems to be a weak correlation between starting position and final rank. However, the graph Average Time Behind Leader vs Final Rank suggests that players who end up ranking high on average remain close to the leader. It indicates that time behind leader might prove important in forecasting rank in the race.

Finally from Figure 12, we can see that player who complete the race maintain similar strategies and remain consistent throughout the race. They also happen to take pitstops near the same time. The strategies only differ slightly towards the end of the race. But for players who do not complete the race, Figure 10, there is more inconsistency. These could be due to external or internal factors.

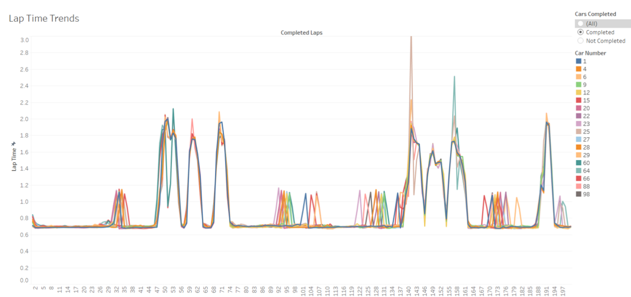

Figure 11: Lap time trends for players who completed the race.

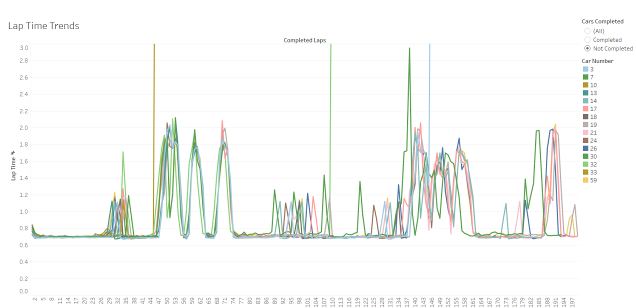

Figure 12: Lap time trends for players who did not complete the race.

4.2 Baseline Model-ARIMA Model

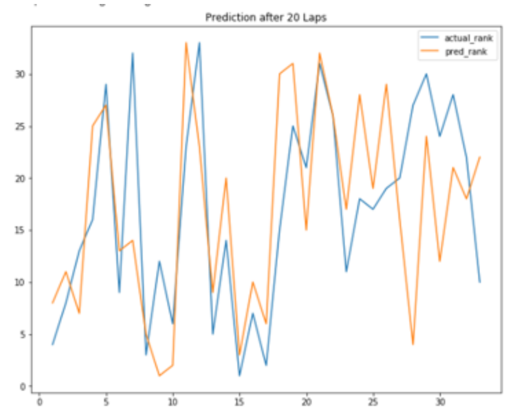

Figure 13: Predictions Vs Actual after 20 laps

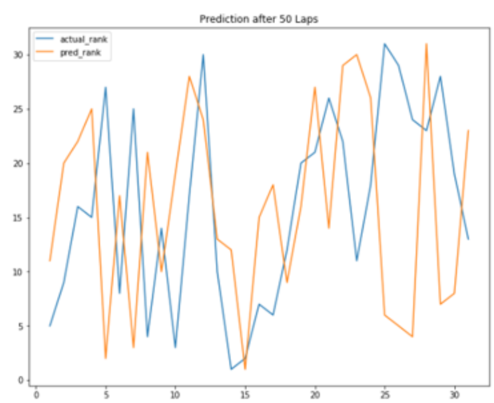

Figure 14: Predictions Vs Actual after 50 laps

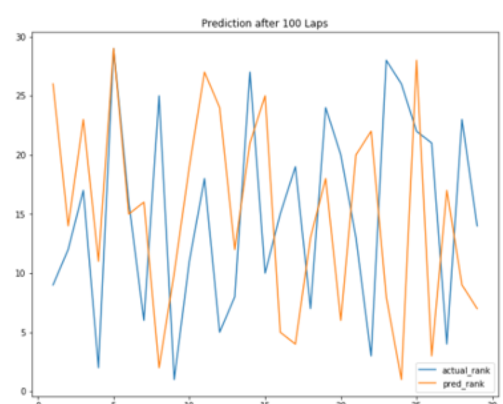

Figure 15: Predictions Vs Actual after 100 laps

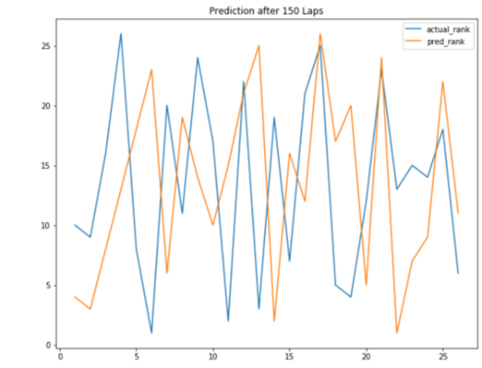

Figure 16: Predictions Vs Actual after 150 laps

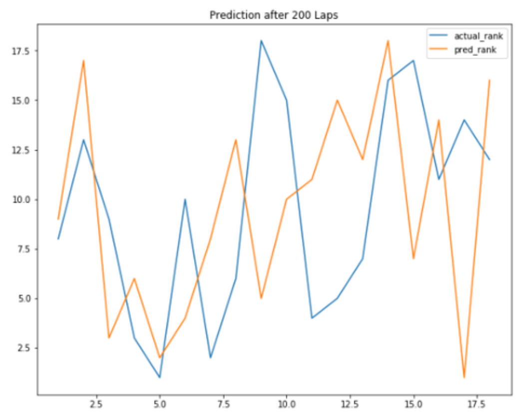

Figure 17: Predictions Vs Actual after 200 laps

From all the above figure we can notice that for same of cars the prediction is quite close but for some other there is significant difference.So this may be due to the assumption that all the cars are considered independently while predicting. So next steps would be to incorporate the dependencies.

4.3 Random and Gradient Forest

1.The accuracy obtained by Random forest is 42% . 2. The accuracy obtained by Gradient Boosting is 44%

4.4 LSTM and Attention based Deep Neural Network

The training took around 1.5 hours on full data and the validation accuracy for both models were not very stable. We were able to achieve highest accuracy of 55% with attention model and 48% without attention model. This variation in accuracy could be due to small differences in the step values within the frame for distance. For instance, one of the values in the first row was 0.998863739 and one of the values in the last row 0.999436234. As we can see the difference is very low this might make the model unstable. Even if we try increasing the frame size or make wider steps the accuracy did not improve. We also did try different configurations for step sizes, frame sizes and the range of ranking prediction. For some models, we saw very high training accuracy > 90% but less validation accuracy and fluctuating validation accuracy. This might be caused due to weight swinging due to values in input data.

For future research, we believe adding high quality telemetry data will help achieve better accuracy as it would carry more information for the model to learn from. Information such as pattern in throttle, gears and acceleration will greatly help to achieve more accuracy. But this data needs to align properly with each other and lap result data.

4.5 Comparison and Conclusion

Sr.No	Model	Accuracy	Stability	Training Time	Hyperparam.	Simplicity
1.	Random Forest	42%	Yes	1.5 Hr	Few(4)	Simple
2.	Gradient Forest	44%	Yes	1.5 Hr	Few(4)	Simple
3.	LSTM	48%	No	30min to 1.5 Hr	Many(>4)	Complex
4.	Attention	55%	Yes	30min to 1.5 Hr	Many(>4)	Complex

(1) From EDA, the starting position, last pit stop displayed correlation with rank
(2) By deploying various models, we found the Seq2Seq Attention Model has performed well while compared to other models.
(3) But when you consider the stability or variation of accuracy then the Random Forest and Gradient Boosting models are stable over other Deep Learning Models.

4.6 Code Repository

(1) LSTM and Attention Notebooks
(2) ARIMA , Bagging and Boosting Models

REFERENCES

[1] Fredrik Armerin, Jonas Hallgren, and Timo Koski. Forecasting ranking in harness racing using probabilities induced by expected positions. *Applied Artificial Intelligence*, pages 1–19, 10 2018.
[2] Randall Chapman and Richard Staelin. Exploiting rank ordered choice set data within the stochastic utility model. *Journal of Marketing Research*, 19, 08 1982.
[3] Wai-Chung Chung, Chuan-Yu Chang, and Chien-Chuan Ko. A svm-based committee machine for prediction of hong kong horse racing. pages 1–4, 08 2017.
[4] Elnaz Davoodi and Alireza Khanteymoori. Horse racing prediction using artificial neural networks. 06 2010.
[5] Aidan Meyler, Geoff Kenny, and Terry Quinn. Forecasting irish inflation using arima models. 1998.
[6] Rangsan Nochai and Titida Nochai. Arima model for forecasting oil palm price. In *Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and applications*, pages 13–15, 2006.
[7] Ping-Feng Pai and Chih-Sheng Lin. A hybrid arima and support vector machines model in stock price forecasting. *Omega*, 33(6):497–505, 2005.
[8] Janett Williams and Yan Li. A case study using neural networks algorithms: Horse racing predictions in jamaica. pages 16–22, 01 2008.