Real Time Rank Prediction using Time Series on IndyCar data

High Performance Big Data Systems: Final Report

Uma Maheswari Gollapudi
MS in Data Science
Indiana University Bloomington
Bloomington Indiana US
ugollap@iu.edu

Sai Sugeeth Kamineni
MS in Computer Science
Indiana University Bloomington
Bloomington Indiana US
skaminen@iu.edu

1. INTRODUCTION

The meteoric rise of Machine Learning has had path-breaking consequences for multiple domains worldwide, particularly having substantial success in making accurate predictions, classifications, and detections, among others. Additionally, there has been a boom in the usage of high-performance computing systems in recent years, with the increasing requirements brought on by large data sizes and processing power. This project is placed at the intersection of both these domains, as we aim to predict real-time ranking estimates of race cars from the IndyCar dataset. Our task comprises extracting the relevant data from the IndyCar data logs, using exploratory data analysis and data preprocessing to isolate the best features, and predicting the real-time ranking of a racecar using an ML model, executed on a distributed environment. We plan to deploy models such as LSTM, Random Forest, LightGBM, among others, using TensorFlow, primarily on Apache Storm. Along the way, we would like to gain insight into building a high-performance computing framework from the ground up and attaining accurate predictions of car ranking.

The Indianapolis 500 is a world-famous, annual car racing tournament, held in Indiana. It is a massive sporting event, bringing in millions in revenue and holding national significance. Our project aims to predict real-time ranking estimates based on cars, telemetry, laps, and pitstop data, as well as other sensor data from the IndyCar log dataset. This real-time prediction allows the crew to develop strategies on the spot for upcoming laps, monitor their cars more closely in case of decreased performance, and increase pitstop timings, as well as benefitting people who indulge in legal race betting. This is an immensely difficult problem and is still being solved around the world – owing to the various factors that can play an integral role in the racecar's performance. Our project is mainly motivated by the above reasons. The dataset has a total of 5,979,749 records among 4 data sets. The telemetry data alone has 5,939,913 records, with 67 records from cars, 910 records from pitstop, and 38,859 records in laps. We will also look at the data log for each race car in 2018 to see if we can identify other crucial sensor data that will be used to maximize prediction accuracy. As we work further on with the data, we realize that in addition to racing statistics, the ranking order correlates with the anomalies that occur during the race and the number of pitstops, and the time taken per pitstop. These bring us into the domain of pitstop prediction and anomaly detection and work as an add-on to the original task of predicting the ranking with respect to lap times and other data. Hence, there are many variables at play that can decide the position and performance of a racecar, making this problem both challenging and exciting to work on. Once we run different models with different settings and parameters, we will identify the best solution for this problem and execute that on Apache Storm to work with real-time streaming data. We plan to evaluate the results using various metrics such as Accuracy, MAE, and MSE, among others. Factoring in the variability and uncertainty that such large datasets bring about, we will establish a prediction framework that can accurately predict each car's rankings. With respect to our goals for this project, as a team we are interested in creating and deploying an end-to-end high performance computing framework, since this is the first such endeavor for both team members.

1.1. Related Work

We found a couple of papers related to rank prediction in racing – Predicting the Outcome of NASCAR Races: The Role of Driver Experience [1], Linear Regression Applied to Race Car Performance Predictions [2] and Do Reliable Predictors Exist for the Outcomes of NASCAR Races? [5]. We also found a report titled Predicting Horse Racing Result with Machine Learning [3], which is a detailed 81-page report that we plan to use as required. Finally, we looked at a paper titled Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. [4] The paper titled Do Reliable Predictors Exist for the Outcomes of NASCAR Races? [5], is an early work on determining the factors affecting the outcomes of races, and while it is fairly intuitive in terms of its results which mainly state that variables related to car speed and the prior success of the driver are correlated positively with the ranking order, it is still substantial. Paper [1] supplements this finding, by stating that starting position and driver experience play an important role in the final outcome of a race. Additionally, papers [3] and [6] are related to racehorse predictions and support the use of both simple and complex ML models to predict the results of racing.

It is important to note that it has been difficult to find adequate previous work on real time rank prediction in race cars using log data, and all these papers mentioned above do not accurately fit our problems needs.

Additionally, the main Python libraries we have used are pandas, NumPy, scikit-learn, TensorFlow, matplotlib among others. We looked at open source code on related analysis and predictions such as Formula One prediction [https://github.com/neiltwist/f1-prediction] and AWS Real Time Racing Analytics using ML [https://aws.amazon.com/blogs/machine-learning/delivering-real-time-racing-analytics-using-machine-learning/].

During the latter part of this project, we were introduced to a novel research paper titled Rank Prediction Forecasting in Car Racing [7], which was a joint effort between a team from IU Bloomington and NEC, Japan. Published on October 23,2020, the paper was heavily related to this project, and proposed the use of RankNet for race car rank forecasting. Essentially, 'RankNet' is 'a combination of encoder-decoder network and separate MLP network that is capable of delivering probabilistic forecasting to model the pit stop events and rank position in car racing'.

2. ARCHITECTURE AND IMPLEMENTATION

For this project, we used the 'completed_laps' record from the 2018 IndyCar log data file, which had about 19427 records. This section details the process of understanding the data, identifying the best features and discussions about the models we intended to use.

2.1. Data Processing

The first step in this project was to perform data cleaning and processing. Here, we worked to remove invalid or incomplete records. To decide which features will help us predict the rank, we will perform exploratory data analysis. We looked at the trends of the car's rankings and the relation of ranking with other variables. We visualized the number of pitstops taken by the winning cars, the number of times the winning cars lead a lap, and so on.

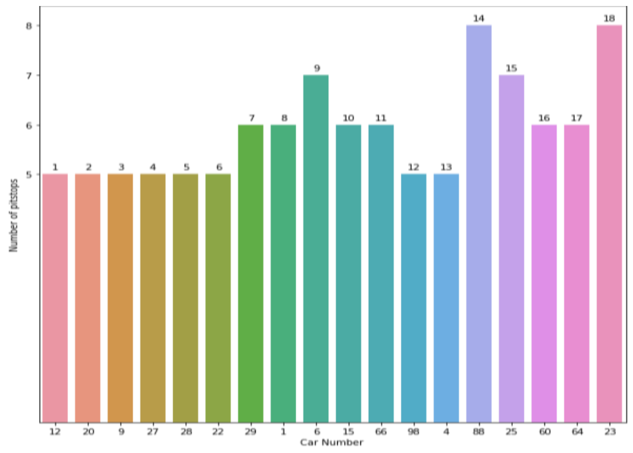


Figure 1.1: EDA result- car ranking versus number of pitstops

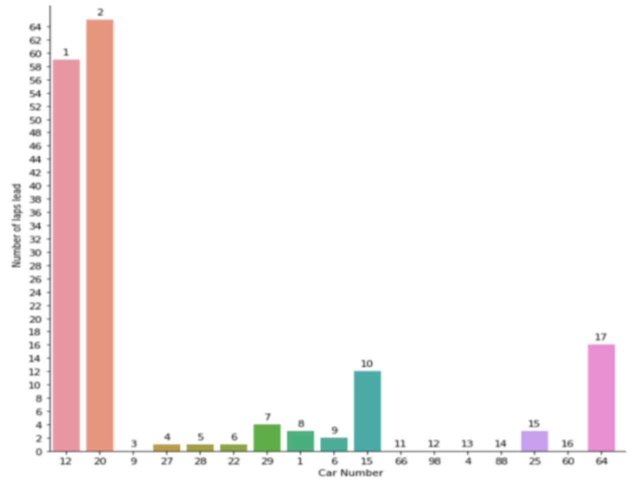
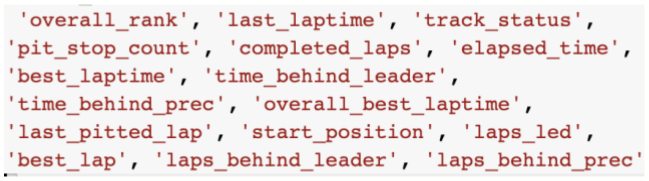


Figure 1.2: EDA result- car ranking versus laps led

From the EDA, we could observe some insights. In Figure 1, when comparing the car ranking versus the number of pitstops, it is observed that the top six cars all had the lowest number of pitstops, and generally as the number of pitstops increase, the car ranking is lower, which seems to be logically sound. However, we can observe in some cases- such as Car 66 and 98, the relationship is weaker. In this case, further research into the duration of the pitstops or other factors would be very useful. In the second figure, one would believe that if a car led a higher number of laps, it would generally be one of the top performers. For example, Car 20, with the highest laps, placed second, while Car 64, which lead a decently high number of laps, placed second to last. This shows that there are other factors affecting the car's ranking in the race. We also performed other analyses, such as looking at the correlation matrices to understand which features provide the best and most comprehensive information about the performance of the race cars. In terms of processing, we incomplete records, removed nan's and looked into cars that completed the race. We looked at the track status, to try and understand the cautionary flag on the track – this is an insight we gained from reading the paper titled 'Rank Prediction Forecasting in Car Racing'. After performing this step, we were able to convert the data into the correct format required for deep learning model as inputs, i.e., timestep inputs for LSTMs. In general, we isolated the 16 best features, all of which are displayed in the image snippet below. Then, we split it into training and validation data to get the final dataset, which we will use to train different models.



```
 'overall_rank', 'last_laptime', 'track_status',
'pit_stop_count', 'completed_laps', 'elapsed_time',
'best_laptime', 'time_behind_leader',
'time_behind_prec', 'overall_best_laptime',
'last_pitted_lap', 'start_position', 'laps_led',
'best_lap', 'laps_behind_leader', 'laps_behind_prec'
```

Figure 2: The 16 features we used for rank prediction

2.2. Models

The next phase involves Modelling, and here we planned to consider numerous Deep Learning models to compare and evaluate their performances against each other to identify the best models for the given data and task. By this stage, the data is prepared and ready to be deployed in models. We deployed high level LSTM's and compared their performance to Random Forest, LightGBM and other simpler models. Additionally, if the time and resources permitted, we would have liked to experiment with attention models, which seem to have a good amount of success with such problems. The idea behind deploying ML and DL models is that this will allow us to know how deep learning models compare with traditional tree-based machine learning models while performing predictions, since our original plan was to create and produce a foundational work depicting the best models to implement for real time rank prediction in race cars.

2.2.1. Deep Learning

We believe that deep learning models like LSTMs and RNNs will perform well for this task because this is a large time series-based dataset. This data can be easily converted into timesteps and supplied as input to such models, allowing them to predict the rank of a racecar based on a given timestep. These models have the ability to retain some memory and information about the performance of the previous input data, and this property of these models will be highly useful for such problems. With respect to LSTM algorithm, the inputs are related to each other whereas in a fully connected neural network for example they are independent. This means that the effect an input has on the result is dependent on its previous inputs as well. This is how LSTMs are able to retain some memory, which can be even longer term when compared to its counterpart - the RNN model. We will need to convert the data into the desired format with necessary amount of time steps. We experimented with LSTM's and worked on with isolating the best parameters that results in the best, most accurate results (this includes both variations of the model's architecture - number of LSTM layers, parameters such as Dropout, Batch normalization, input shape etc., and many hyperparameters - learning rate, batch size, number of epochs). We also added the early stopping feature to prevent overfitting. Essentially, we will be working to find the optimal Deep Learning model to aid us in predicting the race car ranks accurately and quickly. The hyperparameters will be discussed below in the experiments section.

2.2.2. Machine Learning

Machine Learning models are simpler and more commonly used when compared to Deep Learning models for most modelling problems in the real world. Because of their frequent use, we believe that it is a good idea to consider these for our problem as well. As we have seen previously, there has not been a lot of work conducted in the same domain for the same question of accurately predicting the performance of race cars. Hence, we also want to see whether ML models are able to perform similarly well with respect to the more robust and complex DL models. This would make this work far more fundamental, comprehensive and hopefully, accessible for others who would like to conduct similar research. For the ML models- we implemented Random Forest, LightGBM, AdaBoost, among others such as Gradient Boosted Model. We also experimented with various parameters as mentioned above and used this information to decide the best set of models, settings and hyperparameters for this problem**.**

2.3 High Performance Systems

After running these models on a single node, we ran them on real-time big data platforms like Apache Storm to compare their performance. Here are a few key points about Apache Storm, which is comparably complex to set up and implement. Storm has wide language support and uses Spouts and Bolts which form a DAG. Storm is useful for very low latency real-time applications when compared to other interfaces, which have slightly higher latencies. It is used for stream processing. For the purpose of this problem, once we have run our DL and ML models, compared and analyzed performances, we re-implement the best of these models (the LSTM) on Apache Storm for this project, since we have been exposed to it in class.

2.3.1. Storm

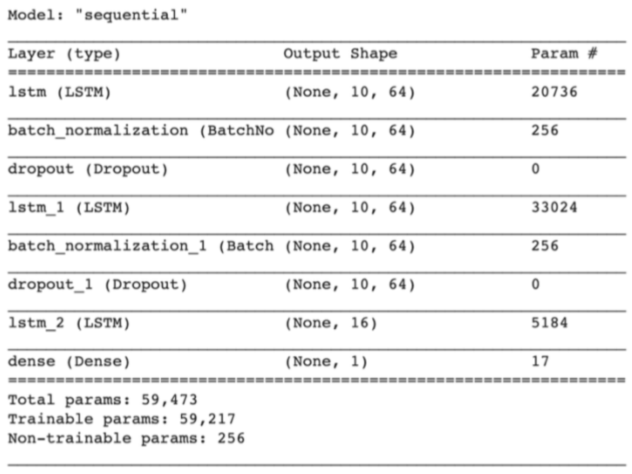
We ran the experiments on single node virtual machine on which Apache Storm was setup. We imported the data in .csv format and the model in .h5 format. We then setup the data spout which extracts the data from the csv file and emits the necessary data in the specified format. The inference bolt receives this data and inputs it to the model which then gives the prediction. This prediction along with other necessary data is then emitted by the inference bolt. The send web server bolt receives this and packages it in the necessary format and then sends it to the webserver. The webserver then creates the visualization of the real time predictions**.**

3. EXPERIMENTS

For this project, we have implemented an optimized LSTM model and Random Forest, AdaBoost, LightGBM and Gradient Boosting models. In this section, we will discuss the parameters we set, the metrics we chose, and the results obtained**.**

3.1. Model Settings - LSTM

After our initial experiments tweaking the model architecture and hyperparameters, we finally used an LSTM with the following architecture and compared how the performance might vary if we changed the loss function from mean absolute error to mean squared error.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 10, 64)            20736
_________________________________________________________________
batch_normalization (BatchNo (None, 10, 64)            256
_________________________________________________________________
dropout (Dropout)            (None, 10, 64)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 10, 64)            33024
_________________________________________________________________
batch_normalization_1 (Batch (None, 10, 64)            256
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 64)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 16)                5184
_________________________________________________________________
dense (Dense)                (None, 1)                 17
=================================================================
Total params: 59,473
Trainable params: 59,217
Non-trainable params: 256
_________________________________________________________________
```

Figure 3: LSTM model architecture

LSTM hyperparameters and settings: Optimizer=Adam, Learning rate=0.001, Loss=Mean Absolute Error, Epochs = 50, Batch size=32, Validation split=0.1.

We took a time step of size 10 and tried predicting the overall rank for the future step (11th step). For the initial simple LSTM, the MAE validation loss was 2.4 and for MSE, we obtained a validation loss of 9.1. Finally, with the optimized LSTM, we obtained a lower MAE validation loss of 2.1. The following plots show the actual performance of Car-12 and compare this with its predicted performance, while comparing the performance of MAE and MSE as metrics. Based on our initial and optimized results, we proceeded with using MAE as our evaluation metric.

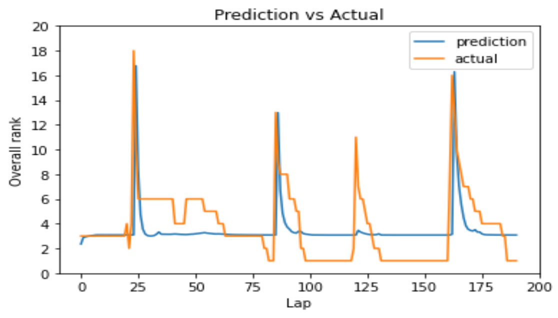


Figure 4: Prediction vs Actual ranking for LSTM with MAE

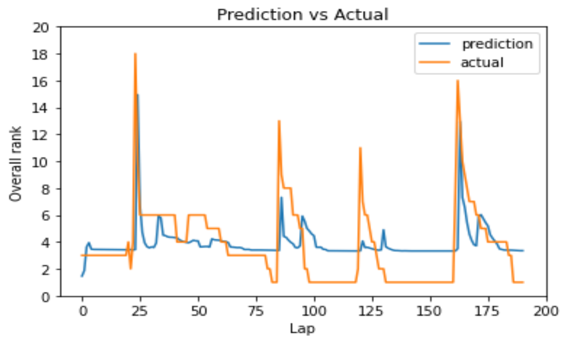


Figure 5: Predicted vs Actual rankings for LSTM with MSE

The LSTM model is able to sufficiently predict the trends of the values, even though there is more to be optimized for the sake of higher performance.

3.2. Model Settings – Machine Learning

We ran four machine learning models, each of which has a different set of optimizations and had slightly different results. These were run directly on the processed data.

3.2.1 LightGBM Regressor

We ran the LightGBM regressor with the following parameters – 500 trees, learning rate = 0.0001, boosting type = gbdt, objective = regression, metric = l1, number of leaves = 50, minimum data = 50 and maximum depth = 10. For this, we obtained a MAE of 8.63, which was 4 times more than that of the optimized LSTM model.

3.2.2 AdaBoost Classifier

We ran the AdaBoost classifier with 200 trees, a maximum depth of 10 and learning rate of 0.01. Other than these values, all the other settings were the default settings. This fetched us an MAE of 4.66, which is slightly better than the LightGBM regressor, but does not match up to the performance of the LSTM model.

3.2.3 Random Forest Regressor

The Random Forest Regressor performed well at the following settings – number of trees = 100, maximum leaf nodes = 20, maximum depth =10 and minimum samples per leaf = 50. The resulting MAE was only 1.63, which is 0.7 times lesser than the MAE obtained by the optimized LSTM model.

3.2.4 Gradient Boosting Regressor

The Gradient Boosting Regressor, which we ran towards the end of the project, was run on all of its default settings – number of trees = 10, learning rate = 0.1 etc. and outperformed all the models in this project, with an MAE of 0.71, beating all optimized machine and deep learning models.

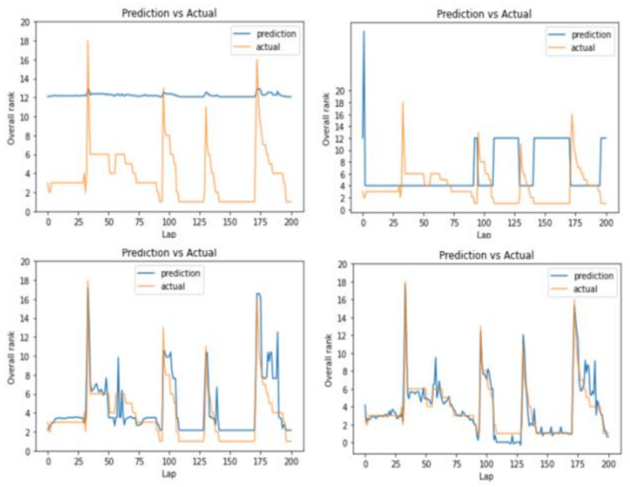


Figure 6: Predicted vs Actual rankings for all ML models : from the top, going clockwise – LightGBM Regressor, AdaBoost Classifier, Random Forest Regressor and Gradient Boosting Regressor.

As we can see, there is clear difference in the performance of the Random Forest and Gradient Boosting regressor, which perform very close predictions.

3.3. Results and Analysis

After running the models on Google Collaboratory, we exported the LSTM model and the cleaned, processed data to Apache Storm, via our virtual machines. We then set up Apache Storm's spouts and bolts. When we ran this program on the Storm interface, we were able to visualize real time car ranking predictions, as we can see in the below images.

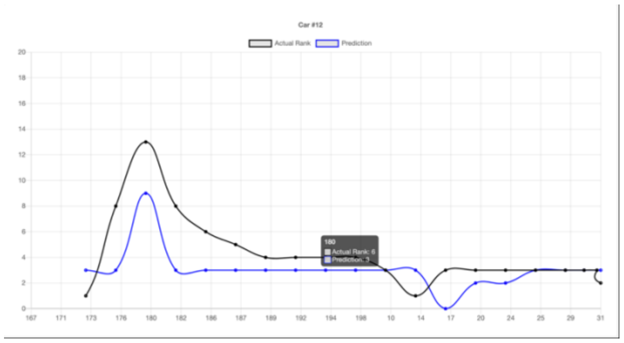


Figure 7.a.: Predicted vs Actual rankings of LSTM on Apache Storm

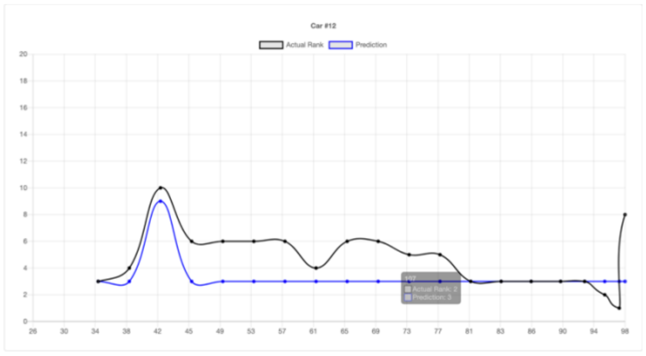


Figure 7.b.: Predicted vs Actual rankings for LSTM on Apache Storm – where actual and predicted scores are very close and even overlap

Analyzing the overall results, we think it's important to comment upon the success of the deep learning model, even though this was an initial attempt at optimizing an LSTM and deploying it on Apache Storm for real time forecasting.

One crucial thing that would have helped us provide further insight and possibly improve the prediction power of this model is including pit stop prediction and anomaly detection to greater levels. Through the use of the track status, we were able to take anomalies in some level of consideration, but a more powerful analysis could have included ensemble models where one model was designated to a certain problem – anomaly detection, pitstop prediction, pit stop interval predictions etc., and the model would then robustly predict the car rankings.

Another insight from this analysis that has stood out, is the surprisingly robust performance of the random forest regressor as well as the gradient boosting regressor, both of which recorded an MAE of only 1.31 and 0.71 respectively, which is much better than the LSTM's best MAE of 2.1. Based on the prediction graph, this appears to be a case of overfitting, but more work has to be done to understand whether the metric used here has a deficiency, or possibly that Random Forests and Gradient Boosting Models are genuinely good for such forecasting.

4. CONCLUSION

We were able to achieve good results with real time car ranking forecasting via optimized LSTM's, which performed 2-4 times better than some machine learning models such as LightGBM Regressor and AdaBoost Classifier. However, one important insight was that certain machine learning models such as random forest and gradient boosting regressors are promising too – to a certain extent. More research and parameter optimization will be useful in this case. Additionally, the project helped us understand the working and usage of Apache Storm, for real time analysis and predictions. For future work, we would like to delve deeper into analyzing the performance of Random Forest and Gradient Boosting regressor, as well as finding better deep learning models to perform the same task. Finally, we also believe it is important to take pitstop interval prediction into direct consideration for rank forecasting.

5. ACKNOWLEDGEMENTS

This project required a good amount of dedication and hard work, and we would like to take the time to acknowledge and appreciate all the faculty involved in ensuring its success. We would like to thank Prof. Judy for providing us with the opportunity to work on this project and in guiding us to becoming more proficient at working with distributed systems. We would also like to thank her for her constant support throughout this course. We are grateful to both Selahattin Akkas and Jiayu Li for their quick responses and advice when we needed some aid, and their immense support throughout the working of this project. The completion of this assignment is a success for both of us, as we were previously unfamiliar with the domain of high-performance computing systems. We would like to extend our gratitude to everyone who played an important role in the completion of this project, and this course as a whole.

6. TASK DISTRIBUTION

For the break-up of work, ideally, we would have liked to split each task – Data Processing, Feature Engineering, modelling, etc., equally among us since we are interested in learning about and implementing the entire pipeline. We worked together when it came to using the distributed environment.

For a clear break up of activities:

No.	**Uma Maheswari G**	**Sai Sugeeth K**
1.	Exploratory Data Analysis	Data Cleaning/Processing
2.	Run initial data on, Random Forest, Gradient Boosting Regressor.	Run initial data on LSTM, LightGBM, Adaboost
3.	Run optimized models– LSTM, LightGBM, Adaboost.	Run optimized models – Random Forest, Gradient Boosting Regressor.
4.	Analyze the performances across both sets of models.	Analyze the performances across both sets of models.
5.	Implement best of DL models –LSTM on Apache Storm.	Implement best of DL models –LSTM on Apache Storm.
6.	Convene to discuss and perform final analyses – what's the best way to perform this task, and what could be done better next time?	Convene to discuss and perform final analyses – what's the best way to perform this task, and what could be done better next time?
7.	Final Report and Presentation	Final Report and Presentation

7. REFERENCES

[1] Allender, M. 2008. Predicting The Outcome Of NASCAR Races: The Role Of Driver Experience. *Journal of Business & Economics Research (JBER)*. 6, 3 (Mar. 2008). DOI: https://doi.org/10.19030/jber.v6i3.2403.

[2] Dwight Southerland. 2016. Southwest Decision Sciences Institute (SWDSI) Conference. In *http://www.swdsi.org/proceedings/2016/*. Oklahoma City: Southwest Decision Sciences Institute. http://www.swdsi.org/proceedings/2016/Papers/Papers/PA015.pdf

[3] Yide LIU. 2017. Predicting Horse Racing Result with Machine Learning. (2017). Retrieved September 19, 2020 from http://www.cse.cuhk.edu.hk/lyu/_media/students/lyu1703_term2report.pdf

[4] Ralf C. Staudemeyer and Eric Rothstein Morris. 2019. Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks. (September 2019). Retrieved September 19, 2020 from https://arxiv.org/abs/1909.09586

[5] Pfitzner, Barry and Tracy D. Rishel, 2005. Do Reliable Predictors Exist for the Outcomes of NASCAR Races? *The Sport Journal*, vol. 8, no. 2. Retrieved September 30, 2020 from https://thesportjournal.org/article/do-reliable-predictors-exist-for-the-outcomes-of-nascar-races/

[6] Elnaz Davoodi and Alireza Khanteymoori. 2010. Horse Racing Predictions using Artificial Neural Networks. Retrieved September 30, 2020 from http://www.wseas.us/e-library/conferences/2010/Iasi/NNECFS/NNECFS-21.pdf

[7] Bo Peng, Jiayu Li, Selahattin Akkas, Fugang Wang, Takuya Araki, Ohno Yoshiyuki, Judy Qiu. 2020. Rank Position Forecasting in Car Racing. (October 2020). Retrieved December 14, 2020 from https://arxiv.org/abs/2010.01707