French Trot Horse Racing Prediction Using Artificial Neural Networks

Santosh Kumar Satapathy
*CIS*6020: Artificial Intelligence*
University of Guelph
Under the Guidance of: Dr. Neil Bruce

***Abstract-* Artificial Neural Networks (ANNs) have demonstrated success in tackling complex problems across various domains. This project applies ANNs to the realm of horse racing prediction, specifically focusing on French Trot races. Utilizing the French Trot Horse Racing Dataset from the CANSSI Horse Racing Prediction Competition, our objective is to craft a Feed Forward Neural Network aimed at predicting the Finish Position of each horse in forthcoming races. Employing advanced Feature Engineering and Modeling techniques, we preprocess and input the data into the Neural Network, fine-tuning its parameters to minimize the Root Mean Squared Error (RMSE) while predicting the Race Beaten Margin for each horse. This project seeks to advance our comprehension of Horse Racing dynamics, understanding key parameters influencing a horse's performance in a race, and simultaneously aims to enhance proficiency in constructing and fine-tuning Neural Networks to achieve predictive accuracy.**

Keywords- neural networks, model architecture, horse race prediction

INTRODUCTION

Artificial Neural Networks (ANNs), inspired by studies in brain modeling, have proven highly effective across diverse applications due to their efficiency and ability to address complex problems. Their versatility extends to areas such as classification, pattern matching, optimization, and time series modeling, with prediction being a crucial aspect of the latter. Traditional methods like regression often encounter difficulties in approximating complex relationships, and their linearity poses limitations. In contrast, ANNs provide a general and adaptable method for prediction, overcoming these challenges.

The widespread success of ANNs in predicting phenomena like weather, travel time, and stock market trends has extended to predicting game results, including animal racing. This project focuses on applying ANNs to forecast horse racing outcomes in the French Trot Horse Racing Dataset. Each horse's performance is predicted using a dedicated ANN to estimate the Beaten Margin in a race. The exploration of optimal neural network architectures is conducted through grid search, comparing different configurations based on the RMSE measure. The chosen architecture minimizes RMSE, ensuring accurate predictions for each horse across multiple runs. This work contributes to the broader understanding of predictive modeling in horse racing, emphasizing the effectiveness of ANNs in capturing the complexities in such intricate domains.

This report is organized in the following Sections:

SECTION 1: UNDERSTANDING FRENCH HORSE TROT RACING

1.1 Trot Racing Overview

Trot racing constitutes a distinctive form of horse racing where horses execute a specific gait, known as a trot. A prevalent manifestation is the harness race, where horses pull a two-wheeled cart (sulky) steered by a driver. Alternatively, mounted trotting races involve jockeys riding saddled horses.

1.2 Trot Racing in France

France stands as the preeminent jurisdiction for trotting races, hosting over 11,000 events annually. The races predominantly feature trotters, distinguished by a diagonal leg movement pattern. Both harness (Attele: Figure 1) and mounted (Monte: Figure 2) trotting races are prevalent, with monte races constituting approximately 20% of all trotting events. Notably, horses can participate in both attele and monte races.

Figure 1: A depiction of a typical French harness race (Attele).

Figure 2: Illustration of a mounted trotting race (monte).

1.3 Characteristics of French Trot Racing

In both attele and monte racing, horses are prohibited from breaking their stride, facing disqualification if this occurs. Age restrictions vary, with racing horses allowed participation until the age of 10, commencing as early as 2 years old.

1.4 Track Surfaces

Trot races in France transpire on three distinct surfaces: grass (turf), sand, and ash and cinder. Horses may exhibit preferences for surfaces, influencing their performance.

1.5 The Start

The start of a French trot race significantly influences its outcome. Two types of starts exist: auto starts and volte starts. Auto starts involve horses lining up behind a motorized gate, while volte starts see runners gathering before entering the track, executing a quarter turn before the official start.

Auto Starts:

A. Horses start in race book order:

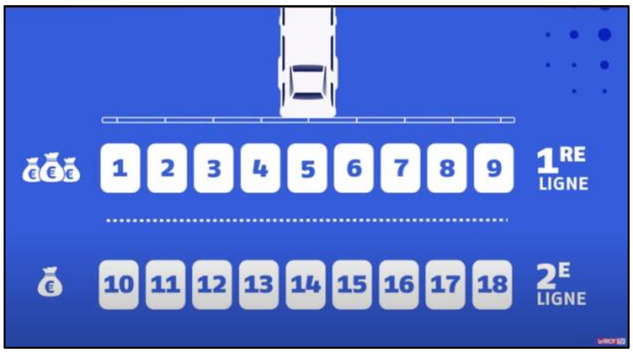


Figure 3: Racing Order.

B. Computer ballot allocates starting positions.
C. Maximum 18 starters, with the first line determined by winnings.

Volte Starts:

A. Runners gather in an adjacent area.
B. Enter the track in an orderly fashion.
C. Volt a quarter turn before the official start.
D. False starts prompt a race reset.
E. Handicaps may be assigned based on winnings.

Starting Line	Handicap Distance
1st line	None
2nd line	25-metres
3rd line	50-metres

Table 1: Handicap distance for each line

F. Three starting lines, with greater winnings requiring more distance.

Understanding these facets provides a foundational insight into the unique world of French Trot Horse Racing, crucial for the subsequent application of Artificial Neural Networks for predictive modeling.

1.6 The Race Dynamics

The primary objective of the rider is to complete the race in the shortest possible time, with the horse maintaining a trot throughout. Securing victory necessitates finishing first without resorting to galloping, as any deviation results in disqualification.

1.7 Rider Tactics

Rider strategies in a trot race are diverse, ranging from seizing an early lead and maintaining it until the finish, to employing a more strategic approach, waiting until the final straight to make a decisive move and surprise competitors.

1.8 Class Restrictions

Race conditions, also known as class restrictions, play a pivotal role in determining the parameters for any given race. Factors such as horse age, sex, earnings, race distance, and specialty are considered in defining the conditions under which the race unfolds.

1.9 Disqualification Rules

The foremost reason for disqualification in trot races is the violation of maintaining a consistent trot, commonly referred to as breaking stride. This strict adherence to the trotting gait ensures fair competition and upholds the integrity of the race. Understanding these race dynamics, tactics, class restrictions, and disqualification rules is fundamental to comprehending the intricacies and challenges faced by both riders and horses in the context of French Trot Horse Racing.

During a trot race, horses are allowed specific actions without facing disqualification, including:

A. Cantering or pacing for up to 15 strides.
B. Executing a trap or aubin for a maximum of 7 strides towards the end of the race.

Disqualification, apart from stride-related infractions, is often associated with individuals affiliated with the horse, such as the jockey, trainer, or owner.

Fault	Definition
Gallop	In the same stride, the horse throws its 2 front legs forward and its 2 hind legs backwards.
Trap	The horse trots with the front legs and gallops with the back legs.
Aubin	Galloping with the front legs and trotting with the back legs.
Pace	The horse moves both legs on the same side at the same time and in the same direction

Table 2: Disqualification Definitions

1.10 Weight Restrictions for Riders

The rider's weight assumes critical significance in French trot racing, particularly in monte races. For monte races, the minimum weight requirement for a rider is contingent upon the horse's age:

Age	Minimum Weight Restriction
3YO	60 kg
4YO	63 kg
5YO (and above)	67 kg

Table 3: Weight Restrictions

1.11 Horse Footwear in French Trot Racing

In French trot racing, horses participate with either horseshoes, plates (constructed from plastic or leather), or without any footwear (barefoot). The regulations regarding the use of horseshoes (or plates) are stringent, permitting only three combinations in any given race:

- Front legs only
- Hind legs only
- Completely unshod

1.12 Prize Money Distribution

Prize money distribution in French trot racing extends to the top 7 finishers in any given race. The allocated prize money serves as an indicative measure of the race's quality, with higher-grade races offering more substantial prize money. Understanding the nuances of horse footwear and the correlation between prize money and race quality contributes to a comprehensive grasp of the intricacies surrounding French Trot Horse Racing.

Finish Position	Percentage of Prizemoney
1st	45%
2nd	25%
3rd	14%
4th	8%
5th	5%
6th	2%
7th	1%

Table 4: Prize Money Distribution

SECTION 2: METHODOLOGY

Our approach involves:

A. *Data Collection and Preprocessing:* Extract and prepare race-related Current Race and Past Races Performance Metrics data from the Horse Racing Prediction Competition dataset.

B. *Individual Horse Modelling*: Segment the dataset for each horse, construct neural network models individually for each horse, and identify correlations between beaten margin and performance metrics.

C. *Temporal Data Sequencing*: Organize horse data chronologically for sequential training to recognize evolving racing patterns.

D. *Incremental Model Training*: Train models sequentially for each horse, focusing on capturing individualized racing dynamics.

E. *Beaten Margin Prediction*: Deploy models to predict beaten margins, sorting results to forecast finishing positions.

F. *Iterative Refinement*: Continuously refine models based on predictions, ensuring adaptability and accuracy in performance forecasting.

This methodology aims to uncover nuanced patterns for accurate predictions in French Trot Horse Racing, optimizing models through iterative refinement.

SECTION 3: DATA STRUCTURE

We fetch the data from CANSSI French Trot Horse Racing Competition which provides us with a dataset that has approximately 1.2M samples of horses and their data for each race and their performance in the same race.

The data has Total of 32+10=42 Features. First 32 features, represent each horse’s metrics in a single upcoming race.

Age Restriction:
The feature represents what is the Age Restriction for the race this horse is participating in. This ranges from 2- to 10-year-old horses. However, most races are between horses that are 3-5 years old (Figure 4).

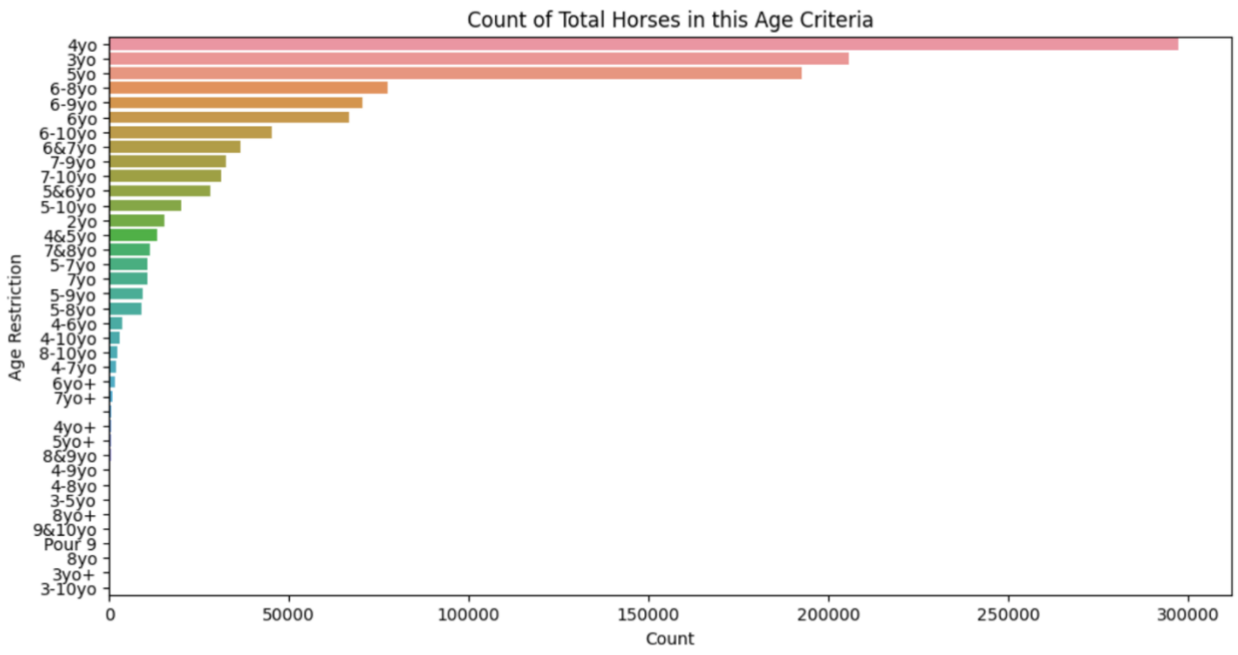


Figure 4: Age Restriction Distribution

This feature was a random string which we cleaned and made two features out of it:

Class Restriction Type: NW, CF, CLM, etc. denote a particular class type restriction for each race and NW is most of the races (Figure 5).

Class Restriction Price:
This value defines the class entry price in dollars for each race.

Starting Line:
This features the Line each horse will be racing on during the race.

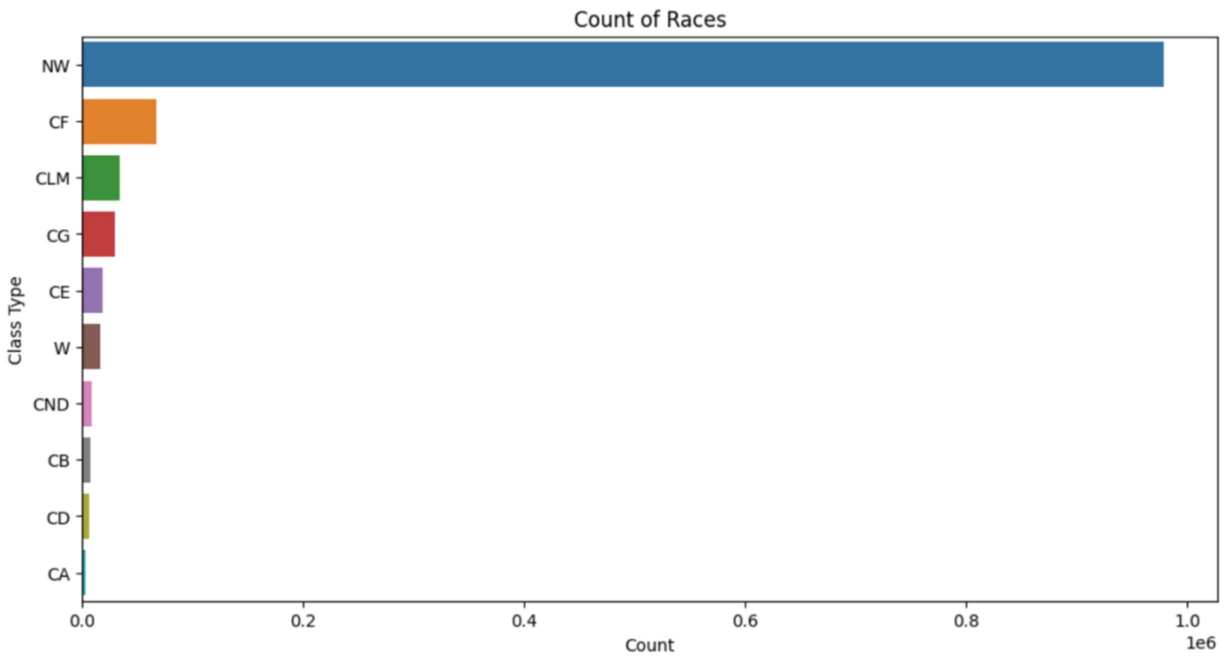


Figure 5: Class Type Distribution

Barrier:
This feature defines the barrier for each horse in each race. This ranges from 0 to 18. In most of the cases, there is no barrier, and the value is 0. We will use this feature as a numerical feature as the relation of the barrier is expected to be linear.

Foaling Country:
This feature defines the Country of the Horse's origin. Most of the horses in this dataset are from France as this race origins in France itself.

Foaling Date:
This feature represents that date of birth of each horse. We will use this feature to predict the Age of the Horse in Days during each race.

Front Shoes:
This feature shows how many front shoes are worn by each horse during the race. A horse can wear up to max 2 or race barefoot with 0 shoes. We will use this feature as a numerical feature as wearing more shoes has a linear relationship to the performance.

Dam ID:
This feature defines the ID of the mother of the Racing Horse. We will use this feature of define certain metrics that would help us predict a horse's performance basis it's mother's all children's performance average.

Course Indicator:
Defines the course type for each race, most of the courses are Normal which is denoted by Null (Figure 6).

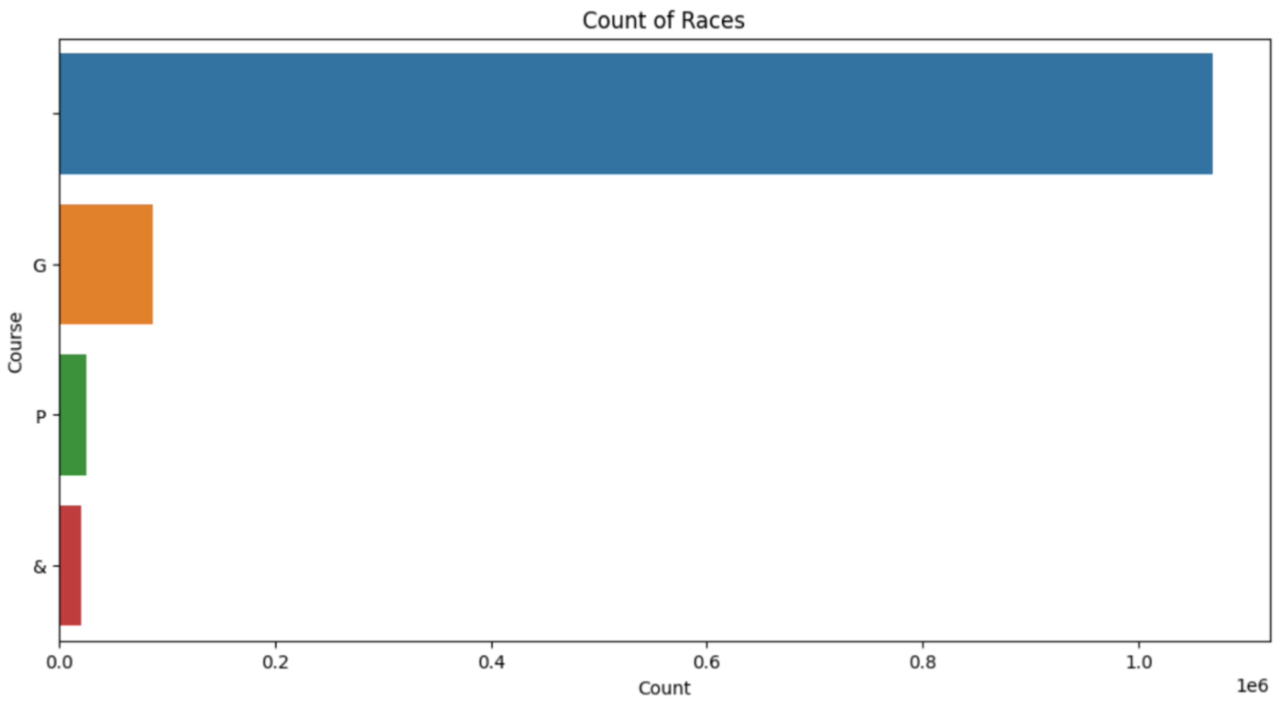


Figure 6: Course Distribution and Null Representation

Distance:
This feature defines the length of the distance to be covered by the horses in each race. It ranges up to 4500m. We will use this as a numerical feature.

Hind Shoes:
This feature shows how many hind shoes are worn by each horse during the race. A horse can wear up to max 2 or race barefoot with 0 shoes. We will use this feature as a numerical feature as wearing more shoes has a linear relationship to the performance.

Horse Age:
This defines the age of the horse during the race. We will also treat this as a numerical feature.

Race Group:
Defines the Group of the Horse if the race has groups. We will Encode this feature.

Race Prize Money:
This feature defines the Total Prize Money associated with each race that will be distributed among the Top 3 Finishing Horses.

Surface:
This feature defines the type of surface of the ground in each race. i.e. Grass, Sand or Ash.

Weight Carried:
This feature defines the weight carried by each horse in each race. Either the horse does not carry any weight, or it carries from around 55-67kg of weight.

Racing Sub Type:
This feature defines the sub type of each race class. We will be denoting TM with 1 and T with 0 to label encode this feature.

Handicap Distance:
This feature shows the Handicap distance of each horse during a race because of a penalty or an award. This distance ranges from -50 to 50m. Most of the races have no handicap distance (Figure 7).

Handicap Type:
This feature defines the type of handicap penalty, which is of 3 types: HCP, CWT and SW. We will label encode this feature.

Race Start Time:
This is an important feature. It defines the exact time the race took place. We will sort our whole dataset by Race Start Time in Ascending order before training the model on each sample one by one in a timely order as the races occurred.

Start Type:
This feature defines the Start Type of each race: Auto-Start or Volte Start. We will be denoting Auto-Start with 0 and Volte with 1 to label encode this feature.

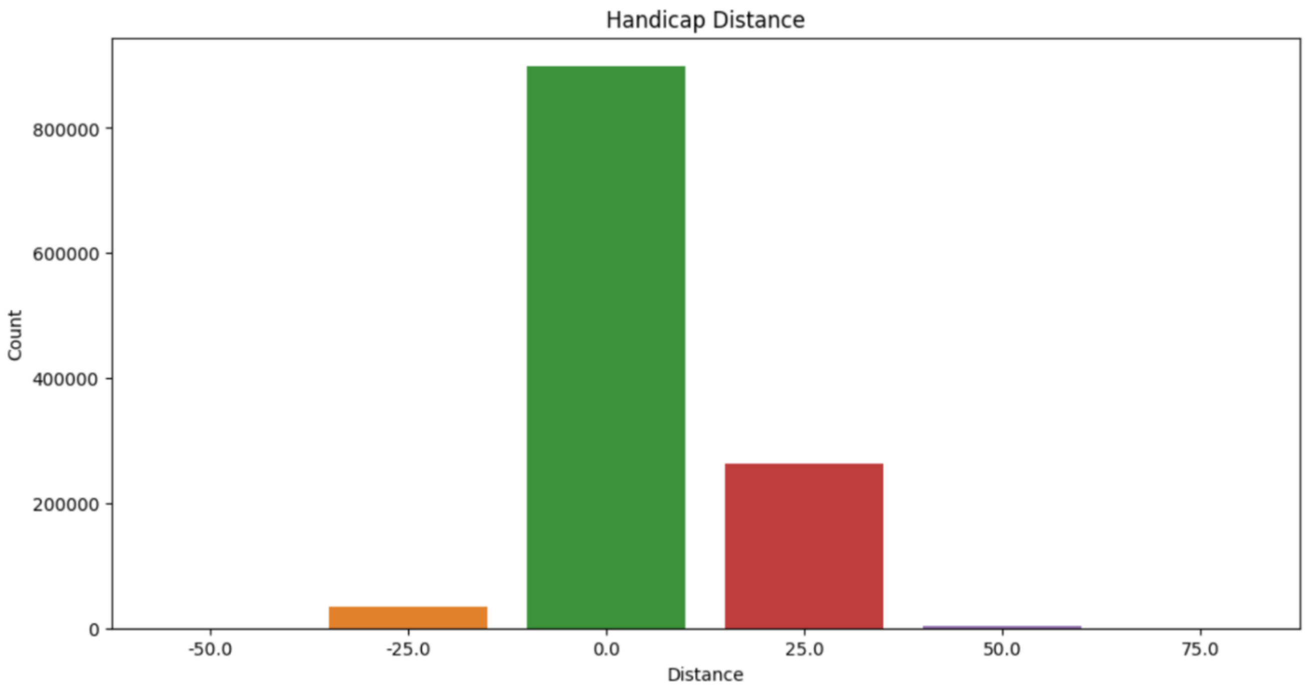


Figure 7: Handicap Distance Trend

Horse ID:
This ID defines the identification for each horse throughout the dataset.

Jockey ID:
This ID defines the identification for the Jockey riding the horse in each race. We will use this feature to extract valuable information related to Jockey's Past Performance.

Track ID:
This feature identifies each Track. We will not be using this feature as the Track's Surface and Past Performances will be captured in other available features as well.

Race ID:
This feature identifies each race uniquely.

Trainer ID:
This ID defines the identification for the Trainer training the horse for a given race. We will use this feature to extract valuable information related to Trainer's Past Performance with other horses.

Sire ID:
This ID defines the identification for the father of each horse. We will use this feature to extract valuable information related to Father's Children's Average Performance.

Saddle Cloth:
This feature defines the Type of Saddle used by each horse during a given race.

Wetness Scale:
This feature defines the wetness level of the ground during the race. This scale ranges from 3-9 in increments of 2. We will use this as a numerical feature.

Gender:
This feature defines the gender of each horse. We will be denoting Female with 0 and Male with 1 to label encode this feature.

Sex Restriction:
This feature defines the restriction on certain horse genders competing for the races. Most races do not have any restrictions (Figure 8).

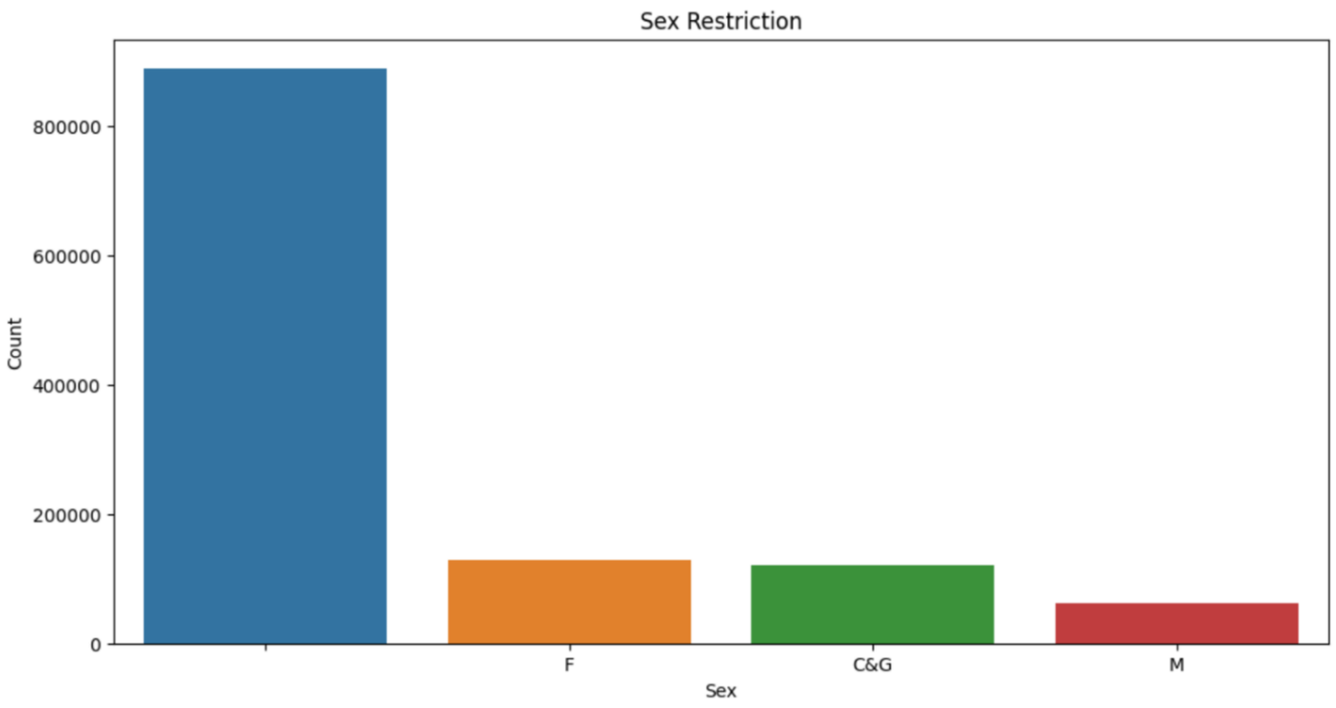


Figure 8: Sex Restriction Distribution

Race Performance Measurement Features

These features define the metrics on which each horse is evaluated to declare a final position in the race:

Finish Position:
This feature defines which position the horse finished in. In case the horse did not finish due to Breaking Stride, Not Participating or other reasons, the positions in these cases are denoted by a Two Character element. We will be encoding this disqualification measure separately and using this Finish Position feature as a numerical feature.

Beaten Margin:
This feature defines the length in metres by which a horse is beaten by the horse finishing first in each race. We will be using this feature as our Label and we will be aiming to predict the beaten margin for each horse in a race and aim to minimise the RMSE in predicting this metric for the races. We will use this instead of Finish Position as this feature helps more accurately measure the performance of the horse instead of only representing what position the horse finished in. This feature ranges from 0 to 350m.

Disqualified:
This feature defines whether the horse was disqualified from the given race.

Prize Money:
This feature defines how much prize money was won by the horse in the race in Dollars.

Race Overall Time:
This feature defines the total time from the start of the race to the end of the last horse finishing the race. This feature will be used as an indication towards the length of the race which must have a relation with the horse's stamina for the race.

No Front Cover:
This feature defines the number of front covers used by the horse in the race.

Position in Running:
This feature defines the starting position of each horse allotted before the race.

Wide Off Rail:
This feature defines the width of the racing strip allotted to the horse.

SECTION 4: *IMPUTING MISSING DATA*

There is no missing data in this dataset. This missing values in certain features are all Categorical Features and Null Values in these features are representing not falling into any category for these features, so we would simply One Hot Encode these features as they are, and the Null Values would then be considered a separate feature by themselves representing not falling into any criteria for these features.

SECTION 5: FEATURE ENGINEERING

Finish Position:

Any horse, who is not participating in a race is considered coming last in the race and hence if any horse is under this category, for Breaking Stride, we are marking its finishing position as 19, which is the last position in any race and similarly, if a horse is not participating, it is considered finishing even later in the race.

This would help us to not Encode the Finish Position feature as it would then become 25 more features, so by label encoding, we save 25 additional features for our model, by considering horse finishing in position 1 to be better as compared to horse finishing in position 2.

Finish Position 2 – Disqualification:

This feature represents the finishing position of each horse in a race from 1 to 20, given that some horses also disqualify by Breaking Stride or Not Participating, these horses are marked by Positions such as "BS", "NP", "UN", etc. To uniquely capture the information of a horse disqualifying in a race, we have created a New Feature named: "FinishPosition2"

This feature is "1", if the horse was Disqualified in each race and "0", if not. This is an important feature for the model as it would help the model understand why the Beaten Margin of that horse in each race was so high, because it was disqualified and hence it would factor in its effect while Predicting performance in the future races.

Beaten Margin:

In case a horse if disqualified for any reason, the Beaten Margin is populated as "999". While working with this same value, we realised that this value is pulling on the Mean value of Beaten Margin so significantly that the Root Mean Squared Error (RMSE) is too high for accurate predictions. To tackle this problem, we are first calculating the beaten margin of the last finishing horse in each race, and in case of any horse that has been disqualified, we are assigning it a new beaten margin value, which is:

= Max (Beaten Margin of the Last Horse in Each Race) + Standard Deviation (Beaten Margin for that race)

By using this, for example, if in a race, the beaten margin is 50 for the last horse, with a STD Deviation of 5, the assigned beaten margin to the losing horse would be 55. By performing this, the prediction on the final model become highly accurate and considerable.

Class Restriction:
This feature was a random string which we cleaned and made two features out of it:

Class Restriction Type:
NW, CF, CLM, etc. denote a particular class type restriction for each race and NW is most of the races.

Class Restriction Price:
This value defines the class entry price in dollars for each race.

Foaling Date:
This feature represents that date of birth of each horse. We will use this feature to predict the Age of the Horse in Days during each race.

Race Horse Count:
By Grouping the dataset on Race ID, we can fetch how many races participated in each race. We will use this as a numerical feature while training our dataset.

Dam Total Child:
This feature is calculated by grouping the dataset on Dam ID and calculating the count of unique Horse IDs to represent how many Children of the Dam participated in racing competitions which represents how well bread the Dam is.

Dam Total Child Races:
This feature is calculated by grouping the dataset on Dam ID and calculating the count of total races that its Children have participated till date. This is again to measure the Dam's genetic effect on the racing horse.

Sire Total Child:
This feature is calculated by grouping the dataset on Sire ID and calculating the count of unique Horse IDs to represent how many Children of the Sire participated in racing competitions which represents how well bread the Sire is.

Sire Total Child Races:
This feature is calculated by grouping the dataset on Sire ID and calculating the count of total races that its Children have participated till date. This is again to measure the Sire's genetic effect on the racing horse.

Jockey Total Races:
This feature represents how many total races the Jockey has participated in till date. This represents the experience of the Jockey.

Jockey Total Horses:
This feature represents how many total horses the Jockey has ridden till date. This also measures the versatility of the Jockey.

Jockey Mean Beaten Margin:
This feature measures the average Beaten Margin for the Jockey's past races and indicates the performance of the Jockey in previous races.

Trainer Total Races:
The feature represents how many races the Trainer has trained the horses for. This is an indicator of Trainer's experience.

Trainer Total Horses:
This feature represents how many Total Horses the Trainer has trained in the past, which also is an indicator of Trainer's Performance.

Trainer Mean Beaten Margin:
The feature measures the average Beaten Margin of the Trainer in the past races of the Horses that this trainer had trained.

PAST 3 RACE PERFORMANCE METRICS:

To train the model on the past racing results of each horse, we will use the Performance Metrics of the past 3 races as Input of the upcoming race. This would help the model to remember the horse's performance in the recent 3 races. However, we will still train the model on each race of each horse, but this would serve as a Buffer Memory for the model while predicting the Beaten Margin for the upcoming race.

DURATION SINCE PAST 3 RACES:

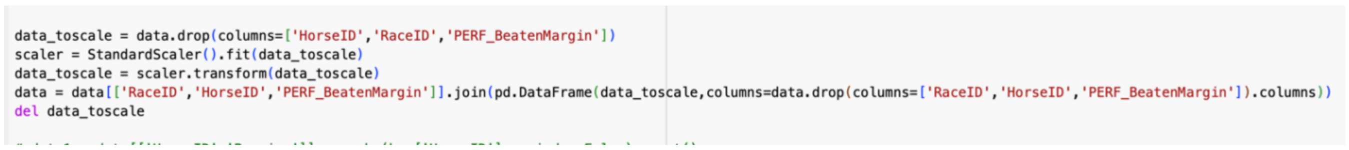
To indicate to the model, how long it has been since the horse's previous races, we will count the number of days from the previous 3 races to the current race day and measure how much recovery time the horse had in-between consecutive races in a competition.

SECTION 6: FEATURE SCALING

To feed the training data into the model, we would attempt to Normalise/Regularise the dataset so that the weights are equally distributed in the network which would improve the model's learning rate and would help the model to converge faster to the desired neuron weights.

A. Standard Scaler:

We attempted to scale the dataset using Standard Scaler (Code Block 1) before feeding it as inputs into the Neural Network input layer. The results obtained were significantly better as compared to not scaling the Training Features.



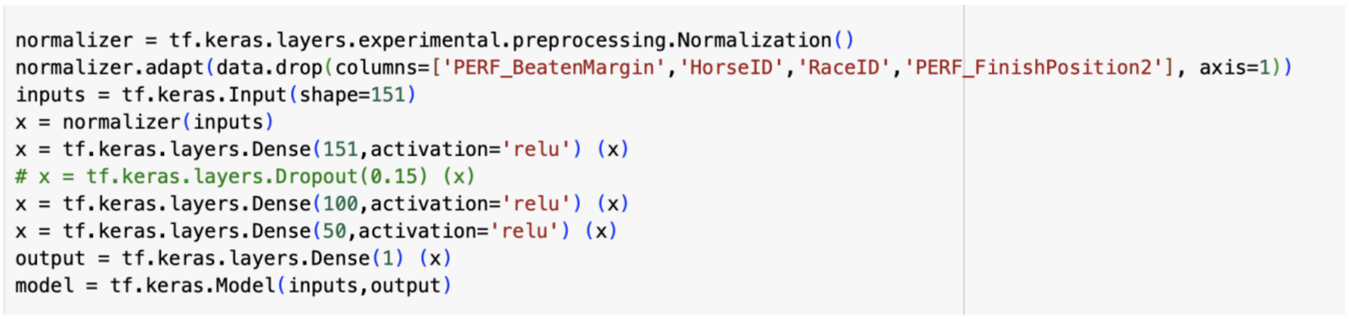
```
data_toscale = data.drop(columns=['HorseID','RaceID','PERF_BeatenMargin'])
scaler = StandardScaler().fit(data_toscale)
data_toscale = scaler.transform(data_toscale)
data = data[['RaceID','HorseID','PERF_BeatenMargin']].join(pd.DataFrame(data_toscale,columns=data.drop(columns=['RaceID','HorseID','PERF_BeatenMargin']).columns))
del data_toscale
```

Code Block 1: Standard Scaler Implementation

A. Normalizer:

We also attempted to scale the dataset by Normalization, which is offered within Keras Preprocessing Library. This Normalization technique, scales the data across each sample, i.e. row by row, as compared to Standard Scaler that scales the data Feature by Feature.

As seen in the second code block below, we first adapt a normalizer instance on our whole Training Data, then we feed the raw data points into our network whose first layer is a Normalization layer which was previously adapted to the whole training data (Code Block 2).



```
normalizer = tf.keras.layers.experimental.preprocessing.Normalization()
normalizer.adapt(data.drop(columns=['PERF_BeatenMargin','HorseID','RaceID','PERF_FinishPosition2'], axis=1))
inputs = tf.keras.Input(shape=151)
x = normalizer(inputs)
x = tf.keras.layers.Dense(151,activation='relu') (x)
# x = tf.keras.layers.Dropout(0.15) (x)
x = tf.keras.layers.Dense(100,activation='relu') (x)
x = tf.keras.layers.Dense(50,activation='relu') (x)
output = tf.keras.layers.Dense(1) (x)
model = tf.keras.Model(inputs,output)
```

Code Block 2: Keras Normalizer Implementation

The overall comparison of these two scaling techniques yielded nearly similar results with Keras Normalizer performing slightly better on our specific dataset which we will be using in our final model.

SECTION 7: SEGMENTATION

Once our Training Data is scaled, we will split the dataset by Horse ID. We are doing this to train a separate model for each horse. By doing this, we will train a unique model for every horse and that model will attempt to unique determine the correlation of each horse with the features we are feeding into the model.

Once this segmentation is done, we will separately train different models for each horse and aim to predict the Beaten Margin for that horse in upcoming races.

Section 8: Network Architecture

In designing our predictive model for French Trot Horse Racing outcomes, we opted for a multilayer feedforward neural network, a robust architecture known for its adaptability to complex patterns. The network's structure was determined through a combination of network pruning and growing methods, ultimately employing the network growing approach to enhance performance.

The network comprises five crucial layers (Figure 9):

Normalization Layer: Serving to normalize input data and facilitate efficient processing.

Input Layer: Each neuron within this layer corresponds to a distinct input signal, forming the foundation for data interpretation.

Three Hidden Layers: These layers consist of neurons dynamically adjusting to encapsulate intricate relationships within the data.

Output Layer: Each neuron in this layer corresponds to a specific output signal, representing the network's predictive output.

An important characteristic of our architecture is its full connectivity, ensuring that each node in any given layer is intricately linked to every other node in the immediately adjacent forward layer.

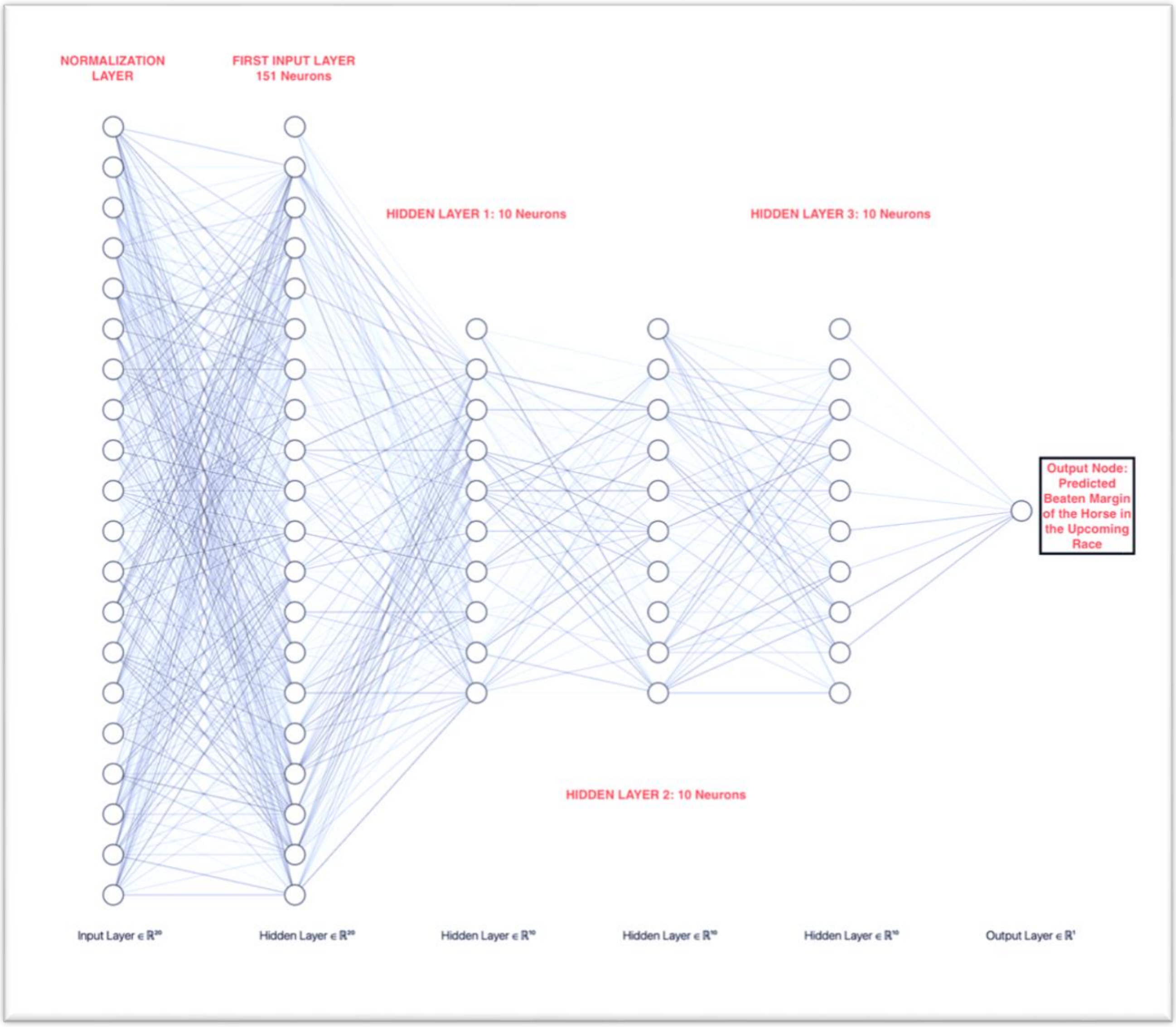

NORMALIZATION LAYER
FIRST INPUT LAYER 151 Neurons
HIDDEN LAYER 1: 10 Neurons
HIDDEN LAYER 3: 10 Neurons
HIDDEN LAYER 2: 10 Neurons
Output Node: Predicted Beaten Margin of the Horse in the Upcoming Race
Input Layer ∈ ℝ²⁰
Hidden Layer ∈ ℝ²⁰
Hidden Layer ∈ ℝ¹⁰
Hidden Layer ∈ ℝ¹⁰
Hidden Layer ∈ ℝ¹⁰
Output Layer ∈ ℝ¹

Figure 9: Network Architecture

Section 9: Network Tuning and Growing

To arrive at the optimal architecture, we used the network growing method. This approach involves incrementally introducing layers until we achieve a structure that yields satisfactory performance, as measured by minimizing the root mean square error (RMSE). Initial exploration involved testing a multilayer perceptron with a single hidden layer and varying numbers of neurons (10, 25, 50, 100).

Subsequently, additional layers were systematically added, and the predicted outcomes of these networks, each with different neuron configurations (e.g., [10,10], [10,25]), were calculated (Table 5).

The structure that emerged as the best performer, offered the minimum RMSE, featured 10 neurons in each of the three hidden layers.

Epochs	Neurons Layer 1	Neurons Layer 2	Neurons Layer 3	Mean of Deviation from Actual Finish Position for Top 3 Finishing Horses	Beaten Margin RMSE in Metres
20	100	100	10	-1.33	7.71
20	**10**	**10**	**10**	**-2.33**	**8.60**
50	10	50	50	-1.00	9.20
50	50	100	50	-3.00	8.15
20	25	10	100	-3.33	9.36
20	10	25	0	-3.67	9.30
50	25	100	100	-1.67	9.69
50	10	25	0	-3.67	9.33
50	50	100	0	-4.00	9.28
20	25	100	10	-3.00	9.71
20	25	25	10	-3.00	9.79
50	25	10	0	-3.00	9.83
20	10	50	10	-5.33	8.08
20	25	25	50	-3.00	9.95
20	25	100	50	-5.33	8.72
20	100	100	100	-5.33	8.87
50	25	10	100	-4.67	9.63
50	25	100	25	-4.67	9.67
50	10	50	100	-5.33	9.30
50	100	100	10	-3.67	9.95
20	10	100	50	-5.67	8.82
20	50	50	25	-5.33	9.38
50	100	25	50	-5.33	9.40
20	25	100	0	-3.00	10.18
50	10	10	10	-5.67	9.07
50	100	50	25	-4.33	10.02
50	25	10	50	-4.33	10.04
20	100	25	25	-3.33	10.25
20	25	10	25	-0.67	10.47
50	25	100	0	-4.33	10.17
20	25	25	0	-5.33	9.84
50	25	50	100	-5.00	9.97
50	100	25	25	-5.00	10.02
50	10	10	0	-6.00	9.62
50	50	50	50	-5.00	10.02
20	25	50	10	-5.67	9.83

Table 5: Model Evaluation Metrics after performing Grid Search

After performing a grid search over these parameters, the lowest RMSE was found on using:

1. A Normalization Layer
2. Input layer with 151 Neurons
3. 3 Hidden Layers with 10 Neurons each
4. Showing the data samples to the model in 20 EPOCHS
5. Using Batch Size of 1. i.e. Feeding each race sample one by one.

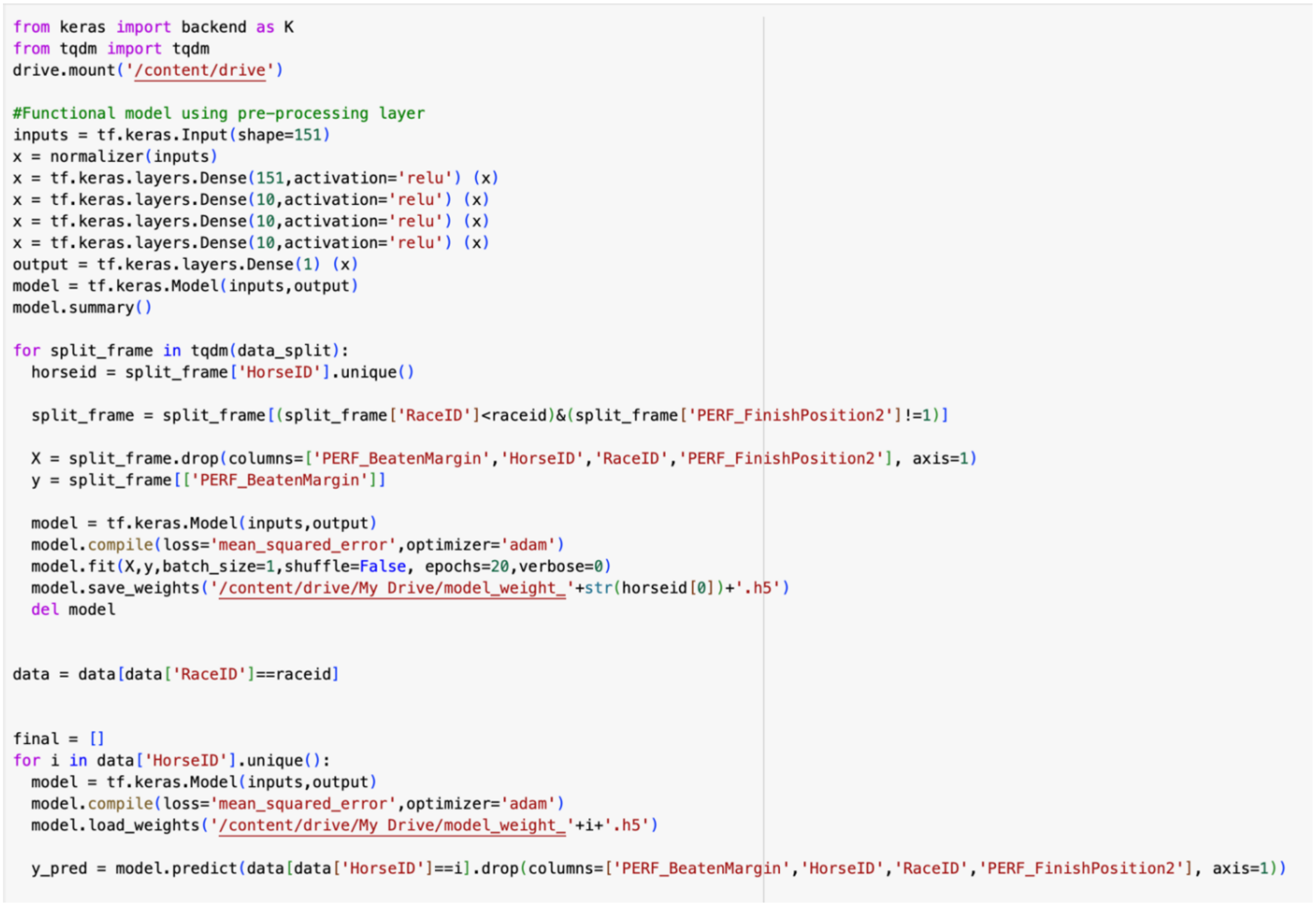
Also, while training the model, we only feed the race history of horses where the horse was NOT DISQUALIFIED. Given that we are already capturing the essence of Disqualification in past races in the PAST PERFORMANCE FEATURES while training our data, we do not need to train over samples where the horse was disqualified due to any reason.

This was done because if we consider the disqualification races, the Beaten Margin for these samples are very high, which pulls the mean significantly and the predictions were not accurate and were giving very high RMSE. After removing these races, the RMSE improved significantly.

SECTION 10: MODEL PREDICTION

To predict the Winner of any race using the model, we first select any given race using its Race ID. The model automatically fetches all the participating horses in that race, fetches the training data for the participating races only, trains the model on each horse separately and uses these trained models for each horse to predict the Beaten Margin for the Race ID we had selected (Code Block 3).

Once the predictions are calculated, basis the lowest predicted Beaten Margin, a Winner is predicted by the Model.



```
from keras import backend as K
from tqdm import tqdm
drive.mount('/content/drive')

#Functional model using pre-processing layer
inputs = tf.keras.Input(shape=151)
x = normalizer(inputs)
x = tf.keras.layers.Dense(151,activation='relu') (x)
x = tf.keras.layers.Dense(10,activation='relu') (x)
x = tf.keras.layers.Dense(10,activation='relu') (x)
x = tf.keras.layers.Dense(10,activation='relu') (x)
output = tf.keras.layers.Dense(1) (x)
model = tf.keras.Model(inputs,output)
model.summary()

for split_frame in tqdm(data_split):
  horseid = split_frame['HorseID'].unique()

  split_frame = split_frame[(split_frame['RaceID']<raceid)&(split_frame['PERF_FinishPosition2']!=1)]

  X = split_frame.drop(columns=['PERF_BeatenMargin','HorseID','RaceID','PERF_FinishPosition2'], axis=1)
  y = split_frame[['PERF_BeatenMargin']]

  model = tf.keras.Model(inputs,output)
  model.compile(loss='mean_squared_error',optimizer='adam')
  model.fit(X,y,batch_size=1,shuffle=False, epochs=20,verbose=0)
  model.save_weights('/content/drive/My Drive/model_weight_'+str(horseid[0])+'.h5')
  del model


data = data[data['RaceID']==raceid]


final = []
for i in data['HorseID'].unique():
  model = tf.keras.Model(inputs,output)
  model.compile(loss='mean_squared_error',optimizer='adam')
  model.load_weights('/content/drive/My Drive/model_weight_'+i+'.h5')

  y_pred = model.predict(data[data['HorseID']==i].drop(columns=['PERF_BeatenMargin','HorseID','RaceID','PERF_FinishPosition2'], axis=1))
```

Code Block 3:

Section 11: Model Evaluation

Upon performing a grid search over training parameters, we found that the lowest RMSE on Beaten Margin for a given Race was 8.15m and average deviation of Predicted Horse Position as compared to the actual Horse Finish Position of 3 positions.

This means that for all the horses in each race, we can predict the Beaten Margin for each horse by +- 8.15m and the position by +-3 positions.

Upon executing this project, we observed that Neural Networks can be used to observe patterns in Horse Racing trends with a significant RMSE value.

Section 12: Improvements

Given the computational power limitations, the current Model can only predict the Outcomes for One Race at a time. Once we feed the Race ID for any race, it only prepares and trains Neural Network models for only the participating horses (15-20 horses in any race) to make the predictions.

By using strong computing machines, we can train the model to feed more than just past 3 races as buffer data while training the model and train for all horses simultaneously and store the calculated weights for faster results for future predictions which would help improve the model without significant delays.

Section 13: Conclusion

During our project, performing an exhaustive grid search unveiled the optimal neural network configuration, featuring a Normalization Layer, an Input Layer with 151 Neurons, and 3 Hidden Layers, each housing 10 Neurons.

Noticeably, the incorporation of a 20-Epoch training period and a Batch Size of 1 demonstrated superior performance in minimizing Root Mean Squared Error (RMSE).

A critical refinement involved excluding races with horse disqualifications during training, leading to a significant enhancement in RMSE accuracy.

By selecting a race using its Race ID, the model dynamically fetches participating horses, trains on race histories, and predicts the Beaten Margin for each horse, ultimately identifying the winner based on the lowest predicted Beaten Margin.

Model evaluation showcased the effectiveness of our approach, with the lowest RMSE on Beaten Margin reaching 8.15m. The average deviation of Predicted Horse Position compared to the actual Horse Finish Position stood at approximately 3 positions, highlighting the model's ability to predict outcomes with notable accuracy.
While the project successfully demonstrated the utility of Neural Networks in discerning patterns within French Horse Trot Racing, there exist opportunities for improvement. Current computational limitations constrain predictions to one race at a time, urging consideration of stronger computing machines to enable simultaneous training for all horses and expedite future predictions.

In conclusion, the project underscores the potential of Neural Networks in horse racing prediction, emphasizing the critical role of structure, and learning algorithms, albeit acknowledging the need for extended training periods and increased computational power for further advancements.

References

1. Davoodi, Elnaz & Khanteymoori, Alireza. (2010): https://www.researchgate.net/publication/228847950_Horse_racing_prediction_using_artificial_neural_networks

2. Jones, A. et al. (2018) - ”Predicting Horse Racing Outcomes Using a Bayesian Network.” This study employs probabilistic modeling to predict horse racing outcomes, but it lacks focus on the unique characteristics of French trotters.