Roy Son
Student # 1005105690
mingyu.son@mail.utoronto.ca

Rishistav Ray
Student # 1006884301
rishi.ray@mail.utoronto.ca

PREDICTING ROULETTE

ABSTRACT

This document outlines the development process of Team 49's project - The Roulette Predictor. With artificial intelligence being incorporated more and more into our daily lives, our team wanted to explore how machine learning, specifically neural networks can influence the gambling industry. Specifically, we wanted to study and predict the outcomes of an online Roulette Simulator to evaluate its accuracy in terms of emulating real-world ball physics. –Total pages: 9

1 INTRODUCTION

1.1 ROULETTE

Roulette is a classic casino game that has become increasingly popular through online gambling. The game is played by spinning a wheel with numbered pockets and a small ball in the opposite direction. The objective is to predict the pocket in which the ball will land.

There are various types of bets that can be placed in roulette, each with different odds and payouts. Betting on a single number has the lowest odds of winning, with a probability of 2.63%, but has the highest payout of 35 to 1. Betting on a group of numbers, such as a row or a column, has higher odds of winning, with a probability of 5.41% and 8.11%, but a lower payout of 2 to 1 or 3 to 1, respectively. The expected value (EV) of a bet in roulette is calculated by multiplying the probability of winning by the payout and subtracting the probability of losing. For example, the EV of a $1 bet on a single number is calculated as follows: (35/1) x 0.0263 - 0.9737 = -$0.027 This means that on average, a player will lose $0.027 for every $1 bet placed on a single number.

1.2 MODEL GOAL

The goal of our project is to determine if it is possible to crack the exact pattern of the motion of the ball and predict exactly where it will land on the index wheel of the roulette wheel. This is an intriguing question, as in the process of creating a neural network, the neural network is simultaneously learning how to reproduce the behaviour of how a physical ball interacts with the environment. Hence the network is also acting as a physics simulator. Furthermore, it allows us to dive deeper into the inner workings of online Roulette simulators and how accurately they replicate the behaviour of an actual physical Roulette Wheel. Since we wish to track the motion of the ball and accordingly predict its future behaviour, it is ideal that we use Machine Learning to progressively learn how a ball's previous speed and location affect its future position and speed and a Recurrent Neural Network can help us achieve exactly that.

2 ILLUSTRATION

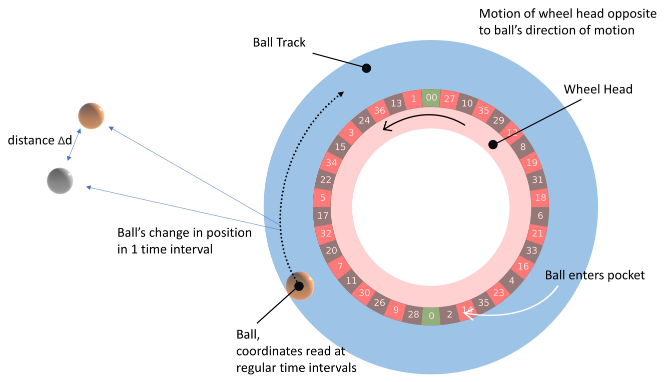

Motion of wheel head opposite to ball's direction of motion
Ball Track
Wheel Head
distance Δd
Ball's change in position in 1 time interval
Ball enters pocket
Ball, coordinates read at regular time intervals

Figure 1: Illustration of a simplified roulette wheel

2.1 DESCRIPTION

Before the ball is released, the wheel head will rotate with a random angular velocity ϕ within a specified range in accordance with the rules of Roulette. The ball will then be pushed into the system in a circular motion with an initial velocity v_i and a centripetal acceleration a_c towards the centre of the wheel head. The ball will gradually land inside one of the numbered pockets as shown in the illustration, indicating the output of the system.

We will track the (x,y) coordinates of the ball and calculate its instantaneous velocity and direction at each position. This will be achieved by taking readings of these coordinates at regular time intervals and dividing the distance between successive readings by the time interval to get the instantaneous velocities.

3 BACKGROUND AND RELATED WORK

3.1 FIRST PRIOR WORK

According to an academic research paper by David Abuhanna, it is possible to skew the odds of returning a positive average return on the Roulette based on statistical data. He attempted to prove that there does exist a relationship between initial conditions such as initial position and velocity.(Abuhanna, 2021)

3.2 SECOND PRIOR WORK

According to Richard Muller, a professor at the University of California, Berkeley, it is possible to increase the chances of a positive average outcome by 15%. They claimed it can be achieved by timing the change in rotational velocity of the ball and wheel separately to determine the range within which the ball will land. However, in this approach, the timing devices and their heuristics have to be calibrated on the specific Roulette Wheel. (Robers, 2016)

3.3 THIRD PRIOR WORK

The American mathematician Edward O. Thorp alongside his MIT colleague Claude Shannon theorized that given the information available during a roulette spin, the odds are no longer random.

They built a wearable computer that allowed them to predict a rough range of where the roulette ball will likely stop spinning.

The casino allows players a few seconds while the ball is still spinning to place bets. During that time, Thorp and Shannon could click a button that registers a full rotation of the spinning ball. After enough clicks a model can be made of the ball's velocity, angular momentum, and landing point. They claimed they achieved an EV of around 20%. (Thorp, 2017)

3.4 Fourth Prior Work

Wu Pinye published an article on Medium on the outcome of roulette based on reinforcement learning.

He used hyper-parameters based on the physical environment of the roulette wheel such as: - The dealer spinning the wheel - The dealer's hand position when spinning the ball - The spin force - The velocity

Wu Pinye ran 90,000 records for training and 10,000 for testing. The neural network returned the most profitable strategy that used betting on 2 to 1 combinations where you can bet on a "column" of the wheel. This returned $122,000 from an investment of $1000. (Pinye)

3.5 Fifth Prior Work

In 2012, inspired by Thorp's work, Michael Small and Chi Kong Tse published a paper in the American Institute of Physics called "Predicting the Outcome of roulette". In this paper, they introduced a model for the motion of a roulette wheel and ball can be sufficiently predicted with knowledge of initial position, velocity, and acceleration. In their paper, they were able to achieve an expected return of 18%.(Michael Small, 2012)

4 Data Processing

Since this project does not have a readily available dataset, the data extraction process had to be automated. This required Computer Vision tactics. The team began by automating the Roulette Simulator with random inputs. The bets don't matter as the inputs to our model will be the initial conditions of the ball which will be discussed in further detail, and the output is the slot number the ball lands on, on the index wheel. We screen-recorded these simulation runs and produced a dataset of 1000 videos. To process this dataset, we employed Computer Vision and frame-by-frame manipulation of the video. We took a single snapshot from the video and began by trying to identify the ball. In order to separate the ball from the image, we used a function called ColourFinder from the OpenCV package and exploited the ball's physical characteristics. We tuned the Hue, Saturation, and Value also known as HSV until the ball was the only thing on the screen. The mask can be seen below.

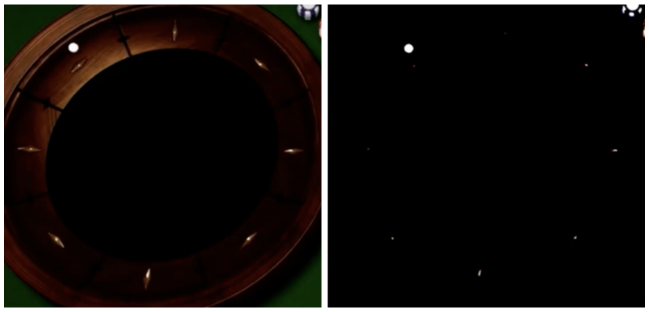


Figure 2: Images displaying roulette wheel before and after ColourFinder function, respectively

We then apply this mask to the frame and deployed the FindContours function to identify the ball by its circular shape. We combined our image manipulations and applied them to the complete video and dataset by automating the process. We also masked the index wheel to remove noise as it had the same colour as the ball and gave us irrelevant values. Next we tracked the (x,y) coordinates of the ball and stored its position at constant time intervals. In order to calculate the instantaneous velocity of the ball we calculated the distance between successive green dots and divide by the constant time interval between 2 green dots which was 100 milliseconds.

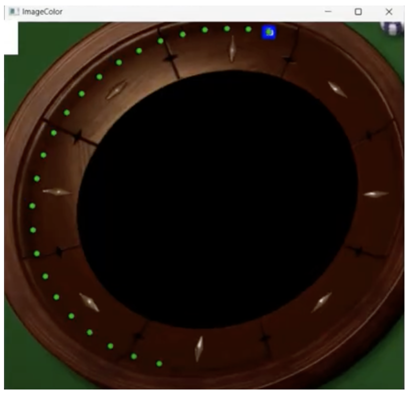


Figure 3: Green dots tracking ball positions

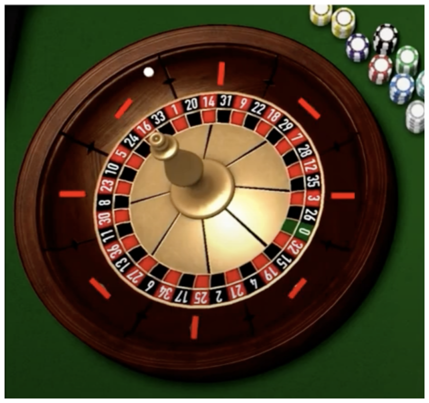


Figure 4: Ball deflector markers

Next, we set markers on the Ball deflectors and detected every time the ball is in the hitbox of a marker. If it was in the hitbox and changed direction relative to its initial direction, we store that in a one-hot encoded array. The purpose of designing the dataset this way is to use its point of collision and velocity at impact, to deploy a neural network that can find patterns in the ball's movement and final location.

Finally, to find out the output of our simulator we used a simple 2-layer CNN that recognizes the digits from the top of the screen trained on cropped images taken from the Roulette Simulator betting mat on the right. This was all a very time-consuming process but gives us a holistic dataset.

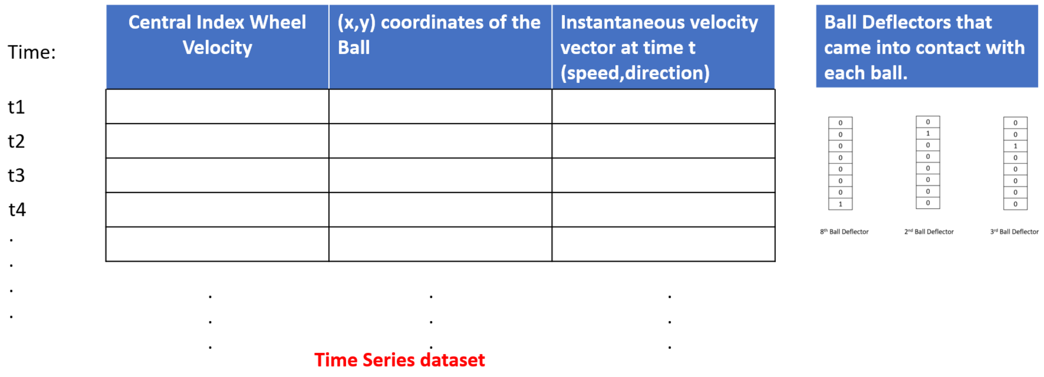

Time:
t1
t2
t3
t4
Central Index Wheel Velocity
(x,y) coordinates of the Ball
Instantaneous velocity vector at time t (speed,direction)
Ball Deflectors that came into contact with each ball.
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0
8th Ball Deflector
2nd Ball Deflector
3rd Ball Deflector
Time Series dataset

Figure 5: Dataset Format

5 Architecture

For the model, we chose a recurrent neural network because we needed to process sequential data, which in our case is a time-series array with additional one-hot encoded arrays. Besides an input vector, our model uses a hidden state vector, which represents the RNN's memory of previous inputs. Essentially, we are attempting to input each successive (x,y) coordinate of the ball as it moves along the Roulette to predict where it will land on the index wheel. We are also using a long-short-term memory or an LSTM network to make sure we conserve the initial velocity and position of the ball as we believe it plays an integral role in where the ball lands.

When developing the architecture, it's important to consider that our input parameters are time-series vectors. This means we pass a list of values measured every 100 milliseconds for each parameter.

These parameters are the (x,y) coordinates of the ball, the instantaneous velocity and direction of motion of the ball when the ball is at point (x,y) and the velocity of the centre index wheel, and one hot encoded vectors for the deflectors the ball collides with.

We begin by implementing a hidden embedding layer ('self.emb') which concatenates inputs of ball coordinates, wheel velocity and ball deflectors and maps them into an embedding space of "input_size + 8." Here "input_size" is 250 as that's the maximum number of tuples in each time-series vector. The "8" is due to dimensional expansion from the additional parameters like centre wheel velocity, the instantaneous velocity of the ball and one-hot encoded vectors. This ensures all the parameters are combined and ready for serving as inputs to our neural network.

We then incorporate a simple LSTM layer (self.rnn) with initial hidden states for each timestep which were initialized to zero. For our hidden LSTM layer, we used "hidden_size" as our hyperparameter to dictate the number of neurons in the hidden LSTM layer. We used the value 256 as our "hidden_size".

Finally, we use a linear layer (self.fc) which takes the last hidden state of the LSTM layer and maps it to 36-dimensional output space (the number of indices on the inner roulette wheel). To capture additional detail in ball movement, we also experimented with an additional LSTM layer, however, the increase in training time made it unfeasible for use. To summarize our model started with an embedding layer, then used two LSTM layers and then used a linear layer for the output.

6 Baseline Model

For our baseline model, we will employ the methodology proposed by Michael Small and Chi Kong Tse in their paper "Predicting the Outcome of Roulette." However, we will use a simple ANN instead of using mathematical equations to model the behaviour of the ball which is tedious and out of the scope of this course. Additionally, we will assume that we have readily available data as Michael Small and Chi Kong Tse haven't released their dataset. (Michael Small, 2012)

The baseline model will require the rotational velocity of the ball timed at the first revolution of the Roulette Wheel and also the Second Revolution of the Roulette Wheel. Similarly, we will need the rotational velocity of the centre index wheel timed at its first and second revolutions. We now have 4 input parameters. For each instance of this dataset with 4 inputs we will assume that we are given an output in terms of a range of numbers from the index wheel from 1 to 36 for example 2 to 7, 8 to 15 etc.

The architecture of the baseline model will use an input layer with 4 neurons, followed by 3 hidden linear layers with the ReLU activation function of size 50, and finally a 36-neuron output layer with the SoftMax activation function. Our output will be the longest consecutive set of activated neurons.

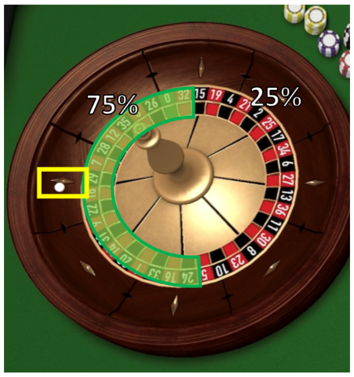
7 Quantitative and Qualitative Results

As mentioned in the presentation, the success rate of our model was 5% at its maximum. Clearly, the model is bad at its task of predicting Roulette outcomes. But in order to understand if the neural network is actually recognizing ball patterns and movements or is simply a network with garbage weights that is inadequately trained, we can deploy the following tactics to quantitatively measure how good or bad the model performs.

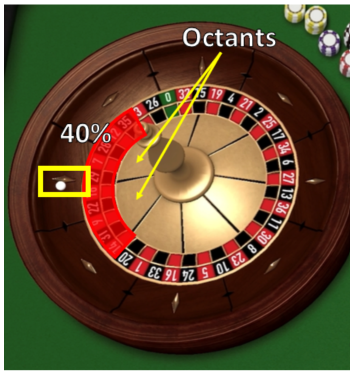
Since we utilized a baseline model that outputs ranges, we used a similar method to evaluate the model. We began by finding out the minimum and maximum values of the predictions for identical input values, which specifically includes the last ball deflector the ball collided with.

Basically, we ran the model on a single unseen data instance for 100 iterations and gathered a distribution of where the ball landed on the index wheel, taking the ball deflector it last collided with as a reference point. The following diagrams illustrate our findings.

The snapshots seen in the Figures are taken at the exact moment the ball collides with its last deflector. This allows us to see in which regions the ball tended to land taking the deflector as a reference. We can see from Figure 5(a), the area highlighted in green depicts the half of the Roulette wheel that is towards the deflector. After running the unseen data instance 100 times, we saw that 75 percent of the time the ball landed in the green region. Furthermore, out of the green region, 40% of the outcomes landed in the closest octants beside the ball deflector it collided with, shown by the red region in Figure 5(b).



(a) 75% landing on colliding deflector side



(b) 40% landing within octants beside deflector

Figure 6: Output distribution after 100 iterations

Finally, upon comparing it with the actual output, despite not being able to find a trend in the deviation between our model's output and the actual output, we found that 38% of the actual output lay in the red region and an 86% of outputs lay in the green region.

If we compare the model to the actual output distribution of the simulator, we can see approximately 10% error rate:

Region	Our Roulette Predictor	RouletteSimulator.net
Green	75%	86%
Red	0.4*75=30%	38%

Table 1: Output comparison

8 Evaluate Model on New Data

To evaluate the model on new data and push the model to its limit we deployed several tactics. As described above the model has a 5% accuracy rate in predicting single outcomes however is fairly close to the Roulette Simulator we used. Hence, we obtained more recordings of the Roulette Simulator as unseen data to test our model. In the first scenario we parallelly ran our model and the Roulette Simulator, and here we will the first 5 instances of our output and which regions the ball lands for either models:

Initial Velocity (m/s)	Red	Green
8.4	Yes	Yes
9.2	No	Yes
7.3	No	No
7.8	Yes	Yes
8.7	No	Yes

(a) Our Roulette Predictor

Initial Velocity (m/s)	Red	Green
8.4	Yes	Yes
9.2	No	Yes
7.3	No	No
7.8	Yes	Yes
8.7	No	Yes

(b) roulettesimulator.net

Table 2: Output region comparison based on first scenario

In the second scenario we parallelly ran both models specifically on scenarios where the ball was deflected from more than one deflector and these were our findings:

In the third scenario we ran both models parallelly on scenarios where the initial velocity of the ball was extremely slow, and the ball landed on a slot before 2 revolutions. These were our findings:

Initial Velocity (m/s)	Red	Green
1 and 2	Yes	Yes
4 and 6	Yes	Yes
1 and 3	No	No
5 and 6	Yes	Yes
7 and 8	Yes	Yes

(a) Our Roulette Predictor

Initial Velocity (m/s)	Red	Green
1 and 2	No	Yes
4 and 6	No	No
1 and 3	No	No
5 and 6	Yes	Yes
7 and 8	No	No

(b) roulettesimulator.net

Table 3: Output region comparison based on second scenario

Initial Velocity (m/s)	Red	Green
4.6	No	No
5.3	No	No
3.8	Yes	Yes
4.8	Yes	Yes
4.1	Yes	Yes

(a) Our Roulette Predictor

Initial Velocity (m/s)	Red	Green
4.6	Yes	Yes
5.3	No	Yes
3.8	No	Yes
4.8	No	Yes
4.1	No	Yes

(b) roulettesimulator.net

Table 4: Output region comparison based on third scenario

From the first scenario, we can conclude that the model has moderate performance within its comfortable range as we saw the regions are identical. Even though it has a 5% accuracy we can see that it fairly resembles the Roulette Simulator we used in terms of its outputs landing in red and green regions. It is important to note that this is only for the "ideal" values of initial velocities which are within the most occurring range of the Roulette Simulator.

The third scenario reinforces this. The models perform differently when the initial velocities are low compared to the usual average values of initial velocity in the Roulette Simulator and there is a mismatch between both models. The reason for this became evident when further analyzing our dataset. Picture this, when training a handwriting recognition model on the MNIST dataset we have an equal number of images for each alphabet. Similarly in our scenario, a high majority of the training data lies in the "normal" range and very few lie in the "outlier" range which explains the model's inability to simulate the motion of a slow-moving ball. To better train the model and make it holistic, we should have categorized our dataset based on different velocity ranges and had an equal number of instances for each range.

Finally, the second scenario is also due to the above problem as very few instances of our dataset have examples where the ball collides with more than one deflector.

All in all, we can conclude that the model can understand patterns and learn ball movement, paths and collisions fairly effectively, however, due to additional variables and outlier scenarios, it fails to narrow down the red and green regions.

9 Discussion

The accuracy of the model is 5% means the model fails to provide the correct slot the ball will land on 95% of the time. This accuracy clearly states that our model does not meet the initial goal of the project.

However, our team does not believe that our model was a failure. As shown from the performance on unseen data and repeated iterations on a single data instance (keeping input constant for each iteration), we saw that our model behaves similarly to the behaviour of the Roulette Simulator we used within a 10% error range for Scenario 1 which is the highest occurring scenario, even though our model is unable to match the performance for outlier values. From this, we can safely conclude that the neural network was effectively trained and can successfully understand how the ball moves, behaves and interacts with the Roulette Wheel. Comparing the outcome distribution regions (Figure 6) we can also conclude that the neural networks is actively trying to reduce ambiguity and not just making random guesses. The output data can be clearly visualized and seen to be skewed towards certain outcomes with reference to the deflectors.

There's a lot that can be learned from this model. Firstly when approaching such a monumental problem, it is better to take a top-down approach and ask broader questions. Instead of attempting to correctly predict a specific slot perhaps a more insightful question would be - "is it possible to predict a range of values given a ball's initial conditions?" or "what role does the deflector play in the ball's outcome?" and even "how do the center wheel and its relative velocity to the ball affect the deviation of where the ball lands?"

Therefore if we had asked these questions, we could have utilized the impact velocity of the ball with defectors and the position of impact as input parameters to the network which would eliminate the need of a time-series vector and a complex RNN that was extremely time-consuming to train because of the massive number of input data points. We could have had a much more effective neural network without having high training times.

Furthermore from the regional analysis demonstrated in Figure 6, we can conclude that the ball's interaction with a deflector plays a much bigger role in determining where it will land compared to its initial velocities. We can see this from Table 4 which shows the inconsistency in our model for less-than-ideal initial values.

Finally, we would like to point out that this was an ambitious project and we knew starting on that it was a near-impossible task. This is a popular problem that a huge number of people have tried to crack unsuccessfully before, but as prospective engineers, it was a great learning experience given that despite not having any pre-made datasets and a 5 percent final accuracy of the model, we found an insightful result in the end.

10 Ethical Considerations

The majority of casinos have strict rules against using any form of artificial intelligence while in a game and legally the Alcohol and Gaming Commission of Ontario(AGCO) has defined cheating at play as " Every person who, with intent to defraud any person, cheats while playing a game or in holding the stakes for a game or in betting". (of Canada, 2019)

However, using deep learning to predict roulette outcomes for research purposes without actual gambling is ethical. Our data set consists of pre-recorded roulette games as such it will not be considered "at play" nor will any predictions made have any effect on any person or entity.

11 Project Difficulty / Quality

The project we chose was high in terms of difficulty due to the reasons mentioned in the Discussion section. Initially, it overwhelmingly exceeded the team's expectations in multiple areas which greatly affected the quality and performance of the model.

11.1 Data Collection

The team expected to collect data from online resources, however, the online projects were secretive in terms of their datasets and all data sources were private. Therefore the team generated its own dataset through the automation of Computer Vision tactics highlighted in the data processing section. This was an extremely time-consuming endeavour and the team had to use close to 20 automation cycles for generating and optimizing the automation process to generate the 1000 tuples, ensuring the data was clean and accurate.

11.2 Physics Engine of Online Simulator

Online simulators do not accurately replicate the conditions of the real casino roulette wheels as the ball's motion will depend on the accuracy and precision of the physics engine used to simulate ball physics and environment interaction. Additionally, since online simulators are programmed using algorithms that are designed to produce random outputs, they may be intentionally skewed toward producing unnatural randomness that may not necessarily arise from the physical characteristics and initial conditions of the ball. To perfectly replicate the randomness the simulator should be transparent about the physics equations and graphics engine so we satisfactorily conclude that it's an unbiased Roulette Simulator.

11.3 Complexity of Roulette

There are many factors that can affect the outcome of a roulette spin, such as the ball's speed, mass, rotation of the wheel, air resistance, and friction between the ball and the wheel. These factors make it extremely difficult to develop a model that can accurately predict the outcome because of the lack of information about the mass of the ball etc. We have already highlighted this limitation in the presentation where we mentioned modelling circular motion as linear motion over a small distance, which is not physically accurate in the real world.

11.4 Data Processing

Due to having a dataset with a high number of data points and parameters for each roll, the processing of the dataset into our model resulted in excessive training time. Since our dataset contains 250 coordinates in each time series vector and 1000 such tuples in the dataset, it caused our model to take multiple hours to train.

11.5 Quality

The quality of our model, as mentioned in section 9, it did not perform well to output accurate predictions of the roulette. However, given multiple factors that increased the difficulty of the project, despite the low accuracy of the model, the team was able to learn and obtain ideas for improving and modifying the model to better suit for calculating predictions. We are also able to find out effective ranges for the output that fairly replicated the output ranges of the roulette simulator.

References

David Abuhanna. Exploring machine learning techniques for predicting the outcome of roulette, 2021. URL `https://digitalscholarship.unlv.edu/cgi/viewcontent.cgiarticle=1026&context=honors_theses`.

Chi Kong Tse Michael Small. Predicting the outcome of roulette. *AIP Chaos*, 2012.

Government of Canada. Disorderly houses, gaming and betting (continued), 2019. URL `https://laws.justice.gc.ca/eng/acts/C-46/page-31.html`.

Wu Pinye. An income back test of three roulette strategies based on reinforcement learning. URL `https://medium.com/ai-academy-taiwan/an-income-back-test-of-three-roulette-strategies-based-on-reinforcement-learning-3`

Brian Robers. Strange but true: How physicists win at roulette, 2016. URL `https://www.engadget.com/2016-08-10-strange-but-true-how-physicists-win-at-roulette.html`.

Edward O. Thorp. *A Man for All Markets*, volume 1. 2017.