King Fahd University of Petroleum and Minerals
College of Sciences
Department of Mathematics and Statistics

MX in Computational Analytics
Math 619: Project

Final Report

Project Title	Simulation and Predictive Modeling for Amusement Parks	
Student Name	**ID**	**Turnitin Score**
Mohammed AlOthimeen	201328390	10 %
Aans Alzahrani	201420780	10 %

Supervisor

Dr. Yahya Osais
Computer Engineering Department

12-08-2021

Abstract

The aim of this project is to justify constructing an IT infrastructure consisting of a smart bracelet and a mobile app via simulating Al-Qiddiya amusement park as a queueing system. The simulation model presented assumes 3 lands and 9 attractions are contained in the Qiddiya amusement park. The reason for using simulation is the lack of historical data due to the Qiddiya amusement park being under construction currently. Two types of simulation runs will be presented, first a simulation run with visitors choosing random lands and attractions, the second run with visitors being directed to lands and attractions. The second run is supposed to improve the operations in the Qiddiya amusement park, where it utilizes a Neural Network predictor model to predict the least wait time viewed by customer before entering a land or an attraction, essentially determining the lands and attractions visitors shall attend. The simulation is implemented via a Python program. The results are statistically analyzed and a comparison of statistical data, as well as performance metrics, is presented between random choice vs. Neural Network predictor choice. In conclusion, the implementation of the proposed IT infrastructure proved to have long term benefits to the operation team and management team, and the utilization of the proposed Neural Network predictor model for land and attraction choice has effectively reduced congestion in the waiting queues for attractions.

Keywords:
Amusement park – Theme park – Queuing system – Python – Qiddiya – Tourism – 2030 Vision – Saudi Arabia – Neural Network – Predict

Table of Contents

List of Figures

List of Tables

1. Introduction

In this project, we shall simulate the Qiddiya Amusement Park using queueing system theory, and proposing the direction of visitors to be handled via a Neural Network (NN for short) predictor model. First, the inner workings of the queueing system will be fully described, and a reduced event graph is inspected to have a better understanding of the full picture of the simulation program. A total of 5 random choice simulation runs are started to create initial data, which we can initially analyze. Statistical analysis and performance metrics are compared between random choice model and the NN predictor model choice model to accurately estimate the benefits of utilizing the NN predictor model for choosing lands and attractions for customers.

2. Problem Statement

The Qiddiya amusement park is currently under construction, and our job is to find out whether providing an IT infrastructure is a justified investment. The IT infrastructure consists of:

- Smart Bracelet
  - Utilizes Radio Frequency Identification (RFID) technologies
  - Used for tracking and capturing data
- Smart Phone Application
  - Online ticket
  - Physical map
  - View expected wait time

The benefit one gets is apparent in the historical data one can gather from the park. This historical data is further analyzed in this report to further and uses for this data will be explored, such as directing visitors inside the amusement park.

The main scope of this project consists of constructing a queueing simulation model similar to one found in amusement parks. The reason a simulation model was chosen is because the park is currently under construction, meaning no operational data can be gathered. The reason for using a queueing model is that a theme park's main functions are analogous to servers in the queueing system theory, and queue delay in an amusement park attraction is a crucial metric to analyze the performance of the amusement park in question. A neural network predictor is used to predict the wait time of customers in attractions, as well as predict the total time spent in a specific land. This NN predictor can be further utilized to serve the amusement park regarding the direction of visitors to specific lands and attractions. This direction of visitors results in more uniform spread of customers across the amusement park, and reduces congestion inside the waiting queues, which leads to higher utilization of theme park attractions and/or facilities, ultimately leading to greater profit.

3. Literature Review

From recent studies describing the processes in theme parks, Liou [1] has constructed and tested a complex scheme for the Theme Park Queueing System (TPQS). Liou also proposed the idea of implementing a NFC card reader – analogous to our RFID bracelet – in the amusement park.

Another study conducted by Athanasios I. Kyritsis and Michel Deriaz [2] have created a neural network-based model that predicts the client's waiting time in a bank. The features used in his study were 4 total features, 3 regarding time and the last feature was the number of clients waiting in the queue at an instance. These features inspired the feature selection for our neural network predictor models.

Data.world [3] provided information regarding the attraction service times and attraction capacitates, of which 9 attractions from Walt Disney World were chosen as parameters to our simulation model.

4. Project Plan

The main tasks are as follows:

- Literature reading and review – Mohammed and Anas

- simulation model of one land and 3 attractions, with arbitrary simulation parameters – Anas
- Gather real world data from online resources – Mohammed
- Extend the initial 1 land random choice model to 3 lands random choice, each land contains 3 attractions with a total of 9 attractions overall, with real world inspired input parameters – Mohammed and Anas
- Define performance metrics and statistical data to be gathered – Mohammed
- Create a Neural Network predictor model to predict waiting times – Anas
- Implementing the Neural Network predictor model in choosing the lands and attractions – Anas
- Comparing the performance data of the random choice model and NN choice model – Mohammed
- Final Presentation – Mohammed and Anas
- Final Report – Mohammed and Anas

A detailed Gant chart is shown below:

Work Breakdown Structure			**Responsibility Assignment Matrix**		**Gantt Chart**																			
Activity/Milestone			**Personnel** (P: Primary Responsibility, S: Secondary Responsibility)		**Time (in weeks)** Start from April 2021 till August 2021																			
No.	**Title**		**Anas**	**Mohammed**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	Start		P	P	■																			
2	Preparation				■																			
2.1		Brain storming ideas	P	P	■	■																		
2.2		Literature reading and review	P	P		■	■	■	■	■														
		Advisor consulting	P	P			■	■	■	■	■													
3	Phase#1																							
3.1		Defining the scope	P	P						■	■	■												
3.2		Random choice 1L, 3Atts	P	S							■	■	■											
3.3		Statistical Analysis	S	P								■	■											
3.4		Advisor consulting									■	■	■											
4	Phase#2																							
4.1		Define performance metrics & Stat. data	S	P									■	■	■									
4.2		Extend to 3L, 9Atts for random choice	P	P										■	■	■	■	■	■					
4.3		Advisor consulting	P	P										■	■	■	■	■	■	■	■			
5	Phase#3																							
5.1		NN Predictor	P	P														■	■	■	■			
5.2		Implementing NN choice	S	P																	■	■		
5.3		Parameter tweaking	P	S																			■	
5.4		Comparison	P	P																			■	■
6	Final phase																							
6.1		Presentation	S	P																	■	■	■	■
6.2		Finalizing	P	S																		■	■	■
6.3		Final Report	P	P																			■	■

Figure 1: Gantt chart

5. Completed Work

- Define events, state variables and activities

Event	Event Abbr.
Theme park Arrival	TPA
Rejection	Rej
Security Check Start	Sec.S
Security Check End	Sec.E
Theme park Enter	TPE
Enter land #	EL#
Enter Queue Attr. ##	EQA##
Start Attraction ##	SA##
Depart Attraction	DA##
Depart land #	DL#
Theme park Dep.	TPD

State Variable	Abbr.	Range of values
Customers in theme park	cust_in_TP	[0, CTP] – Discrete
Security Check queue length	Sec.Check_q	[0, CTP] – Discrete
Customers in queue of Attr. #	q_length_a##	[0, CTP] – Discrete
Attraction ## status – busy or not	server_busy_a##	[True, False] – Discrete
Customers in Lands#	cust_in_L#	[0, CTP] – Discrete

Activity	Delimiting Events
Customer Waiting	EQA## - SA##
Security check	Sec.start – Sec.end
Customer Entertained	SA## - DA##

Table 1: Events, state variables and activities

- Event Graph

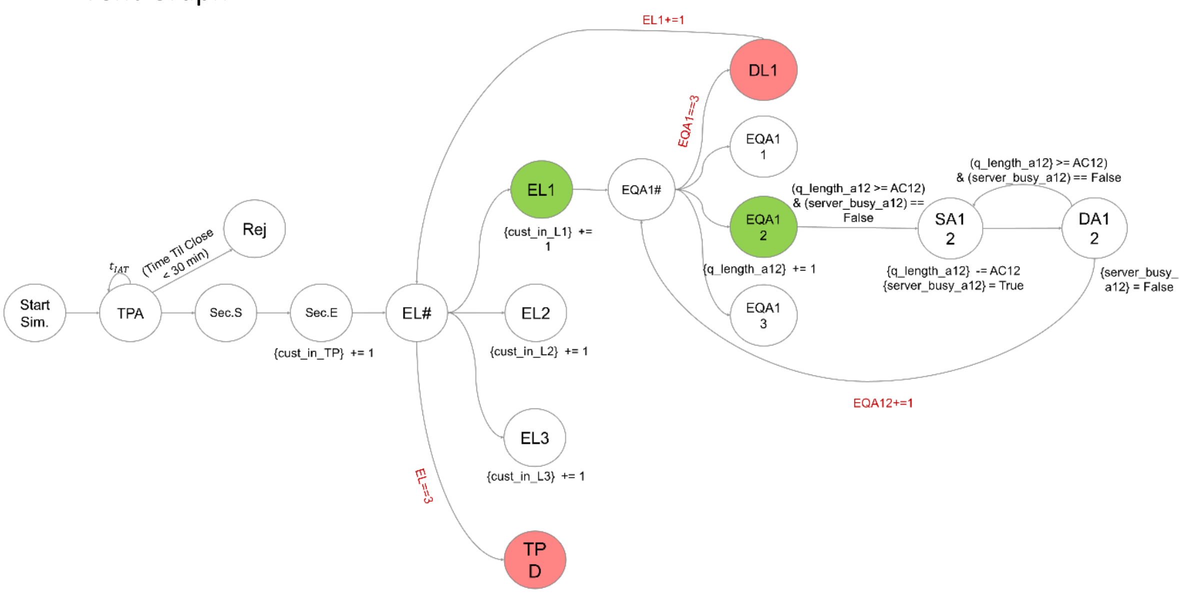


Figure 2: Event Graph

- Coding the Simulation Program in Python

```
if __name__ == '__main__':
    main()
```

```
334   118.82  Security Check 1 End
335   118.82  Land Choice for Customer#(58)
336   118.82  Departure from Theme Park, #(58)
337   119.93  Arrival to Theme Park
338   119.93  Security Check 1 Start
339   120.61  Arrival to Theme Park
340   120.61  Security Check 1 Start
341   120.95  Security Check 1 End
342   120.95  Land Choice for Customer#(59)
343   120.95  Departure from Theme Park, #(59)
344   120.97  Security Check 1 End
345   120.97  Land Choice for Customer#(60)
346   120.97  Customer Enters Land 1, #(60)
347   120.97  Attraction 1x Choice for Customer#(60)
348   120.97  Customer Enters Attraction 11, #(60)
349   121.61  Arrival to Theme Park
350   121.61  Security Check 1 Start
351   122.99  Security Check 1 End
352   122.99  Land Choice for Customer#(61)
353   122.99  Departure from Theme Park, #(61)
```

Figure 3: Snippet of event list

- Calculating the performance metrics

```
#Calculate wait_times for each attractions
for i in range(1, LN+1):
    Lx = 'L'+str(i)
    wait_key = 'wait_' + Lx # Wait  times for each attraction.
    arrs_key = 'arrs_' + Lx
    deps_key = 'deps_' + Lx
    arrs = out_var[arrs_key]
    deps = out_var[deps_key]
    out_var[wait_key] = calc_wait_L(arrs, deps)
    print('out_var[\'' + wait_key + '\'] created. (Index#, ID#, custs in land, wait time, time of arrival)' )
    for j in range(1, AN+1):
        Axx = 'A'+str(i)+str(j)
        wait_key = 'wait_' + Axx # Wait  times for each attraction.
        arrs_key = 'arrs_' + Axx
        deps_key = 'deps_' + Axx
        arrs = out_var[arrs_key]
        deps = out_var[deps_key]
        ST = ST_param[i-1][j-1]
        out_var[wait_key] = calc_wait(arrs, deps, ST)
        print('out_var[\'' + wait_key + '\'] created. (Index#, ID#, queue at entry, wait time, time of arrival)' )
```

```
out_var['Times_in_TP'] has already been created (Index#, ID#, time in TP)
All times  are in minutes
out_var['wait_L1'] created. (Index#, ID#, custs in land, wait time, time of arrival)
out_var['wait_A11'] created. (Index#, ID#, queue at entry, wait time, time of arrival)
out_var['wait_A12'] created. (Index#, ID#, queue at entry, wait time, time of arrival)
out_var['wait_A13'] created. (Index#, ID#, queue at entry, wait time, time of arrival)
out_var['wait_L2'] created. (Index#, ID#, custs in land, wait time, time of arrival)
out_var['wait_A21'] created. (Index#, ID#, queue at entry, wait time, time of arrival)
out_var['wait_A22'] created. (Index#, ID#, queue at entry, wait time, time of arrival)
out_var['wait_A23'] created. (Index#, ID#, queue at entry, wait time, time of arrival)
out_var['wait_L3'] created. (Index#, ID#, custs in land, wait time, time of arrival)
out_var['wait_A31'] created. (Index#, ID#, queue at entry, wait time, time of arrival)
out_var['wait_A32'] created. (Index#, ID#, queue at entry, wait time, time of arrival)
out_var['wait_A33'] created. (Index#, ID#, queue at entry, wait time, time of arrival)
```

Figure 4: Wait time calculations

In [40]:

```python
import math
def plot_q_density_combined(SP):
    global q_density
    q_density = {}
    color = ['b', 'g', 'r', 'c', 'm', 'y', 'lime', 'orange', 'k']
    plt.title('Q Density at each ride')
    plt.xlabel('Time (Hour)')
    plt.ylabel('Q/AC')
    time_list = vectorize_list(SP, 1)
    time_list_min = [x / 3600 for x in time_list]
    for i in range(1, LN+1):
        for j in range(1, AN+1):
            index = (i-1)*(LN) + j
            Axx = 'A'+str(i)+str(j)
            q_density[Axx] = vectorize_list(SP, 2+index)
            plt.plot(time_list_min, q_density[Axx], color = color[index-1])
            print(color[index-1], '->', Axx)
def plot_qd_bar(SP):
    qdl = []
    qdl_titles = []
    for i in range(1, LN+1):
        for j in range(1, AN+1):
            index = (i-1)*(LN) + j
            xx = str(i) + str(j)
            Axx = 'A'+xx
            q_density[Axx] = vectorize_list(SP, 2+index)
            maxqd = max(q_density[Axx])
            qdl.append(maxqd)
            qdl_titles.append(Axx)
```

Figure 5: Queue density code snippet

- Vectorization of inputs to NN

Data Vectorization - Lands

```python
trans_th = 1.5*60 #(Transient Phase Threshold, meaning only times after this are considered in )

q_L1 = vectorize_list(out_var['wait_L1'], 2)
q_L2 = vectorize_list(out_var['wait_L2'], 2)
q_L3 = vectorize_list(out_var['wait_L3'], 2)

t_L1 = vectorize_list(out_var['wait_L1'], 4)
t_L2 = vectorize_list(out_var['wait_L2'], 4)
t_L3 = vectorize_list(out_var['wait_L3'], 4)

x_L1 = []
x_L2 = []
x_L3 = []

y_L1 = []
y_L2 = []
y_L3 = []

for i in range(len(q_L1)):
    if t_L1[i] > trans_th:
        x_L1.append([int(q_L1[i]), int(round(t_L1[i]))])
        y_L1.append(int(round(out_var['wait_L1'][i][3])))

for i in range(len(q_L2)):
    if t_L2[i] > trans_th:
        x_L2.append([int(q_L2[i]), int(round(t_L2[i]))])
        y_L2.append(int(round(out_var['wait_L2'][i][3])))

for i in range(len(q_L3)):
    if t_L3[i] > trans_th:
        x_L3.append([int(q_L3[i]), int(round(t_L3[i]))])
        y_L3.append(int(round(out_var['wait_L3'][i][3])))
```

Figure 6: Data vectorization for lands

6. Computational Experimentation

Random Simulation Run

For a random choice simulation run, the input parameters are as follows:

```
Input Parameters:
_lambda -> 0.15
_STsec -> 0.2
_TBA -> 0.0016666666666666668
_TBL -> 0.0011111111111111111
Tot_Sim_Time -> 36000
TTC_TH -> 7200
ST11 -> 660
AC11 -> 220
ST12 -> 195.0
AC12 -> 46
ST13 -> 150.0
AC13 -> 30
ST21 -> 1245.0
AC21 -> 166
ST22 -> 120
AC22 -> 47
ST23 -> 165.0
AC23 -> 37
ST31 -> 210.0
AC31 -> 140
ST32 -> 150.0
AC32 -> 50
ST33 -> 285.0
AC33 -> 40
Seed ->  None
Sim End Time =  14.2 Hours
Time to simulate =  8.176138799999997 seconds
```

a) Queue Density

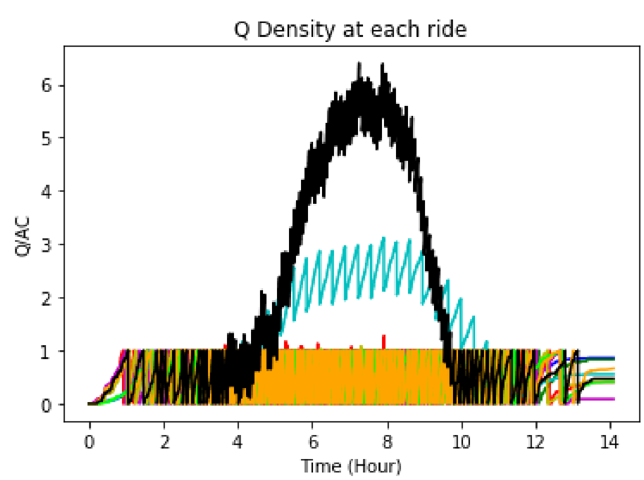


Figure 7: Queue Density of multiple attractions combined

Queue density is the queue length at an attraction at given time, divided by the attraction capacity, Q/AC.

b) Max Queue Density

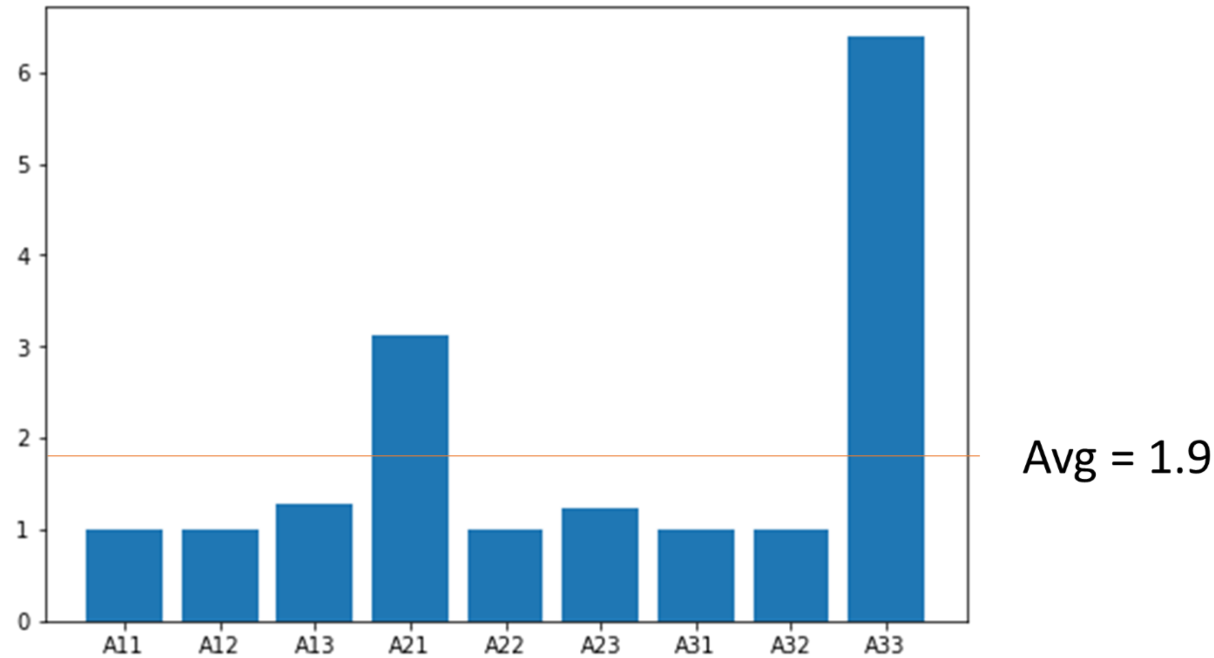
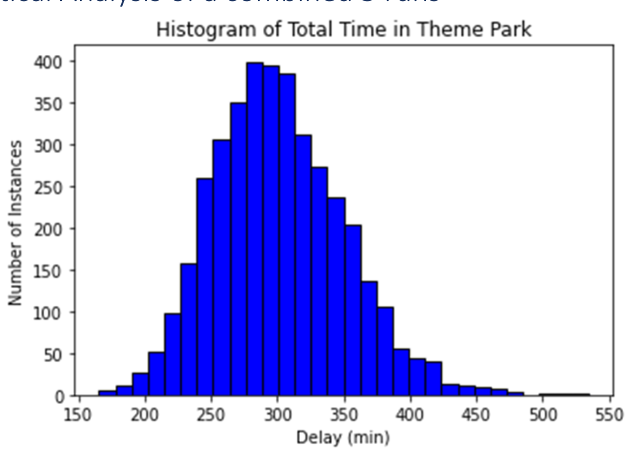


Figure 8: Maximum Queue Density for each Attraction, random model

c) Statistical Analysis of a combined 5 runs




```
Random Simulation Run:
mean =  301.02 - Minutes
stdev =  49.31 - Minutes

Min time spent in park =  165 Minutes
Max time spent in park =  534 Minutes
```

Figure 9: Histogram of total time spent in the theme park, with mean, std, min and max

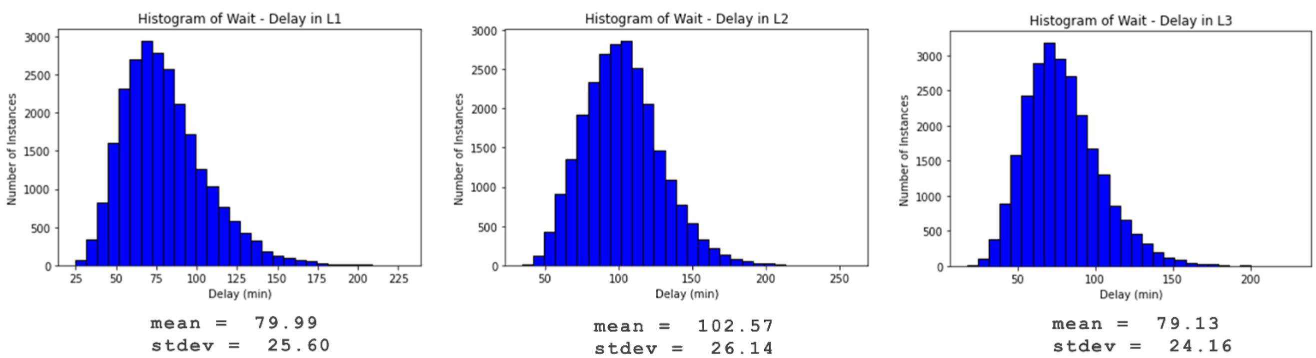


Figure 10: Histograms of total time spent in each land, with mean and std

Neural Network Construction:

There are two main NN models:

Neural Network Input features

a) For Land Time Predictor:

1. Customers in land at time of Entry
2. Entry Time in minutes

b) For Land Time Predictor:

1. Entry Time of day in minutes
2. Service time of Attraction in seconds
3. Attraction Capacity

Land Time Predictor (LTP)

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 2)                 6
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1536
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 65
=================================================================
Total params: 174,087
Trainable params: 174,087
Non-trainable params: 0
_________________________________________________________________
None
```

Attraction Wait Time Predictor (AWT)

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_18 (Dense)             (None, 3)                 12
_________________________________________________________________
dense_19 (Dense)             (None, 512)               2048
_________________________________________________________________
dense_20 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_21 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_22 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_23 (Dense)             (None, 1)                 65
=================================================================
Total params: 174,605
Trainable params: 174,605
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 11: NN models summary

NN Simulation Run

For the NN choice simulation run, the input parameters are as follows:

```
Input Parameters:
_lambda -> 0.15
_STsec -> 0.2
_TBA -> 0.0016666666666666668
_TBL -> 0.0011111111111111111
Tot_Sim_Time -> 36000
TTC_TH -> 7200
ST11 -> 660
AC11 -> 220
ST12 -> 195.0
AC12 -> 46
ST13 -> 150.0
AC13 -> 30
ST21 -> 1245.0
AC21 -> 166
ST22 -> 120
AC22 -> 47
ST23 -> 165.0
AC23 -> 37
ST31 -> 210.0
AC31 -> 140
ST32 -> 150.0
AC32 -> 50
ST33 -> 285.0
AC33 -> 40
Seed ->  None
Sim End Time =  14.68 Hours
Time to simulate =  201.5766910000002 seconds
```

Notice how the actual time for simulation increased greatly due to the many calls to the NN.

A. Queue Density

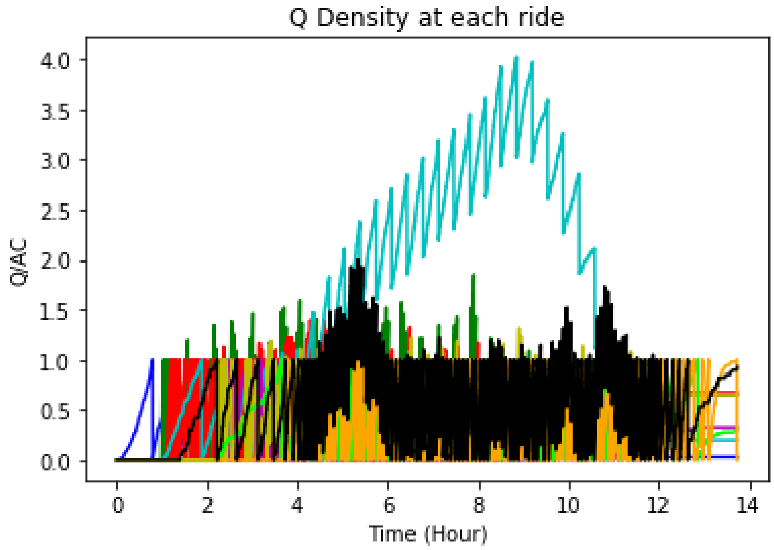


Figure 12: Queue Density of multiple attractions combined

B. Max Queue Density

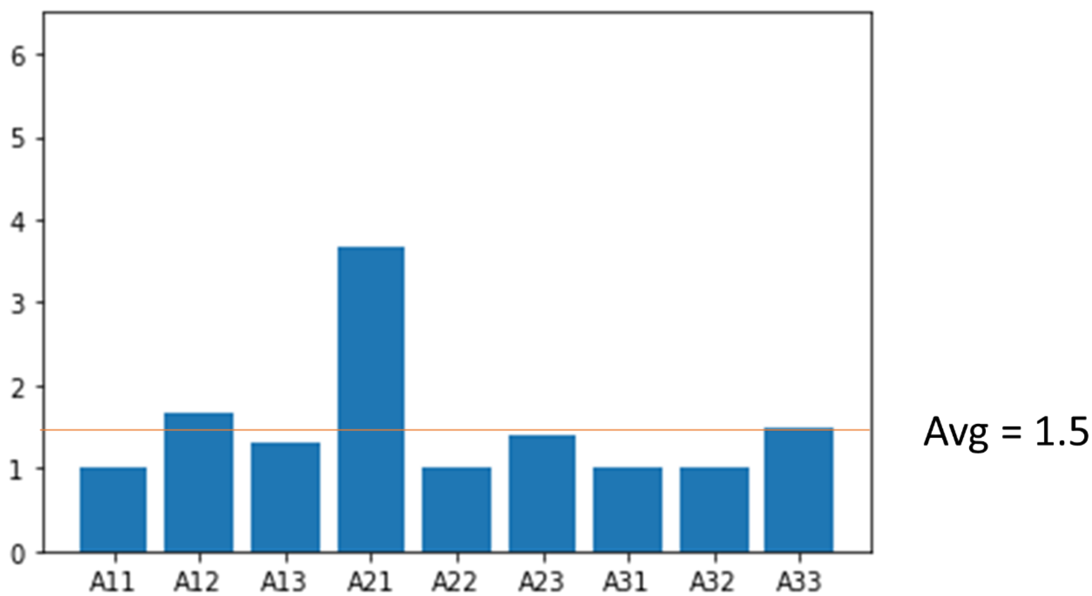
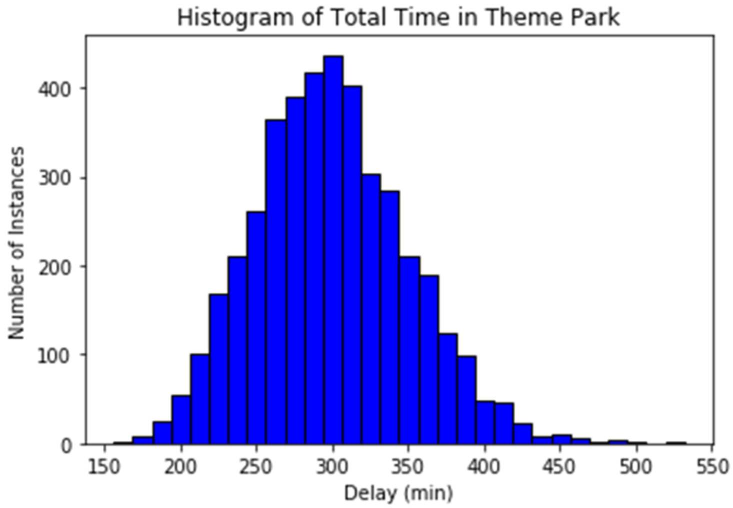


Figure 13: Maximum Queue Density for each Attraction, NN choice model

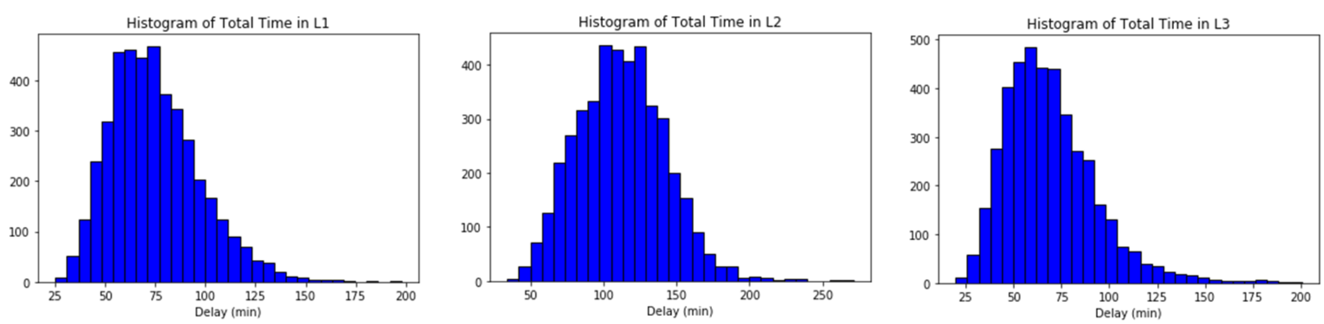
Note how the lines are now more uniform, with a lower maximum.

C. Statistical Analysis of the NN choice



ML assisted Simulation Run:
mean = 288.4 - Minutes
stdev = 50.31 - Minutes
Min time spent in park = 154 Minutes
Max time spent in park = 488 Minutes

Figure 14: Histogram of total time spent in the theme park, with mean, std, min and max



mean = 74.68	mean = 112.12	mean = 69.01
stdev = 22.36	stdev = 30.19	stdev = 24.02

Figure 15: Histograms of total time spent in each land, with mean and std

As apparent above, the total time in the theme park of the NN choice model have seen a slight decrease compared to the random choice model. Another benefit from utilizing the neural network is that it had reduced the average maximum queue density from 1.9 to 1.5, and the maximum queue density decreased from approximately 6 to 4. Another thing to note is that the queues are more uniformly distributed, which can be seen in the slight increase in queue density above 1.

7. Conclusion

We constructed a simulation model and predictive models to study the impact of the introduction of two technologies into the operation of the Qiddiya amusement park, namely the smart bracelet and the mobile application.

We have successfully shown that the quality of service and utilization of services can be improved using predictive model, which resulted in an even distribution of visitors, thus resulting in less crowded sites, aiding in increasing social distancing.

Implementing the two technologies above will assist in the generation of a historical database of visitor behavior

In the end, the simulation run gives a brief understanding of how visitors behave inside an amusement park, which is a useful investment in the long run. The simulation runs have shown interesting results, namely the queue piling on attraction 33 (in black, figure 7)

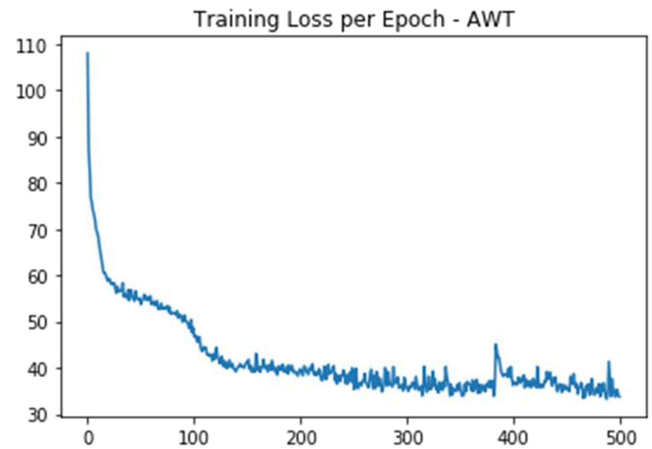
8. References

[1] F.-Y. H. a. Y.-C. L. Liou Chu, "Analysis and Simulation of Theme Park Queuing System," 2014.

[2] A. I. K. a. M. Deriaz, "A Machine Learning Approach to Waiting Time," in *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*, Geneva, Switzerland, 2019.

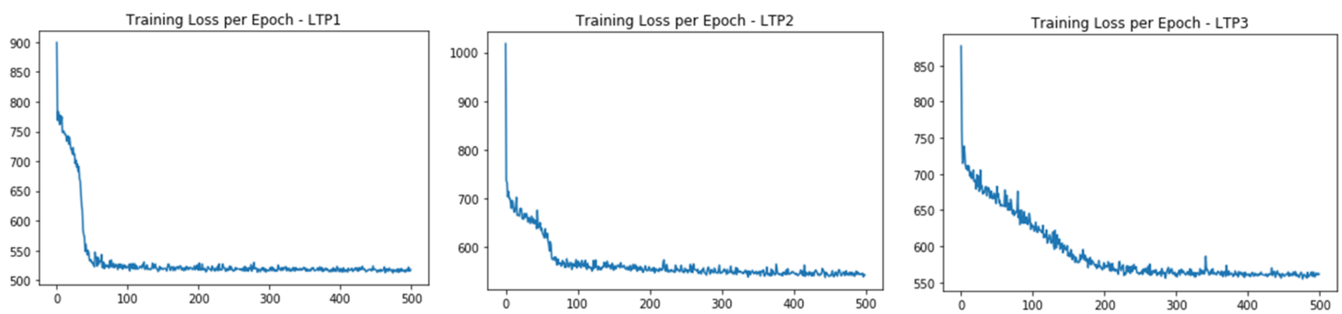
[3] L. Passanisi, "data.world," 2019. [Online]. Available: https://data.world/lynne588/walt-disney-world-ride-data/workspace/file?filename=WDW_Ride_Data_DW.xlsx.

9. Appendices




```
Training score for AWT =  34.29
Test score for AWT =  38.33
```

Figure 16: Training Loss per Epoch, AWT




```
Training score for LTP1 =  532.35
Test score for LTP1 =  486.92
```

```
Training score for LTP2 =  525.53
Test score for LTP2 =  571.99
```

```
Training score for LTP3 =  538.25
Test score for LTP3 =  533.08
```

Figure 17: Training Loss per Epoch, LTP