


## ORIGINAL RESEARCH

# A machine learning-based approach for wait-time estimation in healthcare facilities with multi-stage queues

Amjed Al-Mousa  | Hamza Al-Zubaidi | Mohammad Al-Dweik

Computer Engineering Department, Princess Sumaya University for Technology, Amman, Jordan

**Correspondence**

Amjed Al-Mousa.  
Email: [a.almousa@psut.edu.jo](mailto:a.almousa@psut.edu.jo)

**Funding information**

Systems and Electronic Development FZCO (SEDCO), Grant/Award Number: N/A

**Abstract**

Digital technologies have been contributing to providing quality health care to patients. One aspect of this is providing accurate wait times for patients waiting to be serviced at healthcare facilities. This is naturally a complex problem as there is a multitude of factors that can impact the wait time. However, the problem becomes even more complex if the patient's journey requires visiting multiple stations in the hospital; such as having vital signs taken, doing an ultrasound, and seeing a specialist. The authors aim to provide an accurate method for estimating the wait time by utilising a real dataset of transactions collected from a major hospital over a year. The work employs feature engineering and compares several machine learning-based algorithms to predict patients' waiting times for single-stage and multi-stage services. The Random Forest algorithm achieved the lowest root mean squared error (RMSE) value of 6.69 min among all machine learning algorithms. The results were also compared against a formula-based system used in the industry, and the proposed model outperformed the existing model, showing improvements of 25.1% in RMSE and 18.9% in MAE metrics. These findings indicate a significant improvement in the accuracy of predicting waiting times compared to existing techniques.

**KEYWORDS**

building management systems, data analytics, healthcare facilities, machine learning, smart cities applications, wait-time estimation

## 1 | INTRODUCTION

There has been tremendous focus on improving smart cities' healthcare systems using artificial intelligence and machine learning. This has been applied in controlling the spread of diseases such as COVID-19 in smart cities [1], helping disabled people [2], or in the detection of diseases [3–5]. Another important area of improvement in smart cities is queuing systems, which have become an integral part of any service-oriented organisation. Be it banks, hospitals, or governmental organisations, all use queuing systems to organise the crowds and give a sense of predictability of when the customer expects to be serviced. Recently, numerous incidents have underscored the detrimental consequences of the inadequate organisation of queue systems and insufficient information regarding patient

waiting times. For instance, a problematic occurrence between 2016 and 2019 revealed that nearly 5500 patients lost their lives while enduring extended waiting periods for hospital beds in overburdened medical facilities [6]. Furthermore, during the initial year of the COVID-19 pandemic in England, an alarming escalation of over 4000 additional hospital deaths transpired, attributable to patients enduring protracted waits of up to nine hours without receiving any healthcare services [6]. These distressing events highlight the urgent need for improved queuing systems and enhanced information dissemination to promptly deliver vital healthcare services and prevent such tragic outcomes.

While the ultimate goal of any organisation is to eliminate these queues and have people be served instantaneously, we know this is impossible as it conflicts with maximising resource

This is an open access article under the terms of the [Creative Commons Attribution-NonCommercial-NoDerivs](https://creativecommons.org/licenses/by-nc-nd/4.0/) License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2024 The Authors. *IET Smart Cities* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

utilisation. However, there are several key advantages of providing an accurate waiting time estimation for patients, such as

- Patient satisfaction: Providing patients with an accurate waiting time increases their satisfaction and helps them manage and arrange their other commitments. For example, if the waiting time is high, patients can decide to leave and reschedule the visit.
- Resource management: Knowing the average waiting time can help healthcare providers and administrators better allocate resources and staff to meet patient demand.
- Quality improvement: Measuring and tracking waiting times can help identify bottlenecks and their causes.
- Transparency and Reputation: Providing patients with accurate waiting times can increase transparency and trust in the healthcare system, enhancing the clinic's reputation.

A wide range of factors and features affect the waiting time. These factors include the number of patients waiting, the service or services they need, and the number of healthcare providers available. It also depends on other factors like the hour of the day, day of the week, day of the month, and many others. Waiting times are relatively more straightforward to predict when the patient requires a single service. However, in a multi-stage patient journey, the task becomes more complex. For example, if the patient needs to have his vitals taken by a nurse, do an ultrasound image, and then see the specialist, in such case predicting the waiting time is a more complicated task.

Many traditional queuing systems are formula-based, which is not very flexible and cannot easily adapt to changes in demand or different service needs. Nor can it accurately utilise the many features in the system, which can lead to a waste of resources and more delays. Thus, the motivation for this research is to develop a system that can handle a large number of features, adapt to temporal changes, and does not need complex hard-coded rules that need to be redesigned every time a change is introduced to the system. Machine learning-based solutions are one of the most efficient techniques to compact this. In addition to being able to handle a large number of features, they can handle both single-stage and multi-stage requests seamlessly.

While there are several attempts to tackle this problem using AI-based and ML-based solutions, they were single-stage service queues and often specific to certain types of healthcare facilities. Section 2 presents a detailed literature review.

The general scenario this work attempts is in a hospital or a clinic, where the patient must visit multiple stations before seeing the specialist. For example, one needs to have their vital signs taken in one Hall (or Lab), then proceed to do an ultrasound Echo and finally see the specialist. These stages/stations can be in the same area/hall or different hospital areas. When the patient first arrives and checks in, whether on a kiosk or via a mobile app, the system would need to provide a time estimate of how long this journey will take.

This paper presents an ML-based approach that relies on services the patient needs, features derived from the current hospital status, and temporal features derived from the time of the service request. The machine learning-based model has to predict the estimated waiting time for that patient. Note that the estimated waiting time presents the time from the moment of issuing a ticket until the patient reaches the doctor (the main service/the last stage), considering the waiting and serving times of the sequence of services required before reaching the doctor.

The quality of any machine learning algorithm is based on the data used for training. In this project, an anonymised dataset was provided by a company that operates the queuing systems inside a significant healthcare facility. The dataset includes data collected for almost a year. This dataset is transaction-based, which means it includes a predicted waiting time generated when the tickets are issued to patients and the actual time it took for that patient to be serviced per each customer request or transaction. The predicted time included in the dataset is based on a formula-based method called Average Service Time (AST), which will be discussed in detail in Section 2.1.

To be able to provide meaningful features for any ML-based system, a hospital-status reconstruction was performed. This process aimed to extract statistics regarding the status of the hospital per minute during the entire year and use these statistics as possible features for the ML algorithm. These time reconstruction snapshots are the foundation for generating real-time predictions for single-stage and multistage records. This enables this research to deliver an efficient business solution that optimises the queuing experience. This could be an even more complex problem if we have multiple objectives that we are trying to optimise for Ref. [7]. However, this research focuses on optimising a single factor, which is the wait time.

To summarise, the main contributions of this work are as follows:

- Introduce a real hospital ticketing dataset to the research community.
- Present a method to transform typical transactional data into a dataset that can be used in conjunction with ML methods.
- Utilise feature engineering to develop queue-related and temporal features that improve the wait time prediction.
- Evaluate the prediction accuracy of several ML-based techniques.
- Compare the accuracy of the ML-based technique to the formula-based method.

The document is organised as follows: Section 2 presents the literature review of current approaches. Section 3 describes the problem's methods and mathematical modelling of the problem, as well as the dataset collected and feature processing and engineering applied. The results of the different methods and a quantitative analysis of the results are presented in

Section 4. Finally, Section 5 summarises the results and propositions for future work.

## 2 | BACKGROUND AND LITERATURE REVIEW

This section presents the relevant prior research and findings related to this problem. This context will establish the groundwork for a comprehensive understanding of the mathematical model detailed in the subsequent subsections. First, the AST method, a common technique, is presented. Next, different approaches to waiting time prediction using classical machine learning and deep learning techniques are detailed in the subsequent sections.

### 2.1 | Average service time method

The AST method is a standard method for calculating the waiting time in queuing scenarios. The model predicts the waiting time based on historical data. It uses data from the past week, in addition to the current day—if it satisfies the condition of serving a predefined number of tickets already. The model checks each customer's expected serving order on counters and then calculates the AST for each service request during the past 7 days. Then, it adds the AST of the individual requests of all customers ahead of the current customer to find the total estimated time, as represented in Equation (1). Note that  $K$  is the total number of requests in the queue.

$$\sum_{n=1}^k AST_n \quad (1)$$

For example, if the current transaction has four transactions ahead with AST of 5, 5, 6, and 6 min, then the estimated waiting time for that specific customer should be around 21 min. However, the problem with this model is that the results are inaccurate, as they do not account for the hour of the day, day of the week, or many other factors. This is why the machine learning model might significantly improve this problem.

### 2.2 | Waiting time prediction using classical machine learning algorithms

A. Joseph et al. describe a method for predicting waiting times in radiation oncology using machine learning [8]. The patient's waiting experience is a significant challenge in healthcare, and waiting times for radiation oncology treatment can be particularly unpredictable and stressful for patients. The main features that generated the best-fit model were allocated appointment time, radiotherapy fraction number, median past treatment duration, the number of treatment fields, and previous treatment duration. The authors concluded that using

machine learning could help improve the predictability and accuracy of waiting times in radiation oncology, leading to a better patient experience and potentially reduced stress for staff.

R. Qamili et al. present a framework for improving the efficiency of issue ticketing systems using machine learning techniques [9]. The authors of this paper propose using machine learning techniques to analyse and classify issues to improve the efficiency and effectiveness of issue ticketing systems. The authors evaluate the performance of their intelligent framework for issue ticketing using several metrics, including accuracy, precision, and recall. They find that their framework can achieve high levels of accuracy in classifying issues, with an average accuracy of over 90% across all categories. They also find that their framework has a high level of precision and recall across all categories, with an average of over 85% and 75%, respectively. Overall, the study's results suggest that using machine learning techniques can significantly improve the efficiency and effectiveness of issue ticketing systems. However, the paper does not deal with wait time prediction.

Curtis et al. aimed to utilise machine learning techniques to accurately predict patient wait times and appointment delays at a hospital [10]. The authors note that long wait times and delays can lead to patient dissatisfaction and reduced trust in the healthcare system, and therefore, improving the prediction of these events could potentially lead to better resource allocation and improved patient experiences. The authors obtained data from the electronic health records of a large urban hospital, including information on patient demographics, diagnoses, and appointment details. However, they created a binary classification label for each appointment, indicating whether or not there was a delay of at least 30 min. The authors trained and tested several machine-learning models on the preprocessed data, including logistic regression, decision trees, and random forests. They found that the random forest model performed the best, with an AUC (area under the curve) of 0.78 on the testing data. This indicates that the model can distinguish between appointments with delays and those without delays with high accuracy, but it was not designed to predict the wait time.

Y. Sanit-in and K. R. Saikaew proposed three approaches for predicting customers' waiting time in one-stop service scenarios [11]. These approaches include Queuing theory, the Average time, and a machine learning-based model using the Random Forest algorithm. This research was done based on two datasets. The first one is the Khon Kaen University post office, which has 3480 records, and the second one is the queue logs of the ear, nose, and throat clinic with 1348 records. Both datasets' records were classified based on the duration into five classes: Very short, Short, Medium, Long, and Very long, which transformed the problem into a simple classification problem rather than a more complex regression problem. The results show that the machine learning-based model has the best accuracy of 85.76% and 81.7% in the ear, nose and throat clinic and the Khon Kaen University post office datasets, respectively. The Queue theory got an accuracy of 65.23%

in the ear, nose, and throat clinic dataset, and it was not applied to the second dataset. The average time theory got an accuracy of 64.94% and 68.89% on both datasets, respectively.

A. Rastpour and C. McGregor proposed a machine learning approach to predict patient waiting time in mental health care settings [12]. The authors obtained data from a large mental health care provider in the United States. The data was highly unidentified, with personal identifying information removed, and included information on patient visits, diagnoses, and treatment modalities. They performed data cleaning and preprocessing steps, including removing missing and outlier data, to prepare the data for analysis. The authors trained and tested several machine learning algorithms, including decision tree, random forest, and support vector machine (SVM) models. The authors found that the SVM model performed best, with an overall accuracy of 78.6% and a root mean squared error (RMSE) of 23.5 min. They also found that the model performed better at predicting longer wait times, with 84.2% accuracy for wait times greater than 60 min.

C. Gomes et al. proposed a machine learning approach for predicting the waiting time in a bank queue based on predicting the service time of individual customers [13]. They use support vector regression (SVR) as the machine learning technique to model the relationship between the waiting and predicted service times. To evaluate the effectiveness of the proposed approach, the authors collected data from a bank in Indonesia. The data consists of a training set of 8000 records and a testing set of 2000 records. The authors found that the SVR model achieved an average error of 2.44 min and a RMSE of 3.24 min. They also compared the performance of the SVR model with other machine learning techniques, such as decision tree regression and random forest regression, and they found that the SVR model outperformed these other techniques in terms of prediction accuracy. Overall, the results of this study demonstrate that the proposed approach using SVR is effective for predicting the waiting time in a simple bank queue based on the predicted service time of individual customers.

### 2.3 | Waiting time prediction using deep learning algorithms

I. Kyritsis and Michel Deriaz present a machine-learning approach for predicting waiting times in queuing scenarios [14]. Accurate waiting time predictions can be useful in various settings, such as in service industries where customers may be more likely to return if they have a good experience or in transportation systems where timely predictions can help with scheduling and resource allocation. The authors' work aims to improve upon traditional approaches to waiting time prediction, which may not be as accurate or efficient. The authors preprocess the data and perform feature engineering before training a neural network with two hidden layers on the data. After 500 training epochs, the model achieves a mean absolute error of 3.35 min on the test set. The model is compared to two baseline models and performs significantly better, with a mean absolute error of 28.9% lower than the naive mean model and 27% lower than the naive median model.

H. Hijry and R. Olawoyin suggested a deep learning-based model to predict the Patient Waiting Time in the Queue System of an emergency room [15]. (ER) is an essential component of the healthcare system, providing immediate medical attention to patients with urgent or life-threatening conditions. However, ERs are often overcrowded and have long waiting times, leading to frustration and dissatisfaction among patients and healthcare providers. In this paper, the authors aim to address this problem by using deep learning algorithms to predict patient waiting time in the queue system of the ER. The authors first collected data from an ER hospital in Nigeria to perform the analysis. This data included patient demographics, arrival time, triage category, and waiting time. The authors found that the deep learning model achieved an accuracy of 84.5% and a RMSE of 0.82 min. The authors also found that certain factors, such as the patient's triage category and arrival time, significantly impacted waiting time. These findings suggest that using deep learning algorithms could potentially improve the efficiency and effectiveness of the ER queue system.

Kuo, Yong-Hong, et al. present an integrated approach of machine learning and systems thinking for predicting waiting times in an emergency department [16]. They applied four popular machine learning algorithms (stepwise multiple linear regression, artificial neural networks (ANNs), support vector machines, and gradient boosting machines) to a dataset collected from an emergency department in Hong Kong. They compared the results to a linear regression model as a baseline. The authors found that all four machine learning algorithms outperformed the baseline model, with the stepwise multiple linear regression reducing the mean-square error by almost 15% and the other three algorithms reducing the mean-square error by approximately 20%. The authors also found that introducing the concept of systems thinking led to significant enhancements of the models, with reductions in the mean-square error of 17%–22% due to the utilisation of systems' knowledge. They also note predicting waiting times for less urgent patients is more challenging.

Anussornnitisarn, P., and Limlawan, V. propose the use of ANNs to predict waiting times in queue systems [17]. The design of advanced queue systems is crucial to managing customer flow and reducing waiting times in various industries, such as healthcare, banking, and retail. The authors first collected queue length, arrival rate, and service rate data to train the ANN model from a real-world queue system. The authors found that the ANN model outperformed the traditional model regarding prediction accuracy. The authors also found that increasing the number of hidden layers and neurons in the ANN model improved prediction accuracy, while increasing the learning rate harmed accuracy.

R. P. Satya Hermanto et al. developed a model to predict customer wait times in bank queues using a type of neural network called RPROP [18]. The authors collected data on factors influencing wait times, such as customer demographics, time of day, and teller availability. The authors trained an RPROP neural network and compared its performance to several machine learning models, including linear regression and decision trees. They found that their RPROP model outperformed the others, achieving an accuracy of about 93% in



predicting wait times. They also conducted a sensitivity analysis to evaluate the impact of different factors on wait times. The results revealed that customer demographics and teller availability were the most significant predictors of wait times.

## 2.4 | Summary

Table 1 summarises the latest research in the area of wait time estimation. It can be seen that both classical machine and deep learning algorithms can be used to train waiting-time prediction models. However, one can notice that

- Some of the previous research simplified the wait time prediction problem by transforming it into a classification problem. However, this does not produce an accurate wait estimation.
- Some of the current research considered banking queues. The wait time in the bank queue is typically shorter, so the RMSE or MAE comparison becomes inaccurate.
- Most of the detailed medical queues are considered a specialised queue, not a general hospital with various services.
- None of the existing research approached multistage queues.

Thus, the gap fulfilled by this work will be in addressing the wait time prediction problem in general healthcare facilities with multi-stage queues.

## 2.5 | Machine-learning model

Machine learning can be an effective tool in developing solutions in the medical field [19–21], for security systems [22], battery discharge capacity [23] and book recommendations [24]. Thus, machine learning can take a role in this problem by building a

machine learning-based model that can learn from the data and produce accurate patient waiting time predictions. The model would predict the patient's waiting time after taking a ticket from a ticketing system based on several features that may affect the patient's waiting time. In machine learning, time prediction can be considered a regression problem, where multiple algorithms can be used, such as the Random Forest, SVR, Linear Regression, KNN, Gaussian Regression etc. In addition, ANNs and deep neural networks can be examined in addition to the classical ML models. Figure 1 shows the typical workflow of any machine-learning-based model, where an ML model is trained based on existing data and the problem specifications. Next, the model is tested on unseen new records. If the model's performance is acceptable, it gets deployed into production; otherwise, the errors are analysed to adjust the model hyper-parameters and restart the process. This section details the algorithms used to build different models for this problem.

### 2.5.1 | Ridge regression

Ridge Regression is a regularised linear regression algorithm used to deal with multicollinearity and overfitting problems in the dataset [25]. It is a linear regression model that adds an L2 penalty term to the cost function of the ordinary least squares regression. The L2 penalty term shrinks the magnitude of the coefficients towards zero, leading to a better model with improved generalisation performance and less prone to overfitting. The Ridge Regression model minimises the cost function as shown in the Equation (2)

where

$$Cost = \sum_{i=1}^N \left\{ y_i - \sum_{j=0}^M \beta_j x_{ij} \right\}^2 + \lambda \sum_{j=0}^M w_j^2 \quad (2)$$

**TABLE 1** Summary of literature related to wait time prediction.

Author	Method	Issues	Error
Joseph et al. [8]	Classical ML	Limited to radiation oncology	MAE = 4.6 mins
Qamili et al. [9]	Classical ML	Predict category of delay (3 classes) not a specific wait time	N/A
Curtis et al. [10]	Classical ML	Based on binary classification of the delay ( <b>True:</b> More than 30 min, <b>False:</b> Less than 30 min)	N/A
Sanit-in & Saikaew [11]	Queuing theory & ML	Predicted the delay class (5 classes), Not as a regression problem	N/A
Rastpour & McGregor [12]	Classical ML	Limited to mental health care facility	RMSE = 23.5 mins
Gomes et al. [13]	Classical ML (SVR)	Limited to banking data single-stage queue	RMSE = 3.24 mins
Kyritsis & Deriaz [14]	Deep learning	Single-stage queue	MAE = 3.35 mins
Hijry & Olawoyin [15]	Deep learning	Limited to emergency rooms, which is usually shorter waiting period	RMSE = 0.82 mins
Kuo et al. [16]	Deep learning	Limited to emergency rooms	RMSE = 44.3 mins
Anussornnitisarn & Limlawan [17]	ANN	Single-stage queue with simulated artificial delay	N/A
Hermanto et al. [18]	Deep learning	Limited to banking data	MSE = 811,859s ⇒ RMSE = 15 min MAE = 12:12 min:sec

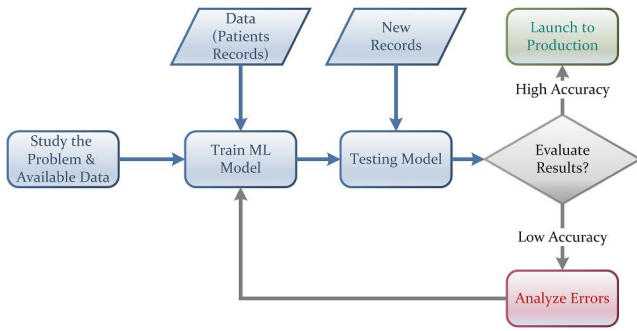


FIGURE 1 Typical workflow of any ML model.

- $N$ : the number of observations in the dataset
- $M$ : the number of features or independent variables in the dataset
- $y_i$ : the target or dependent variable value for the  $i$ th observation
- $x_{ij}$ : the value of the  $j$ th feature for the  $i$ th observation
- $\beta_j$ : the coefficient for the  $j$ th feature
- $\lambda$ : the regularisation parameter that controls the strength of the penalty term.
- $W_j$ : the weight or penalty assigned to the  $j$ th coefficient.

### 2.5.2 | Lasso regression

Lasso regression is an extension of linear regression that performs variable selection and regularisation to prevent overfitting. Lasso regression aims to minimise the sum of squared errors between the predicted and actual values, subject to a constraint on the absolute value of the coefficients.

Lasso regression uses L1 regularisation, which penalises the absolute value of the coefficients, as opposed to ridge regression, which uses L2 regularisation and which penalises the squared value of the coefficients. The L1 penalty shrinks some of the coefficients to zero, effectively performing feature selection by eliminating irrelevant features from the model. The Lasso Regression model minimises the cost function as shown in the Equation (3), with the variables used being the same as those used in the Ridge Regression.

$$Cost = \sum_{i=1}^n \left( y_i - \sum_{j=1}^p x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \quad (3)$$

### 2.5.3 | Elastic net regression

Elastic net regression is another type of linear regression that combines the properties of Lasso regression and Ridge regression. It is often used when there are high-dimensional datasets and multicollinearity among the predictors. In Elastic Net regression, the objective is to minimise the sum of errors between the actual and predicted values, subject to a

constraint on the sum of the absolute values of the regression coefficients (L1 penalty) and the sum of squared values of the regression coefficients (L2 penalty).

The L1 penalty helps produce sparse solutions by shrinking some regression coefficients to zero. In contrast, the L2 penalty helps reduce the effect of multicollinearity among the predictors. The combination of L1 and L2 penalties in Elastic Net regression allows it to select relevant predictors and perform well on datasets with many predictors, some of which may be correlated. The elastic net model minimises the cost function as shown in the Equation (4).

$$Cost = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=1}^p |w_j| + \lambda_2 \sum_{j=1}^p w_j^2 \quad (4)$$

Where

- $n$ : the number of observations in the dataset.
- $y_i$ : the actual target value for the  $i$ th observation.
- $\hat{y}_i$ : the predicted target value for the  $i$ th observation.
- $p$ : the number of predictor variables in the model.
- $W_j$ : the  $j$ th regression coefficient.
- $\lambda_1$  and  $\lambda_2$ : the hyperparameters that control the strength of the L1 and L2 penalties, respectively.

### 2.5.4 | Decision tree regression

A decision tree regressor is a machine learning algorithm that creates a model in the form of a tree to predict a continuous target variable. The algorithm splits the data based on the values of input features in a way that maximises the reduction of the variance of the target variable. The final model consists of a tree-like structure, where each internal node represents a test on a specific feature, each branch represents the outcome of the test, and each leaf node represents a prediction.

The decision tree regressor is a powerful and flexible algorithm that handles numerical and categorical data, interactions, and non-linearities between features. However, it is also prone to overfitting if the tree is too deep or has too many features in the data. Therefore, tuning the parameters wisely is crucial to obtain the best possible results.

### 2.5.5 | Random forest regression

Random Forest Regressor uses an ensemble/combination of decision trees to predict the target variable. Each tree in the forest predicts a numerical value, and the final output is the average of all the predicted values. The algorithm also involves random feature selection, where each tree is trained on a random subset of the features [26]. This helps to decrease the correlation among the trees and enhance the overall performance. This makes the Random Forest Regressor a strong, flexible algorithm that can handle complex regression problems.

## 2.5.6 | Artificial neural network

Artificial neural networks is a machine learning model inspired by the human brain's structure and functioning. A neural network consists of layers of interconnected neurons that process and transmit information if sufficiently activated. The most common type of neural network is the feed-forward neural network, where data flows from an input layer to the output layer. A simple neural network would consist of an input layer, one hidden layer, and an output layer. The input layer receives data from the outside world, and the output layer produces a prediction or classification based on the input. The hidden layer performs intermediate computations on the data, transforming it into a more useful form for making predictions.

Several parameters can be tuned in any neural network, including the number of neurons in each layer, the type of activation function used, the learning rate, the loss function, the weight initialisation, and the optimisation algorithm. Choosing the appropriate values for these parameters is critical for training an effective neural network. It often involves a trial-and-error process, where different combinations of values are tested and evaluated until the optimal set of parameters that can produce the best possible results is found [27].

## 2.5.7 | Deep neural network

DNNs are similar to classic ANN, except that in DNNs the number of hidden layers is two or more. Increasing the number of hidden layers can enhance the efficiency of handling complex problems. However, this additional feature increases the risk of overfitting; thus, the number of hidden layers should be chosen wisely [28].

## 3 | MATERIALS AND METHODS

### 3.1 | Problem formulation & mathematical modelling

As mentioned in Section 1, a complete patient trip to a healthcare facility might require being serviced by multiple services. For example, as shown in Figure 2, if the patient were visiting the cardiology clinic, he/she would need his vital signs checked, have an Echo done, and finally see the specialist in the cardiology clinic. There might be multiple counters providing any of these services. And these services might be served all in the same or multiple halls. Typically, the patients' waiting time starts from issuing the ticket until reaching the last stage (specialist).

In this research project, multiple assumptions and limitations must be taken into consideration which are

- The sequence of services and the halls where these services will be delivered will be known once the patient requests a ticket.

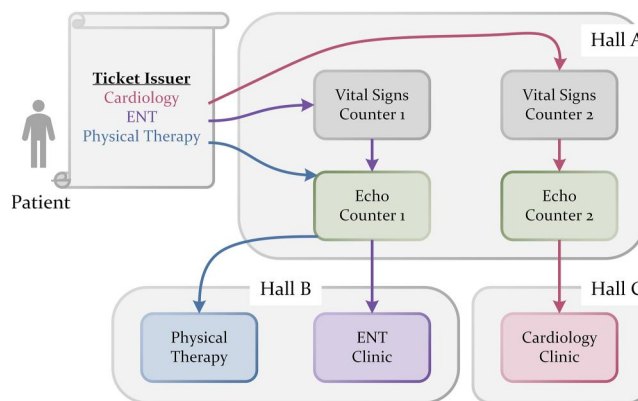


FIGURE 2 Patient's journey based on the reason of visit.

- Patients can be served all services in the same or multiple halls. The total estimated time will be based on adding all services from all halls.
- The user cannot be assigned the same service but in two halls.

The problem with predicting the wait time is that when you issue the ticket, you do not know which counter the patient will be assigned to since the assignment is dynamic. In addition, some of the counters (service stations) handle more than one type of service; for example, the Echo can do a Cardiology Echo or a Urology Echo. If the patient assignment to counters was deterministic, the problem would be much simpler and could be handled by counting the number of people in line for each service.

To prepare the mathematical model, the following variables can be defined:

- $T[id]$ : The total waiting time for a patient (id) until reaching the last stage (doctor).
- $WS(x, hid, cid)$ : The waiting time for the requested service (x) that is being served in the hall (hid) by the counter (cid).
- $SS(x, hid, cid)$ : The serving time for the requested service (x) that is being served in the hall (hid) by the counter (cid).

Based on these variables, Equation 5 can be defined as aggregating the individual services' waiting and serving time. Note that the time may vary based on the number and type of the requested services and the hall the customer can be assigned to.

$$T[id] = \sum_{x=1}^n WS(x, hid, cid) + \sum_{x=1}^{n-1} SS(x, hid, cid) \quad (5)$$

The remaining challenge will be to get an accurate estimate of the waiting and serving time for each service at the time of issuing the waiting ticket. To come up with these estimates, any queuing system should track the status of each counter at any given moment and how many potential patients can be serviced by that counter. More details on this feature engineering will be detailed in Section 3.4.

### 3.2 | Typical data collection

Typical queuing systems store information about completed transactions/services. Table 2 presents a sample of what a typical system would store about the service. Note that the patient may have more than one record in this table if the service requested requires the completion of multiple mini-services. Note that this table cannot be used as the basis for a machine-learning model, as the interest is in the total waiting time, not the individual time of mini-services.

There should also be a means to map counters to services to find the services each counter can serve. This will be very useful in determining each counter's competing services. In addition, typically, there is tracking for the counter status (i.e. whether it is serving, waiting to serve, or even on a break). This usually maps the counter to the individual employee working on the counter.

A dataset from an anonymised hospital was used to validate the proposed model. This hospital has 13 halls (A hall is a group of counters), 223 counters, and provides a total of 303 services. Note that some services can exist in multiple halls, and the counter can serve multiple services. The dataset has almost 300 K patient transactions spanning almost a year, and they follow the structure presented in Table 2. Note that all

**TABLE 2** Significant customer transactions' table columns.

Column name	Description
Queue Branch ID	Branch identity which this transaction belongs to.
Queue Branch visit ID	The visit ID of a transaction that distinguishes between one patient and another, and it can be used to group the transactions with the same visit for the same customer.
Priority	It is the priority of the transaction (patient). Appointments normally get a very high priority.
Service ID	The service of the transaction.
Counter ID	The counter that the ticket was taken specifically to or the counter that serves this
User ID	The user (employee) that the ticket was taken specifically to or the user that serves this transaction.
Waiting start time	This is the time that the customer starts waiting his turn.
Serving start time	The time when the transaction started to serve on some counter, it will be empty if the customer was never served.
Serving end time	The time when the counter end serving the transaction, it will be empty if the customer was never served.
Waiting seconds	The period that the customer was in a waiting state.
Serving seconds	The period that the customer was in a serving state.
Hall ID	The hall ID for this transaction.
Serving type	It indicates how the transaction was closed, for example, served, or not served.

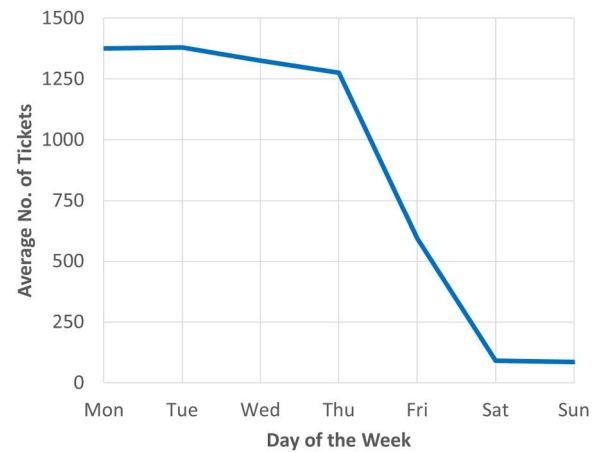
patient names and identifying data have been removed, and only an anonymised ID is used in the table.

### 3.3 | Data insights

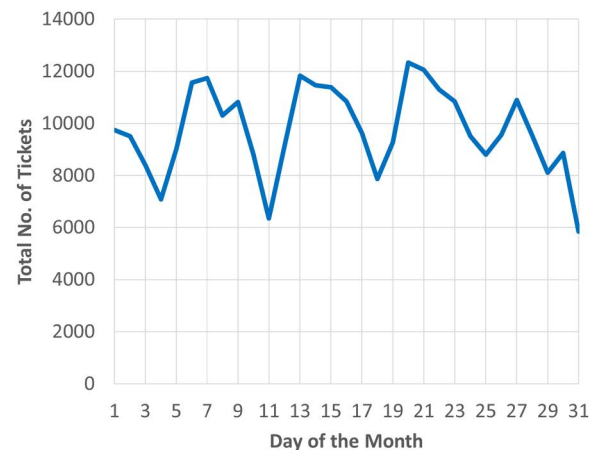
The congestion of the hospital is one of the major pieces of information that can be illustrated by visualisation. Figure 3 shows the hospital's congestion based on the average number of issued tickets per day of the week. As expected, the weekend days have the lowest values. Meanwhile, Sunday and Monday days have the highest values.

Figure 4 shows the congestion based on the total number of issued tickets for any day of the month. The weekend effect can explain the peaks and valleys in the diagram. In addition, for day 31, five months out of 12 do not have this day, and this can illustrate why it has the lowest value.

Figure 5 shows the hospital's congestion based on the average number of issued tickets per hour of the day. It shows that the peak time during the day is in the morning, specifically at 10 am and 11 am.



**FIGURE 3** Hospital's congestion based on the day of the week.



**FIGURE 4** Hospital's congestion based on the day of the month.



As mentioned previously, the hospital has 13 halls. Figures 6–8 analyse the data in these halls. Firstly, Figure 6 shows the congestion of each hall based on the number of issued tickets. It can be noted that Hall A has the majority share of tickets, while Hall M has the least.

Meanwhile, Figure 7 illustrates each hall's supported services. Even though Hall A has lots of tickets, it can be seen that the number of services in that Hall is minimal.

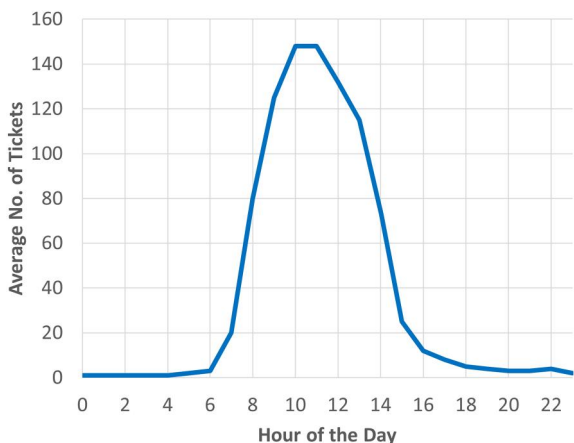


FIGURE 5 Hospital's congestion based on the hour of the day.

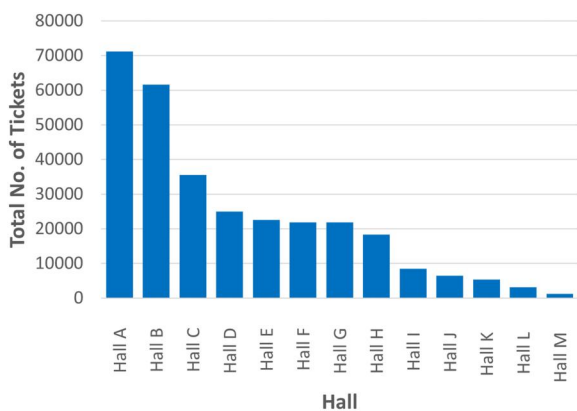


FIGURE 6 Halls' congestion during the year.

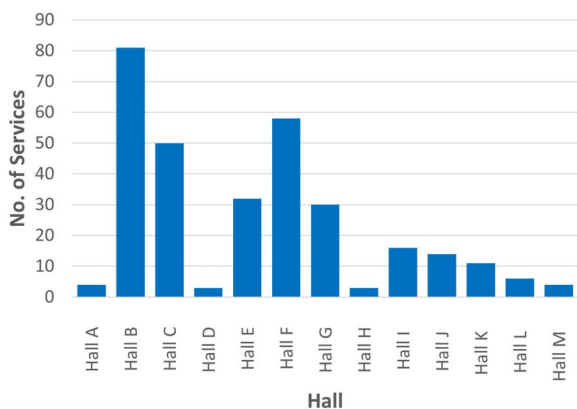


FIGURE 7 Number of supported services in each hall.

Finally, Figure 8 shows the number of available counters in each hall. Halls B and C are the largest, with more than 30 service counters in each.

To understand the length of the services we are dealing with, the issued tickets have been grouped into three classes based on their serving time, and these classes are 'Long' for tickets that took longer than 8 min to be completed, class 'Medium' for tickets that were served in the range of (4–8) minutes and the 'Short' class in the range of (0–3) minutes. Figure 9 shows issued ticket distribution based on these three classes.

Figure 10 shows the 10 most popular services based on the number of issued tickets for each service. Note that the remaining services with at least one request during the year are all combined in the last bar.

Considering the previous analysis, one can conclude the need to add features like Hour of the Day, Day of the Week, Day of the Month, the Hall ID, and the actual service ID. Using such features will be much more valuable than including the timestamp.

### 3.4 | Feature engineering & extraction

#### 3.4.1 | Capturing queue status features

Based on the discussion in Section 3.2, it is clear that there is a need to have an auxiliary data source that provides the

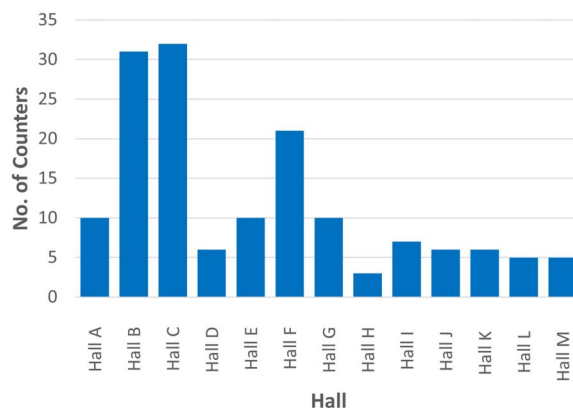


FIGURE 8 Number of available counters in each hall.

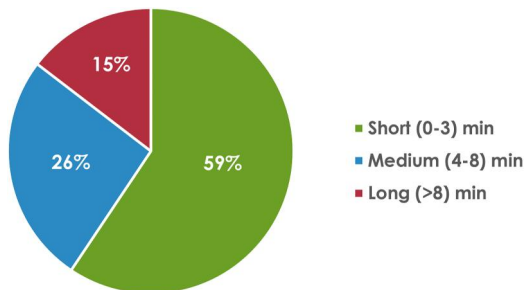


FIGURE 9 Distribution of issued tickets based on the three services classes: long, medium, and short.

real-time status of any healthcare facility. This should include how many patients are requesting any service and how many employees are available to service that type of service. However, it is essential to note that when the patient is waiting for his turn at a counter, the counter could be serving other services that are different from the patient of interest service. Hence, patients with other services serviced by the same counter will affect the patient's waiting time. This calls for the creation of a new feature, which is the number of patients waiting at counters that serve the service of the current patient.

As an illustration, Figure 11 shows a simple scenario of multiple services that can be served on multiple counters. All three counters can service 1, while Service two is serviced by

counters {2,3} only. Let us assume that S1 and S2 are the numbers of patients waiting for Service 1 and Service 2, respectively. And one can assume that these will be roughly distributed evenly on the available counters. Then, the number of patients serviced by each counter will be given by Equations (6–8).

$$\text{Serviced by Counter 1} = \frac{S1}{3} \tag{6}$$

$$\text{Serviced by Counter 2} = \frac{S1}{3} + \frac{S2}{2} \tag{7}$$

$$\text{Serviced by Counter 3} = \frac{S1}{3} + \frac{S2}{2} \tag{8}$$

Thus, the number of patients waiting at counters that serve the service S1, call it AS1, can be found using Equation (9).

$$\begin{aligned} AS1 &= Ceil\left(\frac{S1}{3} + \frac{S1}{3} + \frac{S2}{2} + \frac{S1}{3} + \frac{S2}{2}\right) \\ &= Ceil(1 \times S1 + 1 \times S2) \end{aligned} \tag{9}$$

Similarly, the number of patients waiting at counters that serve the service S2, called AS2, can be found using Equation (10).

$$AS2 = Ceil\left(\frac{S1}{3} + \frac{S2}{2} + \frac{S1}{3} + \frac{S2}{2}\right) = Ceil\left(\frac{2}{3} \times S1 + 1 \times S2\right) \tag{10}$$

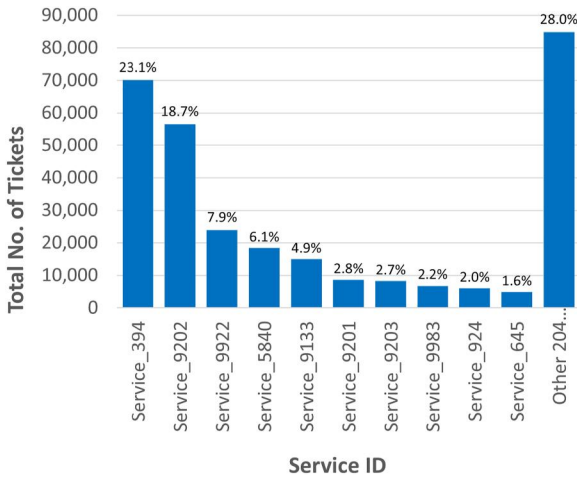


FIGURE 10 The 10 most popular services during the year.

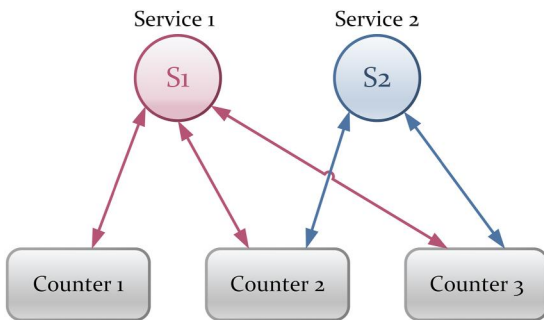


FIGURE 11 Counters with multiple services.

So at this point, three features are now defined per service: the number of patients requesting the service, the number of employees or counters serving that service, and the number of patients waiting at counters that serve the service. These features change constantly and thus should be tracked through time in any institution with such a queuing problem. A new table was created to store the hospital status every minute, and this table will be referred to as 'Hospital\_Status'. This table consists of six attributes, but three of them are repeated for each service, and these attributes are 4, 5, and 6 in Figure 12. In the dataset used, since there are 303 services in the hospital, the number of columns in this table will be  $(3 + 303 \times 3 = 912)$  features.

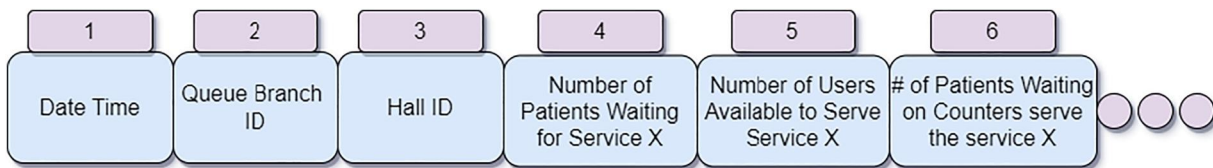


FIGURE 12 Arrangement of columns in the 'Hospital\_Status' table.

### 3.4.2 | Feature selection for the machine learning Model

To train the ML model, one must aggregate all records related to each patient visit to find the total time. Figure 13 illustrates this point clearly.

In addition to the waiting time (label), the ML table should contain information about the patient's requested features and the system status when the patient arrives. These features are shown in Table 3, and their arrangement is presented in Figure 14. Note that three temporal features were extracted from the time when the transaction was done. These are the day of the month, day of the week, and hour of the day. As shown in Figures 3–5, these three features have clear patterns that would impact the wait time. Thus, they were included. One can also note that four of these features are repeated for each service. Thus, in this case, this table's number of features (columns) will be  $(8 + 303 \times 4 = 1220)$ . Using this format, one can see the inherent support for the multi-stage services. Columns (8, 12, 16 ... etc.), which are repeated for all services, indicate whether a single or multiple services are requested. This would pose a challenge if more new services were added, as this would require changing the number of columns in the data table. Algorithm 1, details how the 'ML table' can be generated using the transactions data and the 'Hospital\_Status' data.

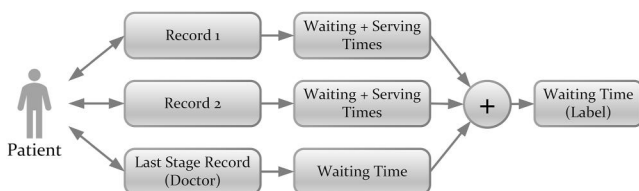


FIGURE 13 The ML label's formation.

TABLE 3 'ML table' features.

Feature	Description	Data type
Visit ID	The visit ID of a transaction that distinguishes between one patient and another, and it can be used to group the transactions with the same visit for the same customer.	Integer
Queue Branch ID	Branch identity which this transaction belongs to.	Integer
Priority	It is the priority of the transaction (patient). Appointments typically get a very high priority.	Integer
Waiting start time	The time that the customer starts waiting his turn.	Datetime
Day of the month	The day of the month when the transaction was done.	Integer
Day of the week	The day of the week when the transaction was done.	Integer
Hour of the day	The hour of the day when the transaction was done.	Integer
Requested service X	A flag if the patient requested this service.	Binary
Number of patients waiting for service X	The number of patients waiting for a specific service in a given time.	Integer
Number of users serving the service X	The number of users (employees) serving a specific service in a given time.	Integer
# of patients waiting on counters that serve service X	The number of patients waiting at counters that serve service X	Integer
<b>Label</b>	<b>The estimated waiting time to reach the doctor.</b>	<b>Time</b>

### Algorithm 1 The algorithm for building the 'machine learning' table

**Data:** *TransactionsData, HospitalStatusData*

**Initialize:**

$i \leftarrow 0;$

$m \leftarrow$  Query the number of unique customer visits;

**for**  $i \leftarrow 1$  **to**  $m$  **do**

  VisitStartTime  $\leftarrow 0;$

  WaitingTime  $\leftarrow 0;$

  ServingTime  $\leftarrow 0;$

  TotalTime  $\leftarrow 0;$

$n \leftarrow$  The number of transactions related to the visit[i];

**if** *TransactionSequence*[i,j]=1 **then**

    | VisitStartTime = TransactionStartTime[i,j];

**end**

**for**  $j \leftarrow 1$  **to**  $n$  **do**

    | WaitingTime = WaitingTime +

    | TransactionWaitingTime[i,j]

**end**

**for**  $j \leftarrow 1$  **to**  $n - 1$  **do**

    | ServingTime = ServingTime +

    | TransactionServingTime[i,j]

**end**

  TotalTime = ServingTime + WaitingTime

StatusRecord  $\leftarrow$  Query HospitalStatusData for status at  
(Time = VisitStartTime)

TimeExtractedFeatures  $\leftarrow$  Feature Extraction Script  
(VisitStartTime)

Insert into ML Table (StatusRecord,  
TimeExtractedFeatures, TotalTime)

**end**

### 3.5 | Data preprocessing

After preparing the data using the previous steps, the data is now ready to be utilised by different machine-learning models. However, several preprocessing steps are detailed below before proceeding with the ML algorithms:

### 3.5.1 | Handling non-numeric features

It is necessary to handle non-numeric features since most machine learning algorithms require numerical input data to make predictions. Therefore, non-numerical features were converted into numerical variables by encoding them so the algorithm could understand them. The label encoding method was used to handle the Boolean flags of services, for example, the feature 'Requested service 1234'. The 'True' value was converted to a numeric value of 1, and the 'False' value was converted to a numeric value of 0.

### 3.5.2 | Handling outliers

Outliers can significantly impact machine learning models as they can skew the results, leading to incorrect conclusions or inaccurate predictions. Thus, outlier records were removed as it is expected that some patients might have had to wait for an unreasonably long time or that a human error could have caused some tickets not to be closed properly.

There are several methods to define the upper and lower statistical limits beyond which a data point would be considered an outlier. Equations (11) and (12) define the Lower Bound and the Upper Bound limits:

$$LB = Q1 - 1.5 \times IQR \tag{11}$$

$$UB = Q3 + 1.5 \times IQR \tag{12}$$

Q1 and Q3 are the first and the third quartiles of the data. The IQR is the Inter-Quartile Range, which is the range between the 25th percentile (Q1) and the 75th percentile (Q3) of the dataset, which contains the middle 50% of the data as defined by Equation (13):

$$IQR = Q3 - Q1 \tag{13}$$

Figure 15 shows the Total waiting time distribution before and after removing the outliers. The upper outlier limit was roughly around 40 min.

### 3.5.3 | Feature scaling

The purpose of feature scaling is to ensure that all features are on a comparable scale and have a similar range of values, which can be important for many machine learning algorithms. Thus, all features were scaled to the range of (0–1) using min-max scaling. Equation (14) illustrates the formula for min-max scaling:

$$X_{Scaled} = \frac{X_{Original} - X_{min}}{X_{max} - X_{min}} \tag{14}$$

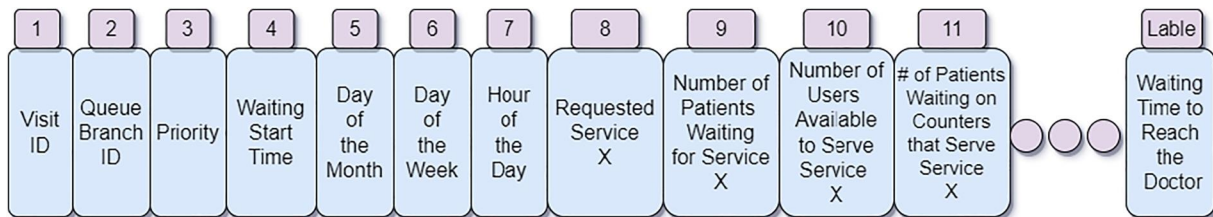


FIGURE 14 Arrangement of columns in the 'ML table'.

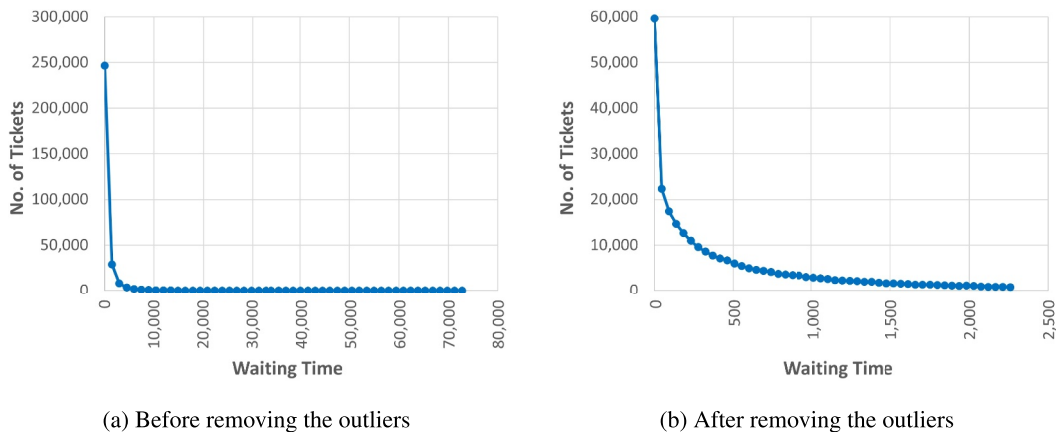


FIGURE 15 Impact of removing outliers on the waiting time distribution.



where  $X_{\min}$  is the minimum original (Unscaled) value, and  $X_{\max}$  is the maximum original (Unscaled) value.

### 3.5.4 | Data splitting

The purpose of splitting the data is to evaluate the model's performance on unseen data. Evaluating the model on the same data used for training can result in overfitting, where the model becomes too specialised to the training data and fails to generalise well to new data. The data was divided into a 70/30 ratio where 70% of the data is used for training and 30% for testing and evaluating the models. In addition, the k-folds cross-validation method was also used in some experiments; 10-folds were used.

## 4 | RESULTS

### 4.1 | Experiment setup

As mentioned, the data used for training and testing will be almost a year's data at a major healthcare facility. The hyperparameters of machine learning algorithms significantly impact the model's performance; thus, they must be chosen optimally. The grid search technique is used to find the optimal set of hyperparameters to produce the best results out of the model. The model's performance is evaluated using a cross-validation technique (5-fold). The combination of hyperparameters that results in the best performance on the validation set is selected as the optimal one, and this set will then be generalised to predict the test data.

Table 4 illustrates the hyperparameters selected for the models used in this work. It is worth noting that these hyperparameters were selected after performing a grid search on many parameter configurations. In addition, Table 5 shows the details of the neural networks used, both ANN and DNN. It can be seen that both have a gradual reduction in the number of inputs. They also have one output at the output layer with no activation function, as expected in regression problems.

The following are the specifications of the machine used to perform the data transformation and the machine-learning models training:

- **Processor:** Intel Core i9 13900 K 13th Gen 24 Core.
- **Memory:** 64 GB DDR-5 5200 MHz.
- **Storage:** 2 TB M.2 NVMe Gen 47,300 MB/s.
- **GPU:** GigaByte GTX1660 Super 6 GB DDR-6 DUAL OC.

For the data engineering and transformation part, PyCharm was used. Meanwhile, Jupyter Notebook was used for the ML model's training and data visualisation. Table 6 shows the software versions used in addition to the Python libraries used in the code.

**TABLE 4** Machine-learning model's parameters.

ML model	Parameter	Values
Ridge regression	alpha	0.1
Lasso regression	fit_intercept	True
Elastic net regression	normalise	False
	random_state	50
	l1_ratio <sup>a</sup>	1
Decision tree	max_depth	7
	max_features	Auto
	min_samples_leaf	5
	random_state	50
Random forest	n_estimators	100
	random_state	50
	max_features	None
ANN/DNN	Optimiser	Nadam
	Learning rate	0.01
	Beta 1	0.9
	Beta 2	0.999
	Kernel initialiser	GlorotUniform
	Loss function	MSE

<sup>a</sup>l1\_ratio applies to Elastic Net Regression Only.

**TABLE 5** Neural networks model's architectures.

Model	Layer	Outputs	Inputs	Activation
ANN <sup>a</sup>	Dense layer 1	128	1217	ReLU
	Dense layer 2	64	128	ReLU
	Output dense layer	1	64	
DNN <sup>b</sup>	Dense layer 1	300	1217	ReLU
	Dense layer 2	150	300	ReLU
	Dense layer 3	64	150	ReLU
	Output dense layer	1	64	

<sup>a</sup>Training using epochs = 15, batch\_size = 10.

<sup>b</sup>Training using epochs = 15, batch\_size = 30.

### 4.2 | Performance metrics

Since this is a regression problem, three metrics were used to evaluate the performance of the models, which are RMSE, MAE, and R2\_score.

- **RMSE:** Route Mean Squared Error. It shows the difference between the prediction values  $\hat{y}_i$  and the actual values  $y_i$ , when the dataset has  $n$  instances, according to Equation (15). In other words, the RMSE illustrates how far the predictions fall from the actual values using Euclidean distance [29].

**TABLE 6** Software & libraries versions used.

Software/Library	Version
Pycharm	2022.3.2 community Ed. w/Python 3.9
Jupyter notebook	6.4.12
numpy	1.23.5
matplotlib	3.5.2
pandas	1.4.4
tensorflow	2.12.0
seaborn	0.11.2
joblib	1.1.0
Pyodbc	4.0.34
scikit-learn	1.0.2
ipython	7.31.1

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}} \quad (15)$$

- MAE: Mean Absolute Error. It measures the average error amount in the predictions compared with the actual values without considering their direction/sign. Equation (16) shows the formula of MAE.

$$MAE = \frac{\sum_{i=1}^n |\hat{y}_i - y_i|}{n} \quad (16)$$

- R-Squared ( $R^2$ ): It is a statistical metric used to determine the proportion of variance in a dependent variable that an independent variable can predict [30]. Equation (17) shows the formula of R-squared. Note that  $\bar{y}_i$  is the mean of all the values.

$$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2} \quad (17)$$

### 4.3 | Results & discussion

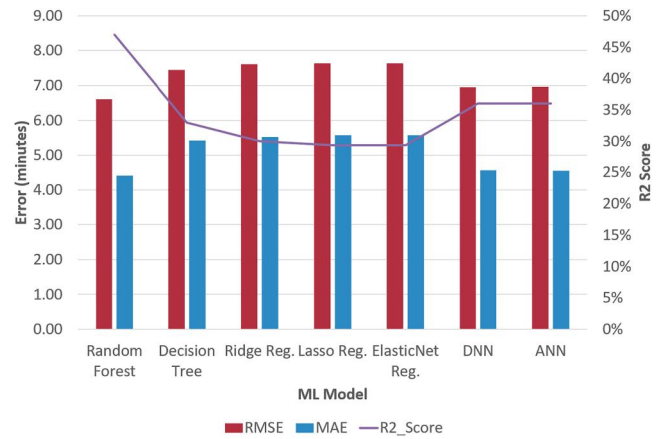
#### 4.3.1 | Results of machine-learning models

Two testing methods were used, one using a simple {70:30} split and the other using 10-fold cross-validation. Table 7 and Figure 16 show the results based on the 70:30 split method. It can be seen that the Random Forest produced the lowest RMSE and lowest MAE numbers, 6.6 and 4.4 min, respectively. In addition, Random Forest has the highest  $R^2$  score of 47%. One can notice that the RMSE is consistently higher than the MAE, which is expected. Also, apart from the Random forest, the classical machine-learning model produced comparable results.

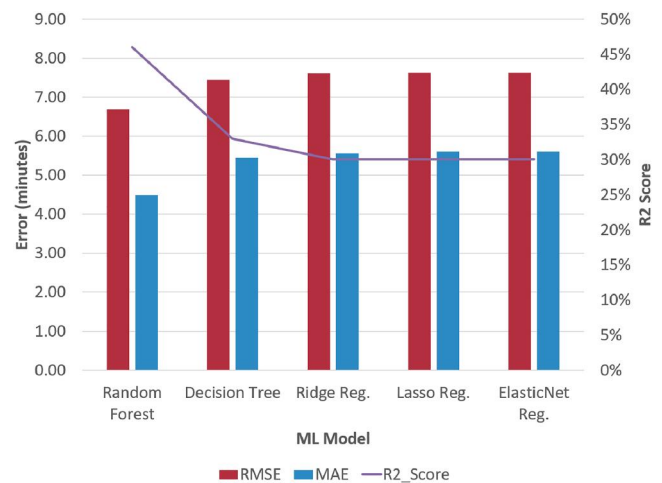
Table 8 and Figure 17 show results based on the 10-fold cross-validation method. Since the data size is enormous, the

**TABLE 7** Results based on the 70:30 split method.

ML model	RMSE mins	MAE mins	$R^2$ _Score
Random forest	6.6	4.4	0.47
Decision tree	7.44	5.42	0.33
Ridge regression	7.61	5.52	0.3
Lasso regression	7.63	5.57	0.294
Elastic net regression	7.63	5.57	0.294
DNN	6.95	4.56	0.36
ANN	6.96	4.54	0.36

**FIGURE 16** Models results with a 70:30 split.**TABLE 8** Results based on the cross-validation method (10-fold).

ML model	RMSE mins	MAE mins	$R^2$ _Score
Random forest	6.686	4.485	0.46
Decision tree	7.45	5.45	0.33
Ridge regression	7.61	5.55	0.3
Lasso regression	7.63	5.6	0.3
Elastic net regression	7.63	5.6	0.3

**FIGURE 17** Models results with a 10-fold cv.

results do not differ much between the two methods. Here also, Random Forest performed the best with an RMSE of 6.69 min and an MAE of 4.49. The  $R^2$ -score decreased slightly to 46%.

Figure 18 compares the actual time values and the predicted values using the Random forest model. The figure shows the best-fit line based on the distribution of the data. One can notice that some predictions are far from the actual time. This is partially expected as this is real data with many factors affecting the actual time. One way to improve the prediction would be not to include the types of services the patient requested but also to figure out who is performing these services in the centre on that day. This could significantly improve the accuracy; however, this could raise lots of privacy concerns about employee monitoring.

Figures 19 and 20 show the RMSE values per hall and per service, respectively. The Halls and services on the x-axis are sorted according to their contribution percentage of the 80 k tickets that represent the testing data. The percentage is shown

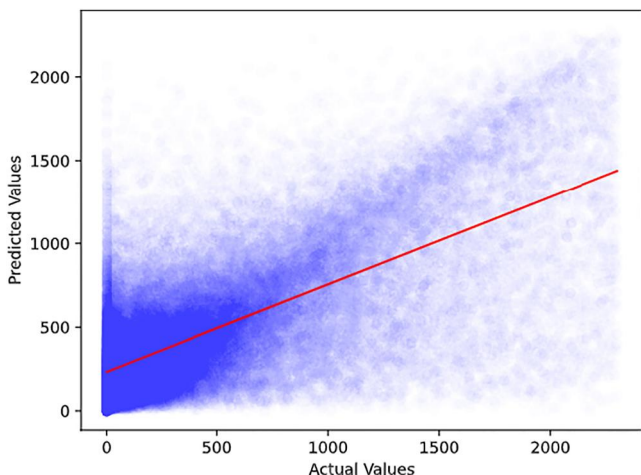


FIGURE 18 Actual versus Random Forest-predicted values (For the entire testing data).

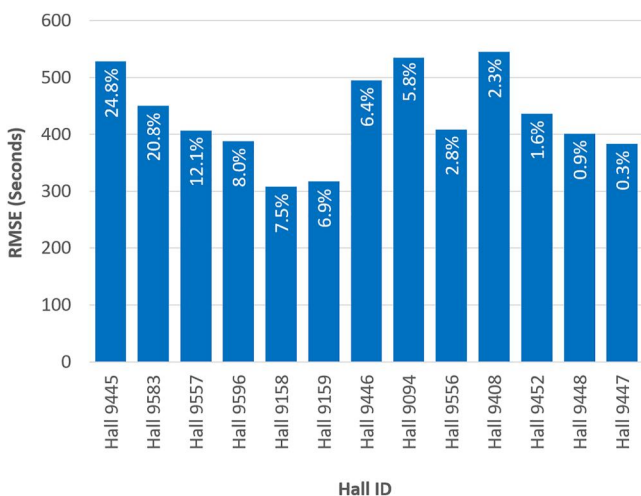


FIGURE 19 RMSE per hall (Single-stage requests).

as the label on top of the bar. Note that in Figure 19, Hall 9445 has the most samples within the test data (24.8%). However, the highest RMSE refers to Hall 9408, which has just 2.3% of the samples within the testing data, which means for this Figure, it is not necessary that the hall with the highest number of samples gets the highest RMSE. Also, the same concept can be mapped to Figure 20. Note that the services that appear in the testing data are 176 services, and this figure shows the RMSE of the top 10 (Most popular) services along with the average RMSE of the remaining 166 services.

### 4.3.2 | Feature importance

One of the most important results in any machine learning-based solution is to understand which features contributed more to the algorithm's decision-making process. Algorithms such as the Random Forest provide a sorted list of features' importance. Figure 21 shows the top 10 significant features based on their importance level.

Note that all the fetched features from the Customer\_Transaction table, which are the Day of the Month, Day of the

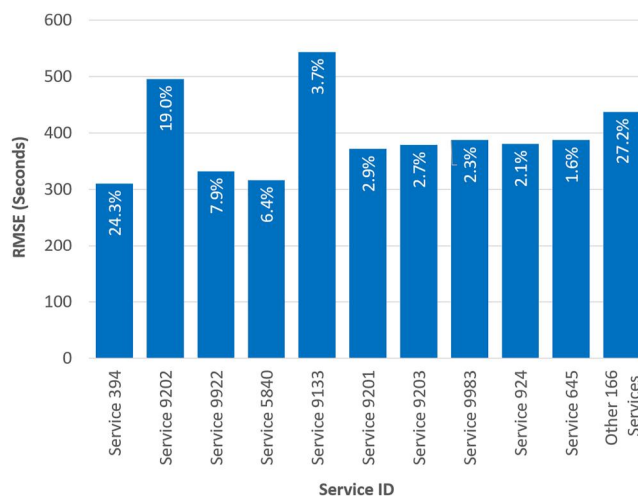


FIGURE 20 RMSE per service (Single-stage requests).

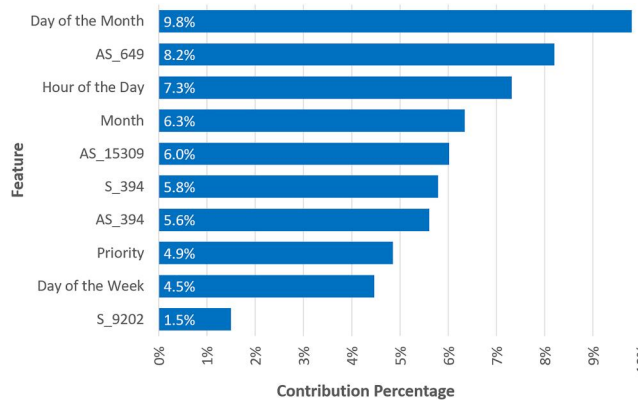


FIGURE 21 Features' importance—top 10 features.

week, Hour of the day, month, and Priority, are within the top 10 features. For the Hospital\_status table's features, their importance can change based on the available data; thus, based on the current data, the status features that significantly affected the model's predictions are shown in the figure.

### 4.3.3 | Proposed ML-based versus AST method

To further investigate the performance of the ML-based method, it is compared to the AST method. The ML models were trained on the entire data except for the last month, which represents the testing portion, as the AST values exist only for the last month, which is 15677 tickets. All of them are single-stage tickets since the AST model predicted only single-stage requests. Table 9 shows the results that demonstrate how superior the ML-based model is to the AST model. Note that the model that produced the best results over the testing data is the DNN.

Based on the aforementioned results, it can be noted that the improvements the ML-based model achieved compared with the AST method model are

- RMSE: ML-based model achieved a 25.1% improvement.
- MAE: ML-based model achieved an 18.9% improvement.

Figure 22 illustrates the comparison between the ML-based and AST methods.

Meanwhile, Figure 23 compares the actual time and the predicted time using the AST method based on the 15 k tickets that were used to establish the comparison between the ML model and the AST method. Note that the red line indicates the best-fit line for the given data. Figure 24 shows a similar

TABLE 9 Machine-learning model versus AST model.

ML-based model		AST model	
RMSE mins	MAE mins	RMSE mins	MAE mins
6.46	4.5	8.62	5.55

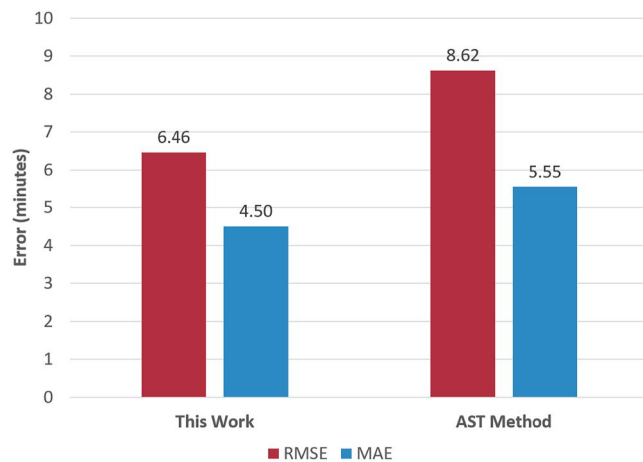


FIGURE 22 ML-based versus AST results.

graph using the ML-based method. It can be seen that the distribution of predictions based on the ML-based model is more linear than the AST method, and this illustrates the improvement in the MAE and RMSE that the ML-based model achieved.

### 4.4 | Limitations

One of the significant limitations of the proposed solution is common to all machine learning solutions. Any ML-based model is as good as the data used to train it, and since the hospital environment is dynamic, there is always a need to keep retraining the model. This retraining process overcomes changes in the environment, such as changes in the employees who perform the services, changes to the health protocol and how services are performed, or even the order of the stages in a multi-stage queue.

It can be seen that the model is flexible and can be generalised to any healthcare facility, as the model did not

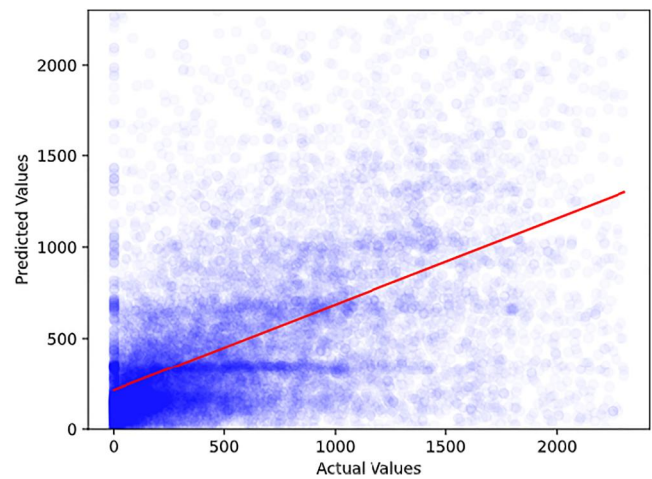


FIGURE 23 Actual versus AST predictions.

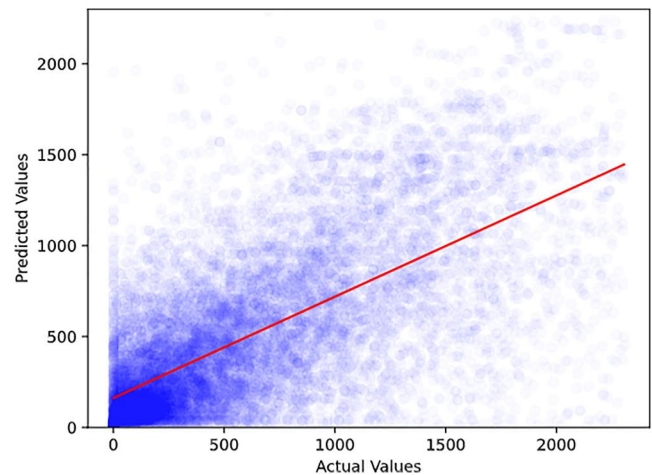


FIGURE 24 Actual versus ML-based predictions.



include any information about the patient's demography or the specific stages in the queue. The only items specific to this healthcare facility were the number of halls and the number of services. These can be easily modified, and the proposed model can be used in any healthcare facility, provided it is retrained using data from the new facility.

## 5 | CONCLUSIONS

This work aimed to design a machine learning model to predict the patients' waiting time in healthcare facilities accurately. The work presented how a typical transactional dataset can be transformed into an ML-ready dataset suitable for wait time prediction and how a set of features can be engineered to enhance prediction accuracy. To evaluate the accuracy of the approach, a diverse set of ML-based algorithms and grid search techniques were employed to find the best algorithm and combination of hyperparameters that would fit this problem, which turned out to be the Random forest with an RMSE of 6.7 min. Moreover, the proposed ML-based model outperformed the currently used AST method. The proposed model achieved a 6.46 RMSE compared with 8.62 RMSE for the AST method. The technique also supports multistage requests, which the AST method does not.

Considering their architecture and daily congestion, it is worth noting that the results reported for this setup can be applied in different enterprises, such as banks, restaurants, and airports. It is not restricted by any clinic type or number of stages, making it applicable to various medical and non-medical applications.

The future scope of this work can include expanding the input features of the model to include more information about the nurses and administrators on duty, which could significantly enhance the accuracy of the prediction. Also, another area for future investigation is if one can identify bottlenecks in the multi-stage queue and be able to reallocate resources automatically.

## AUTHOR CONTRIBUTIONS

**Amjed Al-Mousa:** Conceptualisation; formal analysis; funding acquisition; investigation; methodology; project administration; resources; supervision; writing – review & editing. **Hamza Al-Zubaidi:** Data curation; formal analysis; investigation; methodology; software; validation; visualisation; writing – original draft. **Mohammad Al-Dweik:** Data curation; formal analysis; investigation; methodology; software; validation; writing – original draft.

## ACKNOWLEDGEMENTS

Special thanks to SEDCO company represented by Eng. Hazem Al'aseer, Eng. Majd Jaraa, and Eng. Noor Abdulhameed for their support during the project.

## CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

## DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

## ORCID

Amjed Al-Mousa  <https://orcid.org/0000-0002-6427-1008>

## REFERENCES

- Jaiswal, R., Agarwal, A., Negi, R.: Smart solution for reducing the COVID-19 risk using smart city technology. *IET Smart Cities* 2, 82–88 (2020). <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-smc.2020.0043>
- Lakhan, A., et al.: Digital healthcare framework for patients with disabilities based on deep federated learning schemes. *Comput. Biol. Med.* 169, 107845 (2024). <https://doi.org/10.1016/j.compbimed.2023.107845>
- Rajinikanth, V., Yassine, S., Bukhari, S.: Hand-Sketches based Parkinson's disease screening using lightweight deep-learning with two-fold training and fused optimal features. *Int. Jo. Math. Stat. Comput. Sci.* 2, 9–18 (2023). <https://doi.org/10.59543/ijmscs.v2i.7821>
- Lakhan, A., et al.: Autism Spectrum Disorder detection framework for children based on federated learning integrated CNN-LSTM. *Comput. Biol. Med.* 166, 107539 (2023). <https://doi.org/10.1016/j.compbimed.2023.107539>
- Al-Mousa, A., et al.: Diagnosis of polycystic ovary syndrome using random forest with bagging technique. In: 2023 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), pp. 187–192 (2023)
- Murray, J.: 'they were overwhelmed': families tell of deaths waiting for NHS Urgent Care. *Guardian* (2022). <https://www.theguardian.com/society/2022/feb/01/overwhelmed-families-deaths-waiting-urgent-care-pandemic>. Accessed 20 Dec 2022
- Karl, F., et al.: Multi-objective hyperparameter optimization in machine learning—an overview. *ACM Trans. Evol. Learn. Optim.* 3(12), 1–50 (2023). <https://doi.org/10.1145/3610536>
- Joseph, A., et al.: Predicting waiting times in radiation oncology using machine learning. In: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, pp. 1024–1029 (2017)
- Qamili, R., Shabani, S., Schneider, J.: An intelligent framework for issue ticketing system based on machine learning. In: 2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW), Stockholm, Sweden, pp. 79–86 (2018)
- Curtis, C., et al.: Machine learning for predicting patient wait times and appointment delays. *J. Am. Coll. Radiol.* 15(9), 1310–1316 (2018). <https://doi.org/10.1016/j.jacr.2017.08.021>
- Sanit-in, Y., Saikaew, K.R.: Prediction of waiting time in one-Stop Service. *Int. J. Mac. Learn. Comput.* 9(3), 322–327 (2019). <https://doi.org/10.18178/ijmlc.2019.9.3.805>
- Rastpour, A., McGregor, C.: Predicting patient wait times by using highly deidentified data in mental health care: enhanced Machine Learning Approach. *JMIR Ment. Health* 9(8), e38428 (2022). <https://doi.org/10.2196/38428>
- Gomes, A., Nabil, R.H., Nur, K.: Banking queue waiting time prediction based on predicted service time using support vector regression. In: 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM) (2020)
- Kyritsis, B.I., Deriaz, M.: A machine learning approach to waiting time prediction in queuing scenarios. In: 2019 Second International Conference on Artificial Intelligence for Industries (AI4I) (2019)
- Hijry, H., Olawoyin, R.: Predicting patient waiting time in the queue system using deep learning algorithms in the emergency room. *Int. J. Ind. Eng. Oper. Manag.* 03(01), 33–45 (2021). <https://doi.org/10.46254/ieom.20210103>
- Kuo, Y.-H., et al.: An integrated approach of machine learning and systems thinking for waiting time prediction in an emergency department. *Int. J.*

- Med. Inf. 139, 104143 (2020). <https://doi.org/10.1016/j.ijmedinf.2020.104143>
17. Anussornnitisarn, P., Limlawan, V.: Design of advanced queue system using artificial neural network for waiting time prediction. *Revista Espacios* 44 (2020)
  18. Satya Hermanto, R.P., Suharjito, D., Nugroho, A.: Waiting-time estimation in bank customer queues using RPROP neural networks. *Procedia Comput. Sci.* 135, 35–42 (2018). <https://doi.org/10.1016/j.procs.2018.08.147>
  19. Al-Zubaidi, H., Dweik, M., Al-Mousa, A.: Stroke prediction using machine learning classification methods. In: 2022 International Arab Conference on Information Technology (ACIT), Abu Dhabi, United Arab Emirates, pp. 1–8 (2022). <https://doi.org/10.1109/ACIT57182.2022.10022050>
  20. Alharbi, E., Cherif, A., Nadeem, F.: Adaptive smart eHealth framework for personalized asthma attack prediction and safe Route recommendation. *Smart Cities* 6(5), 2910–2931 (2023). <https://doi.org/10.3390/smartcities6050130>
  21. Hanke, S., et al.: AI-based predictive modelling of the onset and progression of dementia. *Smart Cities* 5(2), 700–714 (2022). <https://doi.org/10.3390/smartcities5020036>
  22. Atari, M., Al-Mousa, A.: A machine-learning based approach for detecting phishing URLs. In: 2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA), San Antonio, TX, USA, pp. 82–88 (2022). <https://doi.org/10.1109/IDSTA55301.2022.9923050>
  23. Vakharia, V., et al.: Estimation of lithium-ion battery discharge capacity by integrating optimized explainable-AI and stacked LSTM model. *Batteries* 9(2), 125 (2023). <https://doi.org/10.3390/batteries9020125>
  24. Khalifeh, S., Al-Mousa, A.: A book recommender system using collaborative filtering method. In: International Conference on Data Science, E-Learning and Information Systems 2021, pp. 131–135 (2021). <https://doi.org/10.1145/3460620.3460744>
  25. Ridge regression. *Mach. Learn. J.* (2020). [Online]. <https://machinelearningjourney.com/index.php/2020/02/13/ridge-regression/>. Accessed: 20-May-2023
  26. Chaya: Random forest regression. *Medium* (2022). [Online]. <https://levelup.gitconnected.com/random-forest-regression-209c0f354c84>. Accessed: 31 May 2023
  27. How many hidden layers and hidden nodes does a neural network need? - technical articles. *All About Circuits.* (2020) [Online]. <https://www.allaboutcircuits.com/technical-articles/how-many-hidden-layers-and-hidden-nodes-does-a-neural-network-need/>. Accessed: 20-May-2023
  28. Johnson, J.: What's a deep neural network? Deep Nets explained. *BMC Blogs* (2020). [Online]. . <https://www.bmc.com/blogs/deep-neural-network/>. Accessed: 20 May 2023
  29. Root mean square error (RMSE). *C3 AI*, 28-Sep-2021. [Online]. <https://c3.ai/glossary/data-science/root-mean-square-error-rmse/> Accessed: 01-May-2023
  30. Abba, I.V.: What is R squared? R2 value meaning and definition. *freeCodeCamp.org*, 28-Mar-2023. [Online]. <https://www.freecodecamp.org/news/what-is-r-squared-r2-value-meaning-and-definition/>. Accessed: 10 May 2023

**How to cite this article:** Al-Mousa, A., Al-Zubaidi, H., Al-Dweik, M.: A machine learning-based approach for wait-time estimation in healthcare facilities with multi-stage queues. *IET Smart Cities*. 1–18 (2024). <https://doi.org/10.1049/smc2.12079>