Predicting Patient Visit Duration with Neural Networks

August 1st, 2025

Wesley Ducharme
Faculty of Computer Science
University of Victoria
Victoria, BC
V00974267

Parker DeBruyne
Faculty of Health
University of Victoria
Victoria, BC
V00837207

Lee Napthine
Faculty of Computer Science
University of Victoria
Victoria, BC
V00986425

Shaban Shala
Faculty of Engineering
University of Victoria
Victoria, BC
V00945372

1 Abstract

Long wait-times in Canadian hospitals are a well-known issue, and Bed Hours Occupied (BHO) remains a critical metric for managing patient flow. Our project explored whether ML—specifically Neural Networks—could predict individual patient visit durations to help forecast BHO more accurately. Using synthetic EHR data from Synthea, we engineered features across encounters, procedures, and patient demographics. Early regression models failed to generalize, largely due to extreme outliers and nonlinear patterns. Inspired by related work using neural models for hospital demand forecasting, we reframed the task as classification. Our final neural network reached 79% accuracy, surpassing our 70% target baseline, and 91% recall, performing well on long-stay cases—those with the highest operational impact. Results suggest that even lightweight neural pipelines can capture useful clinical patterns and offer real potential for improving hospital resource planning.

2 Introduction and Problem Statement

Excessive wait-times in the Canadian Healthcare System have become so problematic and pervasive that the issue is now considered common knowledge. Most Canadians have a friend, family member, or colleague reporting experiences in excess of 6 hours long, and the subject has been covered exhaustively in current literature. Healthcare institutions are optimistic, however, about the improved organizational efficiency dawning on the AI-horizon, and our project sought to experiment with the application potential of Machine Learning (ML) methods in Healthcare.

In particular, patient flow efficiency is often measured in some variation of Patient Bed Hours (PBH), which is a metric that sums the hours that a single patient occupied a hospital bed [1]. Bed Hours Occupied (BHO) extends this as a ward-specific ratio of total PBH to the maximum available staffed bed hours. This BHO metric then informs staffing decisions, equipment purchases, care quality measurements, investment planning, epidemic preparation, health policy adaptation, and much more. Most hospitals rely on historical and moving averages to anticipate BHO trends; the application of ML for BHO forecasting seems an exciting opportunity.

3 Related Work

Morid et al conducted a thorough literature review of different ML models applied in Healthcare as of 2023 [2]. Due to the high-dimensionality and temporal characteristics of Electronic Health Record (EHR) data, they found that the majority of successful ML applications utilized Deep Learning (DL) Neural Networks (NN).

In particular relevance to our project, Khaldi et al trained a hybrid Artificial Neural Network (ANN) model enhanced with Ensemble Empirical Mode Decomposition (EEMD) to forecast the volume of patient arrivals in hospital Emergency Departments—its results significantly outperformed traditional forecasting methods [3]. Our objective differed slightly, as we aimed to predict the duration of individual patient visits in order to inform BHO calculations.

4 Data and Methodology

In this section we will detail our methodology for developing our models. We will start by first going through our preprocessing steps. Then we will outline our initial models and their results, which will lead into why those results made us rethink our approach to the problem. Finally we will explore our final models and the steps we used to train and tune them as well as our testing results.

4.1 Data Source

We used synthetic EHR data generated by Synthea, an open-source tool that simulates realistic but fictional patient records [4]. It models complete medical histories and healthcare interactions based on U.S. care patterns, and exports structured tables in CSVs such as *encounters*, *patients*, and *organizations*.

For our project, we primarily worked on an enriched and feature-engineered version of the *encounters* table, which included the start and end times for each visit. From that, we calculated the duration of each encounter in hours—this became the target variable we aimed to predict. Each duration gives a direct measure of how long a patient occupied a bed, which ties back to our goal of modeling Bed Hours Occupied (BHO).

Since the data was fully synthetic and privacy-safe, it gave us the freedom to experiment with modeling approaches without running into any ethical or regulatory walls.

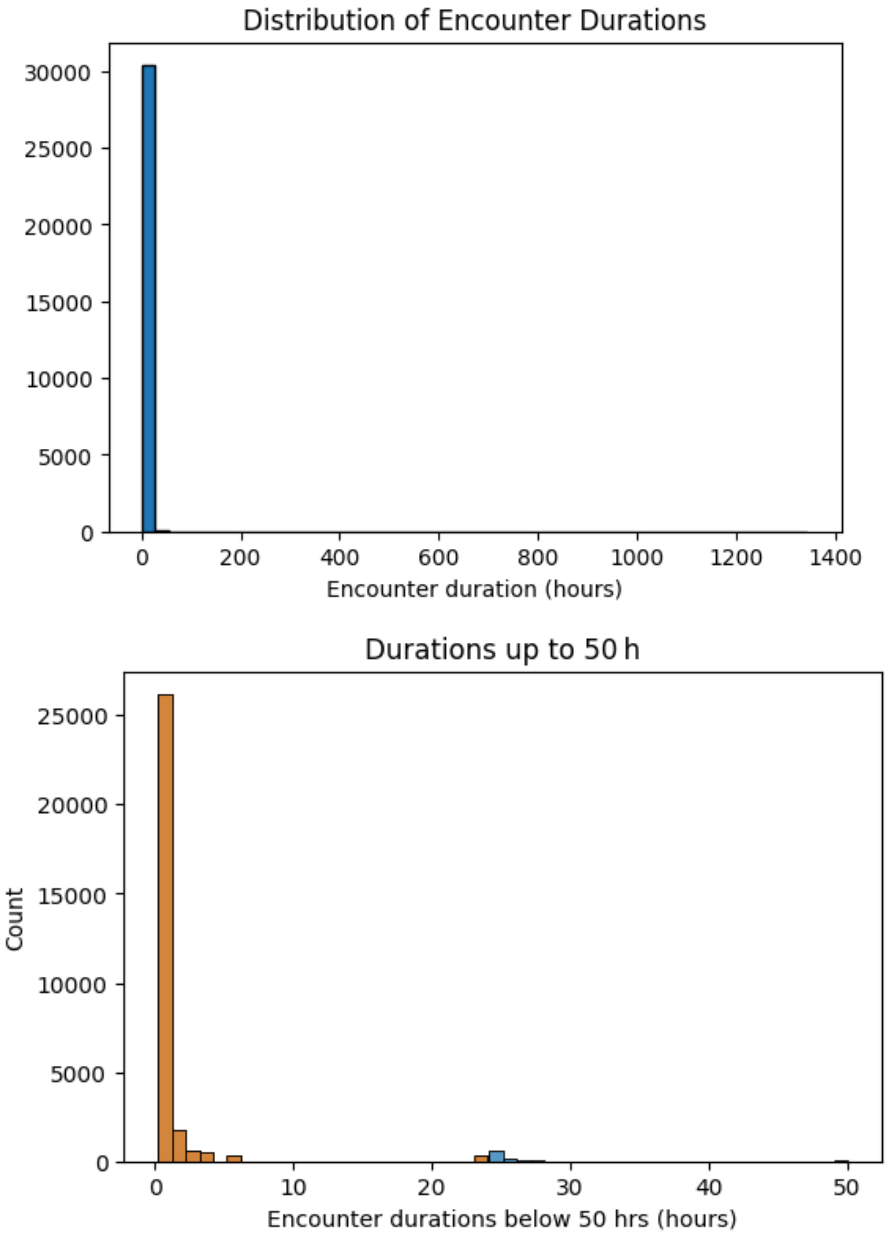
4.2 Initial Preprocessing

Because we had so many datasets, we used Synthea's data dictionary to guide our implementation of a PostgreSQL database that we used to query and engineer our features and our target variable. We calculated our target variable to be the length of a medical encounter in hours (continuous) which we calculated by subtracting the encounter's start time from its end time. We then engineered other features by joining tables together and making features that we thought may yield some relationship with encounter length. Some examples include the number of procedures or surgeries a patient is scheduled to have, reason for the encounter, and whether the encounter is an emergency or not, etc. Once we had compiled all these features into a view we then exported it to a csv which we would then be using as our final dataset for training our models.

After importing the dataset into our notebook file we started by setting up our target and feature set and we instantly ran into a problem with encoding categoricals. Many of our variables that would normally be one-hot-encoded ended up having a very high number of categories which would dramatically increase the width of our feature set. After researching a way to remedy this we found a library called CatBoostEncoder. It encodes categorical variables by using the mean of the target variable for each category but it also adds smoothing (based on the number of samples in each category) which is why it worked well with our high-cardinality categorical features. It also comes with the bonus of preventing overfitting, since we won't be exploding our feature dimensions anymore. After the features were encoded we then lastly scaled the features to get them ready for training our initial models

4.3 Baseline Models

Our initial models included a basic linear regression model, a stochastic gradient descent model, and a simple neural network. We first split the data into 80% training and 20% testing splits to see how well these basic models generalize with minimal tuning. The linear regression model and gradient descent model saw the best results with degree 2 polynomial features and the 1 layer neural network yielded similar results to the linear models. These initial results were very poor, with the linear and neural network models both yielding very low R^2 values (0.00456) and very high RMSE values (10000+). These early results made us question our ability to extract useful information from our dataset. After consulting some graphs of our target variable shown below we saw why our early results were so poor.



Bar Charts of our Target Variable Distribution.

As seen in the graphs above most durations only last around 4-5 hours with very few extreme outliers, which makes sense, as the majority of medical encounters (based on our understanding) are checkups and consultations with some operations and surgeries of varying short lengths. To try to see if our original models may still have some promising results we tried both square root and logarithmic target variable transformations. These transformations yielded equally poor or worse results so we needed to try to find a solution to our outlier problem. This solution initially took the form of capping the extremely large outliers using the IQR method (Quantile3 + IQR * IQR)capping all durations above 50 hours to 50 hours. This immediately led to a sharp increase in performance for our models with the linear regression models yielding a RMSE of 1.42 and R^2 of 0.92. So 92% of the variance in our capped target was being explained by our feature set. However we wanted our model to work on an accurate representation of the

data and be able to recognize those extreme outliers. To do this we elected to change our approach to the problem.

4.4 Change in Approach

We changed our target variable from a continuous variable to a categorical variable. We categorized encounter durations into six classes and aimed to predict the class an encounter would fall under given our specific feature set.
These six categories are outlined below:

- Quick (duration <= 0.25)
- Short (0.25 < duration <= 3)
- Moderate (3 < duration <= 8)
- Long (8 < duration <= 24)
- Very Long (24 < duration <= 50)
- Extreme (duration > 50)

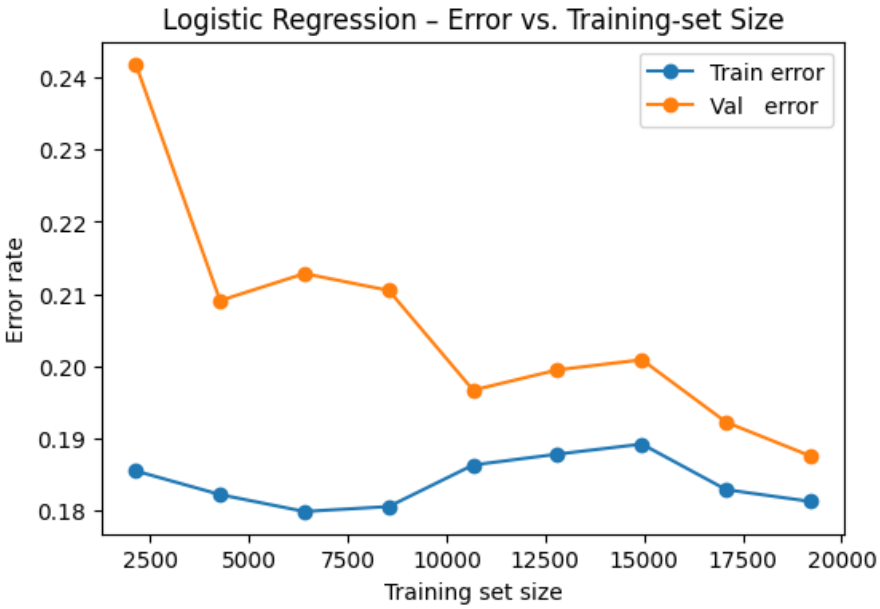
The aim for the change in approach is to retain the improvement in accuracy that capping outliers gave us while still accurately representing the true values of our data as best we could. Our approach now changing to a classification also now leads to us changing our model type selection and in the next section we will outline the development of our final and best performing models.

4.5 Final model development and debugging

The two models we achieved the best results with (and will be comparing) are a multinomial logistic Regression model and a Neural Network model. The workflow between both models was similar. For both, we started by balancing the class weights, since the counts for certain classes varied significantly in our training data. We then split the data into 80% training and 20% testing sets. During model development, we performed 3-fold cross-validation on the training split, ensuring each fold served once as the validation set. After cross-validation, we did one final round of training by further splitting the training set into 80% training and 20% validation subsets to build our learning curves and check for bias and variance in our models. Finally, we retrained each model on the full 80% training data and evaluated its performance on the unseen 20% test set to obtain our final results.

4.5.1 The Logistic Model

When tuning the multinomial logistic model we saw the best accuracy with degree 2 polynomial features. With the model's initial accuracy being 80%, it was looking good performance wise but when we checked our early learning curves it showed that the model exhibited significant overfitting.

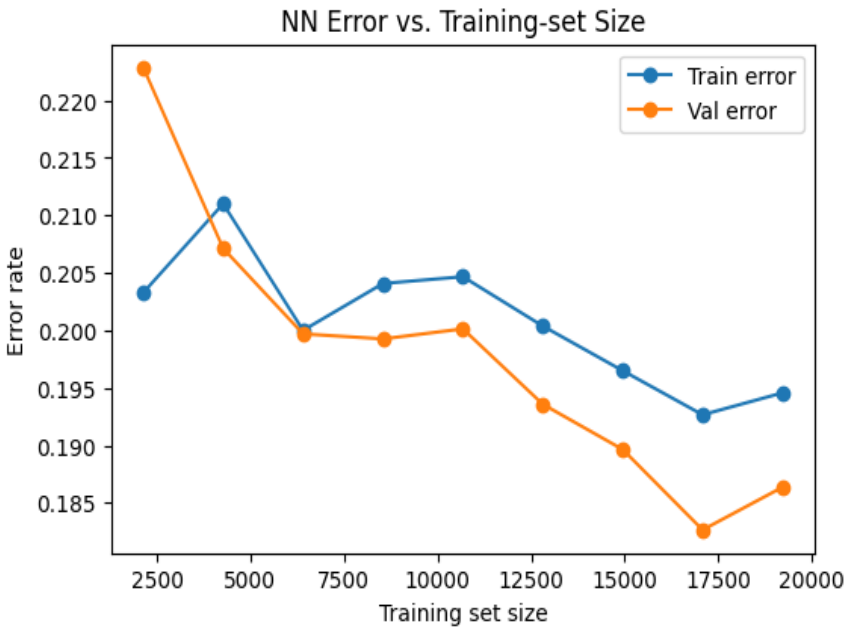


First Logistic Regression Learning Curve.

To try to remedy this we ended up implementing feature reduction in our logistic model pipeline, removing features based on L1 weights from the base feature set before applying polynomial features. This was done in this sequence of steps for runtime efficiency. The feature reduction led to our overfitting problem being fixed but also led to a reduction in overall accuracy down to 78%. It also resulted in a learning curve that failed to converge and the details around our interpretation of these results and the final curve will be covered in our results section of this report.

4.5.2 The Neural Network Model

For our new classification Neural Network model we started with a 2 layer model, first layer starting with 128 neurons and the second layer with 64 neurons. We used RELU activation and a small amount of dropout regularization (10%). We elected to use dropout regularization to help mitigate overfitting. It works by "dropping" a random portion of neurons in each layer, every training iteration. This helps the model generalize since neurons in forward layers will become less dependent on specific outputs from previous neurons. This 2 layer NN resulted in an overall accuracy of 81% which was satisfactory. But when we checked our learning curves for bias and variance and noticed a somewhat similar problem as with the logistic model.



First 2 Layer Neural Network Learning Curve.

As we can see in the above curve the models start to overfit with more data, so we elected to tweak the model's parameters to try to both improve our variance problem and retain close to the same level of accuracy.

We tried a number of different tuning steps, but got the best results after making the following adjustments to the Neural Network in order:

1. Increase dropout regularization to 35% (Fixed overfitting but reduced accuracy to ~68%).
2. Increased the number of hidden layers to three.
3. Adjusted neuron counts: 128 (layer 1), 64 (layer 2), and 32 (layer 3), using ReLU activations.

With these adjustments we found our most balanced / best performing model, The results and learning curves of this model are discussed in our results section below.

5 Results and Analysis

5.1 Evaluation metrics

To evaluate our models, we used the following metrics: accuracy, precision, recall, and F1-score. From these, the metric we thought was most important was recall. Since we aimed to forecast patient hospital stay (bed hours occupied), it was important that we correctly identified cases especially those who would stay for long periods. Failing to predict these cases could lead to insufficient bed or staff availability, making resource planning inefficient potentially leading to resource issues. We chose a baseline accuracy of 70%.

5.2 Logistic Regression Results

Our Logistic Regression model achieved 78% accuracy and an average recall of 89%, which beat our baseline indicating a good overall performance. The model struggled to identify shorter visits ("Short" class), having a recall of 70% which was the lowest from all the classes. The model seemed to have troubles separating mid-range durations, which is likely due to class overlap, or just patterns that the model struggled to learn.

Test set results:

	precision	recall	f1-score
Quick (≤0.25h)	0.79	0.83	0.81
Short (0.25–3h]	0.82	0.70	0.75
Moderate (3–8h]	0.47	0.99	0.63
Long (8–24h]	0.85	0.89	0.87
Very Long (24–50h]	0.71	0.98	0.82
Extreme (>50h)	0.65	0.96	0.78
accuracy			0.78
macro avg	0.71	0.89	0.78
weighted avg	0.79	0.78	0.78

Table showing results of LR

Logistic Regression Curve

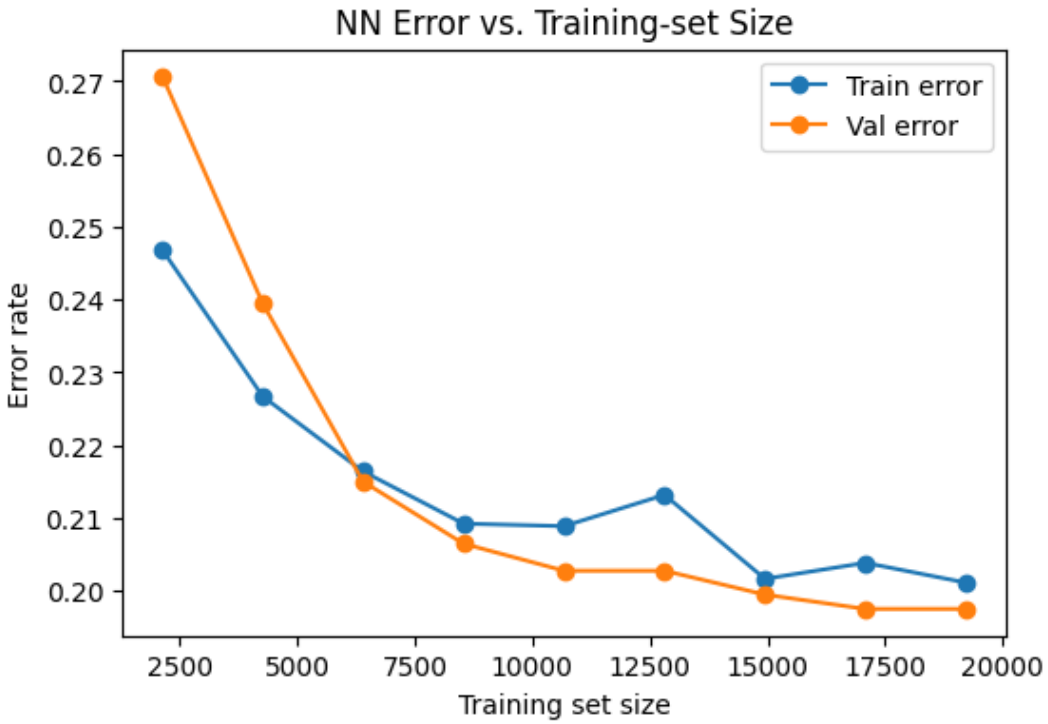
The learning curve for Logistic Regression showed signs of underfitting, however we can see that the graph doesn't plateau therefore this model needs more data to train on as it can keep learning, and may also need further tuning.

5.3 Neural Network Results

Our Neural Network model achieved 79% accuracy and an average recall of 91%, which also beat our baseline of 70% accuracy. This model also struggled to identify shorter visits ("Short" class), having a recall of 59% for that class. This value was significantly lower than the other values, showing that this model had troubles with class overlap, and potentially struggled to learn certain patterns (maybe more difficult or subtle patterns).

	precision	recall	f1-score
Quick (≤0.25h)	0.73	0.96	0.83
Short (0.25–3h]	0.96	0.59	0.73
Moderate (3–8h]	0.50	0.98	0.66
Long (8–24h]	0.81	0.96	0.88
Very Long (24–50h]	0.88	0.98	0.93
Extreme (>50h)	0.48	1.00	0.65
accuracy			0.79
macro avg	0.73	0.91	0.78
weighted avg	0.83	0.79	0.78

Table showing results of NN



Neural Network Learning Curve

The learning curve for the Neural Network model showed some error gaps, but was minimal enough to not affect the model. The graph shows both the training and validation errors plateauing meaning it has completed its training. This indicates the model was able to generalize well to the test data given.

5.4 Results Comparison

Both models performed above the baseline accuracy of 70%, and had some similarities and differences between them. Both models showed struggles when predicting the short duration class. This suggests that shorter hospital stays were harder to distinguish based on the available features, overlap between classes, and may require either additional data, or more targeted feature engineering to improve the performance in this category. Both models also struggled with precision for the moderate class, Logistic Regression having 47% and Neural Network having 50%.

The Neural Network achieved a slightly higher average recall, precision, and accuracy than the logistic regression, with an improvement of 2% for recall, 2% for precision, and 1% for accuracy. While the overall accuracy only increased by 1%, the improvements in recall and precision are important for identifying hospital stays, especially the longer ones. These improvements together with the more stable learning curve suggest that the Neural Network was better at identifying patterns, making it the better model.

6 Discussion and Limitations

6.1 Discussion

Our Neural Network pipeline successfully revealed clinical patterns in the EHR dataset and achieved encouraging result metrics. With a 79% accuracy and an average recall of 89%, the likelihood of missing long-duration patient stay is low, reducing potential under-staffing, overcrowding, or further delays. Our model was particularly useful at predicting correct length times of long-stay cases.

Certain features had a predictable influence on the results of our study and their correlation further increased our confidence in the model's usefulness. Variables such as encounter class, reason for visit, medication costs, and patient age carried the most weight in predictions. This supports that the network was learning from medically useful patterns rather than overfitting on artifacts.

However, performance varied between duration buckets. Both the Logistic Regression and the Neural Network performed best at extreme encounter length (< 15 minutes and > 24 hours), but struggled with mid-range visits (3-8 hours). This suggests that these moderate duration cases likely share certain characteristics, making them harder to distinguish under our current feature set.

The Logistic Regression model, while slightly lower in overall recall (89%) outperformed the Neural Network on shorter visits with a 70% recall. This trade off highlights the potential benefits of a hybrid approach in our future work.

6.2 Limitations

We had only two weeks for feature engineering, model development, and evaluations. This constrained our ability to optimize our data set and dig deeper into the cause of the lower-scores returned from mid-range encounters. The original dataset itself was synthetic and potentially lacked some of the regular noise and bias contained in other authentic data. This may have inflated performance metrics that could have been correctly accounted for given a longer time horizon.

The high-cardinality of certain features ballooned our data set. We used CatBoostEncoder to somewhat mitigate this, but the sheer size of the training data (44 million entries) may have had an effect on the overall interpretability of the results.

7 Conclusion and Future Work

7.1 Conclusion

This study demonstrates that even lightly tuned feedforward Neural Networks, when trained on well-engineered features, can predict patient encounter duration and offer value for estimating Bed Hours Occupied. Reframing encounter length as a classification problem partially remedied the impacts of extreme outliers and helped us derive meaning from our otherwise messy early results.

Our model's ability to identify long visits and its encouraging recall metrics mean it could be a useful tool for capacity planning, as well as insurance risk modeling, and public health cost forecasting.
While results on mid-range visits were mixed, the pipeline showed an ability to learn patterns that were useful for our BHO prediction goals, making it a promising foundation to build on.

7.2 Future Work

Going forward, we would like to expand on our results in the areas of:

- **Improving mid-range predictions**
  Explore clustering misclassified mid-range encounters to gain an understanding of their clinical significance. Based on the findings, engineer new features to capture any inconsistencies that may have skewed predictions.

- **Time-series analysis targeting future encounters**
  Develop recurrent Neural Network models (e.g., GRU, Bi-LSTM) to predict the number of days until a patient's next medical encounter. In addition to our current encounter time target, this would give further context and predictive insight.

- **Fairness and bias**
  Tune our model for fairness metrics, considering both individual and group fairness, and removing unwanted bias that exceeds acceptable thresholds.

Together, these next steps would aim to improve predictions, fairness, and general value of the entire project. With additional time and tuning, our system could become a practical tool for improving patient flow and healthcare resource management.

8 References

[1] L. T. Tierney and K. M. Conroy, “Optimal occupancy in the ICU: A literature review,” Aust. Crit. Care, vol. 27, no. 2, pp. 77–84, May 2014, doi: 10.1016/j.aucc.2013.11.003. Available: https://www.sciencedirect.com/science/article/pii/S1036731413002622?via%3Dihub#sec0010

[2] Morid, M. A., Sheng, O. R. L., and Dunbar, J., “Time Series Prediction Using Deep Learning Methods in Healthcare,” ACM Transactions on Management Information Systems, vol. 14, no. 1, Article 2, Jan. 2023. doi: 10.1145/3531326.
Available: https://dl.acm.org/doi/full/10.1145/3531326

[3] R. Khaldi, A. E. Afia, and R. Chiheb, “Forecasting of weekly patient visits to emergency department: real case study,” in *Proc. Procedia Computer Science*, vol. 148, pp. 532–541, 2019, doi: 10.1016/j.procs.2019.01.026. Available: https://www.sciencedirect.com/science/article/pii/S1877050919300262

[4] Synthea, “Synthetic Patient Population Simulator.” GitHub. [Online]. Available: https://github.com/synthetichealth/synthea [Accessed: May 22, 2025].

[5] “CSV File Data Dictionary,” synthetichealth/synthea GitHub Wiki, GitHub, last modified May 8, 2025. [Online]. Available: https://github.com/synthetichealth/synthea/wiki/CSV-File-Data-Dictionary