C308: NHS AI Predictive Renal Health PoC

Produced for: AI Skunkworks Team within the NHS Transformation Directorate

Against Order: 3100070245

Report No: 72/22/R/153/U – X72-AY-10014-003
August 2022 - Issue 02-000

AUTHORISATION

Author(s):
Timothy Gaultney,
Daniel Heath,
Wren Brick

Approved By

Julia Dawson .....
Project Manager

DISTRIBUTION

		Copy No.
Tom Mcgeoch	ACE/Vivace	1
Scott Linfoot	ACE/Vivace	2
Charlie Coleman	AI Skunkworks	3
Xu Gang	UHL	4
Project File	Roke Manor Research Ltd	Master

DOCUMENT HISTORY

Issue no	Date	Comment
00-001	05 May 22	Created document
00-002	03 Aug 22	Draft following JD comments – prepared for ACE review
00-003	10 Aug 22	Addressed ACE/Roke Review Comments
00-004	17 Aug 22	AI Skunkworks/UHL Review Comments
00-005	23 Aug 22	Integrated remaining review comments
01-000	23 Aug 22	Resolved remaining comments and changes
02-000	30 Aug 22	Updated cover page details

CONTENTS

FIGURES

TABLES

1. EXECUTIVE SUMMARY

Introduction

The research project developed a Proof of Concept (PoC) AI model able to predict the need for Intensive Therapy Unit (ITU) admission, dialysis, or the outcome of death for AKI patients in discrete time windows up to 48h ahead of the event. Positive predictive values and sensitivities achieved are encouraging, and explainability methods have highlighted clinical features necessary for the model to work. Overall, the PoC demonstrates potential for better forecasting of patient needs, and merits further development.

Background

This research project is a commission by ACE in partnership with the National Health Service (NHS) Artificial Intelligence (AI) Lab Skunkworks team, the University Hospitals of Leicester (UHL) NHS Trust, and Roke. UHL is one of the largest acute care trusts in England with over 1,700 acute care beds. They have been using electronic record collection for their patients since 2017. UHL currently identify over 200 patients a day who suffer from sudden changes in kidney functions (formally Acute Kidney Injuries or AKI), 30 of which are admitted to intensive care every month, 50-60 are referred for speciality care, and an overall mortality rate of approximately 30% (66 patients per day). The UHL clinical 'outreach' team actively assess these AKI patients to determine those who may require ITU input, prior to a referral occurring.

The Challenge

Deterioration of AKI patients is hard to predict, leading to outcomes such as the need for emergency admission to Intensive Therapy Unit (ITU) or requiring haemodialysis. The aim of this commission is to predict the likelihood/risk of AKI patients who will require ITU admission, dialysis or to die.

Current tools include AKI alerts, which identify patients who have developed kidney disease based on serial blood tests, and the Early Warning Score (EWS) that collates multiple observations such as blood pressure, pulse and heart rate based on predefined parameters at a single time point. Often these arrive too late to change the course of clinical treatment as the patient's situation has already deteriorated (due to the key identifiers such as Creatinine level reflecting a rate of change, requiring a sufficient lead time to detect), and therefore the opportunity to reduce severity of deterioration is limited. False positives under the current system lead to specialised beds occupied by patients who ultimately do not require them, reducing beds available for those in greater need.

The main objective of this research project is to develop a proof-of-concept tool that can make use of the available datasets to predict AKI patients whose cases are likely to include what have been referred to as adverse outcomes. The tool should look to predict the changing requirements of patients towards these outcomes 24 hours or more before the actual event. This will greatly improve the effectiveness and efficiency of UHL's clinical 'outreach' team, as the limited resource within the team prevents it from being able to assess the volume of patients with AKI alerts in a timely manner. This could mean emergency admissions to ITU are reduced and clinicians are forewarned about deterioration before it occurs, therefore offering the opportunity for earlier intervention.

The Approach

The technical approach by the Roke delivery team is based around extending a highly-cited method from DeepMind in 2019 (Nenad, 2019), which predicted incidences of AKI from a large volume of United States Veterans' healthcare record data which is expected to have partial overlap in structure with the dataset for this project. Though the prediction task here is a step beyond predicting AKIs, the challenge is closely

related and the methodology is expected to be viable after some modification. The codebase for DeepMind's approach was made open source in 2021 (EHR Modeling Framework, 2021), with the exception of their data loading pipeline and occlusion analysis tools, which Roke has defined against the dataset for this project. This project also looks to modify the training objective of the RNN-based model to predict post-AKI classes, rather than simply the incidence of an AKI itself.

Therefore, Roke proposed the use of sequence models, such as Recurrent Neural Networks (RNN) to learn important contextual and temporal qualifiers in a series of clinical tests and other data relating to renal patients for predicting the probability of adverse outcomes. The use of AI to make predictions regarding these class labels on a temporal scale will help clinical staff to make timely decisions regarding the best course of action for a patient. Roke's understanding is that there are tools used by clinical staff to identify patients who have developed kidney disease, however delays in detection for a broad group of patients who experience rapid decline in kidney function make it very difficult to opt in/out the necessary course of clinical treatment for the patient, as their situation has already deteriorated. Specialist beds required to treat these patients are also often filled by patients who do not eventually require specialised treatment, and more accurate prediction of which patients require these beds will ease the management of this resource.

The proposed approach includes providing explainability of the models in terms of feature importance. This may accelerate deployment of similar models at hospitals with dissimilar data available, provide targets for new treatment options, and provide assurance to both machine learning specialists and clinicians.

Project Successes

As tools for predicting the chosen adverse outcomes downstream of AKIs are extremely limited to our knowledge, any sequence classifier performing better than random chance may be of interest. In particular, Positive Predictive Value (PPV) and sensitivity at the 24h prediction window for death events were chosen as our metric to determine our most successful model. Our best model achieved 0.397 PPV and 0.159 Sensitivity against the 24h mortality window which was seen as the key prediction window and adverse outcome by the clinical team. The model also achieves a negative predictive value (NPV) of 0.994 and Specificity of 0.998 *(Note: each of these metrics is ideally as close to 1 as possible)*. The trained models were also suitably performant across the board at other time intervals and for the other two adverse outcomes (every six hours up to 48 hours in the future, with performance across time windows shown in Appendix E.2). Detailed performance at the 24h window for each outcome is given in Section 3.2.5. The performance is seen as encouraging for a Proof of Concept tool and justifies further work to improve the performance.

In addition, application of explainability methods, such as occlusion analysis, have highlighted the key fields which the model needed to reach its predictions, which may enable trusts with some overlap in data to determine whether similar approaches would be viable for them.

1 INTRODUCTION

The purpose of this document is to summarise the work completed to build a proof of concept to demonstrate how Artificial Intelligence (AI) can be used to predict those Acute Kidney Injury (AKI) patients at higher risk of requiring admission to the Intensive Therapy Unit (ITU), needing renal support (dialysis), or are likely to die. This document also contains a series of appendices detailing further technical detail associated with the technical development of the Proof of Concept (PoC).

1.1 PROJECT BACKGROUND

This research project is a commission through the ACE (Accelerated Capability Environment), in partnership with the NHS AI Lab Skunkworks team, the University Hospitals of Leicester (UHL) NHS Trust, and Roke. UHL is one of the largest acute care trusts in England with over 1,700 acute care beds. They have been using electronic record collection for patients since 2017, and therefore have access to a wealth of digital data. UHL currently identify over 200 patients a day who suffer from sudden changes in kidney functions (formally Acute Kidney Injuries or AKI). On average, 30 of these patients are admitted to intensive care every month, 50-60 are referred for speciality care, and there is an overall mortality rate of approximately 30% (66 patients per day).

Deterioration of AKI patients is hard to predict, leading to outcomes such as the need for emergency admission to ITU and need for organ support such as dialysis. The aim of this commission is to understand the likelihood/risk of AKI patients who will need ITU admission, dialysis or may die. It is recognised that the inclusion of death as a predicted outcome may cause emotional distress. However, this is important to clinical staff in helping decide the best course of action for a patient, which may include palliative care at home. Current tools include AKI alerts, which identify patients who have developed kidney disease based on serial blood tests, and the Early Warning Score (EWS) that collates multiple observations such as blood pressure, pulse and heart rate based on predefined parameters at a single time point. Though these do allow for alerting of potentially unwell patients, often these result in cases where patients who do not eventually require dialysis are kept on specialised dialysis beds, while patients in need of dialysis are maintained via other medications until a dialysis bed becomes free, during which time their condition can deteriorate compared to what it could have been, were dialysis administered earlier. UHL currently has a team of 20 ‘outreach’ nurses who specialise in targeting and reviewing potentially deteriorating patients in hospital who may require ITU support. However, although EWS is a very powerful tool in identifying a potentially ill cohort, with over 200 patients a day identified in a hospital with 1,700 acute beds, it is not specific enough to allow for truly guided preventive care to be delivered to those patients who are more likely to deteriorate. Currently ‘outreach’ staff spend a considerable part of their day manually identifying patients who need attention, therefore better risk prediction will help improve efficiency and effectiveness of this identification and assessment of patients.

A tool is needed that can help identify (more than 24hrs in advance) those AKI patients at higher risk of requiring ITU, needing renal support (dialysis), or are likely to die. This will greatly improve the effectiveness and efficiency of the UHL ‘outreach’ team. This could mean emergency admission to ITU is reduced and clinicians are forewarned about deterioration before it occurs, therefore offering the opportunity for intervention to prevent deterioration.

This 12-week research project aimed to deliver such a proof-of-concept tool along with accompanying results and discussion from the experiments undertaken implementing and training the tool.

1.2 CURRENT PROCESS

The current clinical process at UHL to detect and respond to AKI events utilises an existing AKI alert system which uses a defined formula to calculate if an AKI event (of a particular stage) has occurred (NHS, 2022). This system uses creatinine blood pathology results. Though a powerful tool, the decision of whether to

transfer a patient to ITU, or whether dialysis is necessary, relies on a clinician's interpretation of the remaining medical history of a patient, with no strictly defined structure to make the decision.

1.3 PROJECT METHODOLOGY

1.3.1 APPROACH

As Electronic Health Records (EHR) have become more prevalent in recent years, a vast trove of data now exists that may contain undiscovered indicators for necessary treatment downstream of AKI events. This EHR data is often timestamped, and can be extremely specific to a particular hospital (e.g., separate codes for specific wards in specific hospitals, unique medication notes defined alongside prescriptions), or they can be more generic (standard blood test results, routine biometric observations, common medications).

Deep Learning (DL) has emerged in the last decade as an invaluable tool for discovering intricate relationships in large datasets, including those in medical fields and sequential data (Lecun, 2015). DL models are composed of interconnected layers that only require loosely defined mathematical relations at the outset; parameters of the model are fine-tuned by observing real data. The approach used within this project is a *supervised learning* setting, where both input data and labels are used to teach the model to relate new input data to a likely outcome. The input data in this instance takes the form of a patient's health data observations, which has been semi-automatically labelled with any adverse outcome events.

Recurrent Neural Networks (RNNs) are a family of DL models that specialise in time series forecasting, which will be essential in a healthcare setting where new information about a patient will emerge over the course of their stay. Additional details of our network can be found in Appendix C.3.

1.3.2 DATASET AND DATA FLOW

The data originated as Comma Separated Value (CSV) files or Excel Workbooks and was processed into structured JavaScript Object Notation (JSON) records for loading into the model training pipeline. This data was provided in a pseudonymised form (unlinked from hospital patient IDs, with names and dates of birth removed). Figure 1 shows this data flow progression.

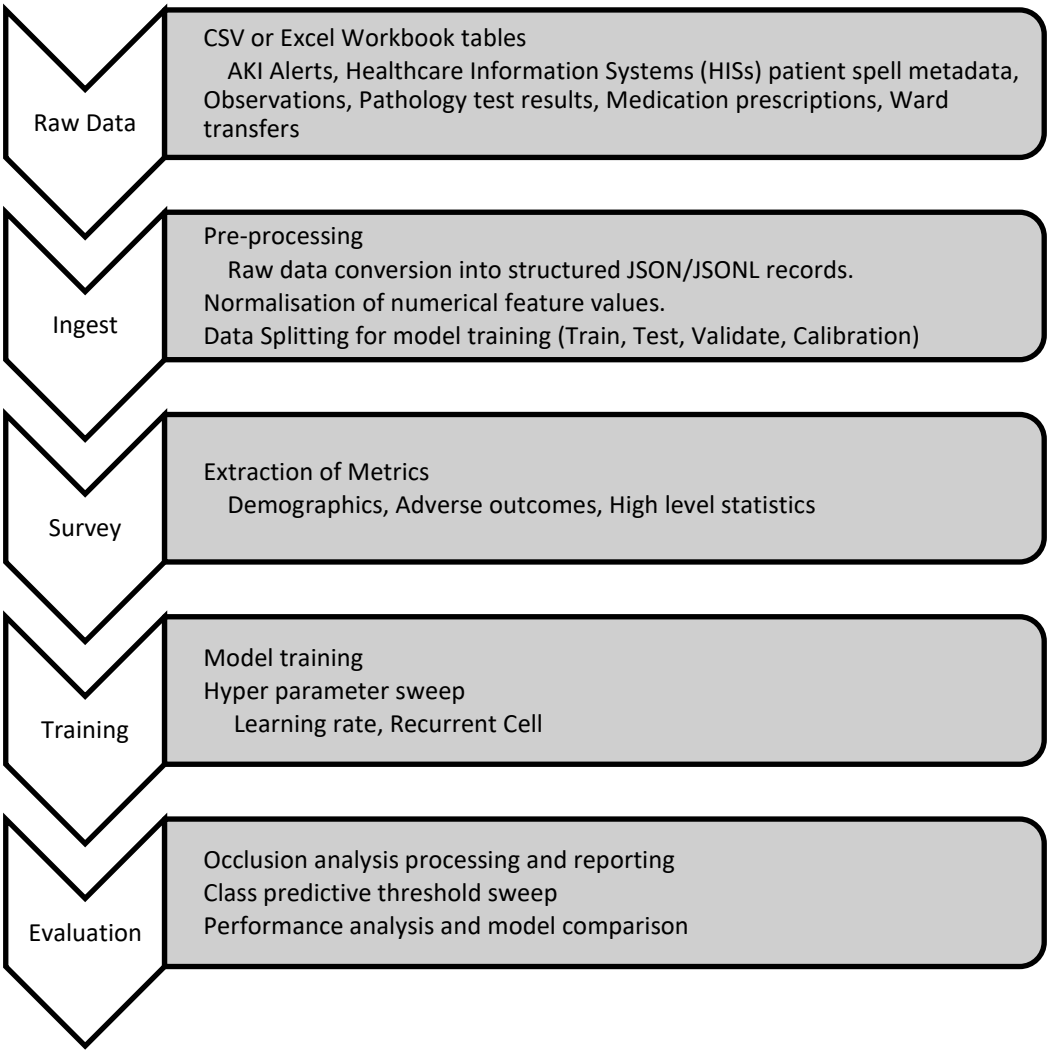


Figure 1: Diagram showing the key data processing stages.

1.3.3 Labelling, Training and Validation

The dataset was semi-automatically labelled using the determined labelling criteria (a combination of available fields and values). These labels are contained within the data structure and used within the model training pipeline. Validation is performed using a specific validation subset of the dataset. This split is determined pre-training and divides the data into train, test, validate, and calibration sets.

Detailed descriptions of data processing steps and loading into the DL model can be found in Appendices C.1 and C.2.

1.4 Project Plan and Timeline

An overview of the project plan is provided in the figure below. The project was planned to be delivered in twelve weeks and was split into six, two-week sprints. An additional seventh sprint was added mid-project to account for inclusion of new data into the model training experiments, prior to the final reporting and handover activities.

1.5 Expected Benefits

The expected benefits of this project are twofold. Firstly, this project aims to increase the understanding of the feasibility of a tool which can help predict such adverse outcomes in a timely manner. Secondly this

project aims to understand further the applicability of expanding on the previous work by Google DeepMind, which may be relevant not only to the software tooling developed under this commission but also may be applicable to other future work in this or other clinical areas.

The expected benefits of the developed PoC tool are primarily to provide a tool which can support clinicians in their decision making, which looks to improve the effectiveness of the UHL outreach team. The use of AI to make predictions regarding these class labels on a temporal scale will help clinical staff to make timely decisions regarding the best course of action for a patient. Roke's understanding is that there are tools used by clinical staff to identify patients who have developed kidney disease, however, delays in detection for a broad group of patients who experience rapid decline in kidney function make it very difficult to opt in/out the necessary course of clinical treatment for the patient, as their situation has already deteriorated. Specialist beds required to treat these patients are also often filled by patients who do not eventually require specialised treatment, and more accurate prediction of which patients require these beds will ease the management of this resource.

The proposed approach includes providing explainability of the models in terms of feature importance. This may accelerate deployment of similar models at hospitals with dissimilar data available, provide targets for new treatment options, and provide assurance to both machine learning specialists and physicians.

2 TECHNICAL SOLUTION

2.1 DEEPMIND EHR PREDICTIONS CODEBASE

Google's DeepMind and University College London (UCL) published work in 2019 on the prediction of AKI using an RNN-based model. The codebase used for DeepMind's approach was made open source in 2021 (EHR Modeling Framework, 2021), with the exception of their data loading pipeline. It is believed that there is sufficient similarity between the DeepMind investigation and the type of problem approached in this commission that the open-source architecture and approach used by DeepMind is applicable and adaptable to the prediction of adverse outcomes for the broader AKI classification problem.

The backbone architecture of the published model is configurable, including options for building block elements relying on Simple Recurrent Units (SRU), Update Gate Recurrent Units (referred to within the DeepMind code and subsequently as UGRNN, or Update Gate Recurrent Neural Network), and Long Short-Term Memory units (LSTM), each a different type of recurrent block that allows handling of sequence predictions in DL models. The aim within this investigation has been to re-use as much of the DeepMind codebase as possible, in order to streamline the development of the desired proof of concept tool.

A bespoke data loader was required to integrate the new dataset, and the model endpoints adjusted to fit the new objectives of the classification problem. The codebase was modified to use a Command Line Interface (CLI) to allow a user to access the key functions within the experimental processing and training pipeline. This includes multiple training experiments which can be executed with flags for setting the training data directory, output directory, and number of training steps. This tooling supported running of a number of combinations of training configurations with and without inclusion of patient metadata. Guidance on these tooling options can be found in Appendix C.5.1 including the expected data and mapping files required.

2.2 DATA PRE-PROCESSING

A series of pre-processing steps were required to convert the provided data tables into time-series record-based structured data. This involved extracting the relevant features from the raw Comma Separated Values (CSV) or Microsoft Excel Workbook tables provided into a structured dataset (in this case JSONL, with a JSON line for each data record. See Appendix C.1.7.2 for structure). The data for this project originated from the UHL clinical systems and covered a wide range of clinical data regarding specific hospital admission spells of interest.

The data comprised of data tables covering the following categories:

- AKI Alerts
- Healthcare Information Systems (HISS) patient spell metadata
- Observations
- Pathology test results
- Medication prescriptions
- Ward transfers

Note: The original data request by UHL specified potential for ~230 pathology test variables. Due to resource and time constraints only 9 were able to be included within the dataset used for this project.

Prior to extracting events and feature values, pre-processing steps (such as removal of duplicates) were applied to the raw data to assist in standardisation of field types and aggregation of fields for more streamlined feature extraction (see Appendix C.1.2).

2.2.1 FEATURE EXTRACTION

This pre-processing stage involved converting the dataset from a tabular form into a form structured more closely to that expected by the DeepMind model, i.e. separation of the data by patient, extraction of medical history prior to the given admission, and ordering events during the given admission by date time. Separation by patient was achieved via the patient spell identification field (a spell ID rather than hospital patient ID due to anonymisation of the data) and used as the master key in matching patient data from different sources—it is understood from the UHL partners that each spell ID should be treated as a unique patient entry. A set of relevant features were selected from the pre-processed dataset and were matched with the relevant date and time information in order to compile a structured set of metadata and sequenced feature changes for each relevant patient spell.

Extracted features include those such as:

- Creatinine (CREA) pathology test results
- Heart rate observations
- Other assorted events during the patient stay (such as ward transfers or medication prescriptions).

See Appendix C.1.3 for a full listing of mapped features and detail on how they are encoded.

Numerical features have been normalised into a standard range. Whilst the DeepMind approach capped these values at the 1st and 99th centiles, in this investigation capping was not applied as it was seen to lose relevant granularity at the extremes (which may correspond to spells with adverse outcomes). See Section 3.1 and Appendix D.2.4 for more detail on the numerical feature distributions and distribution plots.

There are instances where non-numerical values are present within numerical feature fields. This occurs to the largest extent within the pathology data fields. Non-numerical values which have been labelled by the clinical team as “not relevant” are ignored and any remaining non-numerical fields of interest are encoded as invalid values in the batch loader. In the dataset used within this project 7,937,410 numerical values were evaluated during ingest, of which 108,925 were ignored (“not relevant” fields), and 29,647 retained as empty feature values.

Sequence feature changes (events during the patient stay), have their date and time fields standardised during the data ingest. These absolute date times are then converted into relative date time values corresponding to the day of occurrence and a time bin identifier. Events without a populated date time field are ignored. Events are then sorted and grouped into six-hour time bins. See Appendix C.1.4 for further detail on this conversion.

2.2.2 EXCLUSION CRITERIA

A selection of patient spells have been excluded from the delivered dataset if they match the following criteria, as they show pre-existing adverse outcomes:

- First AKI alert occurred when located on a renal or ITU ward.
- Existing dialysis already flagged within the patient spell metadata.

These patients make up approximately 12.8% of patient spells within the original unique spells available. Appendix C.1.5 contains further detail on these criteria and the fields used.

2.2.3 LABELLING

To support classification and time series prediction of the three adverse outcomes, labels must be generated for each data point. These cover each of the adverse outcomes for each predictive time interval

and additional flags to indicate any adverse outcome. The codebase also contains placeholders for the additional labels required for the auxiliary lab result prediction task. See Appendix C.1.6 for more detail.

The labelled outcomes are:

- **Mortality** - The date of discharge and the method of discharge have been used to designate the mortality outcome.
- **Dialysis** - Admission to renal ward does not immediately assume dialysis – therefore dialysis has been detected through admission to a renal ward AND the spell containing metadata ICD-10 (diagnosis) codes relating to dialysis.
- **ITU Admission** - For ITU Admission the ward transfer data allows specific tracking – the following UHL wards have been designated as ITU: FICU, FPIC, and RITU.

2.3 DATA LOADING

A number of steps were needed to convert the EHR predictions codebase to accept the required feature mappings and data structures. A bespoke data loader was built to match up the pre-processed data structure to the format required by the existing model training interface.

2.3.1 MODIFIED DATA LOADING INFRASTRUCTURE

At this point in the data pipeline the pre-processed data was divided into spells with metadata and event information present, with each event spanning a six-hour time window. The events correspond to feature changes and contain the time point when the event occurred, along with the feature ID and category ID for the feature field. The category IDs group similar types of features (ward transfer, location within which observations were taken, a prescribed medication etc.). The dimensionality of unique features was very high, approximately 13,000, owing to some groups of features (such as drug prescriptions) having a few thousand unique possibilities each. The large number of features was due to encoding each categorical feature value as an individual feature with presence value (retaining feature granularity).

Each entry within a particular event was represented within the data loader as a combination of six tensors covering the indexes and values of features which occurred during the event. These relate to categories, numerical features, and feature presence tensors (see Appendix C.2.1 for further detail on these tensors).

These tensors were then concatenated across a new dimension for each time window, and formed the ‘sequence’ information provided to the DL models. When including patient metadata in the data loading, a pair of tensors were generated covering each type of metadata field (such as sex, ethnic origin, method of admission or year of birth). ICD-10 (diagnosis) codes corresponding to the patient stay were also available for inclusion in the metadata. However these were eventually excluded (see Section 3.2.4).

2.3.2 CURRICULUM LEARNING

The order and frequency with which a DL model encounters data within its training set affects its performance. We attempted to use curriculum learning (detailed in Appendix C.2.2) to address some class imbalances that may be present in our dataset (such as imbalances in sex, ethnic origin, age, and outcomes of patients, or of largely differing patient stay lengths), but ultimately found that performance was not improved by curriculum learning’s introduction. Owing to time constraints, further exploration of other implementations of curriculum learning were not performed, as the model appeared to perform to a suitable level without this extension and other technical work was of higher priority. Performance metrics were not taken separately on different classes of patient once a finalised model had been trained (e.g. measuring performance on female patients versus male patients, or the various other possible imbalances), and so it is unknown to what extent biases introduced via these imbalances may be present.

2.4 Model Conversion

The training outcomes for this commission are different to those of the original DeepMind work, therefore new model training tasks were defined within the code framework to correspond to the desired objectives. These related to predicting against the three adverse outcomes (mortality, ITU admission, and dialysis), and against an updated set of predictive time windows (defined as 6 hourly intervals between 6 and 48 hours). The work within this commission used the implementation within the DeepMind codebase which addresses “adverse outcome” level prediction as a basis for new model endpoints.

Additional details of changes made to the DeepMind code can be found in Appendix C.4.

2.5 Model Training

The model training experimental setup is based upon the template provided within the DeepMind codebase. The template was modified to incorporate:

- New configuration parameters required.
- Integration with a new batch loader (designed for the new JSONL data format).
- Additional metric tracking while training.
- Export of performance metrics from the validation dataset while training.
- Ability to run a model training or prediction task against multiple new objectives.

An expanded discussion of each of these points is included in Appendix C.5 .

2.5.1 Hyper-parameter Scanning

Modern DL models can be defined by billions of individual parameters, grouped into layers that may be connected in near-arbitrarily convoluted manners, with each layer structure being potentially independently defined from other layers within a model. This makes the number of combination choices possible for a single DL problem difficult to even estimate. Further options for customisation are possible outside the model architecture, such as learning rate, batch size, optimiser choice, etc. Investigating the optimisation of each of these choices can be a very onerous or infeasible task with each choice requiring anywhere from minutes to months to fully train with an exhaustive search over the complete parameter space is not realistic for any DL model (Li, 2022).

Hyper-parameter scans can search a limited space of options to partially optimise such choices, though a true global optimisation over all parameters is not typically possible. In these investigations a scan was completed over learning rate and recurrent cell choice in the model. For example the DeepMind framework allows configuration of Simple Recurrent Unit (SRU), Update-Gate Recurrent Unit (UGRNN), or Long Short-Term Memory (LSTM) recurrent cell choices. An expanded description of hyper-parameter tuning is given in C.5.2.

2.6 Performance Criteria

Model evaluation for a time series prediction problem such as this involves aggregating the predictive performance of the model at each time step for each sequence of patient events, across a series of patient spells within the dataset.

2.6.1 Performance Metrics

The DeepMind framework includes the capability to generate many common performance metrics whilst training the model. These include typical metrics such as number of true and false positive/negative classifications, positive predictive value, negative predictive value, and area under curve for both receiver-operator and precision-recall curves (see Appendix C.6.1 for a full listing). The raw values for the receiver-operator and precision recall curves were added to the existing performance logging framework during

the project to aid in performance plotting. From these metrics, further performance measures such as sensitivity and specificity can be calculated.

Note: The default performance figures from the DeepMind codebase are in terms of all aggregated time steps processed for the given data split, rather than on a per spell or patient basis. This provides a balanced approach to differing length spells. This has not been adjusted within this commission as the aggregated metrics were determined to be suitable for evaluating the performance, in agreement with the wider commission team.

2.6.2 Model Selection Metric

To evaluate the general performance of a model and determine how it compares to other trained models in a straightforward way, a selection metric calculation has been determined through discussions with the UHL clinical team. The metrics which were preferred by the clinical team were:

- Sensitivity
- Specificity
- Positive Predictive Value (PPV)
- Negative Predictive Value (NPV)

Additionally "Accuracy" was included for reference against the other metrics.

Clinically, the desire was communicated to be that high sensitivity and high positive predictive value would be the ideal outcome, where there are few false negative (missed) cases and few false positive (false alarms). The model is configured to output predictions for a range of 6 hourly time intervals between 6 and 48 hours. When selecting the 'ideal model' when analysing a range of trained models (or a range of models from different stages in a training run) it was determined that the 24-hour prediction interval was the most significant to the clinical team. The performance metrics are also calculated on a per-outcome basis, so in determining the 'ideal' model, the mortality outcome was chosen as the best indicator for initial model comparison and selection.

A single metric was desired which would indicate the general performance of a particular model. Therefore the following scoring metric was chosen:

> *Mean of sensitivity and positive predictive value, for the mortality outcome, at the 24 hour predictive interval.*

Following some experimentation this was revised to be the geometric mean rather than arithmetic mean of the Sensitivity and PPV as this reduces the bias of extremely high or low values.

In this instance applied to the chosen performance metric above:

$$Performance\ Scoring\ Metric = \sqrt{PPV \times Sensitivity} = \sqrt{\frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}$$

(Where TP = Number of true positives, FP = Number of false positives, FN = Number of false negatives)

This approach is similar to the "G-Mean" metric which is commonly used in machine learning performance analysis (see Appendix C.6.3 for further detail).

This metric was used during the performance analysis stage to determine the highest performing model whilst:

- Analysing a particular training run (at which step interval);
- Choosing appropriate hyper-parameter values;
- Selecting an appropriate predictive threshold;
- Providing a point of performance comparison when comparing models trained with differing feature sets or exclusion criteria.

2.6.3 Visualising Model Performance

Visual plots of model performance parameters can be very useful for evaluating the behaviour of the model, particularly for multi-class and time series prediction problems. Within this commission three main plot types were generated.

Confusion type plots were generated which displayed the true/false positives and negatives for a specific outcome for each predictive time interval. See 3.2.5 for these types of output plots.

The PR (Precision-Recall) and ROC (receiver-operator characteristic) plots were also generated for each model of interest. These provide a diagnostic tool for binary classification models. The area under these curves (available within the exported performance metrics) can be used as an additional metric to compare model performance. In this instance the ROC curves can be optimistic on the performance due to the severe class imbalance within the dataset (very few samples of the minority class – such as spells with dialysis).

2.6.4 Visualising Inference Predictions

The proof-of-concept tool allows for running of inference predictions against a trained model. The inference mode within the Command Line Interface (CLI) tool will output a structured file of predictions against each of the outcomes and time intervals. Limited time was available within the project timeline to generate plots from this data, however Figure 2 shows an example of what one of these plots may look like for a specific patient spell.

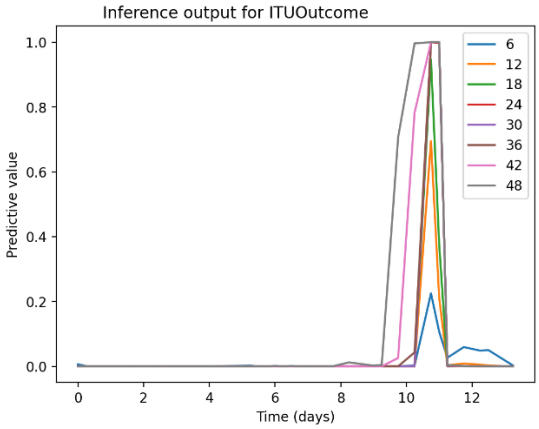


Figure 2: Example inference plot for the ITU outcome showing predictive value over the duration of the patient spell, against each of the predictive time intervals (see legend). Predictive value refers to the probability of belonging to the class ITUOutcome.

2.7 Explainability Analysis

Though DL models can extract information from input data which allows them to make accurate predictions, the reasons why a model chooses to use increasingly more abstracted representations of input data in particular ways throughout its layers can be difficult to extract. The mathematical relations a model builds up will always be an approximation to true relationships, and the approximation will be

impacted by the architecture of the model rather than a 'true' understanding as a human expert would hope for. Attempting this extraction of internal model relations is not necessarily going to yield useful information for further study.

Determining which input features a model managed to make use of, regardless of exactly how it handled them internally, can be more instructive. It can provide a 'minimal set' of features, that could allow others to follow a similar approach, and guide research towards any discovered important features whose import was not previously recognised. A method of evaluating this feature handling is occlusion analysis, whereby the performance (in this case difference in cross-entropy from a baseline) is compared for the instance of each feature being individually masked out in turn. This provides a sort of ranking of individual feature contributions to model predictions. Further details of our approach are given in Appendix C.7, and results in Appendix E.3.1.

2.8 THRESHOLD ANALYSIS

One of the chosen model performance evaluation techniques applied in this commission is a class prediction threshold sweep. This has been configured in the codebase to perform a prediction pass over the validation dataset at varying class prediction thresholds for a specific model. Performance metrics are logged at each threshold value, and can be combined to determine a suitable prediction threshold that provides the desired performance behaviour. Differing class thresholds can be chosen optimally for each class. See Appendix C.8 for further detail on the threshold step values chosen for this analysis.

3 DISCUSSION

The following sections detail the output from each of the experiments conducted during the development of the proof-of-concept tool and the analysis of the final model performance.

3.1 INGESTED DATASET

The dataset used for the project contains Electronic Health Record (EHR) data relating to patient spells with a recorded AKI alert, whilst admitted to UHL between March 2019 and March 2022. This dataset was exported by the UHL data informatics team.

During the project it was determined that a number of dialysis patient spells were incorrectly filtered out pre-delivery of data to the Roke team. The missing data was extracted and delivered to the Roke team prior to the additional seventh sprint in the project. This provided approximately 300 additional spells containing the dialysis outcome. These spells were incorporated following completion of the hyper-parameter tuning activity in order to provide the missing data when performing the remaining key training activities. The dataset statistics shown relate to the revised dataset which includes these missing dialysis spells.

The data ingest identified 24342 unique spell keys and, following filtering, the aggregated dataset includes data from 21225 unique spells. Table 1 shows the distribution of adverse outcomes within the dataset (see “any” row for total adverse outcome spells, as some spells contain multiple outcomes). Any remaining spells not counted within the totals below are flagged by the existing AKI alert system but with no further adverse outcome.

Outcome	Count	%
Mortality	3393	15.99
ITU	854	4.02
Dialysis	315	1.48
Any (any of the above)	4394	20.70

Table 1: Overview of adverse outcome distribution within the ingested dataset.

Specifically for numerical features (such as pathology tests or observations), plots of the distributions were generated to determine a suitable normalisation strategy (see Section 2.2.1). An example of which is shown in Figure 3 below for the creatinine pathology test. It was determined that the most appropriate strategy for this commission dataset was to keep the numerical feature values uncapped when normalising, in order to maintain the granularity at the extremes (creatinine pathology test being a key example).

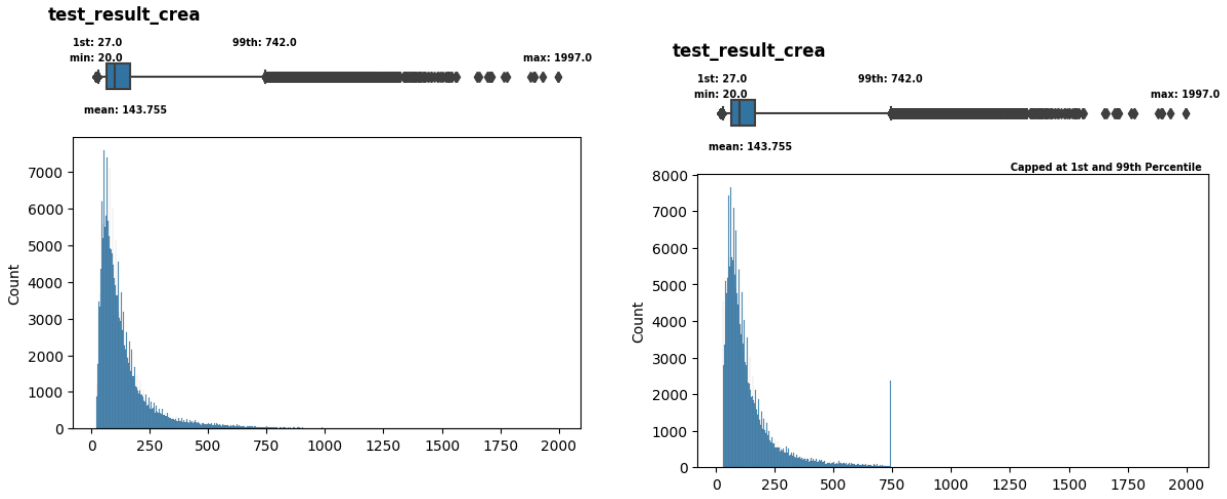


Figure 3: Numerical feature value distribution for the creatinine (CREA) pathology test. Left is uncapped, and right capped at the 1st and 99th centiles. Showing a loss of granularity at the extremely high values if capping is applied. The box plots show the statistics for the raw data (uncapped).

Further detail on the dataset statistics and numerical feature distributions can be found in APPENDIX D.

3.2 TRAINING OUTCOMES

The performance summary below details the end performance of the final trained model, following experiments on hyper-parameter tuning, training duration, and addressing the masking of potential "giveaway" fields from the model during training. Further detail on the training experiments can be found in APPENDIX E.

3.2.1 HYPER-PARAMETER TUNING

Note: the hyper-parameter tuning experiments were performed prior to the addition of additional data relating to the dialysis outcome.

Following the approach in Section 2.5.1, a series of training experiments were completed with each of the desired values for learning rate, and each of the three recurrent cell choices available in the DeepMind codebase. Each training run was run for 60,000 training steps. This was shorter than later training experiments, however due to the time constraints on the project this step length was chosen, and agreed with the relevant stakeholders, to allow a suitable level of model cross-entropy convergence to determine rough performance differences between the variable choices.

3.2.1.1 Learning Rate

A learning rate scan covered learning rate values of 0.0001, 0.0003, 0.001, 0.003, and 0.01. Results are shown in Figure 4.

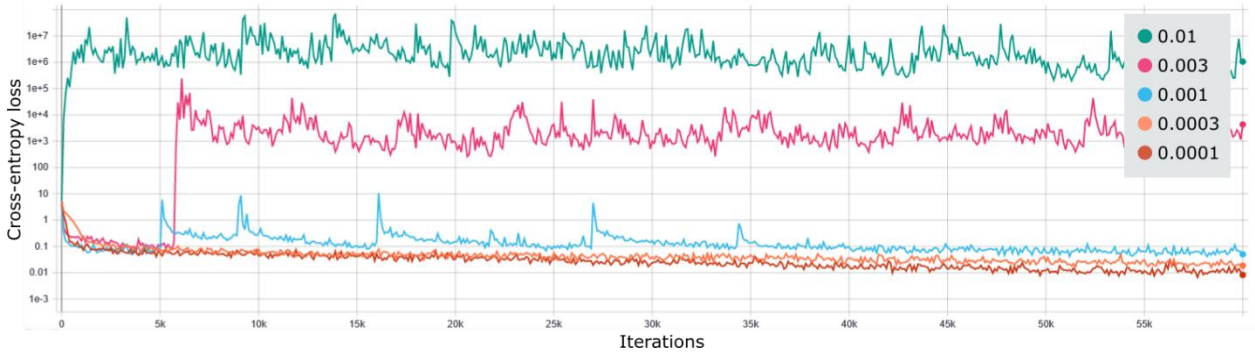


Figure 4: Cross entropy loss (log scale) across 60k training iterations for the same model with 5 different learning rates. All models with learning rates at or below 0.001 descended well, and performed near-equally on downstream performance metrics.

It was found that learning rates above 0.001 have significantly less stable cross-entropy convergence when training. The optimal learning rate for performance against the scoring metrics was determined to be 0.001 (the original DeepMind default value).

3.2.1.2 Recurrent Cell Choice

A training run was completed with each recurrent cell choice available. Resulting loss plots are shown in Figure 5.

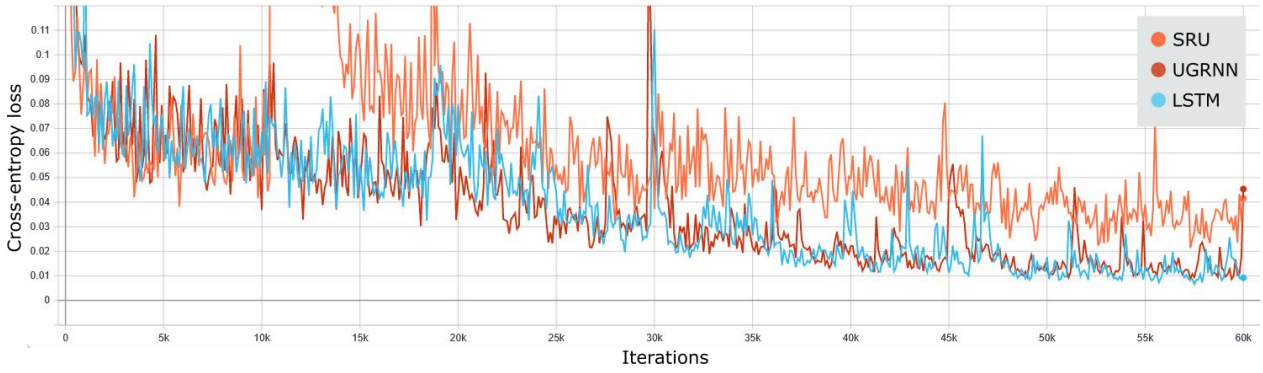


Figure 5: Cross entropy loss across 60k training iterations for the same model with 3 difference recurrent cell choices.

It was found that the cross-entropy loss was comparable between the LSTM and UGRNN cells, with the SRU cell finding fewer false positive values but still scoring lower due to increased false negatives. The LSTM cell performed the best against the chosen scoring metric (see Section 2.6.2).

3.2.2 Determining Training Duration (step count)

The default step count for the DeepMind example experiment was 400,000 steps. A training run of this duration was completed to determine a suitable representative training step count for the new training outcomes and dataset. From this extended training run performance did not greatly increase past 250,000 steps, with some potential overfitting occurring (see appendix E.1.3 for performance plots). Subsequently, for the remaining training experiments, a step duration of 220,000 steps was chosen, and agreed with the relevant stakeholders, as the performance did not measurably increase past this point, and further model check pointing added between approximately 160,000-220,000 steps to capture the performance within the desired training range.

3.2.3 Removing "Giveaway" Fields

Running the occlusion analysis on the baseline trained model found that there were a number of fields in the top 100 features which were of particular interest. These included fields related to the destination of discharge and a number of medication fields with "discharge" appearing in the text. The destination of discharge fields were included in the model for labelling purposes, however were unintentionally retained in the dataset sequence fields. These fields (not insignificant to the model training as seen from the occlusion analysis) may provide the model with information (such as discharge destination) which would not normally be available at inference time. Additionally, when training, the medication fields mentioning discharge may in combination bias the model at certain points in the time sequence to performing better at knowing if a patient is at higher risk of mortality.

Examples of such giveaway fields include:

- **Destination of discharge**: values such as *"home", "died", "nan"* (not yet discharged), *"nhs run care home"*
- **Medication fields**: values such as *"one last iv dose before discharge", "requires haemodialysis prior to discharge", "stat dose before discharge as not having iv"*

It was decided that these fields should be retained for labelling purposes, but masked out from the model at training time, thus removing any potential bias. The baseline performance figures relate to a training run with these "giveaway" fields masked out, to more robustly remove biases from the model training and also ascertain any performance differential when these fields are removed.

3.2.4 Assessing the Inclusion of Context Fields

Through discussion with the clinical team the context of a patient (any metadata, or prior co-morbidities) is seen to be hugely useful in assisting clinicians in their decision making around qualitatively assessing the risk of potential adverse outcomes. It was desired in particular that the co-morbidity data be included in the model training to allow the model to access the potentially relevant prior conditions/diagnosis for a specific patient. The co-morbidity fields (ICD-10 codes provided within the HISS data) were included in a number of model training experiments as context features. Unfortunately through discussion with the clinical team, whilst the ICD-10 codes are useful for labelling dialysis outcomes (see Section 2.2.3), the values within the HISS data are typically populated on discharge/post-spell, and may not be static during a stay, therefore should not be used during model training, as they may include "giveaways" on eventual diagnoses or adverse outcomes before they occur in the time series sequence.

It was found through experimentation (and shown in occlusion analysis) on training runs with the context fields included – that the model did utilise some of these fields to a non-negligible degree (within the top 0.2% of occlusion cross-entropy difference), and that the performance did improve in some areas (expected due to the bias in information provided).

Future work may benefit from accessing the primary care co-morbidities separately to the HISS data, so as to be able to include patient diagnosis prior to a hospital spell. Additionally, if timestamps for coding are available, this would allow the model to be provided with new diagnoses as sequence events.

3.2.5 Baseline Performance

Baseline performance was measured from a 220,000 step training run using the chosen hyper parameter values, without context data included. Table 2 details how the baseline model performs against the chosen key performance and scoring metrics.

	Mortality (24h)	ITU (24h)	Dialysis (24h)
PPV	0.397	0.290	0.636
NPV	0.994	0.999	0.999
Sensitivity	0.159	0.241	0.189
Specificity	0.998	0.999	0.999
Accuracy	0.992	0.997	0.999
Scoring Metric	0.251	0.264	0.347

Table 2: Summary of baseline performance at the chosen evaluation time interval and for a predictive threshold of 0.5, for each of the three adverse outcomes.

From Table 2 it can be seen that broadly (largely due to the large class imbalance) the NPV and Specificity are very high. This is largely biased by the large negative class imbalance, however does still show very encouraging performance (general ability to distinguish negative cases).

When considering the key metrics of PPV and Sensitivity, the performance is encouraging. There is still improvement that would be ideal, however the performance at this time is considered to be reasonable, particularly whilst still in the 'Proof of Concept' stage of development.

When considering the other predictive time intervals trained against, below shows the aggregated counts of true/false positives and true/false negatives for the mortality outcome for the baseline trained model. Plots for the ITU and dialysis outcomes can be found in Appendix E.2.

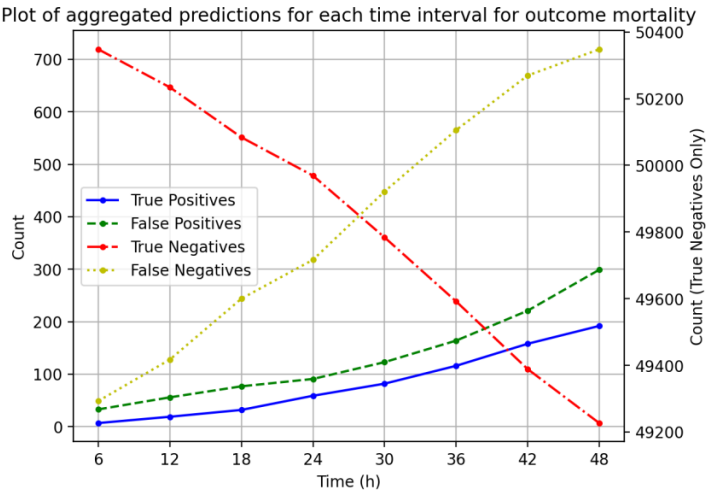


Figure 6: Plot of class predictions for each predictive time interval for the mortality outcome. *Note that these are aggregated values of each time step occurring in each spell within the validation dataset – rather than on a per-spell basis.*

It can be seen from the predictive plot (Figure 6) above that the baseline model exhibits behaviour showing an expected increase in false positives and false negatives with increased interval, as it becomes more difficult to accurately estimate a patient's risk of the adverse outcome. True Positives also see an increase with increased time interval. We typically would not expect the true positive count to increase with each interval as the model predicts further and further into the future, however this could be due to the less

granular date time information for the mortality outcome (exact times not available for labelling, only date of occurrence).

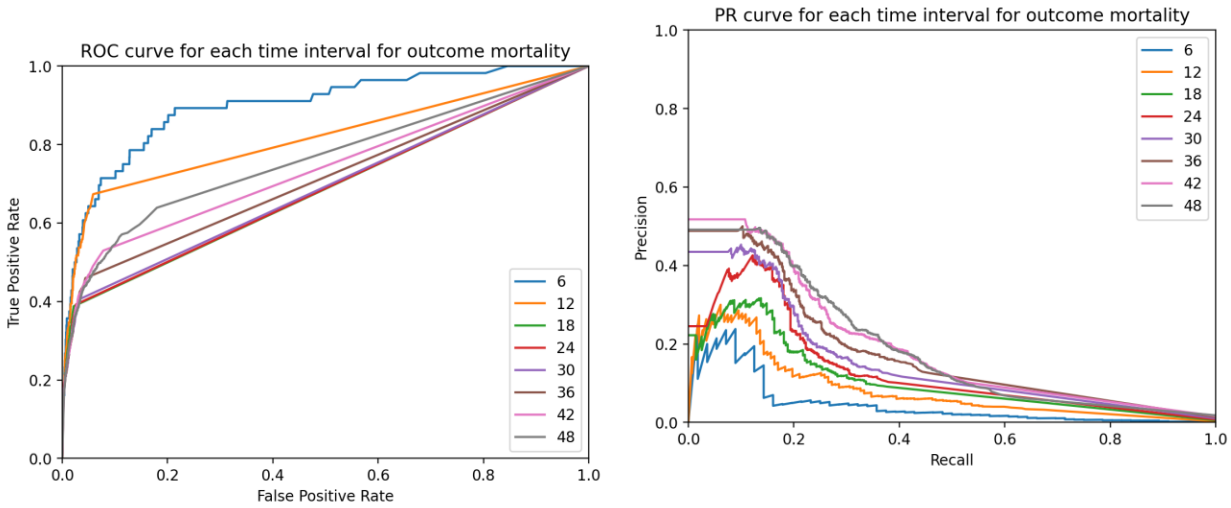


Figure 7: ROC and PR curves for the mortality outcome from the baseline model performance analysis.

The receiver operator characteristic (ROC) curve in Figure 7 above shows the model performance against true and false positives for a range of operating points (predictive class threshold). The plot shows expected decrease in performance predicting further in time interval (line trends close to the centre diagonal from bottom left to top right). As described in C.6.1, the area under this curve can be used as a performance metric, with a greater area indicating better performance.

A similar approach can be used to analyse the precision recall plot above, with the ideal operating point being as far from the bottom left corner as possible. In this instance a very imbalanced dataset leads to a PR curve with this less common 'n' shape.

A sweep of predictive thresholds provides further detail on the optimal operating point of the model.

3.2.5.1 Class Predictive Threshold Sweep

The class prediction thresholds evaluated were initially in steps of 0.01 between 0 and 1. This was broadened to include steps of 0.001 between 0 and 0.01 to gain a clearer picture of performance at the lower end of prediction threshold. This was due to increased performance against multiple adverse outcomes in this range.

When considering the primary outcome (mortality) at the 24h time interval (where the scoring metric has been applied), the highest scores were found at a threshold of 0.49. At this threshold the sensitivity of all three classes shows an increase (see Table 4). In particular the mortality outcome shows increased performance in PPV (fewer false positives).

	Mortality (24h)	ITU (24h)	Dialysis (24h)
PPV	**0.397 (inc)**	0.290	**0.583 (dec)**
NPV	0.994	0.999	0.999
Sensitivity	0.159	0.241	0.189
Specificity	0.998	0.999	0.999
Accuracy	0.992	**0.998 (inc)**	0.999
Score	**0.251 (inc)**	0.264	**0.332 (dec)**

Table 3: Summary of model performance at class prediction threshold of 0.49, the highest scoring operating point against the chosen scoring metric (mortality prediction at 24h interval). Indicates any increase or decrease from baseline performance at 0.5 predictive threshold.

A different threshold may be applied when predicting against each of the adverse outcomes. The highest scoring threshold for the ITU outcome was 0.91, and for dialysis 0.41. For ITU this related to an improvement in PPV, and for Dialysis an improvement in Sensitivity (see Table 4).

	Mortality (24h) Threshold = 0.49	ITU (24h) Threshold = 0.91	Dialysis (24h) Threshold = 0.41
PPV	**0.397 (inc)**	**0.345 (inc)**	**0.615 (dec)**
NPV	0.994	0.999	0.999
Sensitivity	0.159	**0.229 (dec)**	**0.216 (inc)**
Specificity	0.998	0.999	0.999
Accuracy	0.992	**0.998 (inc)**	**0.999**
Score	**0.251 (inc)**	**0.281 (inc)**	**0.364 (inc)**

Table 4: Summary of model performance at class prediction thresholds of 0.49 for Mortality, 0.91 for ITU, and 0.41 for Dialysis. Indicates any increase or decrease from baseline performance at 0.5 predictive threshold.

The details in Table 4 are indicative of the highest performance scores for the chosen trained model against each of the adverse outcomes against the chosen time interval using the specified performance scoring metric (see 2.6.2).

Note: For the dialysis outcome, whilst the validation dataset contains 12 dialysis cases (out of the approximately 300 cases in the whole dataset, this amounts to only 37 "true" dialysis outcome time steps. Therefore whilst the training dataset contains a much greater number of dialysis outcomes, it is difficult to estimate accurately the 'real world' dialysis performance via the evaluation dataset performance metrics.

3.3 PERFORMANCE SUMMARY EVALUATION

From the performance data available it can be seen that the model has been able to begin to learn how to classify each of the adverse outcomes.

Due to the class imbalance in the dataset there is a majority class of true negative time steps. This leads to very high values of NPV and Specificity. However this does show that the model is able to distinguish non-"adverse outcome" time steps with some confidence.

Looking at the key metrics of PPV and Sensitivity, **high** Sensitivity and **high** PPV would be the ideal outcome, where there are few false negative (missed) cases and few false positive (false alarms). In this instance we can see that at the ideal predictive threshold (according to the performance scoring metric) seen in Table 4, and the further plots shown in E.2, the following observations can be made:

- At the fixed prediction threshold of 0.49 (initial optimal operating point), the model achieves Sensitivity of 0.159 for the mortality outcome, 0.241 for the ITU outcome, and 0.189 for the dialysis outcome. In particular the ITU outcome shows very promising performance.
    - ***When accounting for the limited dialysis data, and the difficulties labelling both the dialysis and mortality outcomes, it is not surprising that the ITU outcome scores the highest out of all of the outcomes at the 24 hour predictive time interval.***
- The high Specificity shown suggests the model is very good at correctly identifying time steps with no adverse outcome.
    - ***This is encouraging as if applied clinically, would be less likely to utilise high amounts of clinician screening time on patients with a low chance of an adverse outcome.***
- The lower Sensitivity however supports the analysis that the model is still missing a proportion of the truly positive time steps with adverse outcomes, and that there are false positive classifications occurring. Both of these are not ideal and future development would look to improve the performance for both of these situations, however for this PoC tool, these are still very encouraging values.
- Unfortunately although the updated dataset (with additional dialysis cases), has approximately 300 cases, with the validation dataset split of 5%, only 12 dialysis cases are included in the validation dataset, making it difficult to make conclusions from the performance metric data (too little data for any statistical relevance).

In summary, for each outcome at the 24 hour predictive time interval, at the optimal scoring operating point, the model performs such that:

- **For the mortality outcome at threshold of 0.49, the model correctly identifies 15.9% of "adverse outcome - mortality" time steps, with approximately 2-3 false positives for every correctly classified true time step.**
- **For the ITU outcome at threshold of 0.91 the model correctly identifies 22.9% of "adverse outcome - ITU" time steps, with approximately 3 false positives for every correctly classified true time step.**
- **For the dialysis outcome at threshold of 0.41, the model correctly identifies 21.6% of "adverse outcome - dialysis" time steps, with approximately 0.6 false positives for every correctly classified true time step.**

3.4 EXPLAIN-ABILITY

Running the occlusion analysis pipeline yielded results which may be fascinating both technically and clinically. Prior to the removal of "giveaway" sequence fields the analysis identified the following fields as significant to the model predictions. A higher score indicates that the model performance (in terms of cross entropy) decreases to a greater degree with that feature removed.

Table 5: Table of occlusion analysis fields, showing the top 25 fields from analysis of the baseline trained model, ranked by relative cross entropy from the un-occluded model with all fields included. A

higher value indicates higher significance to the model. *Bold fields indicate giveaway fields masked out in the final model training experiment.*

Feature name	Cross Entropy Differential Value (relative to un-occluded baseline)
sequence_bp_systolic	121932.72
sequence_bp_diastolic	80422.17
sequence_o2_sats	60446.47
sequence_test_result_sodium	55473.04
sequence_test_result_crea	52341.90
sequence_test_result_urea	45998.53
sequence_heart_rate	44177.54
sequence_rr	43012.67
sequence_test_result_potassium	37268.13
sequence_method_of_discharge_on medical advice	33437.52
sequence_test_result_calcium	31804.47
sequence_full_drug_name_antibacterial wash	30919.61
sequence_destination_on_discharge_home	27296.14
sequence_method_of_admission_a&e lri	23923.22
sequence_full_drug_name_sodium chloride 0.9% infusion	22063.42
sequence_full_drug_name_dalteparin sodium 5000units/0.2ml injection	18519.83
sequence_full_drug_name_paracetamol	16692.91
sequence_destination_on_discharge_nan	15604.84
sequence_method_of_discharge_died	15521.39
sequence_full_drug_name_amoxicillin 1g + potassium clavulanate 200mg injection	12358.17
sequence_full_drug_name_mupirocin 2% nasal ointment	10884.88
sequence_transfer_to_specialty_integrated med (acute care)	8372.24
sequence_obs_location_fcdu	8127.46
sequence_full_drug_name_morphine sulfate 10mg/5ml oral liquid	8117.64
sequence_full_drug_name_meropenem 1g injection	7979.06
sequence_transfer_to_ward_fcdu	7438.63

This Table utilises the cross-entropy value, so is not tailored specifically towards a specific outcome, however it does indicate the significance of particular fields as weighted within the model predictions. **This shows that many of the observations and pathology results are the most significant in the current trained model, and that there are a number of medication entries which have been identified as significant in prediction of the adverse outcomes. This may be of particular relevance and interest to clinicians not only in understanding what data may influence the model the most – but also identifying potential relevant data points within the dataset, even without the model in use.**

4 CONCLUSIONS AND RECOMMENDATIONS

The following sections cover the key lessons learnt during the project and recommendations for future work. **Specifically the most key areas related to the provision of suitable data, and the potential to include further data (which was not able to be included in the dataset used for this project) in future investigations.**

4.1 LESSONS LEARNT

Within this commission there are a number of lessons learnt from the duration of the proof-of-concept development. These centre largely around programme schedule and availability of data – as delays in the data delivery impacted the ability to deliver some of the auxiliary objectives/tasks within the original project/sprint plan. This may be remedied in future through more in-depth discussions regarding the data and/or provision of data either prior to the commission, as a dedicated sprint activity (of data "discovery"/evaluation by the delivery team). It is recognised within this commission that the data extraction activity by the data informatics team was greater than initially anticipated, which additionally limited the number of data fields able to be supplied (such as pathology tests).

In addition to this – whilst some aspects are covered in the future work section, the use of the DeepMind codebase provided mixed value. It did dramatically streamline some aspects of the development (such as model architecture implementation, and performance analysis); however, there were a large number of implementation challenges in integration, some of which were known by the Roke team (and understood), and others unknown at the start of the commission (such as the complexity in integration into the batch loader and implementing curriculum learning within the version of the machine learning framework used).

For future, the method of data delivery is now better understood, and a more direct and streamlined approach will be possible with a direct API plug-in to blood results provided online. This would dramatically improve the quality and quantity of data to improve the predictive model. We can also expand into other indicators with full rich data, which should only improve and enhance the predictive algorithms and ML tools.

With online data plug-ins in mind, UHL have indicated that permission can be secured for improved integration of data, which under an agreement can go on to develop a predictive indicator tool to support clinicians in the future. This will have far reaching and positive impacts for patients' outcomes, support for clinicians and a significant reduction in overhead clinical costs and timelines.

4.2 FUTURE WORK

As a result of this PoC project, the insights it has provided Roke's delivery team, and discussions with the wider commission team, a number of areas have been identified which would benefit from further development or consideration for any future extensions to the work. These focus on the improved outcomes of patients who are likely to benefit from earlier intervention for renal support, improved support for clinicians and the supply of quantifiable data to support clinical decision making patient pathways. Also, future directions highlighted here should significantly improve/reduce cost of care and patient planning options, particularly capacity planning.

Further detail on each of these points can be found in Appendix F.1.

The key areas relate to the provision of improved data and re-working the software tooling for future investigations and development. Of these many may provide only marginal performance gains, with the largest improvement in performance likely to be through the dataset construction elements listed below. Many of the software improvements suggested may provide a small amount of performance improvement (in the case of further investigations or adjustments to the model architecture or performance evaluation

tools), however the main points largely focus on improving the robustness of the tooling, providing large engineering improvements to future development activities.

As such the key elements to consider in future work are:

- **Dataset engineering/collection.** There are a number of areas in which the dataset could be improved for future work.
    - **Find improved methods for dialysis labelling** (avoid conflation of previous and current ICD codes). There is a slight flaw in the approach used in this commission to label dialysis outcomes, in that a patient may have received dialysis during a previous visit but not the current visit, and are sent to the renal ward on the current visit. In this case, a dialysis event would be spuriously labelled as occurring. Incorrect labelling will, of course, negatively impact performance in a supervised learning model, both in real-world predictive ability and validity of performance metrics after training—a more robust dialysis event labelling method would then likely improve both of these.
    - **Find improved methods for mortality (better granularity time stamping).** Mortality events are labelled as occurring at the end of a day in the current dataset, regardless of when they actually occur. This will lead to an undesirable punishment of the model if it predicts death to occur at any time window not at the end of a day. While the model may have some capability of understanding time of day with the current encoding of time, and hence learn that mortality events should only be predicted in the last time window of a day, this is clearly contrary to the optimal use case in real-world scenarios—accurate forecasting of the actual time window where death occurs. More accurate time stamping of when mortality events occur would be more suitable to the architecture of the model for training, and more useful for real-world usage (it is recognised that this may be a long-term vision, and may require significant clinical process changes).
    - **Separation of primary care/hospital co-morbidity data** (and potential to extract timestamps for new coding during stay). It is understood after discussion with the UHL team that it should be possible to extract ICD codes from primary care and hospital co-morbidity data, leading to the desired separation between ICD codes relevant to a patient's prior medical history and codes generated during their current stay. Improved timestamping of these codes should also be possible via this route, which could allow ICD codes to be used within the sequential features as well as context features.
    - **Expansion of dataset.** While the dataset for this project contained approximately 20,000 records, which is typically a viable quantity for DL tasks, there was a severe imbalance between positive and negative cases for each of the three outcomes—at a 5% validation split, only a few tens of cases existed for ITU admissions and dialysis outcomes for the purpose of measuring model performance, for instance. It is unknown how varied the presentation of clinical clues to the various outcomes may be that the DL model may discover, but it is possible that with these few validation cases that disparate presentations are occurring in the validation set that were not present in the training set. Increasing the number of positive adverse outcome cases in the training set should mitigate risk of this occurring, i.e. cause the training data to have more complete coverage of possible real-world scenarios.
    - **Further pathology tests** (i.e. infection markers). The data related to pathology tests was limited during this project due to difficulties in exporting a complete list, but it is understood that much more extensive data exists (230 pathology fields specified in original UHL proposal). In particular, blood test markers related to infection were expected by the UHL team to be of high importance in a future expanded dataset. The sheer quantity of pathology data also limited the number of tests that could be included during this project due to export time. If a complete list of desired pathology data were generated, and a data extraction exercise performed without pressure from a parallel model development effort, the looser time constraint on data export would mean a higher number of features could be used.

- **Machine learning implementation and experimentation –** a number of improvements could be made to the architecture/experiments completed.
    - **Re-build tooling for modern ML framework** - The DeepMind codebase was created using TensorFlow 1.15, and re-building the tooling from the ground up is seen as essential for future development (more appropriate frameworks (such as PyTorch) are now available).
    - **Improve efficiencies in batch building and encoder models**, due to CPU bound data batching, and metadata feature encoding strategy.
    - **Increase granularity of feature categories**. Each field, within the full list of approximately 13,000 features, were assigned a feature category. It may be worthwhile to break down some of these categories further to aid the model in associating logically similar features (such as different dosages of the same medication)
    - **Include lab predictions task**. The DeepMind group's original set of objectives included the prediction of some continuous-valued laboratory test results, which was found to improve performance by a few percent, even though the task was auxiliary to the main objective of predicting adverse outcomes.
    - **Try transformers as the recurrent cell**. Shortly before the DeepMind results were published, transformer-based neural networks were developed (Vaswani, 2017), and the transformer, as a building block of DL neural networks, has become popular as a replacement for other recurrent layers such as the UGRNN, LSTM and SRU layers used in this work (Wen, 2022).
    - **Expand hyper parameter tuning scope**. If training efficiencies are achieved via reduced CPU data handling, the breadth of hyper parameters realistic to be tuned is increased simply due to more model training iterations per unit time. Additionally, as a higher dimensionality of hyper parameters are explored, techniques more advanced than those used in this project would prove beneficial. Bayesian optimisation would be a common method for this (Frazier, 2018).
    - **Ability to ingest data from a more automated data source (database/data lake).** There was unexpected complexity in gathering data from various sources prior to handover to the Roke team. It would be valuable to standardise the data gathering approach before attempting to rollout any tool to test-bed users. If the data gathering could become periodic and automated, the model could be routinely updated as new data became available.
    - **Loss function bespoke to our performance metric/time window/outcome of choice.** The model training procedure aims to minimise the cross-entropy between predictions and true adverse outcome labels, with equal weighting across all time windows and adverse outcomes. A loss function that measures the performance scoring metric (see Section 2.6.2) could optimise the model training. Additionally the metric could be tailored to an alternate outcome or time interval (or reweighted).

Across the tooling and experimentation and the dataset collection and processing, there are multiple areas in which this could be extended in future. The primary elements relate to choosing suitable tooling and data encoding for future experimentation and development, and addressing the known issues with how the dataset is compiled and labelled.

4.3 CONCLUSIONS

It's fair to state this project was a success, in demonstrating within a 12 week period that a 24 hour predictive concept model could take the outcomes of this project and go on to be develop a clinical support tool for AKI patients but also for mortality prediction and the possibility of expanding this model into other areas of critical care. The ability to improve forecasting with a set of objective scoring predictors will provide clinicians with an invaluable tool to better support patient outcomes.

It has been shown in this research project that an AI model is able to provide meaningful prediction on the need for ITU admission, dialysis, or the outcome of mortality for AKI patients. The proof of concept has shown the ability to predict in discrete time windows up to 48 hours ahead of the event has been

developed. Positive predictive values and sensitivities achieved are encouraging, and explain ability methods have highlighted clinical features necessary for the model to work.

Overall, the proof of concept demonstrates potential for better forecasting of patient needs, and merits further development.

APPENDIX A REFERENCES AND GLOSSARY

A.1 REFERENCES

Czakon, J. (2022). *F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose?* Retrieved from Neptune AI: https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc

EHR Modeling Framework. (2021). Retrieved from github: https://github.com/google/ehr-predictions

Frazier, P. (2018). A Tutorial on Bayesian Optimization. *ArXiv*.

Lecun, Y. (2015). Deep learning. *Nature*, 436-444.

Li, Y. (2022). Hyper-Tune: Towards Efficient Hyper-parameter Tuning at Scale. *ArXiv*.

Nenad, T. (2019). A clinically applicable approach to continuous prediction of future acute kidney injury. *Nature*, 116-119.

NHS. (2022). *Acute Kidney Injury (AKI) Algorithm*. Retrieved from NHS England: https://www.england.nhs.uk/akiprogramme/aki-algorithm/

Tealab, A. (2018). Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, 334-340.

Vaswani, A. (2017). Attention Is All You Need. *ArXiv*.

Wang, X. (2021). A Survey on Curriculum Learning. *ArXiv*.

Wen, Q. (2022). Transformers in Time Series: A Survey. *ArXiv*.

A.2 GLOSSARY/ABBREVIATIONS

In order of occurrence:

ACE	Accelerated Capability Environment
UHL	University Hospitals of Leicester
AKI	Acute Kidney Injury
ITU	Intensive Therapy Unit
EWS	Early warning score
PoC	Proof of concept
AI	Artificial Intelligence
DL	Deep Learning
RNN	Recurrent Neural Network
CSV	Comma separated value
CLI	Command Line Interface
HISS	Healthcare Information Systems (HISs)
CREA	Creatinine pathology test code
PPV	Positive predictive value
NPV	Negative predictive value
PR	Precision-Recall
ROC	Receiver-Operator Characteristic
SRU	Simple Recurrent Unit
UGRNN	Update-Gate Recurrent Neural Network Unit
LSTM	Long Short-Term Memory
JSON	JavaScript Object Notation (open standard for data interchange, humanly readable)

Technical/Clinical terms:

Tensor	Typically relating to a generalisation of vectors (corresponding to a multi-dimensional array)
Co-morbidities	Describes the existence of more than one disease or condition within the body at a particular time – in this instance this covers patient current/past conditions.

APPENDIX B ASSUMPTIONS

The list of technical assumptions regarding the data or technical approach are listed in Table 6 below.

Table 6: Table of technical assumptions regarding the data or technical approach.

Index	Category	Description	Impact (if incorrectly assumed)
0	Data	Separate spells IDs are treated as separate admissions.	
1	Data	Spell Diagnosis Keys correlate to the ICD-10 coding definitions.	Incorrect usage of ICD-10 code detection.
2	Data	Medication data from nerve centre (NC) and the EPMA system can be treated the same following matching of the appropriate field headers.	May over complicate the model with non-correlating medication features from both systems. May have inconsistent date time or naming.
3	Data	Admissions ID within pathology data correlates to Spell ID on AKI alert list.	Incorrect matching of spell data.
4	Data	Visit ID matches across AKI alerts and Observation data, as a secondary patient spell key.	Incorrect matching of spell data.
5	Data	ICD-10 codes can change/be added to during a patient stay.	Impacts strategy of context feature inclusion.
6	Data - Labelling	The provided ICD-10 codes for dialysis labelling has sufficient coverage.	May miss labelling dialysis cases.
7	Data - Labelling	The presence of a dialysis ICD-10 code in a patient spell co-morbidities, combined with a renal ward transfer constitutes a positive dialysis label.	May miss labelling dialysis cases.
8	Data - Labelling	The destination of discharge indicates mortality through the "died" value.	Unsuitable labelling criteria.
9	Data - Labelling	The provided listing of ITU wards has sufficient coverage for this label.	May miss labelling ITU cases.
10	Data - Exclusion Criteria	Exclusion of patient spells based on existing dialysis listed in HISS data.	Incorrectly excluding patient spells from dataset.
11	Data - Exclusion Criteria	Exclusion of patient spells based on AKI alert on Renal or ITU ward.	Incorrectly excluding patient spells from dataset.

12	Data - Feature Extraction	Events without a date time field are ignored on ingest.	Unable to integrate into time series dataset - impacts breadth of features included in dataset.
13	Data - Feature Extraction	Spell IDs correlate across the delivered dataset.	Contamination of patient spells.
14	Data - Feature Extraction	Numerical feature non-numerical values list of fields to ignore has suitable coverage for the dataset.	There may be some inconsequential non-numerical values still encoded as "not present" entries, rather than ignoring the feature change altogether.

APPENDIX C TECHNICAL SOLUTION

C.1 DATA PRE-PROCESSING

C.1.1 RAW DATA

The raw data was comprised of data tables in CSV or Excel Workbook format. Some data sources were made up of multiple data files of the same structure (due to volume and/or export method). These raw data files covered the following categories, with the following fields available:

- **AKI Alerts**: AKI Update Date, AKI Update Time, Location, AKI, Year of Birth, Ethnic Origin, Sex, Spell ID, HL7VisitID, VisitID, Spell Count, Start date of any HD/RRT , Type of any HD/RRT
- **HISS patient spell metadata**: Spell ID, Method Of Admission, Admission Date, Method Of Discharge, Destination On Discharge, Discharge Date, Spell Wards Key, Stay on Critical Care Ward, Spell Specialty Key, Spell Diagnoses Key Code Description, Spell Procedures Key Code Description, Spells Count
- **Observations**: VisitID, Obsid, Timestamp, Location, Time, BP, Bp Ews, RR, Rr Ews, O2 Sats, O2 Sats Ews, Heart Rate, Heart Rate Ews, Urine, Urine Ews, EWS, Escalated
- **Pathology test results**: Specimen No, Sex, Who Age At Receive Date Years, Who Receive Date, Test Code, Test Expansion, Result, Units, Who Resulted Date Time, Admissions Id
- **Medication prescriptions**: Spell ID, Full Drug Name, Indication, Prescribed Time, Administration Event Time
- **Ward transfers:** Spell ID, Transfers ID, Transfer Sequence Number, Transfer Date, Transfer Time, From Ward, From Ward Code, From Specialty, To Ward, To Ward Code, To Critical Care Ward, To Specialty, Transfer End Date, Transfer End Time, Completed Ward Stay Count

Within the data ingest implementation Excel workbook files may have extra header lines above the table, which can be skipped through a data configuration option on ingest. CSV files may not have any additional non-header rows above table headers.

C.1.2 PRE-PROCESSING FILTERING AND AGGREGATION

Raw data pre-processing steps include:

- **Concatenation of columns** (such as date and time to create a “datetime” column.
- **Splitting of columns** (such as blood pressure into systolic and diastolic values.
- **Renaming of columns** (to map names to valid python identifiers – to assist with feature extraction).
- **Evaluation of duplicates** (to analyse and remove any duplicated entries – particularly when data exported to multiple files overlaps in time periods covered).
- **Formatting of date times** (to map various date time formats to a standard time format).
- **Lookup and unpacking of string values** (allowing extraction of detected lookup strings within text fields into a new column – specifically for extracting values representing presence of specific categorical values in “free” text fields).

C.1.3 FEATURE EXTRACTION

The full list of features extracted from the raw data into the structured dataset are (by type of event/data point):

- **Pathology test results**: CREA, 24hr Urine Phosphate, Chloride, Potassium, Sodium, Calcium, Urea, Urine Phosphate, Vol
- **Observations:** BP (Systolic)**,** BP (Diastolic)**,** Respiratory Rate**,** O2 Saturation**,** Heart Rate
- **Transfer:** To Ward**,** To Speciality
- **Medication:** Full drug name (administration time)

- **Admission:** Method of admission
- **Discharge:** Method of discharge**,** Destination on discharge
- **Metadata:** Year of birth**,** Ethnic Origin**,** Sex**,** Method of admission**,** Diagnosis Keys (for each co-morbidity)

Numerical features are encoded within the dataset as single numerical feature values (such as pathology test results, or observation readings). Categorical variables such as Medication drug name, or transfer location are encoded with each categorical value as an additional feature, with the presence encoded as a Boolean value change. Whilst this increases the dimensionality of the tensors within the model, it decouples the categorical values in feature space (rather than encoding them into a continuous variable).

Numerical features often have non-numerical values present. These are encountered when there is either a failed test or observation, or if there is other context to encode in the data. In this investigation a selection of non-numerical values were intentionally ignored (seen by the clinical team as not relevant, or invalid), with any remaining values encoded as empty numerical feature updates in the batch loader. This is done through the presence tensor. Values excluded are present in the following situations:

- Observations: values such as *“Refused, not concerned”, “Carer refused”, “BP observation not currently indicated”, “Un-recordable, not concerned”*
- Pathology tests with the following values:
  - CREA: *"PDA", "INSUFF", "..", INSUFT", "===", "NGEL", "X-NORESU", "ACC"*
  - 24h Urine Phosphate: *“NOSAM”, “NSR”*
  - Chloride: *“===”*
  - Potassium: *"UNLAB", "BLANK", "ALLUL", "INSUFF", UNSUIT", GHAEM", INSUFT", UNREL", "===", "INCOR", "KOLD", “NGEL", "INAD", EDTA", “OLD", “IBR", "ACC"*
  - Sodium: *"DCHEM", "COMM", "LEAK", "NSR", "UNREL", "GELUL", "UNSUIT", INSUFT", WGEL", “NGEL", "CONTA", "UNLAB", “BLANK", "ALLUL", "INSUFF", “INCOR", "INAD", "NOMACH", "===", "IBR"*
  - Calcium: *"INSUFF", "UNSUIT", "INSUFT", "===", "NGEL", "INAD", “EDTA"*
  - UREA: *"UNLAB", “BLANK", "INSUFF", "UNSUIT", "UNREL", "===", "NGEL", “INAD"*
  - Urine Phosphate: *"NOSAM", "NSR", "==="*
  - Vol: "F", "+", "RANDOM", ".....", "===", "N", "-"

C.1.3.1 Numerical Feature Normalisation

For the numerical features the values are normalised into a standard range. Due to limitations within the DeepMind framework (related to sparse/dense tensor definitions), these values are normalised between `min(val)` and `min(val) + 1` to avoid instances of a “zero” value occurring (underlying methods in the framework DeepMind themselves built with were used to restructure tensors depending on non-zero values, without consideration for ‘valid’ zero values).

The DeepMind approach capped the numerical features using the 1st and 99th centiles. This provides a more even distribution of values to the model. However, through discussion with the clinical team, it was determined that the investigation should not cap these values as for some key pathology results, the resultant distribution post-capping was seen to lose relevant granularity at the extremes (which potentially corresponds to the feature values of patient spells which have adverse outcomes occurring within them). See Appendix D.2.4 for numerical feature distributions.

C.1.4 Time Series Conversion

The date time fields extracted alongside the features for this dataset originate as standard absolute date time entries (such as `dd/mm/yyyy hh:mm:ss`). These were converted into relative date time values of the following form:

- Days since Jan 1 on year of birth
- Time bin corresponding to time of day

The feature change events were grouped into 6h wide time bins with any events with only a year/month/day identifier allocated to a final bin located at the end of the day period. This ensures that data without a specific time is not seen by the model before it occurs (due to the uncertainty of whether the event actually occurred in a future time bin).

C.1.5 EXCLUSION CRITERIA

If the first AKI alert (in time) took place when a patient was located on a renal or ITU ward, this spell would be excluded from the dataset – in this instance the patient will have already been within the remit of the renal health team through alternative means.

The field *"Start date of any HD/RRT"* within the AKI alert data table was used to exclude patient spells with existing renal replacement therapy (RRT/Dialysis), as these are not relevant to predicting the dialysis outcome and only make up a small proportion of spells.

C.1.6 LABELLING

For labelling mortality the labels are unfortunately inherently flawed in that the raw data acquisition does not include granularity of time point more precise than the date of death. This limits the ability of the model to predict outcomes at less than 24hs prior to the event, with the potential of a real event occurring at the start of a 24h period and only being able to predict the outcome within the earlier 24h period rather than up to 48hrs ahead of time. The "method of discharge" field from the HISS data is used, along with the "date of discharge" to label the mortality outcome.

For the dialysis outcome ICD-10 codes are used to assist with the labelling. The following codes in combination with a transfer to a renal ward are used to signify a dialysis event.

Relevant dialysis codes:

"T82.4", "Y60.2", "Y61.2", "Y62.2", "Y84.1", "Z49", "Z49.1", "Z49.2", "Z99.2", T82.4"

Renal ward locations:

"g15a", "g10", "g15n"

This is understandably flawed in that ICD-10 codes are a combination of previous diagnoses a patient had in their medical history, and diagnoses made during the current spell. This raises the possibility of edge cases in the dataset, where a patient may have had dialysis in a previous spell, was admitted to the renal ward on the current spell but did not receive dialysis on the current spell. Although in reality, this should not be flagged as a dialysis outcome as there is currently no way for the data labelling to differentiate between ICD codes relating to previous and current spells, and hence these cases would be spuriously labelled as dialysis outcomes.

For the ITU outcome the following ward codes are used for labelling:

"ficu", "fpic", "ritu"

Once an outcome is labelled for a specific event, these labels are recursively populated across the rest of the patient stay, such that the correct time interval labels are applied (x hours prior to outcome event)..

The labelling process captures:

- Boolean flag for adverse outcome occurring in the patient spell.
- Index of adverse outcome at occurrence (and persisting for any future events)
- Boolean flag for adverse outcome occurring within the next n time periods (matching the bin size for grouping of event updates), for each of the desired outcomes.

For each adverse outcome a series of labelling rules have been applied. Specific features have been chosen which indicate the occurrence of each of the adverse outcomes.

C.1.7 DATA INGEST TOOLING

The data ingest tooling allows for ingestion of a custom dataset into the format required for training a model using the delivered PoC codebase. This uses a configuration JSON file to define the data sources, features, and mappings between the relevant fields and the required labels and various keys.

C.1.7.1 Data Ingest Configuration

Example template for a data ingest configuration file (a further example can be found in the delivered codebase):

```
{
    "sources": [
        {
            "filepath": "<data path>" (can be list of file paths to concatenate-
for files with matching schemas),
            "skiprows": <number of rows to skip>, (excel only)
            "name": "<short name for data file, used as data key>",
            "master_key": "<field to use as master key field>"
            "secondary_key": "<used to state that the secondary key should be
used for this source>"
            "concatenate_columns": [{ (used to concatenate a combination of
columns on data load - assumes strings)
                "field": "<new field name>",
                "values": [ (assumes concatenating in order of list)
                    ... (column names to concatenate)
                ],
                "seperator": <seperator to add between string columns>
            }, ... (can be list of column pairs to combine with
concatenate_columns) ],
            "drop_columns": [] (list of column names from source to drop on load
- for efficiency of indexing)
            "split_columns": [ {
                "value": "<column name>",
                "fields": [
                    ... (list of fields in order to split)
                ],
                "delimiter": "-"
                }, ... (can be list of columns to split with split_columns)
            ],
            "str_lookup": {
                "type": "metadata",
                "filepath": "<filepath>", (lookup table)
                "name": "<name of table>",
                "primary_key": "<primary lookup column>",
                "secondary_key": "<secondary lookp column>",
                "value": "<column to look up values from>",
                "field": "<column name>", (new column to populate with detected
primary keys)
                "delimiter": "," (delimiter to use when detecting more than one
value from lookup list in value column)
            },
            "remove_duplicates": [column name, column name]
        }
        ... (can be list of files)
    ],
```

```
    "mapping": {
        "master_key": {
            "source": "<source for master key",
            "field": "<master key for unique data record", (master key to use to
select row in source table)
            "filter_rules": [ (rules on leaving out records relating to specific
master keys)
                {
                    "description": "<description>",
                    "source": "<source name>",
                    "field": "<master key field>",
                    "query_field": "<query field to look in>",
                    "drop_values": [
                        ... (list of values to detect)
                    ]
                },
                {
                    "description": "<description>",
                    "source": "<source name>",
                    "field": "<master key field>",
                    "query_field": "<query column name>",
                    "remove_if_populated": "True"
                }
            ]
        }
        "secondary_key": {
            "source": "<source for secondary key - must be same as master key>",
            "field": "<secondary key for unique data record>", (must be from same
source as master key, mapped together)
        }
        "numerical_features": [
            <list of numerical output feature names in any order,
            these will map to feature indexes depending on the order>
        ],
        "numerical_feature_exclude_list": {
            "<feature_name>": [
                (list of string values to ignore when finding feature entries)
            ],
            ... (allows multiple features)
        }
        "categorical_features": [ (these features will have name and value
                                    used to assign an index for the value in the
categorical feature
                                    value mapping table)
            {
                "name": "<feature name>",
                "source": "<source sheet>",
                "field": "<field in source>"
            },
        ]
        "metadata": {
            "<field name in new data record>": {
                "source": "<short name of data file (see above)",
                "field": "<name of field to extract>"
            },
            ... (can have multiple metadata keys)
        }
        "events": [
            {
                "category": "<category of event>",
                "source": "<short name of data file (see above)",
                "datetime": "<field in source to use as timestamp>"
                "features": [
                    {
                        "name": "<feature name>",
                        "field": "<field to access in data>",
                        "type": "<numerical or categorical, this will determine
how to handle one hot encoding>"
```

```
                    },
                    ... (multiple allowed)
                ]
            },
            ... (multiple allowed)
        ],
        "labelling": { (labelling is largely based on categorical feature values)
            "labels": {
                "of_interest": "<adverse outcome of interest>",
                "output_of_interest": "<Source data to export for keys of
interest>",
                "master_key_of_interest": "<Master key column for specific
source>",
                "adverse_outcome": {
                    "time_step": "6", (gap in hours between outcome labels)
                    "max_look_ahead": "8", (number of periods to look ahead for
labelling)
                    "labels": {
                        "<adverse outcome name>": {
                            "name": "<feature name to look at>",
                            "values": ["<value of categorical feature>"],
                            "metadata": [
                                <list of metadata feature names to presence check
as secondary rule>
                            ]
                        },
                        ... (multiple adverse outcomes allowed)
                    }
                }
            },
            "numerical_labels": {
            }
        }
    }
}
```

C.1.7.2 Ingested Data Structure

The structure of the ingested data is as a JSON string. In a JSONL data file each record is recorded as a single line comprising of a JSON string. For readability the indented structure is shown below.

```
{
    "record_number": "1",
    "metadata_1": "VALUE_3",
    "year_of_birth": "1200",
    "episodes": [
        {
            "events": {
                "patient_age": "345678",
                "time_of_day": "1",
                "entries": [
                    {
                        "feature_category_idx": "2",
                        "feature_idx": "1",
                        "feature_value": "4"
                    },
                    {
                        "feature_category_idx": "2",
                        "feature_idx": "1",
                        "feature_value": "4"
                    }
                ],
            }
```

```
            "labels": {
            }
            "numerical_labels": {
            }
        }
    ]
}
```

Required metadata fields include `"year_of_birth"` as this is used for date time conversion. The labels dictionary should be populated with key value entries covering each outcome and time interval as a Boolean value of 0 or 1 such as `"adverse_outcome_mortality_within_6h": 0`. See C.1.6 above for labels required.

For the current task the labels dictionary would contain:

```
"labels": {
    "adverse_outcome": "0",
    "adverse_outcome_dialysis_within_12h": "0",
    "adverse_outcome_dialysis_within_18h": "0",
    "adverse_outcome_dialysis_within_24h": "0",
    "adverse_outcome_dialysis_within_30h": "0",
    "adverse_outcome_dialysis_within_36h": "0",
    "adverse_outcome_dialysis_within_42h": "0",
    "adverse_outcome_dialysis_within_48h": "0",
    "adverse_outcome_dialysis_within_6h": "0",
    "adverse_outcome_in_spell": "0",
    "adverse_outcome_itu_within_12h": "0",
    "adverse_outcome_itu_within_18h": "0",
    "adverse_outcome_itu_within_24h": "0",
    "adverse_outcome_itu_within_30h": "0",
    "adverse_outcome_itu_within_36h": "0",
    "adverse_outcome_itu_within_42h": "0",
    "adverse_outcome_itu_within_48h": "0",
    "adverse_outcome_itu_within_6h": "0",
    "adverse_outcome_mortality_within_12h": "0",
    "adverse_outcome_mortality_within_18h": "0",
    "adverse_outcome_mortality_within_24h": "0",
    "adverse_outcome_mortality_within_30h": "0",
    "adverse_outcome_mortality_within_36h": "0",
    "adverse_outcome_mortality_within_42h": "0",
    "adverse_outcome_mortality_within_48h": "0",
    "adverse_outcome_mortality_within_6h": "0",
    "segment_mask": "0"
}
```

This set of labels corresponds to the chosen outcomes and time intervals within this commission.

C.2 Data Loading

C.2.1 Modified Data Loading Infrastructure

At this point in the pipeline, the pre-processed data was separated by spell, metadata was available for each spell, as well as a series of ‘events’, where each event spanned a 6h window, where any number of ‘entries’ could exist within an event. All entries within a single event were treated as having occurred at the same time, had a unique feature index, could hold a numerical value (which was set to 1 for categorical entries such as a ward transfer), and also had a feature *category*.

These categories grouped similar types of entry, such as ward transfer, location within which observations were taken, a prescribed medication etc. While the dimensionality of unique features was very high, approx. 13,000, owing to some groups of features (such as drug prescriptions) having a few thousand unique possibilities each, including a feature category allowed an explicit linking of related field types to the model, without losing the granularity of also having unique features for each possibility. This likely eased to burden of discovering related structures in the data for the model and may have allowed it to make inferences when an unfamiliar feature index was present within a familiar feature category (for example, the model may not have encountered a specific drug prescription before but will at least be able to use the fact that it *is* a drug prescription immediately, based on prior experience of drug prescriptions). Only 20 feature categories were used in this project, and a possible extension would be to experiment with the granularity of categories—differing dosages of a particular drug being allowed their own category, for example, rather than all drugs being treated with the same category.

Each entry within an event was then represented via the combination of six tensors, detailed as follows:

- Indexes category counts: which categories have occurred within this event
- Values category counts: how many times each category occurred within this event
- Indexes numeric: which numerical features have occurred during this event
- Values numeric: the values of the numerical features within this event
- Indexes presence: which features have occurred within this event
- Values presence: a 'severity' value for each feature that has occurred within this event, which was set to 1 for all of our features (this can be used to include features with a limited number of discrete values, e.g. the 1, 2, 3 stages from an AKI score, but were of little use in this project as no features of this type were discovered that were not already covered by continuous values elsewhere)

These tensors were then concatenated across a new dimension for each time window, and formed the 'sequence' information provided to the DL models. A pair of tensors for each type of metadata included in the model was also built, index and value, defining sex, ethnic origin, method of admission, and year of birth. ICD-10 (diagnosis) codes corresponding to the patient stay were also available for inclusion in the metadata. However these were eventually excluded (see Section 3.2.4).

C.2.2 CURRICULUM LEARNING

There was an identified class imbalance problem by the DeepMind authors that caused poorer performance on females than males, due to the use of their veterans dataset, and other confounding class imbalances may be present, such as an age distribution unrealistic of the general population. We refer to this type of class imbalance as input class imbalance, i.e., imbalances in the data fields that are visible to the model at ingest, as opposed to label imbalance, which refers to the proportions of different adverse outcomes (and the null case) present in data labels.

While it is simple to address input class imbalance of this form for a single field, addressing multiple fields is more problematic. We proposed to use curriculum learning to address this (Wang, 2021), in particular hard example mining (HEM), where the model automatically discovers 'difficult' data and adjusts its sampling rate to encounter these difficult cases more frequently.

Our justification for using this technique was that if an input class imbalance was leading to poor performance of the model on a particular population of patients, then encouraging the model to examine these cases more frequently would allow it to handle these underrepresented classes.

In practice, we applied HEM by having the model perform two passes through the full training dataset before recording its own performance against each datum for the purpose of reweighting the encounter rate. On the first pass, the model was not expected to perform well against data early in the dataset, and therefore performance on the first pass would not show a fair comparison of difficulty. Performance was

more stable by the second pass through of the dataset, however, and hence should form a more valid recording of difficulty. We averaged the cross entropy score of the model predictions against each time window and outcome, and used this as a weighting for the probability of encountering the datum in question in future passes, updating the difficulty measure for the datum each time it was encountered, while capping the measured difficulty such that no datum could fall below 5% likelihood of the maximum encounter probability, to avoid any data being lost to training via extremely low sampling probability.

We encountered a loss of performance when enabling curriculum learning during our experiments and brought this to the attention of project partners during the course of the project. As the model was performant without curriculum learning, and time limitations meant that other, higher priority activities may be at risk if we investigated in depth, the use of curriculum learning was deprioritised and not used for any of the results in this report. We theorise that the lowered performance may have been due to overfitting of the model on training examples of positive adverse outcome cases; there was a severe label imbalance in our dataset for adverse outcomes as compared to the null event outcome. We discovered that, when converted to a binary classifier for each of the adverse outcomes later in the project, that even an extremely low threshold of probability led to a good balance between positive predictive value and sensitivity, which would imply that in positive cases for each of the adverse outcomes the model was still predicting a low probability of these events, which would in turn lead to a high cross entropy loss for these cases, and hence the HEM technique would encourage a high degree of oversampling of positive adverse outcome cases, rather than discovering difficult cases that were due to input class imbalances. As only sparse examples of adverse outcomes exist in the dataset, this could have led to memorisation of positive adverse outcome cases in the training set, which would mean the model may not look for generalisable features that could be applied to the validation set, damaging validation performance.

C.3 Deep Learning Model

Further to being a supervised learning problem, the time series forecasting nature of the problem makes casting the problem in a manner suitable for a sequence prediction model a natural choice, e.g. Recurrent Neural Networks (RNNs) (Tealab, 2018). Typically, in these networks, a model will contain ‘recurrent blocks’, i.e. elements that can process time windowed information and make a current prediction for that time window, while feeding information forward to later blocks which have access to information from a later point in time and attempts to make a prediction for that later time window. In this way, a DL model can forecast the probability of an adverse outcome for a patient across configurable time windows, without risk of information ‘leaking backwards’, making it more realistic for use in a real-world system where complete patient information up until an adverse event will not be available.

These are much the same considerations that DeepMind made when formulating their approach in 2019 (Nenad, 2019), and their framework was built for configurability such that some adjustments to their codebase would allow us to leverage the considerable prior work done against the related task of AKI predictions towards predicting patient outcomes downstream of AKIs.

DeepMind’s data loading infrastructure, which was defined against a US veterans’ health record dataset, was not included with their codebase. Our dataset was initially constructed by the UHL team, extracted via SQL into a variety of Excel workbooks, and a considerable portion of the effort on this project was reverse-engineering the data loading infrastructure from descriptions in DeepMind’s published work, and direct inspection of the minimal working example data included in their codebase, which represented an interim stage of their data loading pipeline.

Once our data loading pipeline was constructed, occlusion analysis was used to explain which features in our dataset were most useful to the model. Each feature type, of around 10,000 fields, is masked from the model, and compared to the unmasked output. When masking results in a major change to the model’s output, this indicates that the field is important to the model’s decision. This goes some way to solving the ‘black box’ problem of DL models, where advanced models provide accurate predictions, but without a

clear explanation of how a decision was reached. By highlighting which fields were important, this method of explainability provides a list of priority fields that are likely important to predicting adverse outcomes, and hence may discover fields erstwhile underappreciated by the medical community for further study.

The RNN model architecture used is detailed in Figure 8, after (Nenad, 2019). Context ('historical', in the figure) and sequence ('current step', in the figure) data is combined into a deep embedding via encoder networks, before being passed to stacked RNN layers, and finally being passed to 'fully connected' layers to form a final prediction.

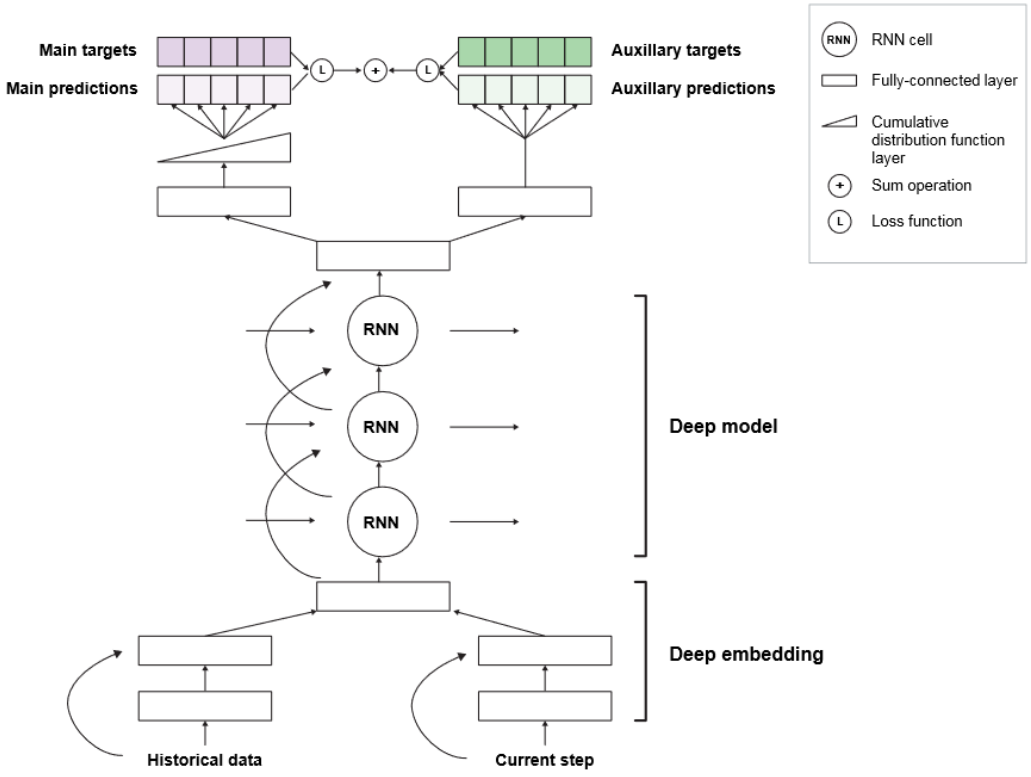


Figure 8: DL model architecture used in this project, after **(Nenad, 2019)**.

'Historical data' in this project is referred to as 'context data', and 'current step' is a single time point in the current stay's sequence data. In this project the auxiliary targets were not used, but would be of interest to reintroduce in future work.

C.4 Model Conversion

As mentioned in 2.4, our training objective relied on tasks built upon the 'adverse outcome task' included in the DeepMind codebase.

For each task, a new set of time windowed predictions are generated by the model, defining probabilities of the adverse outcome for the related task occurring at each time point. DeepMind included a generic 'adverse outcome task', which they used to encode the stage of AKI alert a patient experienced between 1—3, at config-defined time windows during a patient stay. We built upon this task type to include our adverse outcomes: the likelihood of ITU admission, the likelihood of dialysis, and the likelihood of death at any time window during a stay.

The DeepMind codebase had additional task type options that we did not eventually pursue during the project. The option for a continuous value prediction task, which they used to predict the value of a few renal function-related blood test results, and which showed a boost of performance of a few percentage

points in their other tasks when included, was a task type we had initially hoped to find an analogous set of lab results for and include in our training. However, this was deprioritised partway through the project in favour of completing higher priority objectives, with the complexity of enhancing our data loading pipeline to deal with looking ahead in time for the continuous values being the limiting factor in achieving this during the project. DeepMind also included a 'mortality task', which predicted whether a patient died during their stay without attempting to predict the specific time. This was not included, as it provided no clear benefit when our existing time-windowed mortality task was already to be constructed and used.

In addition to these disparate tasks, our dataset also had a subtly different structure to the 'context' data that was to be provided to the model (that is, fields which were true of the patient for the duration of their stay and were known at admission: sex, ethnic origin, method of admission, year of birth, and any ICD codes in their medical history). The DeepMind codebase had been purged of their proprietary data loading infrastructure, and late in the project it was discovered that this included a hard-coded removal of any context data prior to it reaching the DL model. As well as reworking the codebase to avoid this context removal, we needed to deal with the inclusion of the ICD codes as a single item, rather than providing them as unique elements. This was a result of DeepMind providing a separate, relatively simple DL model called an 'encoder' for every unique context item, while only requiring one encoder for the combined sequence data. DeepMind's unique context fields were likely quite limited, with this design in mind, but we were faced with a few thousand possible ICD codes; it would not have likely been viable to have a separate encoder for every context item. We therefore modified the codebase to allow context items to include lists of entries, so that most of our context was treated separately, as DeepMind intended, but ICD codes could be encoded into the model by a single encoder.

Finally, the DeepMind codebase did not include model architecture elements necessary for reloading a trained model and processing new data. We added these elements and built tooling to extract single-patient data from a batched structure at the output of the model into a contiguous set of adverse outcome probabilities for each time window.

C.5 Model Training

The model training experimental setup is based upon the template provided within the DeepMind codebase. The template was modified to incorporate:

- New configuration parameters required.
    - These were typically used to define mappings of features from the JSONL format to numeric values that could be used to build tensors, sets of features to exclude as they were known to be potentially unfair information to include for the model's training, sets of features that needed special handling such as ICD codes, new task definitions, DL model architecture parameters that would allow handling of our specific number of unique features of each type, locations of data on hard drives relative to the codebase.
- Integration with the new JSON Lines batch loader.
    - The example data batch loader provided with the dataset ingested a set of fake protobuffer files that had been pre-processed to tensors by DeepMind's proprietary (and removed) data loading methods. We created an alternative data batch loader that relies on human-readable JSON Lines files, processes them to tensors as required by DL models, and otherwise inherits similar functionality to DeepMind's batch loader.
- Additional metric tracking while training.
    - DeepMind's framework came with tools to record a wide variety of statistical measures of performance of their models against validation data, e.g. true positives, true negatives, false positives, false negatives, and derived quantities such as positive predictive value and sensitivity. We added the inclusion of precision-recall and receiver operator characteristic raw values, for downstream analysis. We additionally included functionality to reload the models and change the binary classifier threshold used to calculate these values, allowing for

threshold sweeps that optimised this threshold beyond the standard choice of 0.5. We modified the default frequency of these metric calculations to align with the frequency of model check pointing, to guarantee that saved models aligned with the performance metrics generated.

- Export of performance metrics from the validation dataset while training.
  - While DeepMind's training procedure calculated many useful metrics routinely during training, they were simply printed to console and not retained. We modified this functionality to retain all performance metrics, which allowed searching through training history for optimal models.
- Ability to run task against multiple new objectives.
  - We modified the DeepMind config handling to allow for different task combinations (e.g. predicting dialysis and/or ITU admission and/or mortality) to be used in separate experiments, where it originally was set to always expect a single adverse outcome in addition to continuous lab value predictions as tasks. This allows fresh models to be trained against prioritised objectives, e.g. only the ITU outcome without attempting to predict other adverse outcomes, with minimal changes to the code.

Within the codebase the CLI tool provided a single point of entry however multiple standalone experiments were retained, which provided traceability of results and the ability to run certain experiments or processing in a standalone manner. This was particularly relevant following discussions on the inclusion or exclusion of context and/or sequence fields which constituted unfair information to provide to the model during training. A clear separation between experiments via individually defined experiments aided in comparing results.

C.5.1 Setting up a Training Experiment

We modified the codebase to contain multiple training experiments which can be executed via command line interface (CLI), with flags for setting the training data directory, output directory, and number of training steps. The output directory need not exist at the commencement of model training, but the data directory should include:

- A training dataset file, with name ingest_records_output_lines_train_uncapped.jsonl
  - This, and other dataset files, follow a nested dictionary structure that defines a single patient spell per line, where each line follows the JSON format.
- A validation dataset file, with name ingest_records_output_lines_validate_uncapped.jsonl
  - A dataset file defining a fraction of the full dataset to be used for evaluation, and selecting a best model
- A test dataset file, with name ingest_records_output_lines_test_uncapped.jsonl
  - A dataset file defining a fraction of the dataset that can be completely unobserved during training, even for measuring best model performance. This and validation data are often treated as one in machine learning problems, as we did for this project, and the data therein can be a copy of the validation data, or left empty.
- A calibration dataset file, with name ingest_records_output_lines_calib.jsonl
  - A dataset file defining an unseen fraction of the data, which was used by DeepMind to linearise their output probabilities of adverse events. This exercise was not deemed necessary in the course of this project, but the DeepMind framework requires such a file to be present for training runs to initialise. Data herein may be a copy of the validation data, or left empty.
- A category mapping file, with name category_mapping.json
  - A definition of all features mapped to integer categories, e.g. all ward transfers may be mapped to one category, all drug prescriptions to another, etc.
- A feature mapping file, with name feature_mapping.json
  - A definition of all sequence features mapped to unique integers.
- A numerical feature mapping file, with name numerical_feature_mapping.json

    - A definition of all sequence features that need handling as continuous values, mapped to the same integers as in feature_mapping.json
- A metadata mapping file, with name metadata_mapping.json
    - A definition of all context features mapped to unique integers (which may overlap with the integers used in feature_mapping.json). These will be separated for encoding via different encoders according to the prefixes 'diagnosis', 'ethnic_origin', 'method_of_admission', 'sex' and 'year_of_birth'. Different fields within these possible features can be defined by appending any valid string, i.e. 'diagnosis_a' and 'diagnosis_b' would both be handled as diagnoses metadata.
- A missing metadata mapping file, with name missing_metadata_mapping.json
    - A definition of metadata fields corresponding to default values when a piece of metadata is missing from a patient spell, with null definitions allowed—some metadata fields from our project partners came with existing defaults, such as 'ethnic_origin_nan', while others did not.
- A sequence giveaways file, with name sequence_giveaways.json
    - A definition of all sequence fields that are to be blocked during data loading, so that the model cannot learn to expect certain outcomes from their presence. These might be, for example, 'method_of_discharge_died', which can be recorded earlier than the daily flag that a spell ended in death, clearly giving unfair advantage to the model when attempting to predict a patient death.

The detailed structure required in each file can be seen via fake data provided with the codebase, in aki-predictions/tests/fixtures. They are also summarised in Appendix C.5.1.

Multiple experiments were retained, rather than creating a single entry-point, largely for traceability of results—discussions on which context and sequence fields constituted unfair information to provide to the model during training extended late into the project, and a clear separation between experiments via separately defined experiments aided in comparing results.

Despite the presence of these multiple training experiments, we suggest that aki_predictions/aki_predictions/training/multiple_adverse_outcomes_w_context_training.py is the most complete experiment, as it is defined such that all available context information is passed to the model, and sequence information is purged during data loading of 'giveaway' fields that would be unfair to expose to the model.

C.5.2 Training Methodology – Hyper Parameter Sweep

The term 'hyperparameters' typically refers to parameter values that are set at the beginning of training and are not affected by the gradient descent mechanism that tunes the vast majority of parameters in a DL model. These would include learning rate, batch size, layer structure (in DeepMind's nomenclature, 'cell type'), etc. Commonly, DL researchers will choose a few such hyperparameters that are expected to impact model performance greatly, and optimise these.

Tuning the learning rate and number of training steps of a model is practically a default step within DL model design, and so devoting some effort towards these was a clear choice at the outset of model training. DeepMind also included configurable options for their choice of 'recurrent cell', with three options accessible in their framework: Simple Recurrent Unit, (SRU) cells, Update Gate Recurrent Neural Network (UGRNN) cells, and long-short term memory (LSTM) cells. Discussion of the structure of each of these is beyond the scope of this report, but broadly they each are a stacked series of network layers, where each layer takes in one time element of sequence information, as well as the output of the previous layer. The specifics of how each layer balance and combine the previous layer's output with its own sequence element defines the separate cell types.

The DeepMind authors conducted a thorough exploration across many hyperparameter options, and even explored non-deep learning models for comparison (Supplementary Sections G and H, (Nenad, 2019)), and found that recurrent DL networks performed best on precision-recall area under curve (PR-AUC) and receiver operating characteristic area under curve (ROC-AUC) (Czakon, 2022), with little difference in performance between recurrent cell choice. While their optimisation efforts were thorough, it is not guaranteed that their optimal choices translate to optimal choices for our problem. Though our compute resources were more limited, such that we could not complete a hyperparameter search over as many options, it was deemed worthwhile to explore some key parameters: learning rate, total training steps, and recurrent cell choice. In the codebase the best other discovered hyperparameters were included as defaults.

As inclusion of context fields was achieved near the end of the project, hyperparameter scans were conducted with sequence data only, and with some sequence fields that were discovered to be 'giveaway' fields later in analysis. However, these giveaway fields were sparse among the highest importance fields discovered, and so the results of our hyperparameter search are expected to still be valid.

C.6 Performance Criteria

C.6.1 Performance Metrics

The DeepMind codebase contained implementation to capture the following performance metrics:

- **true_positives -** True positive count
- **false_positives –** False positive count
- **true_negatives** – True negative count
- **false_negatives –** False negative count
- **rocauc –** Area under curve for receiver operator characteristic
- **prauc –** Area under curve for precision recall curve
- **normalised_prauc –** Normalised precision recall area under curve
- **nb_ex –** total number of examples over which to compute metrics
- **nb_ex_pos –** total number of positive examples in the computations
- **nb_ex_neg –** total number of negative examples in the computations
- **tp –** True positive rate
- **tn –** True negative rate
- **ppv –** Positive predictive value
- **npv –** Negative predictive value
- **acc – A**ccuracy of predictions
- **f1 –** F1 Score (harmonic mean of precision and recall)
- **f2 –** Fbeta-measure with beta value of 2
- **mcc –** Matthews correlation coefficient
- **expected_confidence_calibration_error –** included statistic relating to calibration error (not used in this commission)
- **maximum_confidence_calibration_error –** included statistic relating to calibration error (not used in this commission)
- **expected_risk_calibration_error –** included statistic relating to calibration error (not used in this commission)
- **maximum_risk_calibration_error –** included statistic relating to calibration error (not used in this commission)

The raw PR and ROC values were added during this project to aid in analysis and plotting post-training.

In this instance the "Accuracy" metric refers to the number of correct predictions divided by the total number of predictions. In this instance it corresponds to the aggregated predictions across each time step, rather than on a per-spell basis. The equation is as follows:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

(Where TP = Number of true positives, FP = Number of false positives, FN = Number of false negatives)

C.6.2 VISUALISING MODEL PERFORMANCE

Due to the time series nature of the problem, in estimating against multiple time intervals, there are a number of ways to display the confusion type metric data. This is typically done through the following methods:

- Confusion Matrices
- Precision Recall Curves (PR)
- Receiver Operator Characteristic Curves (ROC)

The first is through individual confusion matrices (showing the actual vs predicted class values). In this instance a single confusion matrix would simply show the True and False Positives and True and False Negatives for a specific predictive time interval. This is defined as the table below:

Example Confusion Matrix		**Predicted**	
		False	**True**
Actual	**False**	True Negative (TN)	False Positive (FP)
	True	False Negative (FN)	True Positive (TP)

Table 7: Confusion Matrix Definition

The values captured within a given confusion matrix (TP, TN, FP, FN) relate directly to values used to generate other metrics and visualisations (such as Precision and Recall).

In this investigation a high level plot of the confusion values was generated which shows the performance at each of the predictive time intervals. An example of such a plot is shown in Figure 9 below:

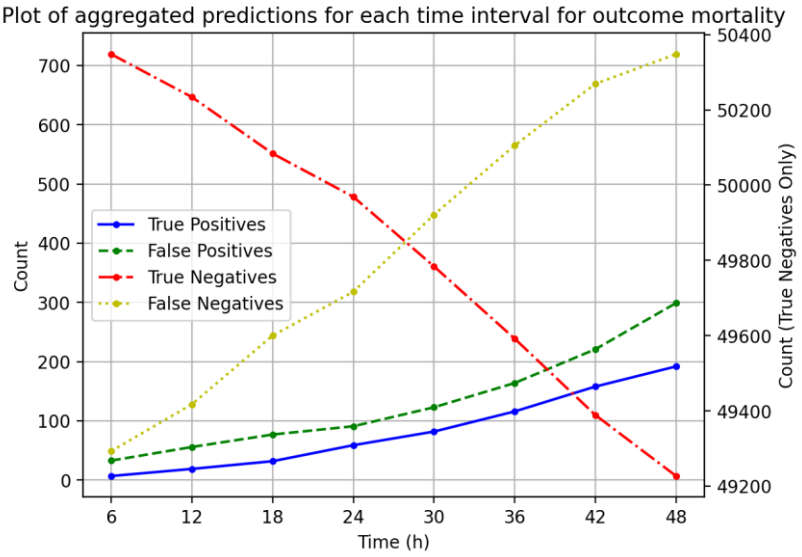


Figure 9: Example confusion plot showing raw confusion values across each time predictive time interval. *Note: the values shown are aggregated counts of class predictions covering each time step evaluated within the evaluation data split, rather than a per-spell count.*

Performance evaluation methods were implemented within the software tooling to allow plotting of PR and ROC curves. These show the performance of the model at different predictive class thresholds. In this instance a single plot shows a different performance line depending on the predictive time interval (performance changes across predictive time interval).

The area under these curves can also be used to evaluate the performance of the model. Examples of these plots are shown in Figure 10 below.

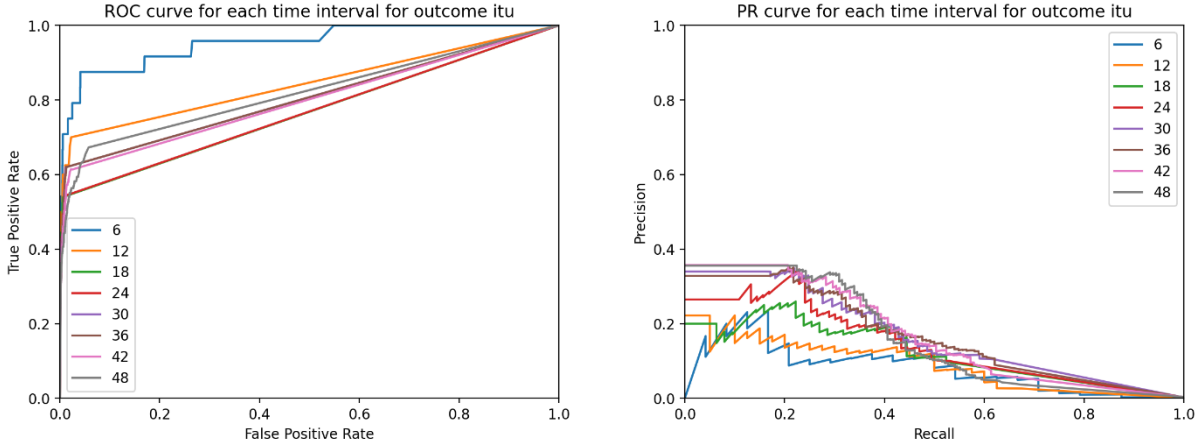


Figure 10: Example PR and ROC curve plots for a given adverse outcome.

C.6.3 Model Selection Metric

The geometric mean (see formula below) is often used for sets of numbers which are often multiplied together or when ratios or area calculations are involved. This provides a value which typically is a more natural compromise between the values. In this instance it provides a less extreme shift in value if one of the values dramatically changes.

$$Geometric\ Mean = \sqrt[n]{x_1 x_2 \dots x_n}$$

In this instance applied to the chosen performance metrics:

$$Performance\ Scoring\ Metric = \sqrt{PPV \times Sensitivity} = \sqrt{\frac{TP}{TP+FP} \times \frac{TP}{TP+FN}}$$

(Where TP = Number of true positives, FP = Number of false positives, FN = Number of false negatives)

Depending on the desired evaluation approach (which outcome is more important, such as addressing false positives, or addressing false negatives) other metrics could be applied for selecting the model, such as F1-Score, Accuracy, or G-Mean (Geometric mean of sensitivity and specificity).

For the chosen performance metric, true negative count is not referenced, as the negative predictive value can provide a suitable indication of true negative performance, and the true negatives (whilst the majority class in this imbalanced dataset), are not the key focus for clinicians.

C.7 Explainability Analysis

'Occlusion analysis' is a common means to achiev feature importance extraction. A trained model processes the validation dataset once more, but with features successively masked from the input. Differences between model output with and without each feature present then are measures of how much the model relied on each feature, and hence the importance of each feature.

The DeepMind team performed this type of analysis by measuring the increase or decrease to risk of an AKI when masking a feature. However, we felt this exact approach may not be appropriate; a single feature may shift risk greater or lower, depending on actual value. Therefore, we instead measured the cross-entropy of the model outputs with and without each feature, with the cross-entropy being a measure of difference between statistical distributions.

C.8 Threshold Analysis

The class predictive threshold sweep used the following threshold set to evaluate the performance of the chosen model at different operating points. A class predictive threshold default of 0.5 was used when extracting the initial performance metrics whilst training. Following selection of a suitable model, a sweep of class predictive threshold values was performed. This selects a more optimal threshold for each of the chosen outcomes.

Initially a sweep of threshold values was performed of values between 0.01 and 1 in 0.01 steps. This was expanded to include the range between 0.001 and 0.01 in 0.001 steps as for some training experiments this range was found to be more applicable.

APPENDIX D DATA SURVEY

D.1 RAW DATA

The raw dataset (pre-spell aggregation) contained a large volume of data relating to over 25 million rows. Not all the data (see Table 8) makes its way into the aggregated dataset, as duplicate entries are removed, and not all spell data is relevant to the investigation (see exclusion criteria).

For further information on the spell exclusion criteria see Section C.1.5.

Data Source	Subset	Total number of entries
AKI Alerts	-	32849
HISS patient spell metadata	-	24457
Observations	-	1691638
Pathology test results	Total	15396024
	Creatinine (CREA)	4090138
	24h Urine Phosphate	131
	Chloride	128402
	Potassium	3254027
	Sodium	3263898
	Calcium	1395026
	UREA	3263620
	Urine Phosphate	782
	Vol	3638
Medication prescriptions	-	8057342
Ward transfers	-	62575

Table 8: Table of Raw Data Statistics

D.2 INGESTED DATASET

The data ingest identified 24342 unique spell keys and following filtering the aggregated dataset includes data from 21225 unique spells.

D.2.1 DATA SPLITS

For each of the adverse outcomes the data splits and corresponding adverse outcome label counts see Table 9 below. During the data splitting, as patient spells can contain multiple adverse outcomes, there has not been any application of distribution enforcement across the data splits. The assumption when ingesting this data volume is that the random selection of spells has been sufficient to adequately represent each outcome class within each split.

It is worth noting that for the small number of dialysis spells this is more challenging statistically, therefore there is a higher differential in percentage of spells with that outcome across the datasets (particularly between validation and calibration which have an equal global split percentage).

	Total		Train		Test		Validate		Calibration	
	Count	%	Count	%	Count	%	Count	%	Count	%
Total	21225	-	16980	80.00	2122	10.00	1061	5.00	1064	5.00
Mortality Outcome	3393	15.99	2715	15.99	347	16.35	178	16.78	153	14.41
ITU Outcome	854	4.02	693	4.08	85	4.01	35	3.30	41	3.86
Dialysis Outcome	315	1.48	242	1.43	34	1.60	12	1.13	27	2.54
Any adverse outcome	4394	20.70	3512	20.68	456	21.49	219	20.64	207	19.49

Table 9: Data splits and corresponding adverse outcome proportions.

D.2.2 NUMERICAL STATISTICS

Further assorted statistics regarding the dataset are detailed in Table 10.

	mean	std	max	min	centile_1	centile_99
Stay length (days)	**14.40**	**17.74**	**427**	**0**	**0**	**82**
Feature changes during stay	**2.09**	**1.01**	**62**	**0**	**1**	**5**
Year of birth	**1949.79**	**17.60**	**2021**	**1916**	**1924**	**1999**
Age in 2022 (for reference)	**72.21**	**17.60**	**106**	**1**	**23**	**98**

Table 10: Numerical statistics covering the ingested dataset.

D.2.3 DEMOGRAPHICS

Table 11 shows the breakdown of the demographic and admissions variables for the dataset.

Context	Field Value	Count	Percentage
Ethnic Origin	WHITE BRITISH	15883	74.83
	ASIAN/ASIAN BRITISH INDIAN	2774	13.07
	WHITE OTHER WHITE BACKGROUND	669	3.15
	ANY OTHER ASIAN BACKGROUND	356	1.68
	ASIAN/ASIAN BRIT BANGLADESHI	93	0.44
	BLACK/BLACK BRITISH CARIBBEAN	184	0.87
	ASIAN/ASIAN BRITISH PAKISTANI	132	0.62
	ANY OTHER ETHNIC GROUP	274	1.29
	WHITE IRISH	192	0.90
	NOT STATED	317	1.49
	BLACK/BLACK BRITISH AFRICAN	119	0.56
	OTHER ETHNIC GROUP CHINESE	59	0.28
	ANY OTHER BLACK BACKGROUND	68	0.32
	ANY OTHER MIXED BACKGROUND	31	0.15
	MIXED WHITE AND ASIAN	26	0.12
	MIXED WHITE & BLACK CARIBBEAN	37	0.17
	MIXED WHITE AND BLACK AFRICAN	11	0.05
Sex	Male	10815	50.95
	Female	10410	49.05
Method of admission	A&E LRI	12966	61.09
	SELF ADMISSION	1088	5.13
	EMERGENCY GP	377	1.78
	EMERGENCY BED BUREAU	1075	5.06
	EMERGENCY IMMEDIATE	2615	12.32
	MATERNITY A PARTUM	384	1.81
	EMERG HOSP TRANSFER	130	0.61
	EMERGENCY OP CLINIC	355	1.67
	WAITING LIST	1334	6.29
	MATERNITY P PARTUM	30	0.14
	PLANNED	424	2.00
	OTHER NON-LRI A&E	67	0.32
	NON-EMERG HOSP TRANS	182	0.86
	NOT STATED (None value)	140	0.66
	BOOKED	54	0.25
	EMERGENCY HOME VISIT	3	0.01
	BORN AT HOME AS INTD	1	0.005

Table 11: Demographic and admission statistics for the ingested dataset.

D.2.4 NUMERICAL FEATURE DISTRIBUTIONS

In total 14 numerical features were included in the dataset for training and evaluation. These cover observations and pathology tests.

For each of the numerical features, the distribution of values has been computed and plotted to show the key statistical values for each feature (pre-normalisation). Plots of these distributions are included below.

Note: Some of the plots do have issues displaying the histogram values, this is likely due to either a very tight distribution (effectively a single line) or a technical issue when rendering the plot. In these instances the box plots do indicate the statistics regarding the feature.

D.2.4.1 Observations

The observation features included are:

- Blood pressure (diastolic)
- Blood pressure (systolic)
- Heart Rate (beats per minute)
- Oxygen Saturation (O2 Sats, %)
- Respiration Rate (RR, respirations per minute)

The distribution and high level statistics for each of these are shown in Figure 11, Figure 12, and Figure 13 below.

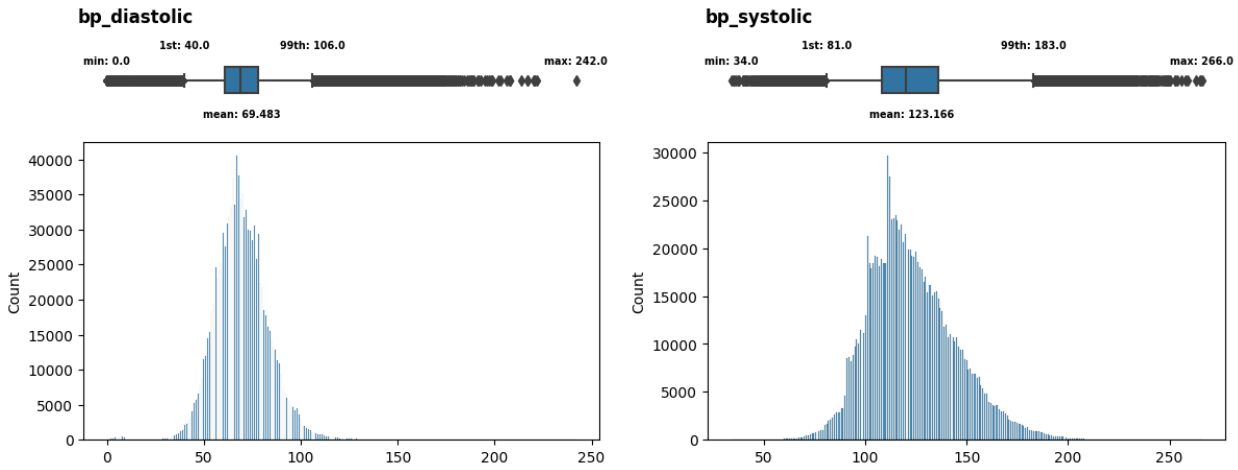


Figure 11: Numerical feature distributions for the Blood Pressure numerical features, of Systolic and Diastolic (logged as separate features).

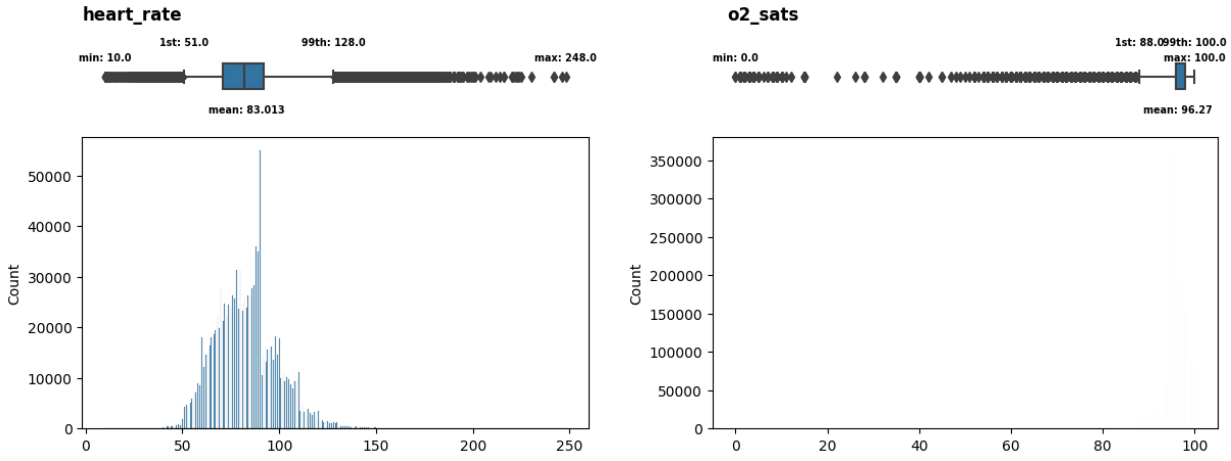


Figure 12: Numerical feature distributions for the Heart Rate and Oxygen Saturation features.

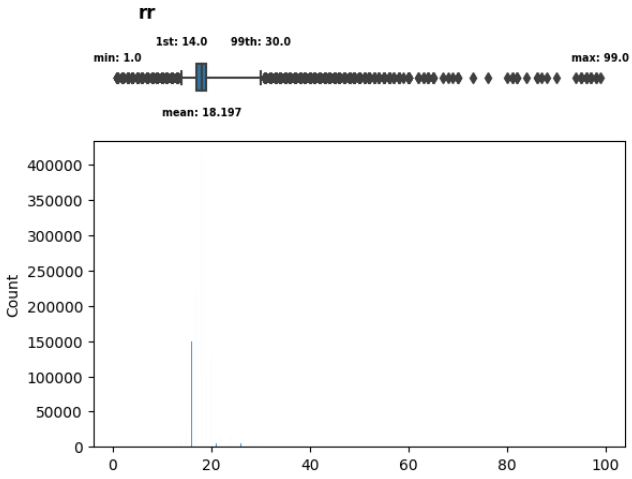


Figure 13: Numerical feature distribution for the Respiratory Rate feature.

D.2.4.2 Pathology Tests

The pathology test features included are:

- 24h Urine Phosphate
- Urine Phosphate
- Calcium
- Chloride
- Creatinine (CREA)
- Potassium
- Sodium
- Urea
- Vol (volume of urine)

The distribution and high level statistics for each of these are shown in Figure 14, Figure 15, Figure 16, and Figure 17 below.

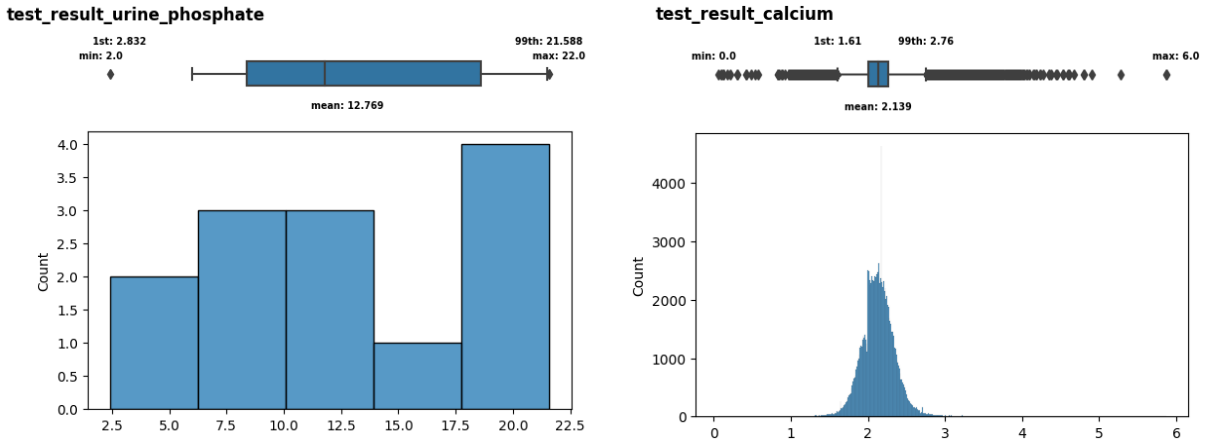


Figure 14: Numerical feature distribution for Urine Phosphate and Calcium.

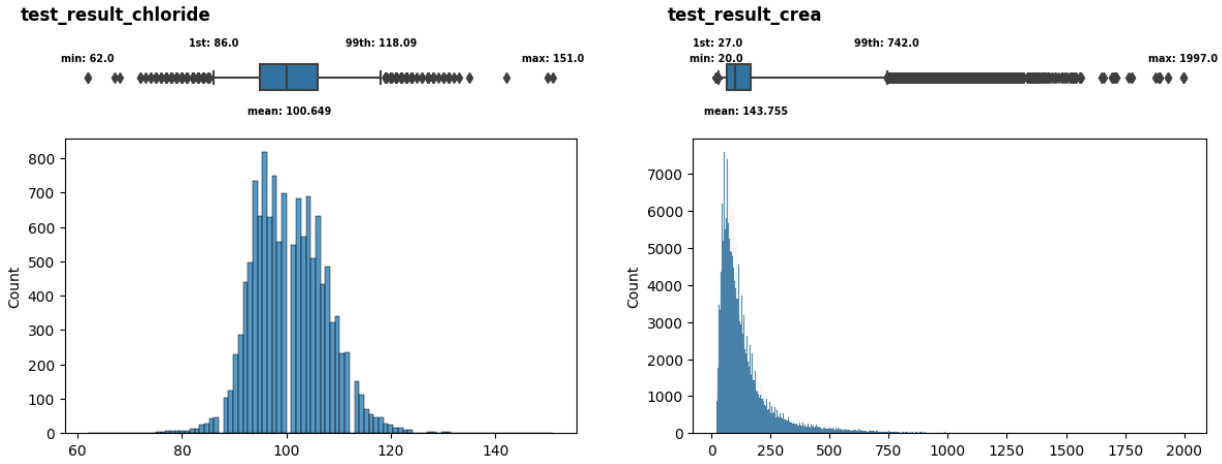


Figure 15: Numerical feature distribution for Chloride and Creatinine.

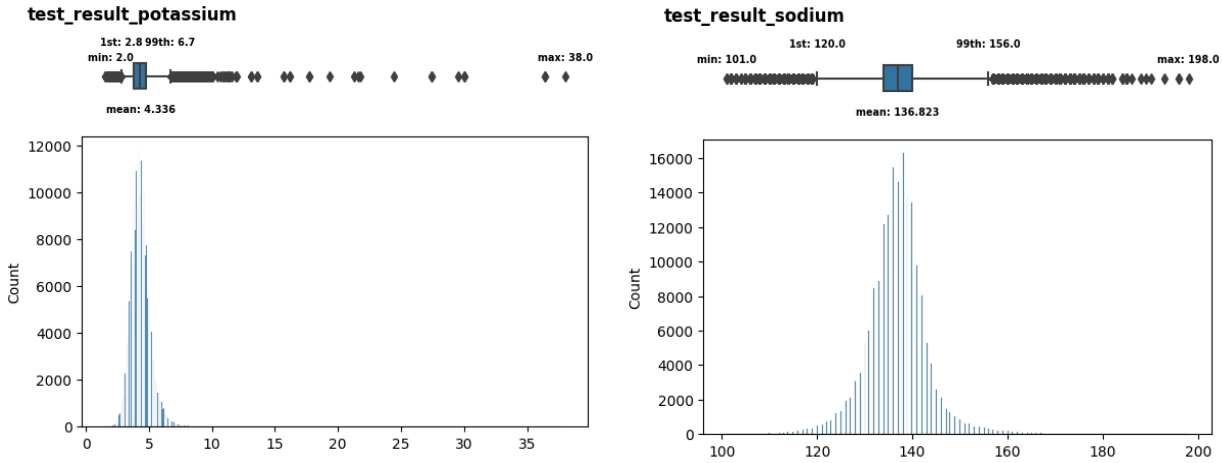


Figure 16: Numerical feature distribution for Potassium and Sodium.

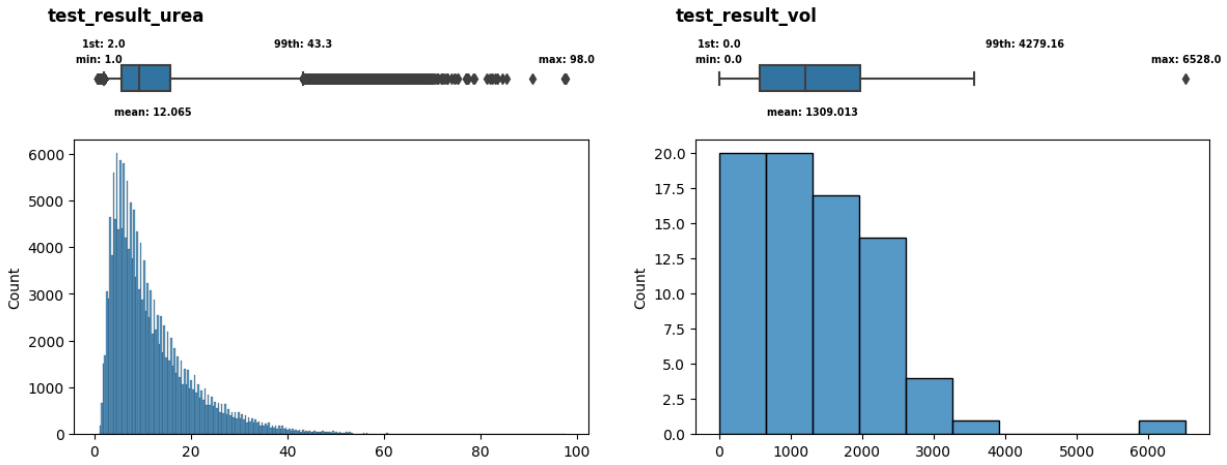


Figure 17: Numerical feature distribution for Urea and Vol.

APPENDIX E TRAINING OUTCOMES

Additional technical material relating to the experimental outcomes (additional training experiments and resultant performance plots, statistics, and commentary).

E.1 HYPER-PARAMETER TUNING

E.1.1 LEARNING RATE

For the learning rate hyper parameter tuning experiment, five different learning rate values we chosen.

The scoring metric for each of these rates is shown in the table below:

Metric	0.0001	0.0003	0.001	0.003	0.01
PPV	0.235	0.228	**0.257**	0.024	0.127
NPV	0.995	0.995	0.995	0.994	0.995
Sensitivity	0.226	0.202	**0.236**	0.003	0.144
Specificity	0.995	0.996	0.996	**0.999**	0.994
Accuracy	0.991	0.991	0.991	**0.993**	0.989
Score	0.230	0.215	**0.246**	0.009	0.135

Table 12: Table of performance scores for each learning rate value, for the Mortality outcome at the 24 hour predictive time interval, for the default class predictive threshold of 0.5. These values correspond to a training run of 60,000 training steps. Bold figures indicate the highest found for the highlighted metric in this test.

It can be seen from the values in Table 12 above that the learning rate value of 0.001 shows the highest performance score and both the highest PPV value and highest Sensitivity of all of the learning rates tested. This value has subsequently been used for all further training runs.

E.1.2 RECURRENT CELL CHOICE

For each of the recurrent cell choices the performance of the model was evaluated. This performance is shown in the table below:

Metric	SRU	UGRNN	LSTM
PPV	0.171	0.217	**0.342**
NPV	0.995	0.995	0.995
Sensitivity	**0.216**	0.140	0.134
Specificity	0.994	0.997	**0.998**
Accuracy	0.989	0.992	**0.993**
Score	0.192	0.175	**0.214**

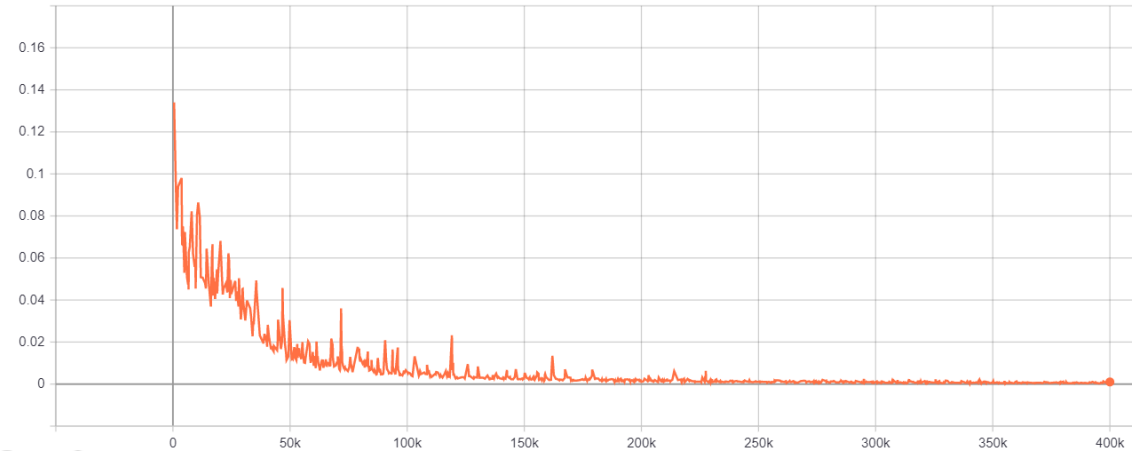
Table 13: Table of performance scores for each recurrent cell choice, for the Mortality outcome at the 24 hour predictive time interval, for the default class predictive threshold of 0.5. These values correspond to a training run of 60,000 training steps. Bold figures indicate the highest found for the highlighted metric in this test.

It can be seen from the values in Table 13 above that the LSTM recurrent cell choice shows the highest performance score and also the highest PPV, Specificity and Accuracy. The SRU recurrent cell does show

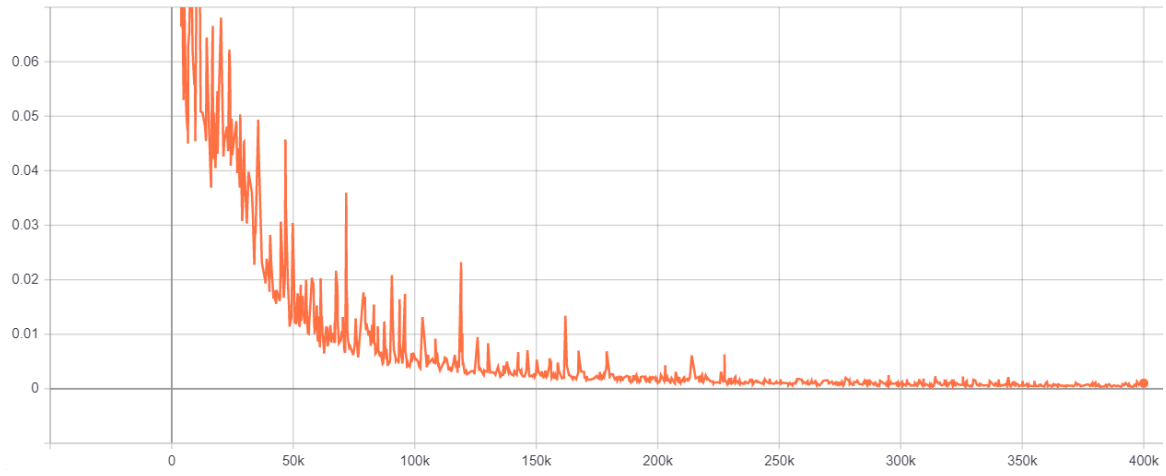
better performance against False Negatives (higher sensitivity), however the number of false positive predictions is increased, leading to a lower performance score.

E.1.3 Determining Training Duration (step count)

From the extended training run, whilst the cross-entropy continues to decline throughout, from the performance metrics the optimal model lies in the 150,000-250,000 step range. Whilst the cross-entropy continues to fall slightly by 300,000-400,000 steps there is an increase in false positive values for the mortality outcome. This suggests that the model is somewhat overfitting on the training data whilst not gaining further improvements on the validation data split when trained past 250,000 steps. The performance could potentially be improved in future through adjustment in hyper-parameter values and further training.



(a)



(b)

Figure 18: Cross-entropy training loss for the extended 400,000 step training run. (a) Full loss plot. (b) Zoomed plot showing late stage training loss in further detail.

Figure 18 shows the cross-entropy training loss for this training experiment. The plot is supported by the values in Table 14 indicating a plateau in performance and loss past the 250,000 step point.

Metric	152k	200k	252k	300k	352k	400k
PPV	0.267	0.233	0.172	0.180	0.152	0.152
NPV	0.993	0.993	0.993	0.993	0.993	0.993
Sensitivity	0.117	0.119	0.101	0.114	0.119	0.119
Specificity	0.998	0.997	0.996	0.996	0.993	0.995
Accuracy	0.991	0.990	0.990	0.989	0.988	0.988
Score	0.176	0.167	0.131	0.143	0.134	0.135

Table 14: Table of performance metrics for an extended training run, showing periodic performance data.

E.2 BASELINE PERFORMANCE

For the baseline model (trained without the context features included, and with giveaway values removed), the following plots show the model performance characteristics for each of the adverse outcomes, along with the relevant performance statistics. These complement the analysis in Section 3.2.5 of the main report, and are included largely for reference.

E.2.1 MORTALITY OUTCOME

The plots in Figure 19 show many of the key performance characteristics of the baseline model for the mortality outcome.

The aggregated time step predictions can be viewed as a confusion matrix as follows:

		Predicted	
		False	**True**
Actual	**False**	49949	111
	True	317	60

Table 15: Confusion matrix for the Mortality outcome at the 24 hour predictive time interval for the baseline trained model, at the default predictive class threshold of 0.5. Represents aggregated time step predictions.

From this the following confusion plot can be generated (and for each predictive time interval).

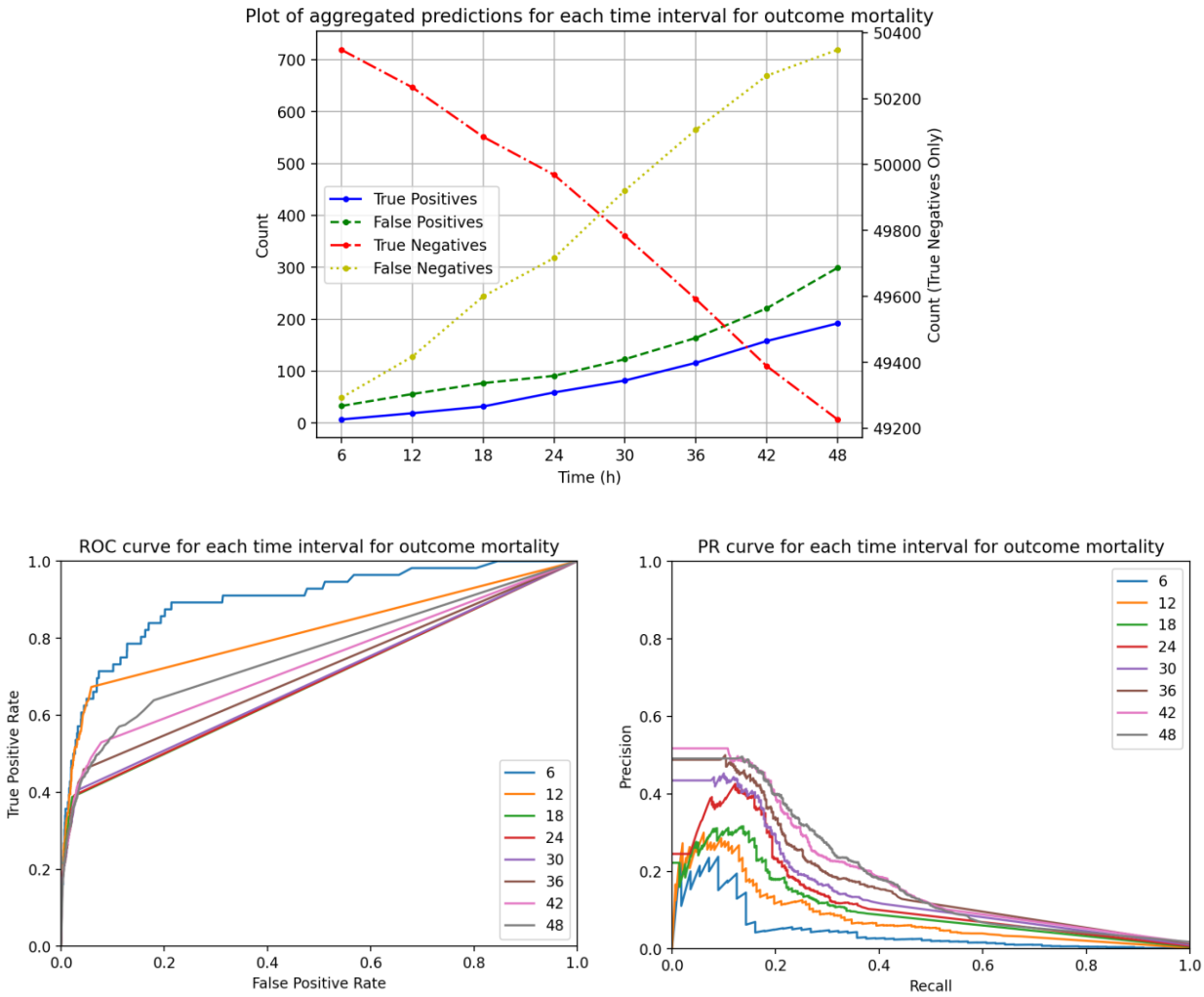


Figure 19: Performance plots for the mortality outcome for the baseline trained model.

E.2.2 ITU OUTCOME

The plots in Figure 20 show many of the key performance characteristics of the baseline model for the ITU outcome.

The aggregated time step predictions can be viewed as a confusion matrix as follows:

		Predicted	
		False	**True**
Actual	**False**	50298	56
	True	71	12

Table 16: Confusion matrix for the ITU outcome at the 24 hour predictive time interval for the baseline trained model, at the default predictive class threshold of 0.5. Represents aggregated time step predictions.

From this the following confusion plot can be generated (and for each predictive time interval).

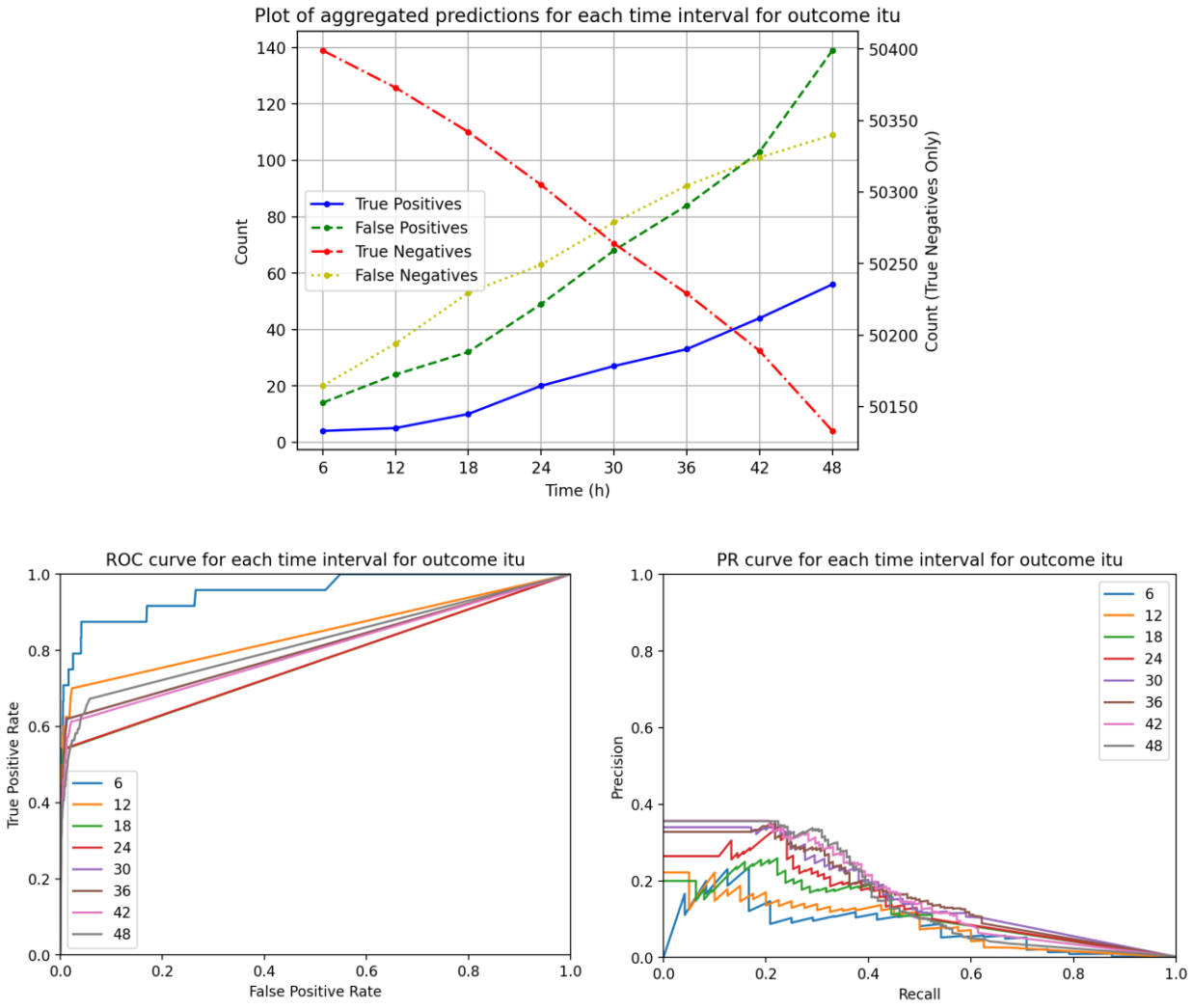


Figure 20: Performance plots for the ITU outcome for the baseline trained model.

E.2.3 DIALYSIS OUTCOME

The plots in Figure 21 show many of the key performance characteristics of the baseline model for the dialysis outcome.

The aggregated time step predictions can be viewed as a confusion matrix as follows:

		Predicted	
		False	**True**
Actual	**False**	50396	4
	True	34	3

Table 17: Confusion matrix for the Dialysis outcome at the 24 hour predictive time interval for the baseline trained model, at the default predictive class threshold of 0.5. Represents aggregated time step predictions.

From this the following confusion plot can be generated (and for each predictive time interval).

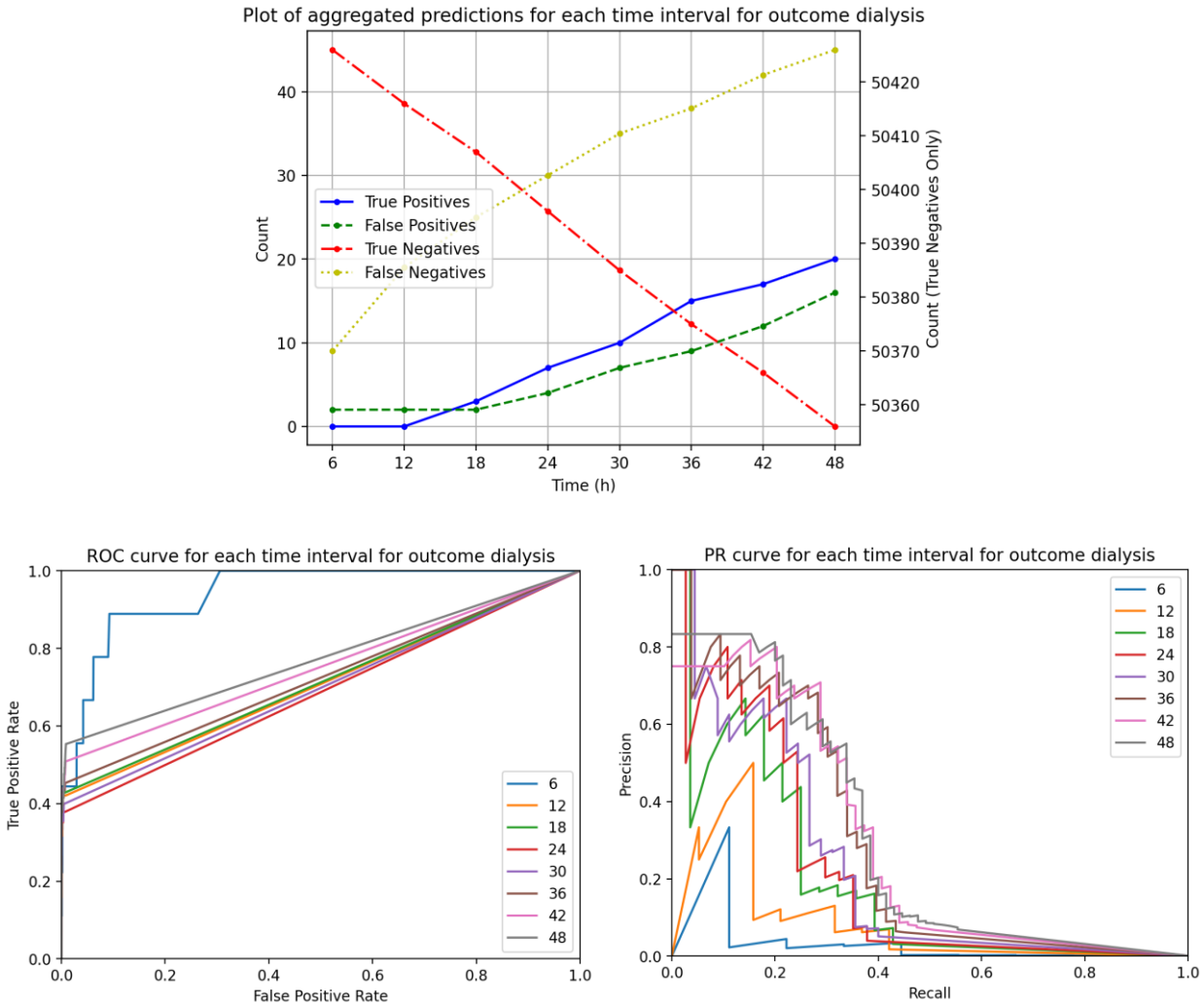


Figure 21: Performance plots for the Dialysis outcome for the baseline trained model.

E.3 EXPLAIN-ABILITY

E.3.1 OCCLUSION OUTPUT

The occlusion output from the baseline trained model (*without the "giveaway" fields masked*) provided the following top 100 fields of interest. The prefix "_sequence" indicates that the field is a sequence feature (rather than a context feature) relating to feature change events during a patient spell (rather than fixed in the spell context. It is not surprising that many of the included pathology tests and observations occur in the top of the table of significant values, as the influence of medications or other categorical variables are spread out in separate variables. Future investigations may benefit from also looking at the aggregated performance change when masking entire categories of features in groups.

Table 18: Table of occlusion output values for the baseline trained model (contains some giveaway fields as discussed)

Feature Field	Cross Entropy Difference from un-occluded value
sequence_bp_systolic	121932.72
sequence_bp_diastolic	80422.17

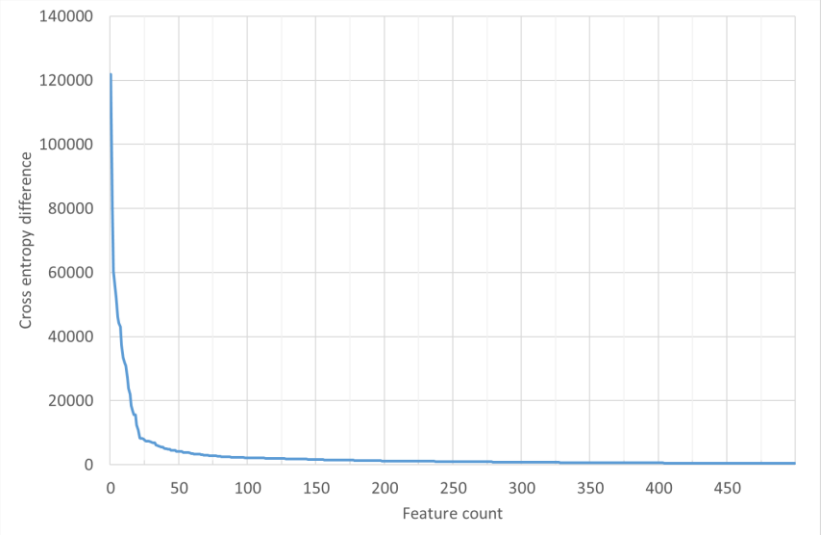
sequence_o2_sats	60446.47
sequence_test_result_sodium	55473.04
sequence_test_result_crea	52341.90
sequence_test_result_urea	45998.53
sequence_heart_rate	44177.54
sequence_rr	43012.67
sequence_test_result_potassium	37268.13
sequence_method_of_discharge_on medical advice	33437.52
sequence_test_result_calcium	31804.47
sequence_full_drug_name_antibacterial wash	30919.61
sequence_destination_on_discharge_home	27296.14
sequence_method_of_admission_a&e lri	23923.22
sequence_full_drug_name_sodium chloride 0.9% infusion	22063.42
sequence_full_drug_name_dalteparin sodium 5000units/0.2ml injection	18519.83
sequence_full_drug_name_paracetamol	16692.91
sequence_destination_on_discharge_nan	15604.84
sequence_method_of_discharge_died	15521.39
sequence_full_drug_name_amoxicillin 1g + potassium clavulanate 200mg injection	12358.17
sequence_full_drug_name_mupirocin 2% nasal ointment	10884.88
sequence_transfer_to_specialty_integrated med (acute care)	8372.24
sequence_obs_location_fcdu	8127.46
sequence_full_drug_name_morphine sulfate 10mg/5ml oral liquid	8117.64
sequence_full_drug_name_meropenem 1g injection	7979.06
sequence_transfer_to_ward_fcdu	7438.63
sequence_transfer_to_specialty_geriatric medicine	7426.56
sequence_test_result_chloride	7373.48
sequence_full_drug_name_macrogol 3350 13.125g/1sachet oral powder	7307.12
sequence_full_drug_name_anti-embolic stockings (aes)	7195.05
sequence_full_drug_name_morphine sulfate 10mg/1ml injection	6976.27
sequence_obs_location_ramu	6809.68
sequence_method_of_admission_emergency immediate	6779.47
sequence_full_drug_name_sodium lactate compound (hartmann ’s solution) 1000ml (2324) infusion	6220.00
sequence_full_drug_name_glucose 4% sodium chloride 0.18% infusion	6015.52
sequence_full_drug_name_furosemide 40mg tablet	5819.89
sequence_full_drug_name_snack	5701.78
sequence_transfer_to_specialty_cardiology	5456.89

sequence_transfer_to_ward_ramu	5450.99
sequence_transfer_to_specialty_general medicine	5166.59
sequence_transfer_to_specialty_respiratory medicine	4993.53
sequence_full_drug_name_sando-k effervescent tablet	4938.90
sequence_full_drug_name_magnesium 243mg oral powder	4902.57
sequence_full_drug_name_amoxicillin 500mg + potassium clavulanate 125mg tablet	4826.72
sequence_full_drug_name_sennoside tablet	4561.83
sequence_full_drug_name_fortisip compact liquid	4537.20
sequence_full_drug_name_midazolam 10mg/2ml injection	4496.13
sequence_transfer_to_specialty_general surgery	4488.93
sequence_full_drug_name_glucose 5% infusion	4186.65
sequence_full_drug_name_furosemide 50mg/5ml injection	4136.42
sequence_full_drug_name_aspirin 75mg dispersible tablet	4110.67
sequence_full_drug_name_co-amoxiclav	4108.40
sequence_full_drug_name_piperacillin 4g + tazobactam 500mg infusion	4100.01
sequence_full_drug_name_dalteparin sodium 2500units/0.2ml injection	3894.44
sequence_full_drug_name_bioxtra dry mouth oral gel	3820.02
sequence_full_drug_name_salbutamol sulfate 2.5mg/2.5ml nebuliser solution	3796.91
sequence_full_drug_name_sodium chloride 0.9% injection	3772.36
sequence_full_drug_name_lansoprazole 30mg gastro-resistant capsule	3765.43
sequence_full_drug_name_pabrinex intravenous high potency injection	3569.77
sequence_full_drug_name_fortisip compact protein liquid	3514.70
sequence_obs_location_racb	3418.76
sequence_full_drug_name_clopidogrel 75mg tablet	3385.75
sequence_full_drug_name_doxycycline 100mg capsule	3357.37
sequence_obs_location_refu	3334.15
sequence_full_drug_name_lactulose syrup	3286.19
sequence_full_drug_name_cyclizine	3279.19
sequence_obs_location_rafu	3167.71
sequence_full_drug_name_atorvastatin 40mg tablet	3058.52
sequence_full_drug_name_amlodipine 5mg tablet	3023.28
sequence_obs_location_guea	2943.94
sequence_full_drug_name_lansoprazole 15mg gastro-resistant capsule	2935.40
sequence_full_drug_name_sodium picosulfate 5mg/5ml oral liquid	2926.19
sequence_full_drug_name_sodium lactate compound (hartmanns solution)	2826.58

sequence_transfer_to_ward_rsau	2815.84
sequence_full_drug_name_oxygen	2808.15
sequence_obs_location_r15	2803.63
sequence_obs_location_rsau	2777.03
sequence_obs_location_r07	2740.59
sequence_full_drug_name_ondansetron	2693.06
sequence_transfer_to_ward_racb	2641.19
sequence_full_drug_name_bumetanide 1mg tablet	2598.36
sequence_full_drug_name_antiembolic stockings (aes)	2495.22
sequence_full_drug_name_adcal -d3 chewable tablet	2437.00
sequence_full_drug_name_furosemide	2424.75
sequence_full_drug_name_dalteparin	2421.28
sequence_full_drug_name_antibacterial hairwash	2418.58
sequence_full_drug_name_vitamin b co strong tablets (actavis)	2400.66
sequence_full_drug_name_dexamethasone	2394.91
sequence_full_drug_name_sodium chloride 0.9% nebuliser solution	2364.99
sequence_obs_location_r23	2361.06
sequence_full_drug_name_ranitidine 150mg tablet	2352.94
sequence_obs_location_g28	2336.64
sequence_obs_location_r42	2331.00
sequence_full_drug_name_folic acid 5mg tablet	2320.98
sequence_full_drug_name_actrapid insulin vial 100 units/ml injection solution	2316.60
sequence_method_of_admission_emergency bed bureau	2293.02
sequence_full_drug_name_bisoprolol fumarate 5mg tablet	2290.59
sequence_full_drug_name_lansoprazole gastroresistant capsules	2270.12
sequence_transfer_to_ward_gstu	2206.59
sequence_full_drug_name_spironolactone 25mg tablet	2202.37

E.3.2 Field Significance Plot

When plotting cross entropy differential from the un-occluded value for each feature, it can be seen that the most significant individual fields lie in roughly the top 100-500 fields out of the approximately 13000 fields provided to the model. Many of these fields are medication field values encoded categorically. The 99.8^{th} percentile is at a value of 7339. This covers roughly the top 25 fields. The 99^{th} percentile covers the top 137 fields.



APPENDIX F FURTHER RECOMMENDATIONS

Expansion of lessons learnt and future work with more of the technical detail.

F.1 FUTURE WORK

Corresponding to the areas covered in Section 4.2, further detail on each point is covered below.

- Increase granularity of feature categories
  - The full list of approximately 13,000 features were each assigned a feature category, to give the model explicit information that certain sets of features were likely more related than others, e.g. drug prescriptions all belonged to one category, transfers between particular wards were one category, etc. It may be worthwhile to break down some of these categories further, e.g. currently several different dosages of the same drug will belong to the same category as different dosages of a separate drug. By separating the drug category further, so that all dosages of one drug are allotted their own category, it may make it simpler for the model to draw correlations when a particular type of drug is administered. Though currently each drug also receives its own unique field label that the model will observe, and hence it is possible the model would learn groupings of different dosages of the same drug anyway, if rare dosages or custom notes in the field name exist then it is likely the model will not learn to handle these as being similar to more generic dosages of the same drug, and instead will only know 'some drug was administered'.
- Include lab predictions task
  - The DeepMind group's original set of objectives included the prediction of some continuous-valued laboratory test results, which was found to improve performance by a few percent, even though the task was auxiliary to the main objective of predicting adverse outcomes. Enhancing the data conversion pipeline for this project such that similar continuous-valued fields become present in our label features would enable a similar approach—this feature of the project was only de-scoped due to time constraints, it should be possible to extend the pipeline with the dataset as it existed for this project. The model was left with placeholder continuous-value fields of the form 'lab_x_in_nh', where 'x' would be the lab result of interest and 'nh' would be an integer number of hours forward in time that the label related to, so much of the tensor construction from our data loader should handle this stage if appropriate field names are changed within the JsonDMBatchGenerator class. An additional labs_task would also have to be included in the shared_config_kwargs within _get_config methods of training experiment entry points.
- Try transformers as the recurrent cell
  - Shortly before the DeepMind results were published, transformer-based neural networks were developed (Vaswani, 2017), and the transformer, as a building block of DL neural networks, has become popular as a replacement for other recurrent layers such as the UGRNN, LSTM and SRU layers used in this work (Wen, 2022). While the earlier layer types attempt to prioritise which information from earlier time points is passed forward to later cells in varyingly complex ways, transformer layers treat all time points prior to the window of prediction equally, meaning that they are likely to handle very long patient stays more capably than cell types used in this work. Including transformers as a cell choice in the current framework would have been quite challenging, but more modern DL libraries would make rebuilding with transformers as an option much more feasible.
- Find improved methods for dialysis labelling (avoid conflation of previous and current ICD codes)
  - Dialysis events in this project were labelled via the combined detection of a patient moving to a renal ward, and the ICD code for dialysis appearing in their metadata. There is a slight flaw in this approach, in that a patient may have received dialysis during a previous visit but not the current visit, and are sent to the renal ward on the current visit. In this case, a dialysis event would be spuriously labelled as occurring. Incorrect labelling will, of course, negatively

impact performance in a supervised learning model, both in real-world predictive ability and validity of performance metrics after training—a more robust dialysis event labelling method would then likely improve both of these.

- Find improved methods for mortality (better granularity time stamping)
    - Mortality events are labelled as occurring at the end of a day in the current dataset, regardless of when they actually occur. This will lead to an undesirable punishment of the model if it predicts death to occur at any time window not at the end of a day. While the model may have some capability of understanding time of day with the current encoding of time, and hence learn that mortality events should only be predicted in the last time window of a day, this is clearly contrary to the optimal use case in real-world scenarios—accurate forecasting of the actual time window where death occurs. More accurate time stamping of when mortality events occur would be more suitable to the architecture of the model for training, and more useful for real-world usage.
- Re-build tooling for modern ML framework
    - The DeepMind codebase was created using TensorFlow 1.15, with various support libraries designed to improve usability of this version of TensorFlow. The codebase was completed shortly before a major update to TensorFlow that overhauled a great many features, as well as PyTorch growing in popularity as a DL framework. Little to no development by the DL community is now done in TensorFlow 1.15, and extending the DeepMind tooling incurs considerable time and difficulty compared to equivalent operations in a more modern framework. If any further changes to the DL model architecture are desired for future work, the entire model should be redefined in a modern DL framework.
- Improve efficiencies in batch building and encoder models
    - In DL, CPU processing is often used to build batches of data from files on disk before sending to the GPU for more complex operations. This typically leads to a considerable speed-up in training, with 10-100x being common. The DeepMind codebase handles an extended series of data transformations on the CPU before transferring to GPU, likely leading to a bottleneck that meant only a minor speed up was observed between CPU and GPU training. If an overhaul of the codebase is performed, ensuring this bottleneck is avoided would be a priority item.
    - The codebase creates a separate encoder for each context item in a patient's metadata, in addition to one for the sequence data of a patient's stay. Little time was available in the project to experiment with independently resizing these encoders, and it is likely the number of parameters available in the model for each context item is far beyond requirements. Though the encoders reflect a relatively small portion of the model, designing with this independent encoder handling in mind would be worthwhile during a rebuild in a modern DL framework.
- Expand hyperparameter tuning scope
    - If training efficiencies are achieved via reduced CPU data handling, the breadth of hyperparameters realistic to be tuned is increased simply due to more model training iterations per unit time. Additionally, as a higher dimensionality of hyperparameters are explored, techniques more advanced than those used in this project would prove beneficial. Bayesian optimisation is a common neural architecture search method that remains viable on limited compute resource, for instance, and balances the search in hyperparameter space between unknown regions and regions with high performance (Frazier, 2018).
- Ability to ingest data from a more automated data source (database/data lake)
    - There was unexpected complexity in gathering data from various sources prior to handover to the Roke team. It would be valuable to standardise the data gathering approach before attempting to rollout any tool to test-bed users. If the data gathering could become periodic and automated, the model could be routinely updated as new data became available. This may be as simple as any new data fitting under the existing fields being used to expand the training set, or include automated retraining with inclusion of new drugs and other events as they become available in the ever-evolving medical field.

- Separation of primary care/hospital co-morbidity data (and potential to extract timestamps for new coding during stay).
    - It is understood after discussion with the UHL team that it should be possible to extract ICD codes from primary care and hospital co-morbidity data, leading to the desired separation between ICD codes relevant to a patient's prior medical history and codes generated during their current stay. Improved timestamping of these codes should also be possible via this route, which could allow ICD codes to be used within the sequential features as well as context features.
- Expansion of dataset
    - While the dataset for this project contained approximately 20,000 records, which is typically a viable quantity for DL tasks, there was a severe imbalance between positive and negative cases for each of the three outcomes—at a 5% validation split, only a few tens of cases existed for ITU admissions and dialysis outcomes for the purpose of measuring model performance, for instance. It is unknown how varied the presentation of clinical clues to the various outcomes may be that the DL model may discover, but it is possible that with this few validation cases that disparate presentations are occurring in the validation set that were not present in the training set. Increasing the number of positive adverse outcome cases in the training set should mitigate risk of this occurring, i.e. cause the training data to have more complete coverage of possible real-world scenarios.
    - Further pathology tests (i.e. infection markers)
    - The data related to pathology tests was limited during this project due to difficulties in exporting a complete list, but it is understood that much more extensive data exists. In particular, blood test markers related to infection were expected by the UHL team to be of high importance in a future expanded dataset.
- Limited time in current commission to extract the volume of pathology data available. Would like to improve this in future to expand the dataset.
    - The sheer quantity of pathology data also limited the number of tests that could be included during this project due to export time. If a complete list of desired pathology data were generated, and a data extraction exercise performed without pressure from a parallel model development effort, the looser time constraint on data export would mean a higher number of features could be used.
- Loss function bespoke to our performance metric/time window/outcome of choice
    - The model training procedure aims to minimise the cross-entropy between predictions and true adverse outcome labels, with equal weighting across all time windows and adverse outcomes. Late in the project, a geometric mean of sensitivity and positive predictive value at the 24h prediction window for mortality only was agreed upon as the most important metric for model selection. A loss function that measures this quantity batch-wise during training could optimise the model directly for this more specific metric, or a reweighting of the loss across the various time windows/outcomes could lend higher priority to this outcome-time window combination, while retaining some focus on others. In either case, it is likely that adjusting towards the prioritised metric during training would yield improved performance in said metric.

This page intentionally left blank

Roke Manor Research Ltd
Roke Manor, Romsey
Hampshire, SO51 0ZN, UK

T: +44 (0)1794 833000
F: +44 (0)1794 833433
info@roke.co.uk
www.roke.co.uk

Approved to BS EN ISO 9001
Reg No Q05609