INFOB3APML: Assignment 2

Group 2

Aksel Can Sozudogru, Coby Simmons Lorenzo Marogna Minna Vainikainen
Tomás Tavares

December 2023

Abstract

In this comprehensive analysis of hospital patient flow, this report investigates various regression algorithms and assess their performance in modeling the remaining time for a trace. Specifically, it delves into the exploration of last-state and aggregation encoding methods. Subsequently, the document details the model selection process, focusing on the Regression Tree (RT), k-Nearest Neighbours (kNN) regression, and Random Forest (RF) regression algorithms. Finally, the report concludes with an evaluation and comparison of all the methods. The findings indicate that the RF algorithm with last-state encoding outperforms the other models in the key Mean Absolute Error metric.

1 Introduction

Process models are used by organizations to document and analyze their processes. A process model describes the behavior of a process, which consists of related activities performed in a repeatable manner to achieve a goal. Specifically, predictive process modelling involves the forecasting of future outcomes of process cases using their partial event logs as an input to a model trained on the event logs of complete cases (Teinemaa et al. 2018).

The area of predictive process monitoring has gained significant interest in recent years, motivated by a growing ability to handle large volume and velocity of data, as well as the significant productivity improvements that can be gained by predicting the course of a business process.

In Mannhardt and Blinde (2017), data was collected and enriched such that an event log was produced for the trajectory of patients with symptoms of sepsis. It was shown that the patient flow in a hospital can be represented using process models. Using a subset of this event log, this research aims to predict the remaining time in a trace for a case, given the data of a single event (possibly enriched with data from previous events in a trace). This is equivalent to predicting a patient's remaining time in hospital.

Given the ability to accurately predict a patient's remaining time, hospital's might be able to optimise their processes so that they can operate at a higher level of productivity.

2 Data

The dataset is a subset of the data collected in Mannhardt and Blinde (2017). This dataset comprises a comprehensive collection of real-life events associated with sepsis cases in a hospital setting. It captures the intricate details of patient pathways through the hospital, tracked by the hospital's Enterprise Resource Planning system.

The subset used contains the event where the activity is between ER Registration and ER Sepsis Triage, including these activities. This subset is significant in its coverage, logging 6 different activities for approximately 1000 cases - resulting in approximately 5200 events.

The dataset has 37 features which fall broadly into eight categories:

- **Identifiers:** Used to identify the subject, not used as features for prediction.
- **Activity:** The activity type associated with an event.
- **Timestamps:** The timestamp of an event took place and other features that are derived from this (e.g., `month`, `weekday`, ...). To protect the anonymity of subjects, the timestamps of events were randomised, although the time between events within a trace was not changed. Thus, the only use for the timestamp features is in engineering the duration features, so the timestamp features can be discarded once this takes place.
- **Durations:** The seconds elapsed since the first event (`elapsed`), the seconds remaining until the final event (`remtime`), and the minutes between the current event and the last event.

	Age	CRP	LacticAcid	Leucocytes
count	5174	5174	5174	5174
mean	70.29	113.91	2.09	13.84
std	17.22	87.30	1.16	12.89
min	20.00	5.00	0.40	0.20
25%	60.00	50.00	1.50	9.40
50%	75.00	113.91	2.09	13.40
75%	85.00	136.00	2.09	15.30
max	90.00	573.00	11.00	296.20

Table 1: Summary statistics for numeric variables

- **Binary:** Various checkbox criteria from, for example, the triage document or the Systemic Inflammatory Response Syndrome (SIRS) criteria.
- **(Multi-)Categorical:** The only non-binary categorical feature is `Diagnose`, with 57 unique values.
- **Numeric:** The subject's age transformed into bins of width 5 (`Age`), and numerical measurements taken from the patient, including leucocyte count (`Leucocytes`), C-reactive protein (`CRP`), Lactic acid level (`LacticAcid`)

These attributes provide a detailed view into the medical assessments, treatments, and clinical criteria relevant to sepsis cases, making this dataset a valuable resource for research in medical informatics, patient care optimization, and hospital process management.

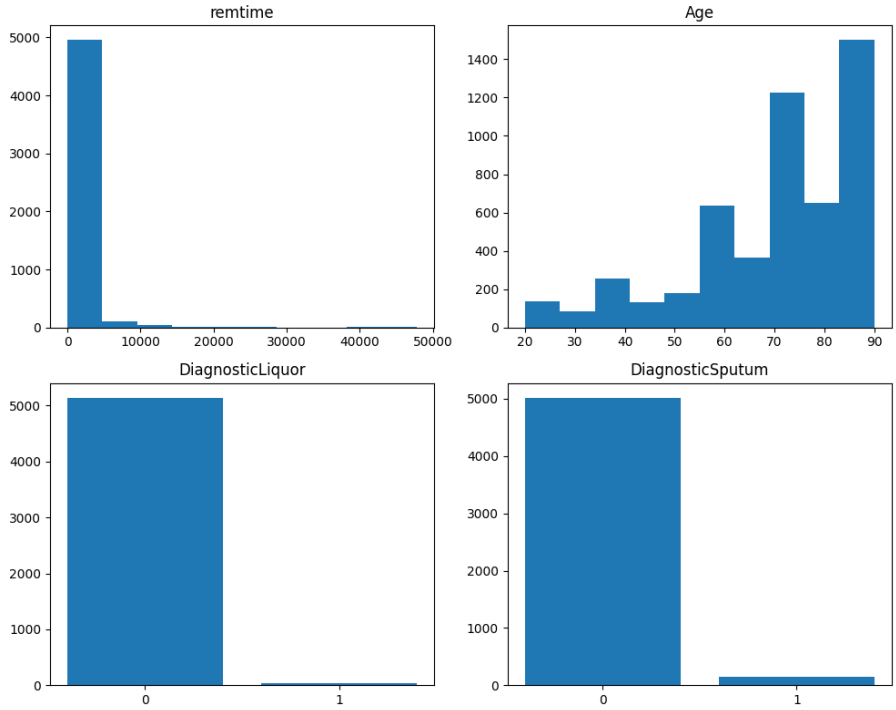

Figure 1: Univariate distributions

To explore the data further, simple summary statistics for the numeric variables were calculated, as seen in Table 1. The distributions for all binary, categorical, and numeric variables were also visualised. In Figure 1, it can be observed that age is right-skewed, which is to be expected, since infection and subsequent sepsis is more likely to occur in older people. Most of the (binary) diagnostic features were skewed towards "false". For each of the SIRS criteria, more subjects recorded "true" than "false" and most patients recorded "true" for having a suspected infection and requiring an intravenous infusion. The features storing diagnostic measurements (`CRP`, `LacticAcid`, `Leucocytes`) were left-skewed, but this is likely because missing values are encoded as 0 for these features. Every feature (except Activity) had missing values.

Furthermore, it can be seen in Figure 2 that the remaining time attribute (i.e., the dependent variable) has a large number of upper outliers. This is important to consider during the model selection phase of the research.

Additionally, a correlation heatmap was plotted (Figure 3). The diagnotic features were quite highly correlated with each other, which makes intuitive sense, since typically a number of diagnostic tests are ordered

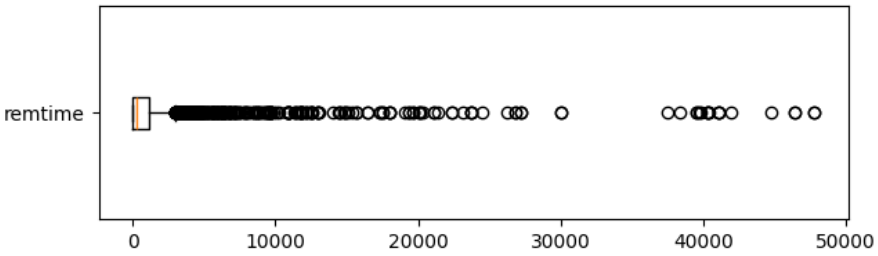

Figure 2: Boxplot of remaining time feature

at one time. Similarly, the SIRS features were highly correlated with each other. The SIRS features were also highly correlated with some diagnostic features, and from both sets of features, there were a number that were highly correlated with the infection suspected and infusion required features. To improve the readability, the correlation heatmap in the figure is only reporting feature with remarkable correlation with some other feature.

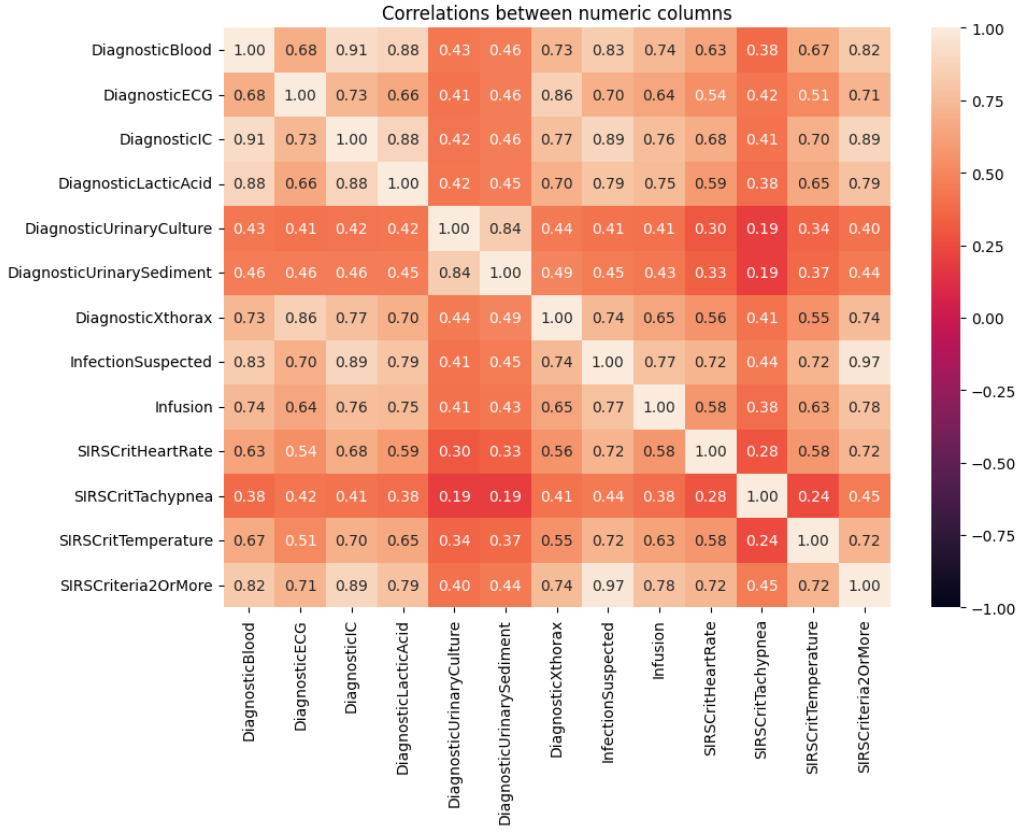

Figure 3: Correlation heatmap

When splitting our data between the train and test sets, we ensured that records of the same case are kept within the same split. If records from the same trace are included in both the train and test sets, then the model might perform well on the test set, but might not perform well when exposed to new data (i.e., the model will be overfit). The size of the training set is the 80% of the whole dataset, ensuring a sufficient amount of instances both for training and accurate testing.

3 Method and Experimental Setup

3.1 Data Pre-Processing & Feature Selection

For this dataset, the pre-processing was heavily focused on removing missing values. For all the binary and multi-categorical features, the values were the same throughout all events in a trace (except for missing values). Thus, missing values were filled with the consistent values for all other events in the trace. There were two cases where all values within the trace were missing, so these were dropped from the dataset. For numerical features, the values changed throughout the trace. These missing values are filled with the next available value

within a trace. Following this, some values for the last event in a trace were still missing. These missing values were filled with the most recent available value within a trace. For numeric values that were still missing (i.e., when a particular type of measurement was not taken within a trace), the mean value for that attribute was taken to fill missing values.

Additionally, to prepare the data for regression, the binary columns were encoded as 0 and 1, and the `Diagnose` column was one-hot encoded.

It was also required to encode the activity column as a binary feature vector, as proposed in Teinemaa et al. (2018). Two approaches were taken:

- **Last (2) State Encoding:** For each event, the current activity and the most recent activity are one-hot encoded.
- **Aggregation Encoding:** For each event, the current activity is one-hot encoded and the the cumulative sum of the feature vectors is taken over the elapsed time within a case.

3.2 Model Selection

Given the supervised nature of this prediction task, and the fact that the dependent variable is numeric and continuous, this research focuses on regression algorithms. For each model, a grid search of paramters was performed in order to choose the most optimal model. Additionally, we also employed 5-fold cross-validation on the grid search to ensure the robustness of the models. Given the size of the split for our set (80% - 20%), the cross-validation models will be trained on a training set that corresponds to the 80% of the 80% and validated on the remaining 20% of the 80%.

3.2.1 Regression Trees

Regression trees (RTs) are a variant of decision trees used for predicting continuous variables. RTs operate using recursive binary splitting to minimize variance within nodes. Each internal node denotes a decision rule on a feature, leading to leaf nodes that output numerical predictions.

A key benefit of RTs is that they provide clear and intuitive decision rules, making them easy to understand and interpret, even for non-experts. Additionally, RTs can capture and predict non-linear relationships without any transformation or preprocessing and do not make any assumptions about the underlying data distribution (i.e., they are non-parametric). On our dataset, regression trees were found to be computationally efficient, with training taking no longer than a couple of seconds.

However, it should be noted that RTs are prone to higher bias and variance than other methods, given that it is a single model rather than an ensemble of models. Part of this is due to the fact that numerous features or complex trees can result in overfitting.

To start with, a naïve RT was trained with the default parameters as a baseline. The default parameters do not set a maximum depth, leading to a very complex tree with a depth of 31. A complex tree can often lead to overfitting and poor interpretability for the model, as well as increased computational intensity in prediction. Thus, it was decided to tune the `max_depth` and `min_samples_per_leaf` parameters using a grid search with the following ranges:

- max_depth: $\{1, 2, 3, ..., 8, 9\}$
- min_samples_leaf: $\{1^2, 2^2, ..., 12^2, 13^2\}$

3.2.2 kNN Regression

K-Nearest Neighbors (kNN) regression is a non-parametric method used for predicting continuous variables by averaging the values of the k nearest neighbors. 'Nearest' is defined using a distance metric like Euclidean distance (i.e., the distance between points in an n-dimensional space).

A notable advantage of kNN regression is its simplicity and effectiveness in cases where the relationship between variables is complex and not easily captured by parametric models. Since it does not make any assumptions about the underlying data distribution, kNN can be particularly useful in scenarios where the data is highly irregular. On our dataset, kNN regression provided a flexible approach, easily adjusting to varying patterns in the data. For these reasons, the use of this algorithm is well-suited for our dataset, and our hypothesis is that it will perform well in this specific use case.

It is essential to note that, despite its advantages, this algorithm has its drawbacks. It can be computationally expensive and slow, requiring storage of the entire dataset and calculations of distances between all pairs of points. However, our dataset seems to be small enough to avoid any major computational inefficiencies.

Sensitivity to noise, outliers, and irrelevant features, along with susceptibility to the curse of dimensionality in high-dimensional spaces, such as our dataset, are among its limitations.

Moreover, the choice of k and the distance metric can significantly impact performance, requiring careful tuning. kNN is also sensitive to the scale of the data, often necessitating feature scaling for optimal performance. Similar to regression trees, we initially trained a naive model using default parameters — 5 neighbors and uniform weights — serving as our performance baseline. To identify the optimal model for our dataset, we fine-tuned the number of neighbors and weight parameters within a specified parameter space:

- n_neighbors: $\{2^0, 2^1, ..., 2^{\lfloor \log(\text{max\_neighbors}) \rfloor}\}$
- weights: [`uniform`, `distance`], where "uniform" weights all neighbours equally and "distance" weights all neighbours based on distance.

3.2.3 Random Forest [Bonus]

Random Forest (RF) regression is both a supervised learning and ensemble algorithm. It is supervised in that it learns the mappings between inputs and outputs during training. Ensemble algorithms combine multiple machine learning algorithms to make more accurate predictions than any underlying algorithm could on its own.

It has been found that ensemble algorithms perform better than their singular counterparts. This is based in the on the common societal practice of the 'seeking a second opinion' (Polikar 2006).

RF regression creates an ensemble of RTs, each created by taking random samples of the datasets features and observations. When provided a new instance to predict, each of the RTs in the RF make a prediction, and then the outcomes are averaged to produce the RFs final prediction.

RF is capable of reducing the correlation between trees through bootstrap aggregation (bagging) which helps avoid overfitting. Additionally, RF can handle large volumes of data, including biased datasets. The ensembling of trees enhances RF's performance, accuracy, and fitting ability. By averaging across trees, RF becomes a robust algorithm capable of dealing with noisy and imbalanced data. This is particularly relevant for our dataset, since the dependent variable (remaining time in process) has a large number of upper outliers.

Therefore, RF is particularly effective with large datasets, especially when dealing with numerous categorical independent variables.

Initially, a naïve RF was trained with the default parameters as a baseline. The default parameters do not set a maximum depth, leading the nodes to be expanded until all leaves are pure or until all leaves contain less than `min_samples_split` samples which can result in an over-complicated tree. More complex models are often not better, and simpler models usually generalize better to new data. Thus, it was decided to tune the `max_depth` and `min_samples_leaf` parameters using a grid search with the following ranges:

- max_depth: $\{1, 6, 11, ..., 91, 96\}$
- min_samples_leaf: $\{1, 3, 5, 7, 9\}$

Additionally, a variant of our RF model was trained on data that had undergone feature reduction. Focusing on the most relevant features can lead to more efficient, interpretable, and accurate models. However, it is important to apply feature reduction judiciously to avoid losing valuable information that could be pivotal to the learning process. As seen in Figure 4, the `Diagnose` feature does not have much importance, so this feature was dropped.

4 Evaluation & Discussion

We chose four standard metrics to evaluate our models.

- Mean Absolute Error (MAE). The average error of predicted values $\hat{y}_i$ compared to actual values y_i, independent of the direction of error.

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

- Mean Squared Error (MSE). The average of squared error of predicted values compared to actual values. Squaring results in an understimation of the model in the case of outliers, large errors are amplified.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Root Mean Squared Error (RMSE).

- R^2. Highlights the proportion of variance in the dependent variable that can be explained by the independent variable.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{MAE(\text{model})}{MAE(\text{baseline})}$$

For MAE, MSE and RMSE, a lower score is better. For R^2, a higher value is better. It should be noted that the R^2 model shows how well the model fits the training data. Thus, it is not a reliable indicator of whether a model has been overfit. Additionally, MSE and RMSE tend to amplify the large errors that might arise in the case of outliers in the class variable. The remaining time variable in our dataset has a large number of upper outliers. Thus, we have chosen MAE as the key metric for parameter tuning.

4.1 Regression Trees

Encoding	Min. Samples/Leaf	Depth	MAE	MSE	RMSE	R^2
Last-2-State	31	16	1091.25	6826311.97	2612.72	0.52
Last-2-State	1	40	1194.94	14126716.30	3758.55	0.01
Aggregated State	31	16	968.51	6152153.47	2480.35	0.57
Aggregated State	1	45	1242.76	16765261.83	4094.54	-0.18

Table 2: Regression Trees: Error metrics for different parameter configurations

The type of encoding has an influence in the quality of the results. On one hand, the aggregated state encoding can summarize all prior events effectively, but it does not provide the order in which said events happened. On the other hand, the late-2-state encoding focuses more on the temporal sequence of events but only stores the two most recent events.

For the RT model, the aggregated state encoding is performing better, perhaps because it results in less features than the last-2 state encoding method, and decision trees perform better with low-dimensional data.

Additionally, it seems that increasing the minimum samples per leaf node parameter prevents the tree from becoming too complex, thus, increasing the generalizability of the model to the test data.

Regarding the maximum depth of the tree, it can be seen that the lack of a limit in the depth led the default model to grow to significant depth. A deep tree can be an indicator of overfitting, since the tree is grown to fit the training data very accurately, but cannot generalise well to new data.

In figure 4 it is possible to observe how the model reacts to a different encode of the data. While some aspects like the kind of activity and the Leucocytes' level are important for both models, the elapsed time loses it's importance with the aggregation encoding.

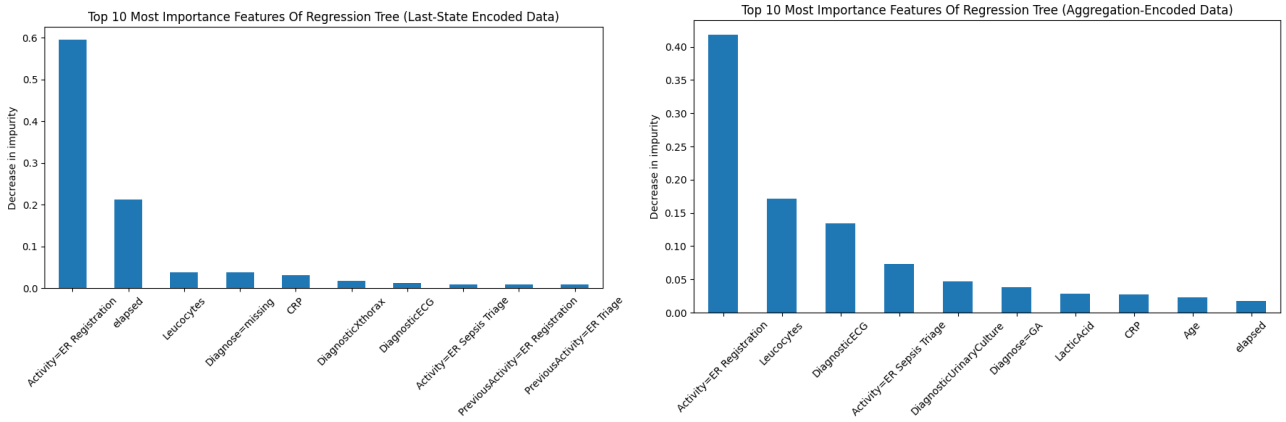

Figure 4: Feature importance for regression trees

4.2 kNN Regression

To find the best model parameters for kNN Regression, we fine-tuned the number of neighbors and weight parameters, as specified in Section 4.2, using grid search. The obtained results are summarized in Table 3.

A small value of k means that the model is more sensitive to noise and outliers, while a large value of k means that the model is more smooth and stable, but may have high bias. Ideally, a good value of k is usually

Encoding	Weights	Num. Neighbours	MAE	MSE	RMSE	R^2
Last-2-State	distance	256	1302.90	13150102.30	3626.31	0.08
Last-2-State	uniform	5	1429.42	15083063.84	3883.69	-0.06
Aggregated State	distance	256	1302.75	13148969.96	3626.15	0.08
Aggregated State	uniform	5	1421.93	15017276.74	3875.21	-0.06

Table 3: kNN Regression: Error metrics for different parameter configurations

somewhere in between, resulting in a model that effectively balances the bias-variance trade-off. For both Last-2-State and Aggregated State encoding the same parameters are said to be optimal: number of neighbors as 256 and distance based weights. This contradicts the intuition of finding a value 'somewhere in between' as 256 falls on the higher end of the k range. Although counter-intuitive the use of a high k value, it is an effective strategy when paired with distance based weights.

By looking at the obtained results we can conclude that the kNN Regression algorithms is ill-suited for the task at hand given its poor performance. This can be attributed to the lack of normalization, which may cause some variables to have more influence than others on the distance calculation. To investigate this, we conducted an experiment by normalizing the data using Min-Max Scaling. The results showed a slight improvement in the model's MSE, RMSE, and R^2, but not in MAE. This indicates that while normalization plays a role in performance, it is not the sole factor contributing to the observed issues. Moreover, we are working with a high-dimensional dataset augmented with either Last-2-State or Aggregated State encoding, further increasing the number of dimensions of the data. As explained in Section 4.2, one of the weaknesses of kNN regression is its poor performance under high-dimensional datasets. This characteristic could be the primary cause of the algorithm's unsatisfactory performance.

4.3 Random Forest [Bonus]

As mentioned in Section 3.2.3, we fine-tuned our parameters for Random Forest Regressor model to decrease the error scores, which the results are shown in Table 4.

Encoding	Feature Sel.	Max. Depth	Min. Samples Leaf	MAE	MSE	RMSE	R^2
Last-2-State	False	None	1	1094.84	6820378.09	2611.59	0.52
Last-2-State	False	11	7	963.70	5462739.26	2337.25	0.62
Last-2-State	True	21	1	1177.39	8857465.39	2976.15	0.38
Aggregated State	False	None	1	1023.83	7350108.45	2711.11	0.48
Aggregated State	False	21	1	1001.87	6695898.60	2587.64	0.53
Aggregated State	True	21	1	1001.87	6695898.60	2587.64	0.53

Table 4: Random Forest Regression: Error metrics for different parameter configurations

As seen in the table, we not only tuned our `max_depth` and `min_samples_leaf` parameters but also, implemented feature selection with the aim of decreasing error and improving efficiency in the model. For Last-2-State encoding, enabling feature selection resulted worse model performance. Thus, the `Diagnose` feature might have contained useful information for the model. In contrast, for Aggregated State encoding, feature selection had no impact on performance metrics, as evidenced by identical error values in the corresponding configurations. This might be an indicator of the fact that when Aggregated State encoding was implemented without feature selection, the model did not use the `Diagnose` feature anyway.

The best results for both encoding methods are achieved through parameter tuning. As the table shows, the best result for Last-2-State encoding is achieved when `max_depth` is 11 and `min_sample_leaf` is 7. These findings lend further evidence to our previous assumption that more complex trees result in overfitting. In the future, it might be useful to allocate more time and computing power to parameter tuning for RF in order to further improve the model.

4.4 Performance Comparison

Table 5 provides a summary of each parameter-tuned model for each encoding. During this research, MAE has been the metric of choice for evaluating a model during the parameter tuning, as it corresponds to the average error of every instance in the test set with the predicted value, independently from the sign and thereby contributes to a great interpretability. By comparing the MAE for aggregated state encoding of the models, we can see that the Random Forest Regression was the best performing model.

For the aggregated state encoding, the RF regression performs better in cross-validation but worse on the training data, relative to the RT method. This possibly indicating overfitting. Since we would have expected

RF to perform better than RT (given that RF is an ensemble of RTs), we see an opportunity to improve the model with more paramter tuning.

As mentioned in Section 4.2 points out, kNN is not a suitable model for this task, since the optimal parameters contradict the intuition that the k value should not be too large. Additionally, kNN handles outliers poorly, and our class variable contains a large number of outliers. This is reflected in the comparison result, where kNN Regression has relatively to the other models the highest error and an R^2 score that indicates a bad fit of the data.

When evaluating the result of last-2-state encoding Random Forest has the lowest error for both cross validation and the test data. This model also has the highest R^2 score (0.62), which indicates that it is the model that best fits the data. The best performing model for predicting remaining time on our dataset showed to be Random Forest regression with last-2-state encoding. One possible justification for this might be the fact that, during training, Random Forest builds multiple RTs that each take a sample of the available features. Thus, RF can effectively handle the large number of features created during the last-state encoding process.

5 Conclusion

The result was obtained by comparing the three regression algorithms´ error measures in terms of MAE, RMSE, MSE and R^2. It was found that the best performing model was trained with the Random Forest Regression algorithm using last-2-state encoding. Random Forest builds multiple RTs that each take a sample of the available features. Thus, RF can effectively handle the large number of features captured in last-state encoding. For Regression Trees, we found that the lack of a limit on the maximum depth of the model led to overfitting and poorer performance. kNN Regression showed to not be suitable for this task, since the result contradicts with the intuition of the k value. The low model performance of kNN Regression can further be attributed to the lack of normalization, which may cause some variables to have more influence than others on the distance calculation.

Different models tried to predict and some of them performed better than others. Unsurprisingly, parameter tuning helped to achieve better results, as it tests a large number of models, and chooses the optimal one for the whole dataset. In the future, we might be able to invest more time and computing power in order to test more parameters and a greater range for each parameter, which might lead to improvements in our models.

References

Mannhardt, F. and D. Blinde (2017). "Analyzing the trajectories of patients with sepsis using process mining". English. In: *RADAR+EMISA 2017, Essen, Germany, June 12-13, 2017*. CEUR Workshop Proceedings. RADAR + EMISA 2017 ; Conference date: 12-06-2017 Through 13-06-2017. CEUR-WS.org, pp. 72–80.

Polikar, R. (2006). "Ensemble based systems in decision making". In: *IEEE Circuits and Systems Magazine* 6.3, pp. 21–45. DOI: 10.1109/MCAS.2006.1688199.

Teinemaa, Irene et al. (2018). *Outcome-Oriented Predictive Process Monitoring: Review and Benchmark*. arXiv: 1707.06766 [cs.AI].

Appendix

Encoding	Model	Mean CV MAE	Mean CV RMSE	Mean CV MSE	Mean CV R2	Test MAE	Test RMSE	Test MSE	Test R2
Aggregated state encoding	Random Forest Regression	1125.34	2902.86	8661919.65	0.26	1001.87	2587.64	6695898.60	0.53
Aggregated state encoding	Random Forest Regression (Feature Selection)	1125.34	2902.86	8661919.65	0.26	1001.87	2587.64	6695898.60	0.53
Aggregated state encoding	Regression Tree	1163.61	3141.50	10079416.64	0.12	968.51	2480.35	6152153.47	0.57
Aggregated state encoding	kNN Regression	1390.44	3625.60	13538864.93	-0.19	1302.75	3626.15	13148969.96	0.08
Last-2-state encoding	Random Forest Regression	1159.89	2948.73	8977721.77	0.23	963.70	2337.25	5462739.26	0.62
Last-2-state encoding	Random Forest Regression (Feature Selection)	1233.51	3100.58	9831496.35	0.15	1177.39	2976.15	8857465.39	0.38
Last-2-state encoding	Regression Tree	1236.68	3234.85	10677223.27	0.07	1091.25	2612.72	6826311.97	0.52
Last-2-state encoding	kNN Regression	1390.95	3626.67	13546427.14	-0.20	1302.90	3626.31	13150102.30	0.08

Table 5: Summary of results

Group Members & Contributions

- **Coby Simmons:**
  - Code: Tasks 1 & 2
  - Report: Sections 2, 3.1
- **Lorenzo Marogna:**
  - Code: Task 3
  - Report: Sections 3.2.1, 4, 4.1 and Abstract
- **Tomás Tavares:**
  - Code: Task 4
  - Report: Sections 3.2.2, 4.2
- **Minna Vainikainen:**
  - Report: Sections 1, 4.4, 5
- **Aksel Sozudogru:**
  - Code: compilation of work into submission notebook, bonus task
  - Report: Section 3.2.3, 4.3