Predicting Incoming Hospital Patients Using Weather Data and Neural Networks

Tofte Tjörneryd
Predicting Incoming Hospital Patients Using Weather Data and Neural Networks

Tofte Tjömeryd

Abstract

Personnel shortages in hospitals are a problem, and one way to alleviate that is to make better use of the existing personnel. This thesis examines whether Intensive Care patient arrivals can accurately be forecast using machine learning models, as well as examining the usefulness of calendar and climate data for the task. The two models used in this study were the Temporal FusionTransformer (TFT) and NeuralProphet (NP). The models were trained using hospital statistics from the city of Gävle, Sweden, as well as climate data from the same region. Results showed that calendar data such as weekday and the week of the year had highest importance on the forecast, but temperature, and specifically the lowest recorded temperature, were also contributing to the forecast. The TFT model forecasts were significantly better compared to the NP model, a mean absolute error of 0.5053 compared to 0.8322. This is in relation to data were the vast majority of target variable values ranged between 0 and 3. This shows some promise for the purpose of the task.
1 Introduction

As the Swedish population grows older, more people get sick and require medical aid. Therefore a greater need for hospital personnel has arisen. This need has not been met, and in Sweden there is a shortage both among specialized doctors and nurse personnel [8].

While the simplest solution to this personnel shortage is to acquire more personnel through raising wages or other means, the shortage can be combated in other ways. One example is to accurately forecast hospital load, and assign workers accordingly. This can reduce hospital overload and make better use of the already existing personnel.

Use of calendar and weather data to predict hospital load is not a new concept[2]. Particularly the case of forecasting emergency department visits has been studied in the past[5]. While the factors of date and day of week have been shown to be reliable for forecasting emergency department visits, studies examining the impact of climate variables have shown conflicting results. Some studies have shown that particularly the daily maximum and minimum temperature can positively correlate with Emergency Room (ER) visits, while other studies have not gotten any different results when incorporating climate data together with calendar data as opposed to only using calendar data [3, 5]. One possible reason for this is a difference in climate between the geographical locations the studies were conducted at.

While only a small portion of ER visits lead to Intensive Care Unit (ICU) treatment, there is a link between the two. As such, methods used to forecast ER visits might be usable for ICU forecasting. Furthermore, the most common cause of ICU treatment in Sweden as of 2021 is intoxication [11], a variable that is not improbable to correlate to some degree with calendar and weather data.

In order to use forecasting methods to predict future events many methods make use of time series, which is a set of chronologically ordered observations [1]. Such forecasting methods could then be used to forecast target attributes based on an estimated effect of predictor variables present in the time series, which in this study are weather and calendar data. This study will aim to use and examine two different machine learning models, which will be used to forecast the daily number of hospital ICU admissions based on weather and calendar data.

2 Theory

In this study, two different AI models are used. The Temporal Fusion Transformer (TFT)[4] and NeuralProphet (NP)[9]. Both are deep learning AI models specialized for forecasting problems and trained on time-series data.

2.1 Training and Loss

When training deep learning AI models, there needs to be a method to measure the performance of the model. Usually the goal of neural networks is to minimize the error. As such the method of calculating the error is important for the performance of the model. An AI model can be used for either classification or regression problems and the error between the two types differ. Classification is when the model is trying to predict a label. An example of this would be a model that categorizes pictures into ‘contains a cat’ and ‘does not contain a cat’. Regression is instead when the model predicts a quantity for a problem. For example predicting the sales value of a property based on its attributes. In this study the problem is a time-series forecasting problem, which is a type of regression problem where the input variables are ordered by their time value. The error of a model is called loss, and the function for calculating the loss is called a loss function [7].

2.2 Temporal Fusion Transformer

The TFT model was introduced in 2020 by Bryan et al. [4], and was proven to be a very effective model for time series forecasting, outperforming many other relevant models [4]. It distinguishes itself
from other earlier time series forecasting models primarily through its implementation and usage of attention and transformers architecture.

### 2.2.1 Transformers and Attention

The transformer architecture was introduced in the paper ’Attention is all you need’ in 2017 [10]. This architecture was introduced by showing off its capabilities for translation tasks, and in the paper its performance is examined using the WMT 2014 English-to-German translation task, and the WMT 2014 English-to-French translation task. In comparison to earlier model architectures such as Long short-term memory (LSTM) and Recurrent Neural Networks (RNN), the Transformer architecture was significantly better in their tests.

As most other state of the art neural sequence transduction models, the TFT architecture have an encoder-decoder structure. In the case of transformers, what this means is that an encoder maps an input sequence of symbol representations to an output sequence of continuous representations, for example mapping a word to a vector. The decoder then uses this continuous representation to generate an output sequence one element at a time. The model works in steps, and is auto regressive. For each step, the output generated so far is used as additional input for generating the next element in the output.

As shown in Figure 1, the transformer architecture uses two embeddings in order to generate output probabilities each step. Put in the context of translating a sentence from one language to another, the input passing through the Input Embedding is the whole sentence in the original language, while the input passing through the Output Embedding are the words generated so far in the target language.

![Figure 1: The Transformer Architecture](image)

Attention as a mechanism is a way to enable near unlimited long term memory for an AI machine learning model. Attention itself can be explained as the ‘focus’ or ‘weight’ a model places on previ-
ously generated content, which happens recurrently while it generates its output. The Transformer architecture applies self-attention in the Multi-Head Attention layers.

Transformers use a scaled dot-product attention. The input for this consists of queries, keys and values. The queries and keys are vectors that have the dimension $d_k$ while values is a vector with dimension $d_v$. The scaled dot-product attention starts by multiplying the queries in matrix $Q$ with the keys in matrix $K$. The result is then scaled by multiplying with $1/\sqrt{d_k}$. A SoftMax function is then applied to the result in order to obtain the weights. Lastly, the resulting weights from the SoftMax function is applied to the matrix $V$, which contains all values $v$.

The Multi-Head Attention is then built using this single attention function. It’s done by linearly projecting the queries, keys, and values $h$ times. Using a different learned projection for each time. On each of the projections, the single attention mechanism is applied in parallel and the different outputs are then concatenated and projected again which results in the final values for the mechanism.

Figure 2: Scaled dot-product as presented in Attention is All you Need [10]

2.2.2 The TFT Architecture

The Temporal Fusion Transformer is an implementation of the Transformer architecture into a forecasting AI model. It’s a deep neural network model for multi-horizon time series forecasting that utilizes attention with it’s use of transformers [4].

Figure 3: Multi-horizon forecast overview [4]

As seen in Figure 3, the overall structure of a multi-horizon forecast model is based on using the past
targets (historical values of target variable) and observed input (e.g. other data values) from the data. These are used together with the known inputs (e.g. holidays and events) and static covariates (e.g. city or country) to generate a forecast with the same time interval as the past targets.

A particular goal of the TFT model is to improve performance for multi-horizon forecasting. Which means that the model forecasts multiple future time steps as opposed to forecasting only the next upcoming time step, for example forecasting a week as opposed to the next day when time steps represents days in the data.

A notable feature of the TFT model is its support of encoding data features as being either:

- Time-varying or static
- Known or Unknown
- Real or categorical

Time-varying attributes are attributes that vary by time while static attributes do not change. Known attributes are attributes whose values are known at upcoming timesteps, while unknown are not. Real attributes have a numerical value, while categorical attributes have categorical values.

The TFT model uses a modified version of the multi-head attention mechanism from the transformer architecture which is called Interpretable Multi-head Attention, where the goal of the modification is to increase explainability. It is modified so that values are shared in each head. That is, for multiple attention heads each head will have its own keys and queries but the values will be shared between all heads. Additive aggregation is then employed to all heads. This allows for a single set of attention weights which can be used for interpreting the model.

![Figure 4: A complete overview of the TFT architecture](image)

### 2.2.3 Variable Importance and Selection

One way the TFT architecture can help with interpreting the model is by calculating and showing variable importance. This is a way of showing what variables are most important for the model when forecasting. The importance of a variable is based on the selection weights in the variable selection network of the model. A higher importance means that the variable had a higher impact on the
forecast, while a low importance means that the variable had a low impact and subsequently is not important for the forecast.

For each of the input types: static, past, and future, a separate variable selection block is implemented. The first step of the variable selection is to formulate each input variable into a vector. Entity embeddings are used on the categorical data, and linear transformations on the continuous variables. Although static, past, and future inputs have different weights in separate variable selection networks the structure of the variable selection network remains in the same form for each input.

\[
\tilde{\xi}_t = \sum_{j=1}^{m_X} v^{(j)}_{\chi_t} \xi^{(j)}_t
\]

2.2.4 Loss

In order to determine how well a trained model performs, TFT uses quantile loss. This loss function is used because TFT generates prediction intervals together with the forecast. This is helpful in explaining the range of predictions from the model.
\[
\mathcal{L}(\Omega, \mathbf{W}) = \sum_{y_t \in \Omega} \sum_{q \in \mathbf{Q}} \sum_{T=1}^{T_{max}} \frac{QL(y_t, \hat{y}(q, t-T), q)}{M_{T_{max}}}
\]

\[
QL(y, \hat{y}, q) = q(y - \hat{y})_+ + (1 - q)(\hat{y} - y)_+
\]

\(\Omega\) is here the domain of training data which contains \(M\) samples. \(\mathbf{Q}\) is the set of output quantiles, which by default are \(\{0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98\}\) in the TFT model. \(\mathbf{W}\) is representing the weights of the TFT. \(\mathcal{L}\) is the summed loss and \(QL\) is the loss function for a singular quantile value. In the \(QL\) function the \((,)_+\) is equal to \(\max(0,)\).

The strength of the quantile loss function is that depending on the quantile value, over and under predicting will be penalized differently. A quantile value of 0.5 will be similar to a mean absolute error, in that over and under predicting will have the same loss for the same difference between predicted and actual values. A quantile value of 0.9 would instead heavily penalize predicted values lower than actual because the loss value is multiplied by a factor of 0.9, while a predicted value higher than actual would have a loss value multiplied by a factor of 0.1. Inversely, a quantile value of 0.1 would heavily penalize predicted values higher than actual values. Because the training loss of the model is the sum of all quantile loss, the loss value of the model will have a meaning similar to mean absolute error loss if the quantile values used have an average value of 0.5.

The TFT model can use many different quantile values in order to display quantile ranges in its forecasts. This can help with explaining the forecast and give ranges of certainty to the predicted values.
2.3 Neural Prophet

Neural Prophet is a time-series modeling library, based on neural networks and implemented with PyTorch. It was introduced as a successor to the forecasting tool Facebook Prophet in the paper "NeuralProphet: Explainable Forecasting at Scale" in 2021[9]. NeuralProphet is at its core a modular model, consisting of different modules which all add an additive component to the forecast. The complete model is described as:

\[ \hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t) \]  

(3)

In which,
- \( T(t) \) = Trend at time \( t \)
- \( S(t) \) = Seasonal effects at time \( t \)
- \( E(t) \) = Event and holiday effects at time \( t \)
- \( F(t) \) = Future regressor effect at time \( t \)
- \( A(t) \) = Auto-regression effects at time \( t \)
- \( L(t) \) = Lagged regressor effects at time \( t \)

Each of these modular components can be individually chosen to be used or not used for a model forecast, and all used components contribute to the final forecast.

2.3.1 Trend

The trend value represents the overall positive or negative trend of the forecast target value over time. For example, the price of a commodity will fluctuate based on many different variables, but inflation will cause a constant positive trend on prices.

NeuralProphet uses a classic method for modeling trend, with one addition. The classic approach consists of using an offset \( m \) and a growth rate \( k \). For a specific time point \( t_1 \) the trend is calculated as the multiplication of the growth rate and the difference between the initial time point \( t_0 \) and \( t_1 \), and lastly adding the offset \( m \)

\[ T(t_1) = m + k \times (t_1 - t_0) \]  

(4)

On top of this, NP allows the growth rate to change at a finite number of points. This means that between these changepoints the trend is as defined in equation 4, while at the points the growth rate can change. The number of changepoints can be chosen when training the model. When a forecast is made for the unobserved future, the growth rate for that time is linearly extrapolated from the final growth rate.

2.3.2 Seasonality

Seasonality, or seasonal variation, refers to the way time-series data can vary according to repeating cycles over time. Values can follow a pattern within a specific period. By default, Neural Prophet supports daily, weekly, and yearly seasonality and can account for, find, and explain the seasonality of the data. In this study, only yearly and weekly seasonality will be examined, because the data is daily and as such no cyclic pattern can be found for a day.

The way Neural Prophet implements seasonality is by using Fourier terms as shown in the equation below. The Fourier terms are defined as sine, cosine pairs, and a number of Fourier terms are defined for each seasonality. In the equation for seasonality, \( t \) is the time step, \( p \) is determined by the seasonality...
and data, so yearly seasonality with daily data would have $p = 365.25$. $K$ is the number of Fourier terms.

$$S_p(t) = \sum_{j=1}^{k} (a_j \ast \cos(\frac{2\pi jt}{p}) + b_j \ast \sin(\frac{2\pi jt}{p}))$$

(5)

Seasonality can be either additive or multiplicative in its effect on forecasting, and for this study additive seasonality was used. Multiplicative seasonality is used when the seasonal pattern is growing or shrinking by the season.

2.3.3 Regressors

Auto-regression can be described as a time series model that uses past values as input to a regression equation in order to predict the value at the upcoming time step. The number of past values used for a forecast is a chosen number $p$, where from time $t$ the values $y_{t-1}...y_{t-p}$ are used. This is usually referred to as the order of AR model. The AR module in NP can be used to forecast $h$ steps into the future, this is referred to as the forecast horizon. For a time $t$, the input to the module are the $p$ last values, and the outputs are $h$ values, one for each forecast step. The forecast horizon also determines the number of predicted values there are for a time $t$. Because a forecast is made from each time $t$, and $h$ number of values are generated, each timestep will have $h$ different forecasted values. One from $t-1$, one from $t-2$, and so on all the way to the oldest forecast from $t-p$.

NP also supports Deep AR. This implements a fully connected neural network that is trained for the AR module.

Lagged regressors are the way Neural Prophet handles variables that are used to correlate other observed variables to the target time series. These are values that are known only for past time steps and are also referred to as covariates, comparable to the Time-varying unknown variables in the TFT model. For a specific time of the forecasting, all previous values are available for the forecasting. Therefore the number of available values is equal to the timestep minus one.

Future regressors are values that are completely known, both future and past.

2.3.4 Loss

For the Neural Prophet model, two different kinds of loss are used: Mean Absolute Error (MAE) and Root-Mean-Square Deviation (RMSE).

MAE is a common loss function for regression problems, especially forecasting. The function is defined by the average absolute difference between the predicted and the actual value. Mathematically it is defined as:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - x_i|$$

(6)

RMSE is similar to MAE, in that it works by taking the squared root of the average squared difference in actual value and predicted value. It is mathematically defined as:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - x_i)^2}$$

(7)

These two loss functions excel at different things. RMSE is better suited when large errors are especially costly and as such it is important to minimize them. Because of this however RMSE is sensitive to outliers in the data, as large errors will substantially increase the overall error. MAE is better at handling outliers in data.
3 Method

3.1 Design

For this study, two kinds of predictive variables are used, calendar and weather data. This data is used to train deep learning AI Forecasting models in order to forecast the daily number of patients arriving at the ICU in a city of Sweden. The goal is to examine if weather and calendar data can be used to forecast ICU arrivals by training an AI model to forecast ICU arrivals as accurately as possible.

3.2 Limitations

In this study, only calendar and weather data are used to train the AI models. This is not expected to be enough to train a model ready for public use. There are many variables that can contribute to a person needing space at the ICU, and not all are covered by the data used in this study.

3.3 Study region

For this study, weather and calendar data are used as predictive attributes. The study uses data collected from the region around the city Gävle in Sweden. The weather data is collected from data published by the Swedish Meteorological and Hydrological Institute (SMHI). This weather data had been collected by weather stations used by SMHI situated in the Gävle area, and is collected each day at the same time. ICU patient data is collected from data published by the Swedish Intensive Care Registry (SIR), also from hospitals in Gävle.

3.4 Data attributes

The data used in this study consists of daily readings for weather data and daily admissions to the ICU in the timespan between January 2nd, 2010 and December 28th, 2018. Because the COVID-19 pandemic have significantly affected ICU crowding, data from that period has not been used. Each data point contains the feature values for a specific date, and the data is arranged sequentially with one data point for each day in the timespan.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DailyTreatments</td>
<td>The number of patients admitted to ICU that day</td>
</tr>
<tr>
<td>minLuftTemperatur</td>
<td>The minimum temperature measured in the last 24 hours</td>
</tr>
<tr>
<td>maxLuftTemperatur</td>
<td>The maximum temperature measured in the last 24 hours</td>
</tr>
<tr>
<td>averageTemperature</td>
<td>The average temperature of the last 24 hours</td>
</tr>
<tr>
<td>precipitation</td>
<td>The precipitation measured in the last 24 hours, in mm</td>
</tr>
<tr>
<td>weekday</td>
<td>A categorical feature of the day of the week</td>
</tr>
<tr>
<td>month</td>
<td>A categorical feature of the month of the year</td>
</tr>
<tr>
<td>dayOfYear</td>
<td>A categorical feature of the day of the year</td>
</tr>
<tr>
<td>weekOfYear</td>
<td>A categorical feature of the week of the year</td>
</tr>
</tbody>
</table>

Both models use Pandas, a data analysis library built for Python [6]. The Pandas library allows loading the data into data frames consisting of rows and columns that the models can use for training. Because this study handles time series the columns denote dates and rows denote the attributes.

3.5 Temporal Fusion Transformer Training

After preprocessing all data into Pandas DataFrame objects the time series dataset is defined. The generated forecast wanted is for the next seven days, therefore the max_prediction_length is set to 7 and the min_prediction_length is set to 1. The max_encoder_length is set to 30 to allow the model 30 days history length. A validation dataloader is created with a batch size of 64. The default values for quantile loss are used.

In order to find the best hyperparameters, the optimize_hyperparameters function was used. This function is built in to the pytorch implementation of TFT and can be used to help optimize the
hyperparameter values. The final values used for training the model were as follows:

```python
trainer = pl.Trainer(
    max_epochs=300,
    accelerator="gpu",
    enable_model_summary=True,
    gradient_clip_val=0.3548916898081371,
    limit_train_batches=30,
    callbacks=[lr_logger, early_stop_callback],
    logger=logger,
)

tft = TemporalFusionTransformer.from_dataset(
    dataset,
    learning_rate=0.022610477215010583,
    hidden_size=109,
    attention_head_size=4,
    dropout=0.28889329643372097,
    hidden_continuous_size=90,
    loss=QuantileLoss(),
    log_interval=10,
    optimizer="Ranger",
    reduce_on_plateau_patience=4,
)
```

### 3.6 NeuralProphet Training

The data is first preprocessed into pandas DataFrame objects in the same way as for the TFT model. The model is then defined as:

```python
model = NeuralProphet(
    n_changepoints=10,
    yearly_seasonality=True,
    weekly_seasonality=True,
    daily_seasonality=False,
    n_lags=30,
    n_forecasts=7,
    learning_rate=0.02
)
```

For the model, n_changepoints controls how many times the trend changes over the dataset. The yearly and weekly seasonality is used for forecasting. In order to have 30 days of lookback and a 7 day forecast, n_lags is set to 30 and n_forecasts is set to 7. Using n_lags also enables auto-regression for the model. Lastly the learning rate used was 0.02.

Precipitation, MinLufttemperatur, MaxLuftTemmperatur, and averageTemperature are added as lagged regressors. Twenty percent of the data is used as validation data, the remaining 80 percent for training.

For this study no future regressors were used, and neither was holiday data.
4 Results

In this section the performance of the models are presented, as well as some attributes of the models that may give insight into the data and forecasts. Because the models have had many different forms during implementation, only the best performing models are presented in this section. The criteria of Best Performing is decided by the model with least loss, and as such the most accurate forecasts.

4.1 TFT Results

The best performing TFT model uses an encoder length of 30 and forecasts seven days ahead. On the validation dataset the model has a MAE of 0.5053. Figure 7 shows an example of a relatively good forecast from the model. The blue line shows the actual values of the data, while the orange line shows the forecast with a quantile value of 0.5. The orange gradient around the orange line shows the forecast made using the other quantile values, and as such shows a range of possible values as determined by the model. The faded out line shows the models attention, also shown more clearly in Figure 8. This is a visual representation of where the attention of the model where for the forecast, i.e. what importance specific past values had for this forecast. As is usual the more recent time points have higher attention weights. There is however an odd spike in attention for the time point 28 days past. This is particularly odd because the importance of the month variable is low for the encoder. No explanation for this was found in the study.

Figure 7: A forecast from the TFT model.

From the model the encoder and decoder variable importance can also be shown, as seen in figure 9 and 10. These are the variable importance weights generated and used by the variable selection network as described in section 2.2.4. The encoder variable importance shows the importance of past covariates, past targets, and past future covariates. The decoder variable importance contains the future covariates importance. For example, these results show that the week of the year of past time points have a very high importance, and the weekday for the forecast time points have the highest importance.

Figure 8: The attention of the forecast.
4.2 NeuralProphet Results

For the NP model, some variations were done when testing that used hidden layers. The results for these models ranged from being much worse to being just slightly worse. For this reason neither the methodology nor the results from those models are presented.

The results of the forecast from the NP model can be shown in two ways. Figure 11 shows the models forecast for the complete data range. Figure 12 shows a forecast slice over a seven day range. In both graphs there are seven different forecasts for each data point. This is because of how the model works when selecting the forecast horizon, as explained in 2.3.3. The black dots on the graphs show the actual data values and the blue lines are the forecasted values.

Figure 11: A complete forecast from the NP model.

Figure 12: A seven day forecast from the NP model.

Figure 13: Training loss of the NP model.
Figure 14: Trends of the data extracted using the NP model.

Figure 13 shows the MAE training loss of the NP model over the training epochs, and how it changed during training. The final MAE value is 0.8322. The final RMSE loss value is 1.0773.

The trends of the data in figure 14 shows how the n_changepoints value affected the trend component of the forecast for different date periods of the data. The trend component saw a slight increase in 2011, and stayed the same for the rest of the data. This shows that the trend component had very little to no effect on the forecast.

Figure 15: Seasonality of the data extracted using the NP model.

The seasonality data in Figure 15 show how much the seasonality component affected the forecast made. This shows both the yearly and weekly seasonality. The vertical axis show the value added to the forecast depending on the date. For example a forecast for a Tuesday would have a value of about 0.19 added from the weekly seasonality component. The values of the seasonality component are mostly low. The largest effect of seasonality would be for example a Tuesday in October adding about 0.45 to the forecast, and a Saturday in late December subtracting a similar value. The impact of seasonality on the forecast value was overall low.
5 Discussion & Conclusion

Using data collected from the city Gävle region in Sweden, this study aimed to train two machine learning models into accurately forecasting ICU arrivals at the hospitals of the same city. For this study the Temporal Fusion Transformer and the NeuralProphet AI models were used. From the results, it is clear that the TFT model outperformed the NP model when it comes to making accurate forecasts. The TFT model have significantly lower MAE loss value compared to the NP model, 0.5053 as opposed to 0.8322. For the purpose of creating accurate forecasts the NP model is not usable. The forecast of the NP model heavily favors forecasts close to the median value of 1, and as such is not much better than simply expecting the median value each day. Compare this to the forecast of the TFT model, while the forecast example in figure 7 is one of the better forecasts, it shows that the model is capable of making valuable forecasts that while not completely accurate, are close. The question is whether it is good enough.

It could also be said that the results of the TFT forecast are more interpretable and better suited for the task when used by hospitals. The quantile loss used by the TFT model gives ranges of certainty in the forecasted value. The quantile values used could also be changed so that the model more heavily penalizes under-predictions, and as such reduces the amount of forecasts that would leave the ICU understaffed at the cost of some accuracy.

The variable importance results can give insight into which variables are unimportant and which variables can help. For future work this can also help decide what variables are worth collecting and using. As predicted calendar variables have most importance for both past and future covariates, with both weekday and week of the year acquiring high importance. Of the weather variables, the temperature have quite a high importance. Though significantly less than the calendar variables. Precipitation have not acquire high importance and seems to play a negligible part for ICU arrival forecasts in this study and as such should not be used for future studies. Neither the NP model nor the TFT model used event and holiday data for the forecast. This is something that could improve the forecast, however the expected impact is low.

The seasonality data from the NP model supports the importance of the weekday and week of year variables. The yearly seasonality shows that the high and low points of the year differ. Although no clear seasonal seasonality, such as an increase over summer and decrease in the winter, it does show clearly how some times of year have an increase or decrease of ICU arrivals compared to an average. The weekly seasonality also show a clear increase of ICU arrivals on weekdays and a decrease on weekends. However, the total impact of seasonality for the NP model forecast is somewhat low.

A challenge when forecasting ICU arrivals during this study is the low amount of daily arrivals. Since the usual ICU daily arrivals are between zero and two, chance and random events have a higher impact on the ICU load. Additionally, a forecast being off by only one person means that the forecast error is very large. The usability of the model also quickly decreases, since an option would be to just always expect the median of one ICU arrival daily and the model needs to outperform that. Were this study to be applied to a larger region or hospital the results might improve as the larger amount of people brings the data closer to averages. However, casting the net too wide would make the model meaningless, since the purpose of the model is to accurately forecast ICU arrivals so that personnel can be better assigned.
References


