# Discovering Machine Learning Models and Features for Predicting Scheduled Patient Examination Volume at the MGH

*This project was carried out as part of the TechLabs “Digital shaper Program” in Aachen (Winter Term 2020)*

**Team members**: Johannes Ehls, Wassim Masmoudi, Hanna van den Munkhof, Kathrin Nieswiec, Xin Shen, Konstantin Wehmeyer **Mentor**: René Ahlsdorf

**Special Acknowledgement**

We would like to thank the Massachusetts Generel Hospital, especially Dr. Oleg Pianykh and Mr. Richard Zhang, who made this project possible not only by providing us both with the problem and the dataset, but also by supporting us with advice and support. We greatly appreciated the opportunity to work with real data from this field.



**1.1 Introduction**

Our project was executed in collaboration with the Massachusetts General Hospital in Boston and aimed to optimize the planning of diagnostics exams. More precisely, we needed to predict the number of patient exams that would be completed at different time points in the future (1 week, 2 weeks, 4 weeks and 2 months) based on basic logistic data (e.g. number of exams ordered by the referring physicians, number of rescheduled examinations, number of processed examinations, number of examinations scheduled — all at different time points in the past), plus features engineered by ourselves. Increasing the predictability of the actual number of exams that will be needed on a certain day would greatly improve staff planning and save resources.

The goal of this project was not only to learn a basic understanding of how to work with and analyze structured data, but also to analyze the differences between prediction models built with traditional machine learning and deep learning algorithms. For this reason, this project was offered as a combination project in the Artificial Intelligence as well as Data Science tracks. Kathrin, Johannes and Konstantin were responsible for the creation of the deep learning based models and Hanna, Xin and Wassim for the traditional machine learning models.

This blogpost is structured as follows: Section 1 briefly introduces and outlines the structure of the dataset, conducted feature engineering and key metrics for model evaluation. Section 2 presents the models and resulting prediction results from machine learning. Section 3 presents the models and the results based on the same test sets from the deep learning domain. Section 4 concludes with a brief summary of the project and offers a short outlook on further steps.

**1.2 Dataset & Feature Engineering**

*Subproject: Wassim & Rene*

**1.1.2 Dataset Structure and Feature Engineering**

Our dataset contained the accumulated counts of ordered, planned, scheduled, performed, rescheduled/postponed and actually performed patient examinations from one department of Massachusetts General Hospital on 438 days in the period of October 2018 to July 2020. The respective patient counts on e.g. rescheduled examinations referred to the accumulated values of the last 7,14,28 and 56 days for the observed day/sample in the dataset. This resulted in 25 features that, in addition to the previously mentioned feature types, also included the respective day of the year (relative to the beginning of the year) to take into account the seasonality of the scheduled examination. Subsequently, the goal has been to predict the actual examinations conducted for the upcoming 7 (CFuture1), 14 (CFuture2), 28 (CFuture3), and 56 days (CFuture4), given the previous features for the observed day (hereafter referred to as core features).

In addition to these features, we also added weather data, artificial features to display the days of the week, season, etc., paydays, and, for purely experimental reasons, stock market data of the Dow Jones to the feature set (hereafter referred to as “all features”). After further preprocessing steps such as deleting invalid rows, we checked the dataset for existing correlations between the core and engineered features and CFuture counts.

To test the correlation of our added features with the variations of the patients in the hospital the linear regression package from scikit learn was used and the coefficients of determination were analyzed and visualized using matplotlib.  
Weather data such as the temperature and the conditions, stock market variation based on the Dow Jones (experimental feature), the variation related to holidays, airport traffic and the paycheck days in the Boston metropolitan area were individually analyzed in terms of their correlation with the CFuture values. Unfortunately, no interesting correlations were discovered between most of the artificially added features and CFuture values (except the stock data surprisingly), hence we will not discuss this part further.

The prediction results of the trained models also showed that key metrics are better when trained with the core featurs than with all features, which may be explained by the low amount of daily data samples (only 438 rows). Additionally, adding more features exacerbates the imbalance between number of samples and amount of features.

**1.1.3 Validation Strategy**

For analyzing our patient examination forecast models, we required both a testing strategy and metrics which allowed fair comparison of the models. Since our dataset falls within the COVID pandemic and accordingly deviates greatly from the normal examination distribution before the onset of the crisis, we removed the COVID data from the entire dataset starting 03/13/2020 and evaluated our models accordingly in two phases: — 1) Training on 90% (approx. one year) of the data without the COVID pandemic and validate on the remaining 10% without COVID pandemic (October 2019 to February 2020) — 2) Training on the previous training and testing sets and validating on the COVID data to measure the generalizability of the models. In addition, in phase 1), in the case of XGBoost, we also applied the walk-forward strategy to measure whether the model achieves better forecasts with the presence of new daily data points.

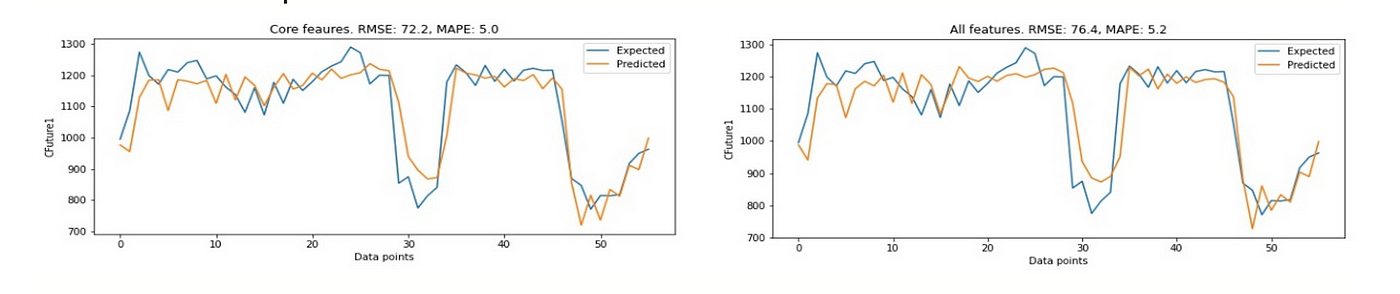
As key metrics, we chose [Mean absolute percentage error (MAPE)](https://en.wikipedia.org/wiki/Mean_absolute_percentage_error)and [Root Mean Square Error (RMSE)](https://en.wikipedia.org/wiki/Root-mean-square_deviation). The former metric is useful to look at the prediction error of the models relative to the actual investigation that happened, and the latter to interpret the deviation from the ground-truth examinations in absolute terms. This combination allowed for all CFuture values a fair comparison of the models and interpretation of the models’ performance. In addition, the exact shape of the prediction curves relative to the ground truth shapes on the test set was used as another subjective metric.

Since some of the models had difficulty predicting CFuture3 and CFuture4 values due to the small amount of data, we focus mostly on CFuture1 (patient counts for the upcoming week) and CFuture2 (next two weeks) in the following report. Also, unless otherwise stated, all models were first trained and analyzed on the dataset without the COVID pandemic first, and were subsequently evaluated on the COVID data to demonstrate their generalizability.

**2.1. Machine Learning Approach**

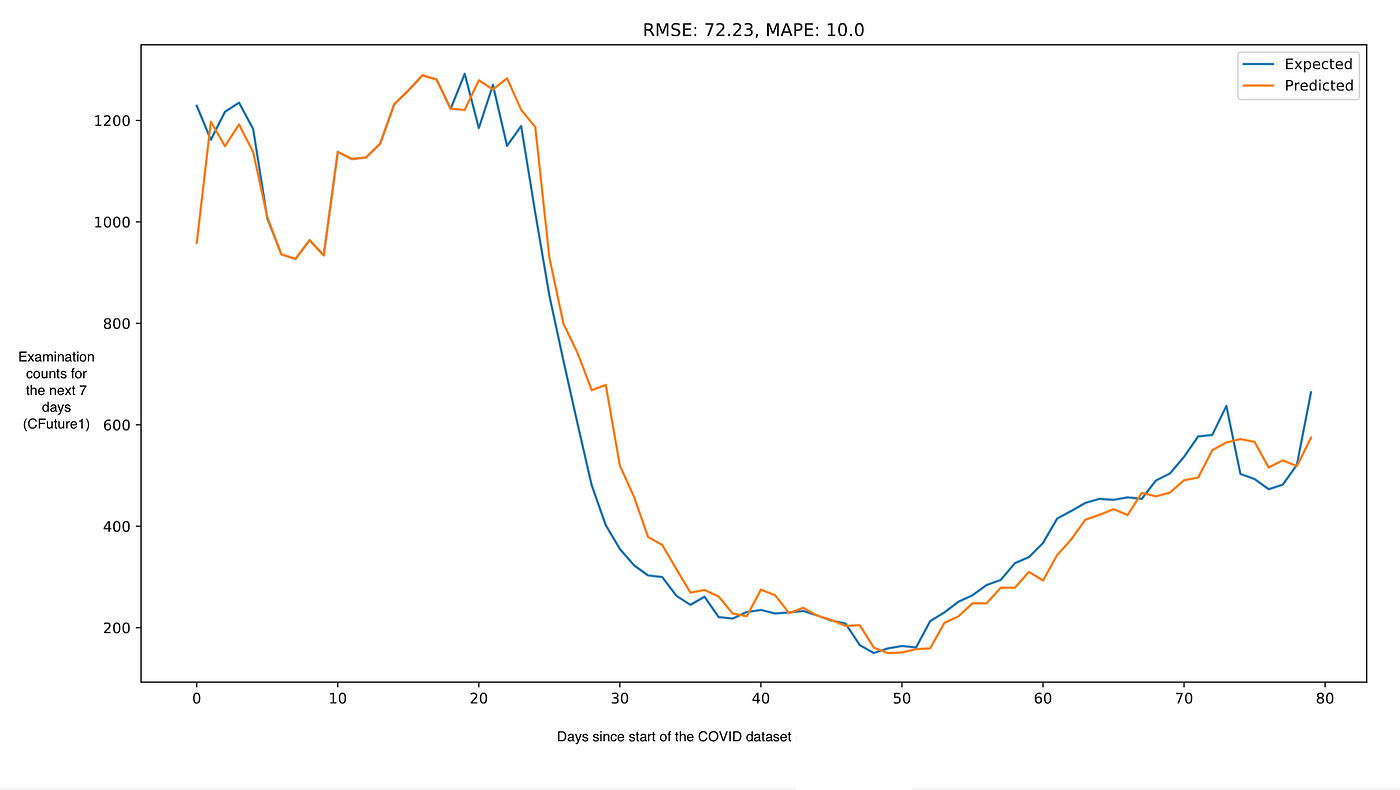
**2.1.1 XGBoost**

*Subproject: Hanna* Extreme Gradient Boosting is currently one of the most popular machine learning models as it is very powerful without having the disadvantages of a complex model, because it combines several weak learners into one richer model. It consists of a decision tree ensemble that in addition to deciding on a feature threshold at each split/leaf, assigns scores to each leaf outcome. For parameter optimization we first tried GridSearchCV, but as its output was incorrect — consistently setting the learning rate too small — for some reason we still don’t understand, we switched to manually optimizing the three most important parameters (eta, max\_depth and gamma) which worked well enough. After parameter optimization, the final model was trained and tested using walk-forward-validation for time-series data. This means that all features of the previous 6 days (best out of 6, 10 and 20) were used for prediction, and training and testing took place by adding the test data one by one to the training set after each prediction.



*Fig. 1:* *The results for the core (i.e. original) and augmented features on CFuture1 using walk-forward validation.*

Finally, the model was tested on the held-out Covid19-data to check its generalizability. XGBoost performed quite well, slightly better when only the core features (MAPE=5.0% on CFuture1) were included than with all features (MAPE=5.2% on CFuture1), and a lot worse on the COVID19-data (MAPE=24.3% on CFuture1) yet closely following the curve of the real data. Limiting the model to our engineered features also led to a worse performance (MAPE=9.5% on CFuture1), yet adding two important core features improved this performance greatly (MAPE=5.8% on CFuture1), which shows that those two features are very important for prediction. The fact that this model performed better than the neural networks, using mostly those two core features, was very surprising. Thus we learned that it’s sometimes better to use simpler models, especially in the health care field as results need to be interpretable in order for a model to be accepted.



*Fig. 2:* *The results for the time during the COVID outbreak after training the model on data of 2019.*

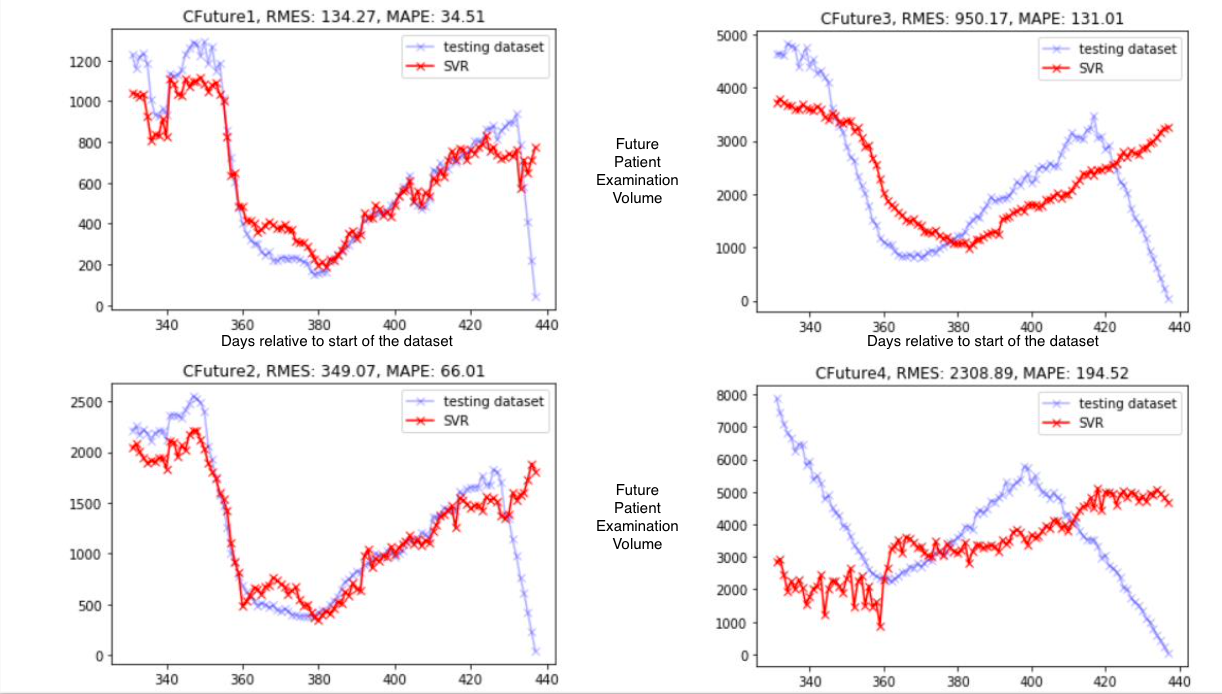
**2.1.2 Support Vector Regression**

*Subproject: Xin* Support Vector Machines (SVM), a traditional machine learning algorithm, has been widely implemented for various classification problems. Support Vector Regression, on the other hand, is considered as the counterpart of the SVM for effective real-value estimation. The core concept is to determine the hyperplane in a higher dimension to fit the dataset by considering the defined error margin.

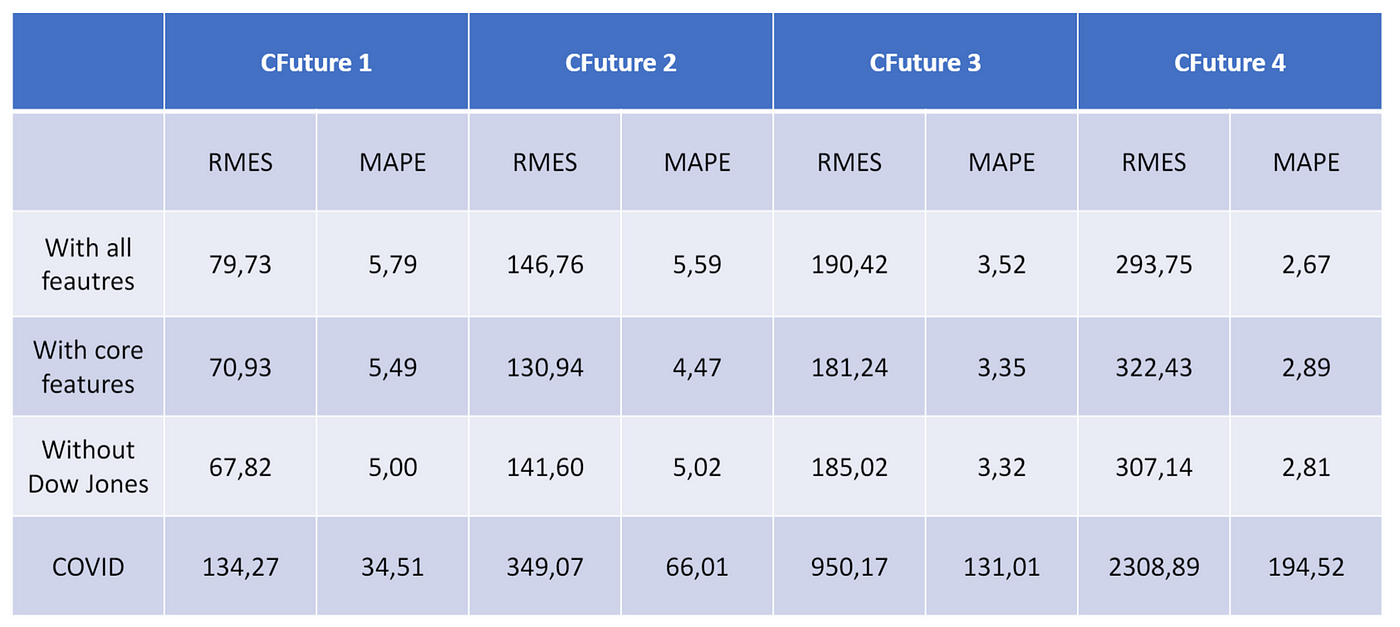
The radial basis function is chosen for the SVR model. Similar to the XG-boost, the rest hyper-parameters of the SVR estimator need to be tuned firstly, including gamma, C and epsilon with the usage of the grid search method. After the determination of the robust parameters, the fitted curve is visualized along with the training dataset. As for the CFuture 1 and 3, the curves demonstrate a good fit for a series of data points. However, the curves cannot reflect the trend of the dataset of CFuture 2 and 4. With a manual increase in parameter C, the curves perform significantly better in general.

The trained model is evaluated on the small-scaled testing dataset. It is interesting to observe that the same value is predicted for each day while applying the trained model on the four scenarios CFuture 1–4. Considering the sensitivity of the SVR to the normalized dataset, the data scaling technique is hereby utilized to arrange the input dataset in a standard normal distribution. The curve fitting results are significantly improved afterward both on the training and testing dataset.

Aside from the testing dataset, the trained model is additionally applied to the COVID dataset. The predicted results demonstrate an outstanding performance for CFuture 1 and 2. However, the prediction with obvious time-window shift can be clearly observed for CFuture 3. As for CFuture 4, the prediction fails in capturing the rapid downward tendency. RMES and MAPE are employed in the project in order to critically evaluate the testing results, seen in Table 1. Overall, the prediction accuracy is constantly decreased from CFuture 1 to CFuture 4. The trained SVR model outperforms using the test dataset before the COVID outbreak (cf. Figure 4).



*Fig. 4: The SVR predictions for CFuture1–4 after the COVID outbreak. The SVR model was only training on pre-COVID data.*



*Tab. 1: The results for the SVR models and different CFuture examination loads.* ….

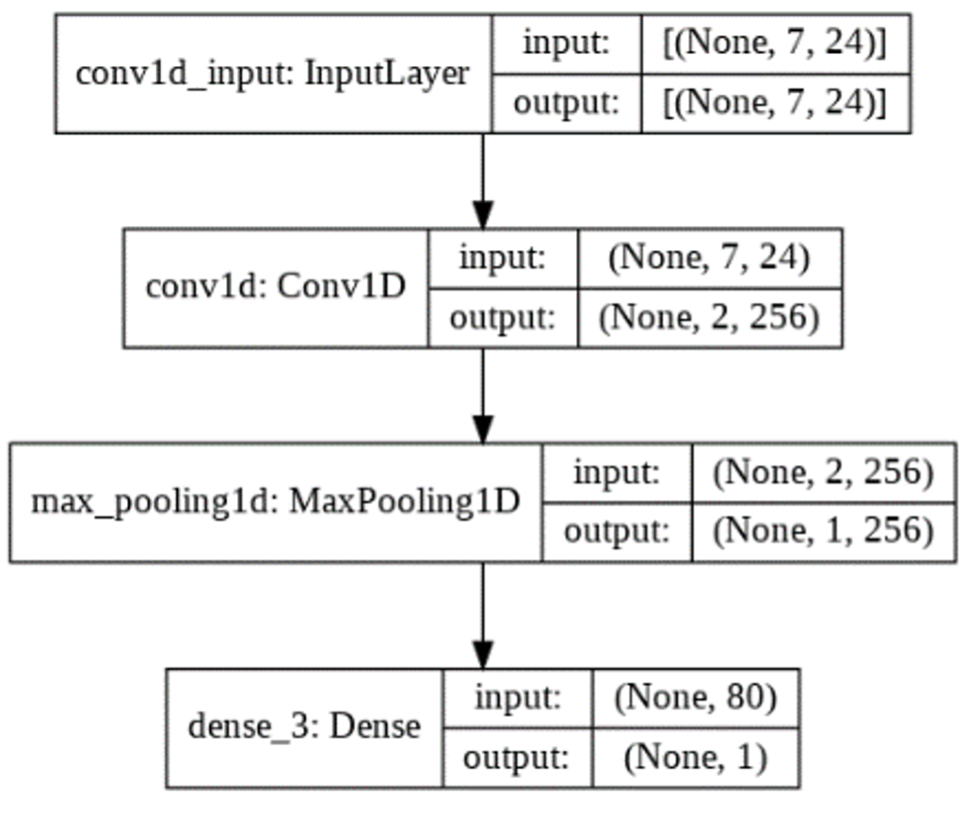
**2.2. Deep Learning Approach**

In order to compare the capabilities of deep learning based models in comparison to the traditional machine learning approaches presented before, we trained and evaluted CNN and LSTM based algorithms on the same training and tests sets which were used before, both on non-COVID and COVID examination data.

**2.2.1 CNN-based Models**

*Subproject: Konstantin*

Although traditionally developed for two-dimensional image data, CNNs can also be used to model time series forecasting problems. The CNN model learns a function that maps a sequence of past observations as input to an output observation. We have used Keras, a library that provides a Python interface for neural networks, to set up our deep learning models. For a first try, we defined an architecture containing one convolutional layer followed by a max pooling layer and a dense layer to interpret the input feature. An output layer predicts a single numerical value, the patient examinations count for the next 7 days (or 14/28/56 days depending on which output variable the model was trained on). A schema of the first model is shown in Figure 5. We hoped that the convolutional layer would recognize certain patterns to outperform the previous traditional modeling approaches. If, for example, the number of rescheduled examinations increases significantly, a (e.g. difference-) filter would detect this and therefore could predict a lower examinations count. As mentioned in the introduction, we had two different versions of datasets, one with all the additional features (such as weather data or one hot encoded calendric features) and one with just the core features provided from the hospital. Trained initially on the enlarged dataset, the performance of the model was rather bad, but trained on the original dataset, the results were quite promising, so we went on to tune the hyperparameters on the core-features only dataset.

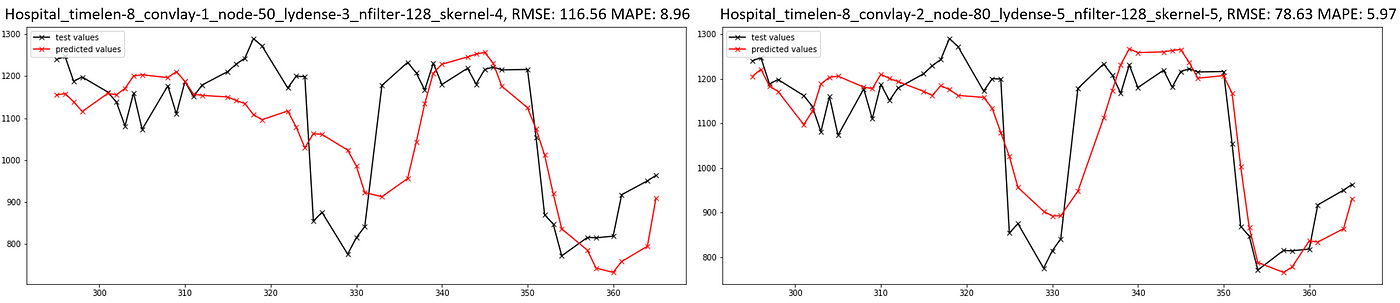


*Fig. 5: Architecture of the CNN model — Layers: InputLayer, Conv1D (CNN layer), MaxPooling1D, Dense.*

**2.2.2 Hyperparameter Tuning of the CNN Model**

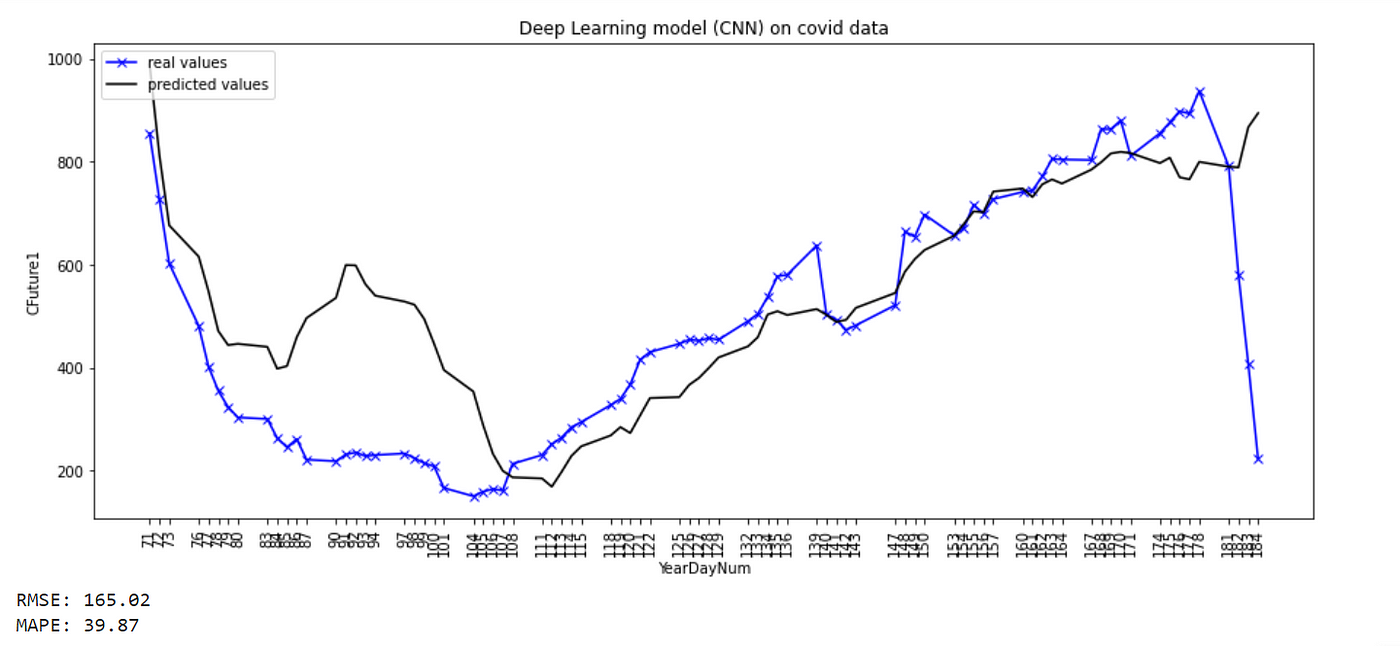
*Subproject: Kathrin*

The architecture of a neural network has a great impact of the accuracy of model predictions. Architectural design choices or hyperparameters are, for example, the number of layers, types of layers, number of neurons used, etc. During a training process, the individual weights between the input and output values are varied in order to make the most accurate prediction possible. However, the architecture and hyperparameters are not changed during model training, which means that the choice of hyperparameters has a major impact on prediction accuracy. For accurate prediction results, the hyperparameters must be optimized for the particular model. Hyperparameter optimization can be done in different ways, manually or automatically. In manual optimization, different model configurations are tried by hand with trial and error. However, this poses a major challenge for complex models or long training times and furthermore consumes a lot of time on has to spend each time a new configuration shall be tested. For automatic optimization, on the other hand, tuners such as Grid Search or Kera’s tuner can be used, or one can design a tuning procedure to furthermore also do optimizations on the model architecture. In the TechLabs project, automated hyperparameter optimization was performed by implementing an individually programmed routine, which creates multiple model types. These model types are created based on list of options with subsequent hyperparameters as: Number of convolution layers, convolution filters, kernel size, dense layers, dense nodes, an the timeseries. Parameter spaces were defined for the hyperparameters and the models were tested. When inspecting the different outputs for the prediction of CFuture1, the importance of different hyper parameters optimization is shown. For each of the models, 360 (core features) and 288 (all features) models were tested, and the best models were determined based on the RMSE and MAPE values, which were calculated by testing each created model on a separate validation dataset. For the prediction of CFuture1 values with the core features, a model with the following architecture was found to be a suitable tested model, with the values for the RMSE of 78.63 and the MAPE of 5.97%: Timeseries=8, convolution layer=2, convolution filter=128, kernel size=5, dense layers=5, dense nodes=80(cf. Figure 6). To underline the importance of hyperparameter optimization, the worst tested model gets a RMSE of 116.56 and a MAPE of 8.96% with the following architecture: Timeseries=8, convolution layer=1, convolution filter=128, kernel size=4, dense layers=3, dense nodes=50(cf. Figure 6). Using all the features as inputs, the model was also tested over the whole parameter space. The best model with the values RMSE 72.30 and MAPE 5.16% was based on the following architecture: Timeseries=8, convolution layer=1, convolution filter=128, kernel size=3, dense layers=5, dense nodes=50.



*Fig. 6: Tested models with using the core features. Left: The worst tested model, Right: The suitable tested model.*

The hyperparameter tuning process was repeated for the second model that was trained on the training- and testset and validated on the COVID data. Figure 7 shows the forecast results after the start of the Corona pandemic.



*Fig. 7: Optimized CNN architecture trained on the core features of the training- & testset and validated on the COVID dataset.*

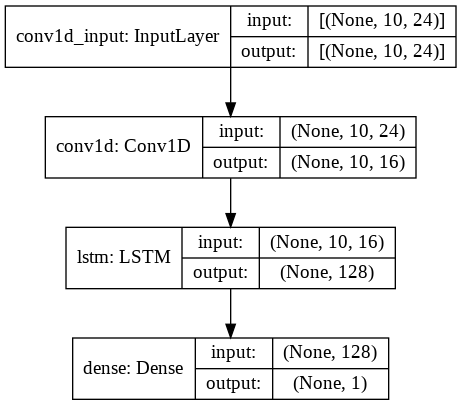
The model is able to follow the overall trend of the patient examination drop after the start of the Corona pandemic, but is not able to predict the short-time incline starting at 87th day of 2020 (March 27). In comparison to the model results of Section 2.1, the CNN-based models performs worse, especially when the dynamics of the input features change drastically. Nonetheless, the model successfully forecasts the increasing patient examination counts starting at day 111 (April 21).

**2.2.3 LSTM-CNN Model**

*Subproject: Johannes*

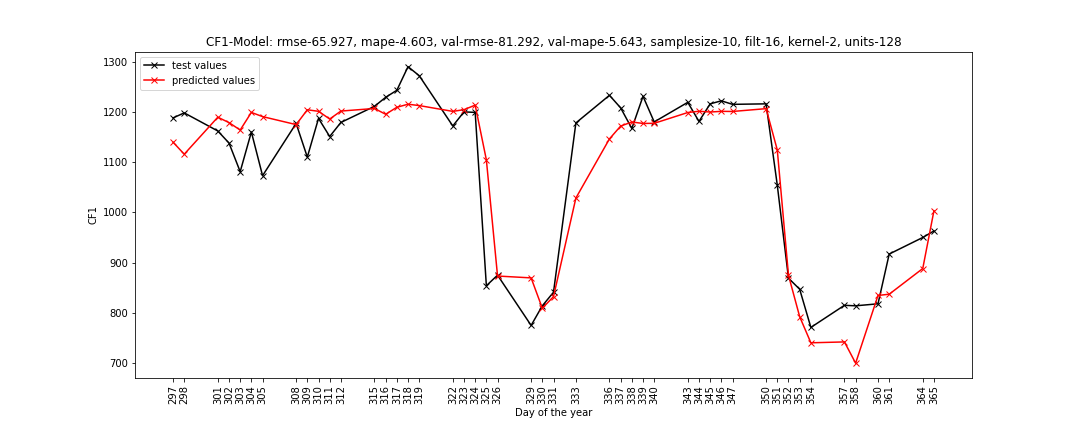
The second AI-Model that we trained was a CNN-LSTM model which combined one CNN layer with one LSTM layer. The intuition behind this architecture is the following: — The **CNN** (**C**onvolutional **N**eural **N**etwork) technology was developed and is often used for image classification. We used CNN as a first layer type with the intent to filter out the most important features which then are the input for the second layer an LSTM layer. — The **LSTM** (**L**ong **S**hort-**T**erm **M**emory) layer was, in contrast to the first layer, a recurrent layer with memory. This means that it does not only rely on the current sample to predict the examination count, but also on the predictions of the previous examinationcounts. Our intent was to capture patterns on a temporal axis from the filtered features.

Last, the architecture was completed with a Dense layer which outputs one single value and, of course, an input layer which forwards the values to the CNN layer (named Conv1D).

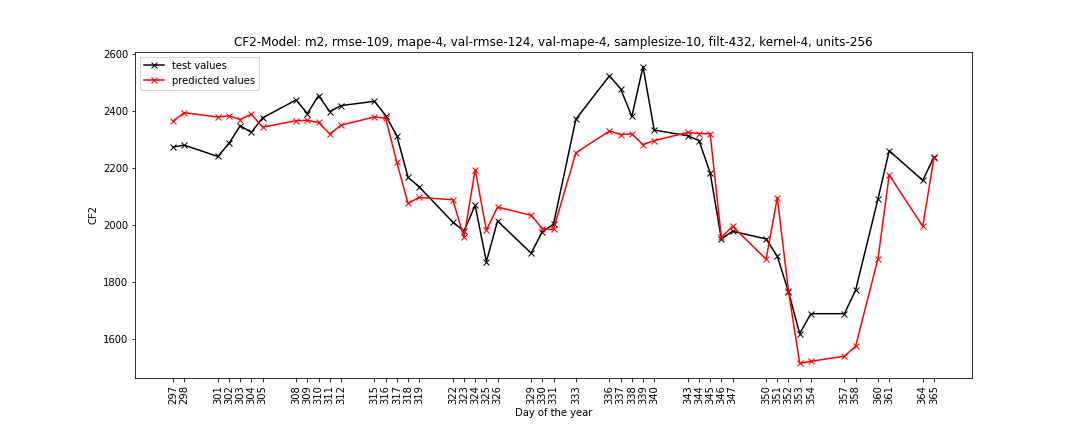


*Fig. 8: Architecture of the CNN model — Layers: InputLayer, Conv1D (CNN layer), LSTM, Dense.*

To implement this model in Python we used the the same libraries (Tensorflow and Keras) as for the CNN model before. The models were trained with an eighty twenty train-validation-split (80% test and 20% validation data). We also used early stopping and restored the parameters of the best epoch to obtain the best result. As we already assumed, we experienced the LSTM to be more difficult to train due to it’s recurrent architecture. Apart from the fact that the LSTM uses more parameters which takes more time to train, we were facing some issues with weights beeing equal to zero which had the consequence that the model predicted the same value for every sample. In contrast to the first CNN model, to tune the hyperparameters of this CNN-LSTM model RandomSearch from the Keras Tuner library was used. For the CNN layer, the number of filters and the kernel size were tuned and for the LSTM layer, the number of units was tuned. In Figure 9, the results obtained by applying the best models for each CFuture1 and CFuture2 on the test dataset are presented.

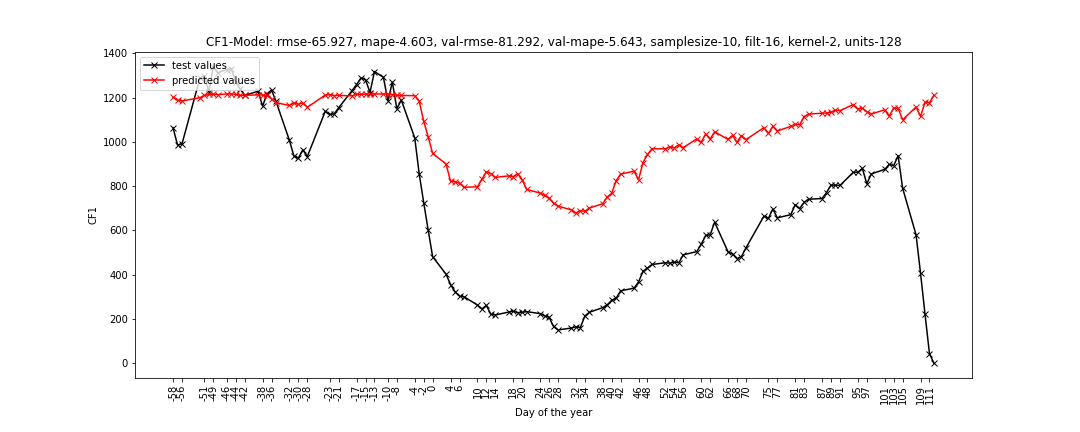


*Fig. 9: Prediction of CFuture1 on the test set.*



*Fig. 10: Prediction of CFuture2 on the test set.*

The first graph (cf. Figure 9) shows the prediction for CFuture1 on the test dataset. This yields an RMSE of 65.926 and an MAPE of 4.603%. The second graph (cf. Figure 10) shows the prediction for CFuture2 on the test dataset. This yields an RMSE of 109.339 and an MAPE of 4.143%. (On the x-axis the number of the ‘Day of the year’ in 2019 is plotted.) In comparison between CNN and CNN-LSTM, one can see that both models are able to catch major trends. Nevertheless, whereas the CNN trys to capture the small fluctuations in the data, of which we know that it is important in this domain, the CNN-LSTM is not able to predict those and rather predicts some averaging value for those days. This slow response of the CNN-LSTM gets even worse for predicting CFuture in COVID times (cf. Figure 11). On this data, the CNN performs better but both perform worse than classic machine learning models.



*Fig. 11: The optimized CNN-LSTM model trained on the training- and testset and validated on the COVID dataset.*

Against our assumption that classic machine learning techniques would outperform the Neural Network approaches in time series forecasting, we got surprisingly good results and found out that our two models can keep up with the classic models and sometimes even slightly outperform those. However, since both CNN and especially CNN-LSTM models need more time and effort than classic models need for modeling and training, it is questionable if it pays off to invest more effort if one can get almost the same results with more lightweight methods, also given their better interpretability.

**3 Conclusion**

**3.1 Conclusion**

In this project, we successfully trained and fairly validated different models from different machine learning fields for predicting CFuture1 (~ 5% deviation on average for the next 7 days) and CFuture2 (~ 4.5% deviation on average for the next 14 days) for the problem at hand of predicting Patient Examination Volumes of one department at the Massachusetts General Hospital (MGH). In doing so, we not only learned much about data analysis, model training and evaluation in the time series data domain, but also specifically took away the following learnings:

* *Never underestimate linear regression on structured data:* When we added linear regression as a baseline for our predictions, we were surprised at how small the gaps in target metrics were between much more powerful algorithms such as XGBoost & SVR and linear regression. This probably has to do with the importance of further engineering the core features, e.g. adding delta or delta-delta features, in order to allow the more powerful models to exploit patterns between these augmented features and the core features for improved predictions. Unfortunately, we did not have time for this in the end, but it was nevertheless a valuable learning for us.
* *Our models performed unexpectedly well on truly unseen data*: As mentioned before, models such as XGBoost, SVR, and the deep learning models were able to predict the impact of the COVID crisis on the examination counts for the next 7 and 14 days surprisingly well despite their pre-training on pre-crisis data. This could indicate that these models were able to extract important relationships in the dataset, but requires a deeper analysis. Umtimately, we were surprised how accurately, for example, XGBoost could predict examination counts for a week ahead (CFuture1).
* *Deep learning is not the optimal solution for predictions using structured data*: As we have seen both for predicting future examinations for a week ahead (CFuture1), and for predicting values at the onset of the COVID pandemic, traditional models such as linear regression, XGBoost and SVR performed better than the various deep learning based algorithms on our key metrics (even though these models trained unexpectedly well despite little data). This is of course not the case for other scenarios with more complex input data such as images and videos, but is often the case for structured data, also when interpetability of the algorithms is desired.

**3.2 Mentor Conclusion**

I would like to thank the whole team for the great semester and am glad that the knowledge gained during TechLabs could be applied on such a good level in a short time window in this non-beginner friendly project. I was also able to take some knowledge for myself after this semester, both for in supporting a small research project with a team, but also in the technical area of timeseries forecasting.

**3.3 Team Member Acknowledgements**

Throughout the Techlabs journey, the knowledge related to data science is significantly enhanced by completing the track as well as the practical project. Moreover, the MGH project provides students with the opportunities to conduct a direct conversation with the director of medical analytics at MGH hospital, which is a valuable experience. The well-organized event, reliable assistant and passionate team members are the key attributes of a successful team project.

**Contact Details**

***Team:***

* Johannes Ehls: [LinkedIn](https://de.linkedin.com/in/johannes-ehls)
* Wassim Masmoudi: [LinkedIn](https://www.linkedin.com/in/wassim-masmoudi-958b69ab/)
* Hanna van den Munkhof: [LinkedIn](https://www.linkedin.com/in/hanna-van-den-munkhof-27837550/)
* Kathrin Nieswiec: LinkedIn
* Xin Shen: [LinkedIn](http://www.linkedin.com/in/xin-shen-)
* Konstantin Wehmeyer [LinkedIn](http://linkedin.com/in/konstantin-wehmeyer-233b88128)

***Mentor:***

* Rene Ahlsdorf [LinkedIn](https://www.linkedin.com/in/rene-ahlsdorf/)

*TechLabs Aachen e.V. reserves the right not to be responsible for the topicality, correctness, completeness or quality of the information provided. All references are made to the best of the authors’ knowledge and belief. If, contrary to expectation, a violation of copyright law should occur, please contact*[*journey.ac@techlabs.org*](mailto:journey.ac@techlabs.org)*so that the corresponding item can be removed.*