TEAM PROJECT - Final Report 2025

Continuous Glucose level prediction

Marta Animucka¹, Anna Kaczyńska¹, Juliusz Korc¹, Tu Nguyen¹, Stanisław Podogrocki¹,

Marcel Wieczorek¹, Olaf Żurawski¹

1Lodz University of Technology, International Faculty of Engineering, Lodz, Poland

Supervisors: Daniel Jodko (e-mail: daniel.jodko@p.lodz.pl), Michał Kulak (e-mail: michal.kulak@p.lodz.pl).

ABSTRACT The project focuses on applying machine learning techniques to tackle prediction of dangerous glucose levels in type 1 diabetes. For this purpose, we used an open-source *O’Malley* dataset of glucose level measurements from 168 patients. The purpose is to provide a reliable method to help patients react in time, in the event of their bodies entering a critical diabetic state. And the main goal is to determine which algorithm, Long Short-Term Memory (LSTM) or AutoRegressive Integrated Moving Average (ARIMA), performs better for this task. Our results indicate that the ARIMA model outperforms LSTM in terms of prediction RMSE, making it a promising choice for future applications.

KEYWORDS ARIMA, glucose level prediction, LSTM, machine learning, type 1 diabetes, O’Malley

I. INTRODUCTION

Type 1 diabetes is a chronic condition that affects millions of people. It requires lifelong management of blood glucose levels to prevent dangerous health complications. According to estimates from 2021, approximately 537 million people worldwide suffered from it [1]. The condition affects individuals and challenges healthcare systems in preventing dangerous blood sugar levels. This highlights the urgent need for advanced tools, which would help in managing the crisis.

The project is performed as part of Process Modeling for Industry 4.0 course. The Fourth Industrial Revolution is characterized by rapid improvements in data technologies and artificial intelligence. This gives a great number of opportunities to address challenges through new, innovative solutions.

We are inspired by the need to connect healthcare and technology to create impactful and groundbreaking systems. We aim to demonstrate how machine learning can be used to improve quality of life for individuals with type 1 diabetes. Using a dataset called O’Malley from 2021, which comprises six months of continuous glucose monitoring data from 168 patients [2], we implemented and evaluated two predictive models: Long Short-Term Memory (LSTM) [3] and AutoRegressive Integrated Moving Average (ARIMA) [4]. LSTM is a deep learning method known for handling complex time-series data, while ARIMA is a statistical forecasting technique. By comparing these approaches, we aim to identify the most suitable model for this application.

Using advanced machine learning techniques, this project underlines the importance of predictive models in healthcare applications. The insights gained from comparing LSTM and ARIMA models not only highlight the strengths and limitations of each approach but also uncover area for improving predictive tools to better suit the needs of patients with type 1 diabetes. Such innovations can support health management, lower the risks associated with glucose level fluctuations, and improve the quality of life for millions of people.

II.  PROBLEM FORMULATION

A.  STATE OF THE ART

Due to the large number of people affected by the type 1 Diabetes there are many teams currently working on developing the best solutions. By researching the recent scientific papers, we were able to determine most popular machine learning models used for predictions of glucose levels.

Traditional machine learning models, such as linear regression and Ridge regression [11], were among the first approaches used to model glucose trends but because of their simplicity, they were unable to capture the complex and non-linear changes in glucose levels. None of these models considers the dynamic and time dependent nature of glucose metabolism. However, these were only the first approaches to Data Science itself, which explains why it was not optimized for this problem. Next generation of glucose prediction models utilized methods created specifically with time-series in mind, such as Autoregressive Integrated Moving Average (ARIMA) or Seasonal Autoregressive Integrated Moving Average (SARIMA). Those models brought a significant improvement in short-term glucose predictions, resulting in predictions accurate enough for real life applications. They excel in capturing trends, and in case of SARIMA also seasonality, which due to the nature of diabetes, like rapid changes after meals, is a significant factor in terms of predicting the future levels. Since future glucose levels are strongly connected to their immediate past values, in case of short-term predictions, these models were most commonly used among researchers [12].

The next generation of models used for Glucose Predictions came with the introduction of Deep Learning models. The whole field of Neural Networks, Recurrent Neural Networks (RNN) were designed specifically with sequential data in mind. The research of those models was extremely accelerated due to their application in Stock predictions, which highly benefited Glucose levels predictions. Examples of RNN used for this task include Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) [5]. Those models contribute to the clear majority of the models used for the predictions, with most of the current research focusing on optimizing and modifying them. Models based on LSTM architecture also proved to be superior in the context of learning physiological relationship between user entered inputs, such as consumed substances, and glucose prediction [6]. LSTM also show a promising ability to retain information over long sequencies, allowing for prediction over significantly longer times than traditional Machine Learning algorithms.

The newest invention in the field of Deep Learning, Transformer based architecture was also employed to the task of predicting glucose levels [7]. While it proved to be slightly superior to current Deep Learning models, the increase in accuracy is only significant in very certain cases, when the input from the Continuous Glucose Monitoring (CGM) device is smoothed or imputed.

B.  MODEL SELECTION AND JUSTIFICATION

Based on our research of existing models and other scientific papers [9], [10] we concluded that LSTM and ARIMA models would most fit our project. They are both characterized by great performance on continuous time series data which makes them well suited for glucose level predictions.

Root Mean Square Error (RMSE) is one of the most common performance evaluation methods for regression machine learning models. It was the main method of model evaluation and comparison in the previously cited literature. We will stick to it throughout our project. RMSE can be calculated using the formula in Eq.1:

(1)

It calculates the average difference between predicted and actual values. The lower the RMSE, the better the predictions of the model.

ARIMA model was chosen on account of its high performance during short-term predictions of glucose levels. It captures trends, which makes it efficient in detecting their sudden fluctuations. They change dynamically and often show short-term correlations, thus the autoregressive and moving components of the ARIMA model, make it perform proficiently in predicting values in the next few time steps. Moreover, ARIMA shows high interpretability which is an important factor in medical applications.

We chose the LSTM model owing to its good performance dealing with long-term and non-linear relationships. The glucose level may be affected by many different factors, such as the meals a person eats throughout the day or even conditions like pregnancy. The model performs effectively with such time evolving complex patterns. Its memory cell mechanism keeps important information from the past, which makes it more efficient than its conventional alternatives for medium and long-term predictions. Additionally, it performs better with continuous time series, so it is a great choice for working with continuous glucose monitoring (CGM) data.

We also researched a type of model called Extended Long Short-Term Memory (XLSTM). It’s a new version of LSTM, which shows better memory retention and efficiency. The first scientific paper about the model’s architecture was published at the beginning of December 2024 [8]. It has not yet been used for the glucose level prediction, but it’s supposed to generally perform better than LSTM in most cases. We might continue to work on this model after completing the current stage of the project.

The choice of ARIMA and LSTM models should make our results accurate and convenient for interpretation, which is an important factor while working with medical data. A possible extension of our project may include making a combined model consisting of both ARIMA and LSTM components. Another possibility to improve the results of the previous approaches would be to implement more complex models such as XLSTM.

III. SOLUTION IMPLEMENTATION

A. LSTM

The LSTM model can recognize patterns in a time series and use them to expand it. As input it takes a vector of data points and as output it returns another vector. For it to train it must be supplied with a predetermined set of such vectors. You can imagine a company that tracks the number of sales they make each month. They see that every time their sales drop three months in a row, they grow in the next two after that. Now whenever the company comes up against three months of downturn, they know what to expect next. The same way the model uses previously prepared input and output vectors to adjust and optimize its weights, so that later it can use that information to make its own predictions. To make them accurately, it needs a very large number of such data vectors. A training dataset is simply a large collection of input and output vectors from which the model learns.

A purple circles with numbers

AI-generated content may be incorrect.Moving on to the implementation process. In our case, we must create a training set for our particular application. As mentioned previously, the data we use in the project is a very large time series of glucose level measurements that were made continuously every 5 minutes for six months. They are equally separated in time. Which means, they can easily be distributed without any additional scaling. In fact, a time axis is not needed, because elapsed time can be calculated from the number of datapoints in a vector.

A group of purple circles with white numbers



AI-generated content may be incorrect.First the whole series is arranged in a very long one-dimensional vector of values, see FIGURE 1.

FIGURE 1. An exemplary representation of an interval of the long one-dimensional vector of glucose measurements.

It’s time to build the input. Different models are good at processing input series of some certain length ranges, while being worse at processing some others. In fact, the sweet spot for the length of the input vector, can even vary in the same type of a model depending on the data it is supplied with. Every dataset is characterized by its own distinct features and patterns. In testing, we concluded that for our LSTM, the data set that we used, and the prediction length we expect, about 40-minute input intervals were the best. To represent a 40-minute glucose level progression we need 8 consecutive measurements. Every data point in the long one-dimensional series was considered as a starting point of a new vector of length 8. These were all put into arrays (input vectors) and stored all together in another array. That’s what makes our input training set.

Now, the output. First, we must specify how long an expansion we want the model to produce. To make the least space for mistake it must be as short as possible. Conversely, the model is trained to predict future glucose levels, so that it can foresee dangerous rises or drops, and warn the user early enough, so they have time to react. Rapid acting insulin has an onset time of 15 minutes. Which means the user must be warned at least 15 minutes before their insulin levels pass the dangerous threshold. That’s the length of expansion we need. Each output vector, corresponding to each input vector must represent a time series of 3 data points. The ones that in the long one-dimensional vector come after the initial 8 from the corresponding input array. Moreover, we found that, if the model is pushed to predict the first data point in the output vector to be the same value as the last data point in the input vector, the further data points are predicted more accurately. Thus, each output vector was prepended an additional value at the beginning that is a duplicate of the last one in its corresponding input array. That makes 4 values in total. If we look at the 11 datapoints in FIGURE 1 we can represent them as a single input vector that starts at value 172 and a corresponding single output vector that starts at value 155. Such distribution can be seen in FIGURE 2.

**FIGURE 2**. **LSTM input and output vector visualization. The 8 initial values represent an input set. The next 4 form the output. The first output value is a duplicate of the last input value. It’s colored in cyan.**

This logic can be implemented in a python function as presented below:

def df\_to\_XY(df, window\_size=8, time\_to\_predict=3):

NP = df.to\_numpy()

X = []

Y = []

for i in range(len(NP) - window\_size-time\_to\_predict):

row = [[a] for a in NP[i:i+window\_size]]

X.append(row)

Y.append(NP[i+window\_size-1:i+window\_size+time\_to\_predict])

return np.array(X), np.array(Y)

The parameter **df** is a data frame containing all the consecutive glucose measurements, **X** is the array with model input vectors, and **Y** is the array with model outputs for training. The **window\_size** is the range of how many data points the model is supposed to base its predictions on. And **time\_to\_predict** is how many data points the model is supposed to predict. As explained before, the code creates every possible consecutive interval of the size **window\_size**, and appends it to **X**, while in the same iteration creating the next interval of size **time\_to\_predict** plus one duplicated node form the inputinterval and appends it to **Y.**

Later the arrays are distributed into the training, testing and validation sets, where 80% of the data goes to the training set, 10% goes to the testing set and the last 10% is used for validation. It was implemented as indicated below:

X, Y = df\_to\_XY(values)

train\_size = int(X.shape[0] \* 0.8)

validation\_size = train\_size + int((int(X.shape[0])-train\_size) / 2)

X\_train, Y\_train = X[:train\_size], Y[:train\_size]

X\_val, Y\_val = X[train\_size:validation\_size], Y[train\_size: validation\_size]

X\_test, Y\_test = X[validation\_size:], Y[validation\_size:]

The model contains an input layer, 64 LSTM units in a single layer, and two dense layers to reduce dimensionality for the output. It optimizes to reduce the RMSE of the prediction just as in the case of ARIMA. This structure can be seen below:

model1 = Sequential()

model1.add(InputLayer((8, 1)))

model1.add(LSTM(64))

model1.add(Dense(8, 'relu'))

model1.add(Dense(4))

# model1.summary()

cp = ModelCheckpoint('modelLSTM/model.keras', save\_best\_only=True)

model1.compile(loss=MeanSquaredError(), optimizer=Adam( learning\_rate=0.0001), metrics=[RootMeanSquaredError()])

model1.fit(X\_train, Y\_train, validation\_data=(X\_val, Y\_val), epochs=100, callbacks=[cp])

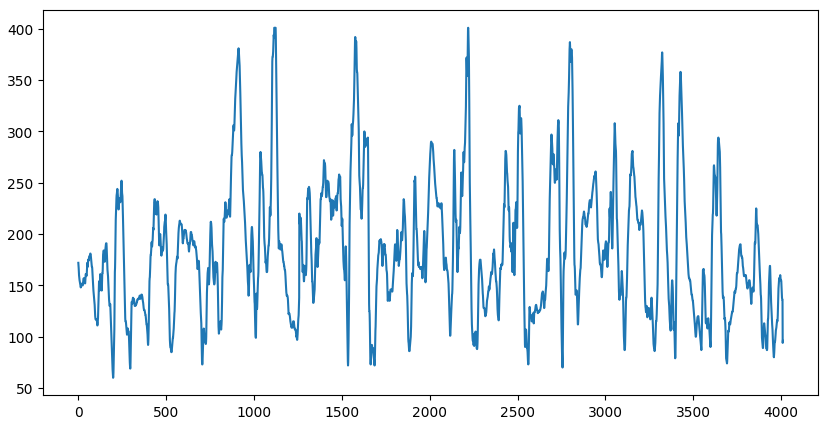
B. ARIMA

Since there were no signs of seasonality and extra exogenous variables on our dataset (generated by the code below and presented in FIGURE 3), we decided to go with the traditional ARIMA model. It is a statistical analysis model, which takes the input of a time series to forecast future data points that are best fitted to that time series. The Auto Regressive term (AR) represents that the evolving variable of interest is regressed on its prior values. The Integrated term (I) represents the differencing of raw observations to allow the time series to become stationary. The Moving Average term (MA) indicates that the regression error is a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The code below shows how we generated FIGURE 3.

Y = glukoze\_levels.iloc[:,1]

y.plot(figsize=(10, 5))

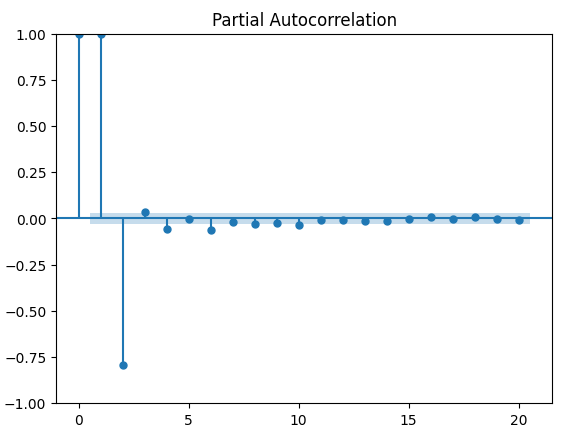
plt.show()

**FIGURE 3.  Example of glucose levels for one patient.**

After analyzing the data using the autocorrelation and partial autocorrelation plots (generated by the code below and presented in FIGURE 4), we decided to use the set of parameters (2,0,0) since only the AR term had a significant impact on the time series and more specifically, lags number 0, 1 and 2. Parameters (2,0,0) indicate that the Auto Regressive term is set to 2 while Integrated parameter is 0, since the function does not need to be integrated to become stationary. Moving Average being 0 indicates that past errors do not affect future values.

plot\_pacf(glukoze\_levels.iloc[:,1], lags=20)

plt.show()

**FIGURE 4.  Autocorrelation plot.**

Similar to our approach with the LSTM model, we tried to predict the glucose level for 4 future nodes (15 minutes). Initially, we chose a 40-minute interval to run the model (similar with LSTM), but the results were disappointing considering the prediction could not capture the current trend of the time series and the Root Mean Square Error (RMSE) was high. Based on that, we alternated the model into 4 hours of training data – 15 minutes of prediction, and the results were more promising in both ways of the evaluation that were mentioned. The execution time of the program was also acceptable, and the size of training data could not be enormous considering it could capture more outliers, making the model inaccurate.

The code for the ARIMA model is presented below. As previously mentioned, the split for training data – predictions were 4 hours – 15 minutes, and after predicting one data point, we appended it into the training data and remove the first data point in the training set, making the training set always be the same length.

my\_order = (2,0,0)

predicitions2 = list()

for i in range(4):

model2 = ARIMA(train\_data, order=my\_order)

model\_fit2 = model2.fit()

output = model\_fit2.forecast()

predictions2.append(output.iloc[0])

train\_data = lim\_glukoze\_levels[i+1:48+i+1][‘CGM’]

The code below shows the comparison between the predictions and the actual glucose levels, along with the Root Mean Square Error. We take predicted data from previous functions and using root of mean\_squared\_error function we calculate rooted MSE which in the end is the RMSE.

test\_data = test\_data.reset\_index(drop=True)

rmse = sqrt(

mean\_squared\_error(test\_data, predictions2)

)

rmse = sqrt(mean\_squared\_error(test\_data, predictions2)))

plt.plot(test\_data, label = ‘test’)

plt.plot(

predictions2, color = ‘red’, label=’predictions’

)

plt.xticks([0,1,2,3])

plt.legend()

plt.title(

plt.title(

‘Patient {}, RMSE = {:.2f}’.format(ptid, rmse)

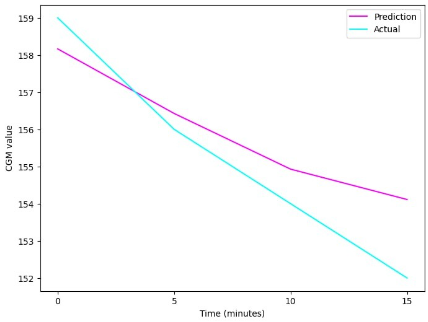
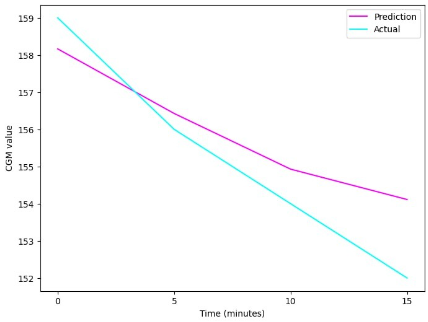
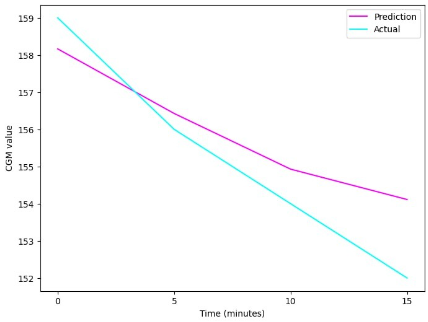
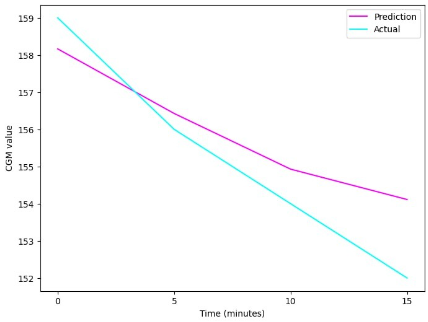
)

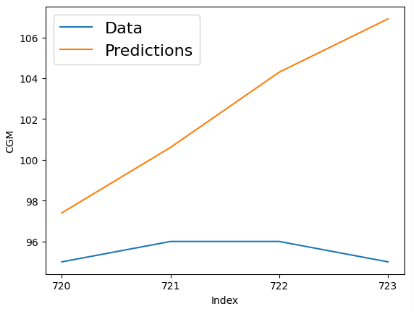
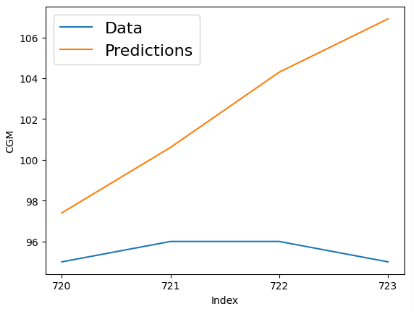
plt.show()

It is worth mentioning that presented code creates a graph that is scaled according to the differences between the predicted and actual values and can present almost perfect predictions as less accurate than they really are. That is why we advise to pay attention to the range of the scaling.

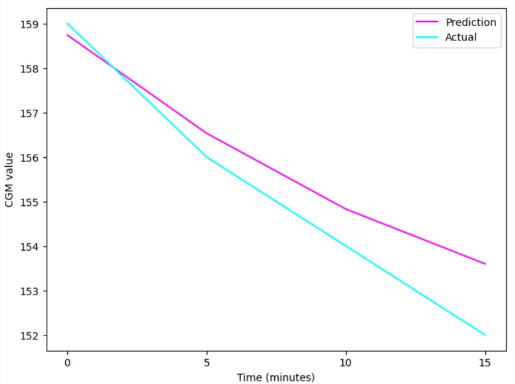
IV.  WAYS OF VERIFICATION

After implementing both ARIMA and LSTM models we decided to compare them. While initially the LSTM performed better, it took some adjustments for ARIMA to at least match or even outperform LSTM. The first comparison results are shown in FIGURE 5 and FIGURE 6:

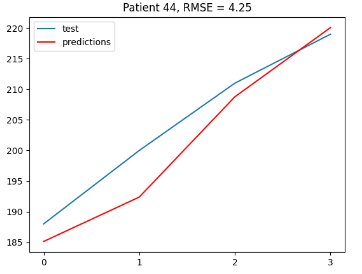
**FIGURE 5.  Early prediction of glucose levels by LSTM model.**

**FIGURE 6.  Early prediction of glucose levels by ARIMA model.**

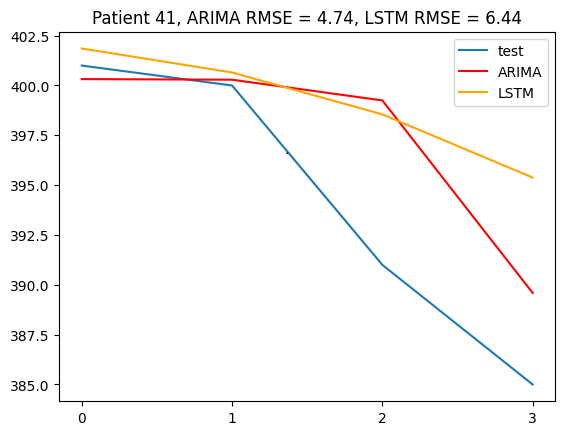
Looking at the example images we can see that even though both models were still in need of improvements, ARIMA was generally way more chaotic. It was caused by trying to force both models to predict based on the previous 35 minutes. While LSTM could learn and at that time needed more data, ARIMA is a purely mathematical model and small ‘training’ size affected ARIMA much more. The average RMSE of LSTM was around 6.39 while ARIMA was a bit higher with some values even being as low as 0.77 which showed its unreliability. After that we decided to improve both models, LSTM by giving it more data to train on and ARIMA by adjusting parameters and changing the size of the training data. The results were as shown in FIGURE 7 and FIGURE 8:



**FIGURE 7.  Final prediction of glucose levels by LSTM model.**

**FIGURE 8.  Final prediction of glucose levels by ARIMA model**.

After making adjustments to both models, the results improved with ARIMA significantly. In the end ARIMA performed better at predicting 15-minute intervals while we believe that LSTM might perform better for longer ranges. Following that, we can say that ARIMA outperforms LSTM in short ranges with LSTM having higher potential for longer intervals. Additionally, ARIMA requires a longer past time interval in order to make a prediction while LSTM requires shorter one, although it needs a relatively big dataset to train on. A graph comparing both models at the same time predicted time intervals is shown below in FIGURE 9:



**FIGURE 9.  Final comparison of ARIMA and LSTM models.**

V. CONCLUSIONS AND PERSPECTIVES

Type 1 diabetes is a condition that would certainly be less dangerous and harmful for health and well-being with the help of monitoring systems based on machine learning. During our project we compared two approaches, and found that ARIMA algorithm performs better on a database of our choice. When looking at the plot comparing both models (Fig. 9), we can see that ARIMA is more sensitive, responding to small changes in the value of CGM faster than LSTM. We can also see that in our comparison ARIMA obtained slightly better RMSE score of 4.74, compared to LSTM’s score of 6.44. This results correspond to results of other scientific articles comparing both models [12], with predictions of ARIMA following the actual values much more accurately. As for the RMSE score, the ratio found in the literature [12] also confirms that LSTM performs worse, with it’s score being almost 3.5 times higher. This is a much greater difference than in our case, however the models trained in this article used only one month of CGM data, which does explain the poor results of LSTM model, which needs a lot of training data to tune to the changes of values.

It is also important to mention that we did not train our models on the whole dataset we have found, the reason being that it contained too much information and datapoints for the devices we used to comprehend. We were only able to train the LSTM model on 50 out of 168 patients and since the ARIMA one was solely based on mathematical calculations, it did not need to be trained. ARIMA was just presented with time intervals to base its computations on to create the predictions.

Some members of the group also tried to use the improved version of the latter model – xLSTM which was relatively new to the industry and could outperform ARIMA, but the research was too novel and complex, which caused that the project had to be terminated when it comes to this improvement. We do think, however, that the xLSTM model can be promising and worth pursuing, just not with tools and experience we have currently.

Even so, the project has proven to be successful, and we gained truly qualitive knowledge about conducting medical projects as well as working with machine learning models. We are of the opinion that we did provide two models which are precise enough to help predict the blood sugar level over time and avoid its critical values. Those models can be used as a part of an app that, connected with a glucose monitor, could alert the user to try adjusting their blood sugar level.

REFERENCES

1. *Armocida B, Monasta L. (2024) The Burden of Type 1 and Type 2 Diabetes Among Adolescents and Young Adults in 24 Western European Countries, 1990–2019: Results From the Global Burden of Disease Study 2019*
2. [*https://github.com/IrinaStatsLab/Awesome-CGM/wiki/*](https://github.com/IrinaStatsLab/Awesome-CGM/wiki/)
3. *Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation*
4. *Liu Q, Liu X, Jiang B, Yang W. Forecasting incidence of hemorrhagic fever with renal syndrome in China using ARIMA model. BMC Infect Dis. 2011*
5. *W. Mack, U. Bharadwaj, S. Chakrabarty and E. A. P. Habets, "Signal-Aware Broadband DOA Estimation Using Attention Mechanisms," ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*
6. *Prendin, F., Pavan, J., Cappon, G. et al. The importance of interpreting machine learning models for blood glucose prediction in diabetes: an analysis using SHAP. Sci Rep 13, 16865 (2023)*
7. *Acuna, Edgar & Aparicio, Roxana & Palomino, Velcy. (2023). Analyzing the Performance of Transformers for the Prediction of the Blood Glucose Level Considering Imputation and Smoothing. Big Data and Cognitive Computing.*
8. *Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, Sepp Hochreiter (2024) xLSTM:* *Extended Long Short-Term Memory*
9. *van Doorn WPTM, Foreman YD, Schaper NC, Savelberg HHCM, Koster A, van der Kallen CJH, et al. (2021) Machine learning-based glucose prediction with use of continuous glucose and physical activity monitoring data: The Maastricht Study. PLoS ONE 16(6): e0253125*
10. *Ming Cheng, Xingjian Diao, Ziyi Zhou, Yanjun Cui, Wenjun Liu, Shitong Cheng (2024) Toward Short-Term Glucose Prediction Solely Based on CGM Time Series*
11. *Mravik, Miloš & Th, Vetriselvi & Venkatachalam, Kv & Šarac, Marko & Bacanin, Nebojsa & Adamović, Saša. (2021). Diabetes Prediction Algorithm Using Recursive Ridge Regression L2. Computers, Materials & Continua. 71. 457-471. 10.32604/cmc.2022.020687.*
12. *Umapathi Krishnamoorthy, V Karthika, M K Mathumitha, Hitesh Panchal, Vijay Kumar S Jatti, Abhinav Kumar, Learned prediction of cholesterol and glucose using ARIMA and LSTM models – A comparison, Results in Control and Optimization, Volume 14, 2024, 100362, ISSN 2666-7207.*