# Summary

[**Summary 2**](#_djo5olqv0zjr)

[**I. Introduction with a description of the dataset 3**](#_7d1wcpkqtd20)

[A. Overview of the problem 3](#_3zq0iarwgfm3)

[B. Description of the dataset 3](#_kozextypq6k2)

[C. Description of the protocol 4](#_vxxqkdoa9k61)

[**II. Theory and method 4**](#_z7dk3irvzis0)

[A. Data analysis 4](#_xt7xnsnzw7jo)

[1. Data representation 4](#_z8x40i64862c)

[2. Stationarity 5](#_bea32nh0je6k)

[a. Moving statistics 5](#_akd04rmedd0g)

[b. ADF and KPSS test 8](#_a7pg5tk6rm65)

[3. Correlation coefficient 8](#_sm9bcufuc8vq)

[B. Preprocessing of data 9](#_89z2dwwecylo)

[1. Outliers and NaN values 9](#_r4zfil379zds)

[2. Interpolation 10](#_n5iybqfnld5l)

[3. PAT inversion 11](#_a8364ud9akmd)

[4. Normalization 11](#_41g0ixn6ckc3)

[C. Machine Learning and Deep Learning models 12](#_95nmsee8cbqn)

[1. Regression models 12](#_kzq7izk9ai7a)

[2. Cross-validation methods 13](#_3kwduyxz5o3v)

[3. Metrics 14](#_yelfsj3t568w)

[**III. Interpretation of the results 15**](#_q18qd1kgzbe5)

[A. Machine Learning models 15](#_7e743dldsbma)

[1. Impact of parameters 15](#_d49h6py5yvso)

[2. Choice and design of the best model 18](#_2d2207dkpw14)

[3. Going deeper into the models outputs (predictions vs expectations): 18](#_xyzg1w9udn22)

[B. Deep Learning models 19](#_owz3eu4njqws)

[**IV. Conclusion and recommendations for future development of study 20**](#_yv9wigob2ryl)

[**Appendix 22**](#_bxas7mvj9shm)

[Appendix 1 : Preprocess and computation of the accuracy (models.py): 22](#_d7hsxxedn50)

[Appendix 2 : Preprocess and computation of the accuracy (models.py): 22](#_qtvqhe3iqpg4)

[**Bibliography 27**](#_3hg1t0b44vt5)

# 

# Introduction

## Overview of the problem

Blood pressure (BP) is an important cardiovascular health indicator, as hypertension (high blood pressure) is a major mortality factor. It is usually measured by cuff monitoring, but this is uncomfortable for the patient, limited to daily routine, and it provides sparse measurements in time. Therefore, it is desirable to improve the method of blood pressure measurement, to develop a non-invasive, continuous, and portable BP monitoring device. Preliminary results (trial still running) have shown that building an accurate BP meter requires exploring the full range of BP in response to physiological mechanisms.

This project investigates the potential of Machine Learning (ML) based BP estimation from pulse arrival time (PAT), which is a feature derived from optical photoplethysmography (PPG) signals, computed for each heartbeat. The estimation of BP from PAT is a difficult and uncertain process, because… In part II, an essential part of the investigation is to understand the relationship between the two signals, in terms of stationarity analysis and how to correctly perform the preprocessing of the data. Further, the project explores how to fit machine learning (ML) model to the data and perform cross-validation, regarding the fact that we are dealing with time series. In part III, the performance of prediction by the different models are evaluated with the root mean square (RMSE) and R^2 as statistical metrics. Further, investigation on deep learning models is also done, both with a neural network taking time into account and not. To be cont/improved. Finishing sentence of final goal / recommondations/findings

## Description of the dataset

The data set used in this project was collected with one subject performing several physical activities in the context of a clinical trial organized by CEA researchers. The protocol is detailed in the following section.

The dataset is an excel file with 8 columns corresponding to the different metrics measured for the subject. There are 2342 rows, where each row corresponds to an acquisition point, or in other words, one row for each heartbeat of the subject.

The columns are the following :

* Date : date of the acquisition with the precise time
* Delay (s) : time between the acquisition and the beginning of the process
* Raw PAT (s) : raw data acquired
* PAT filtered (s) : PAT filtered manually to remove the outliers and the points without sense
* Mask : mask used to filtered the PAT
* Blood pressure systolic (mmHG)
* Blood pressure mean (mmHg)
* Blood pressure diastolic (mmHg)

For this project, the mainly concerned data is the blood pressure systolic (named blood pressure in the report for simplicity) and the PAT filtered.

## Description of the protocol

Based on preliminary results (trial still running), it has been shown that building an accurate BP meter requires exploring a full range of BP in response to various physical mechanisms. This is the reason that the research protocol for this particular project combines relaxing and physical activities. The protocol for the research is provided in Table (..):

|  |  |
| --- | --- |
| **Description** | **Duration(s)** |
| Rest with PPG on the left arm | 306.7070 |
| rest with PPG on the right arm | 388.3910 |
| 3 min of static pilates ring, followed by rest | 376.6710 |
| 3 min of dynamic pilates ring, followed by rest | 207.0430 |
| rest | 109.6990 |
| 3 min of mental calculation, followed by guided relaxation | 564.9720 |

*Table 1 : Protocol description*

### 

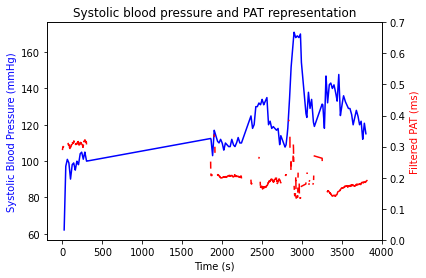
# Theory and method

## Data analysis

### Data representation

The first part of the project is dedicated to getting an understanding of the data and the relationship between the blood pressure and the PAT filtered.

To do a first-hand visualization of the two features we plot the systolic blood pressure (BP) as a function of the PAT filtered, as shown in Figure 1. From this plot, we observe that the PAT is not continuous. This is due to the fact that some measurements with values outside a certain threshold range have been removed, resulting in “holes” in the data. For the modeling part, we have to make this continuous, which is done by linearly interpolating the data. The explanation for how this is performed is given in the part describing the preprocessing of the data.

.

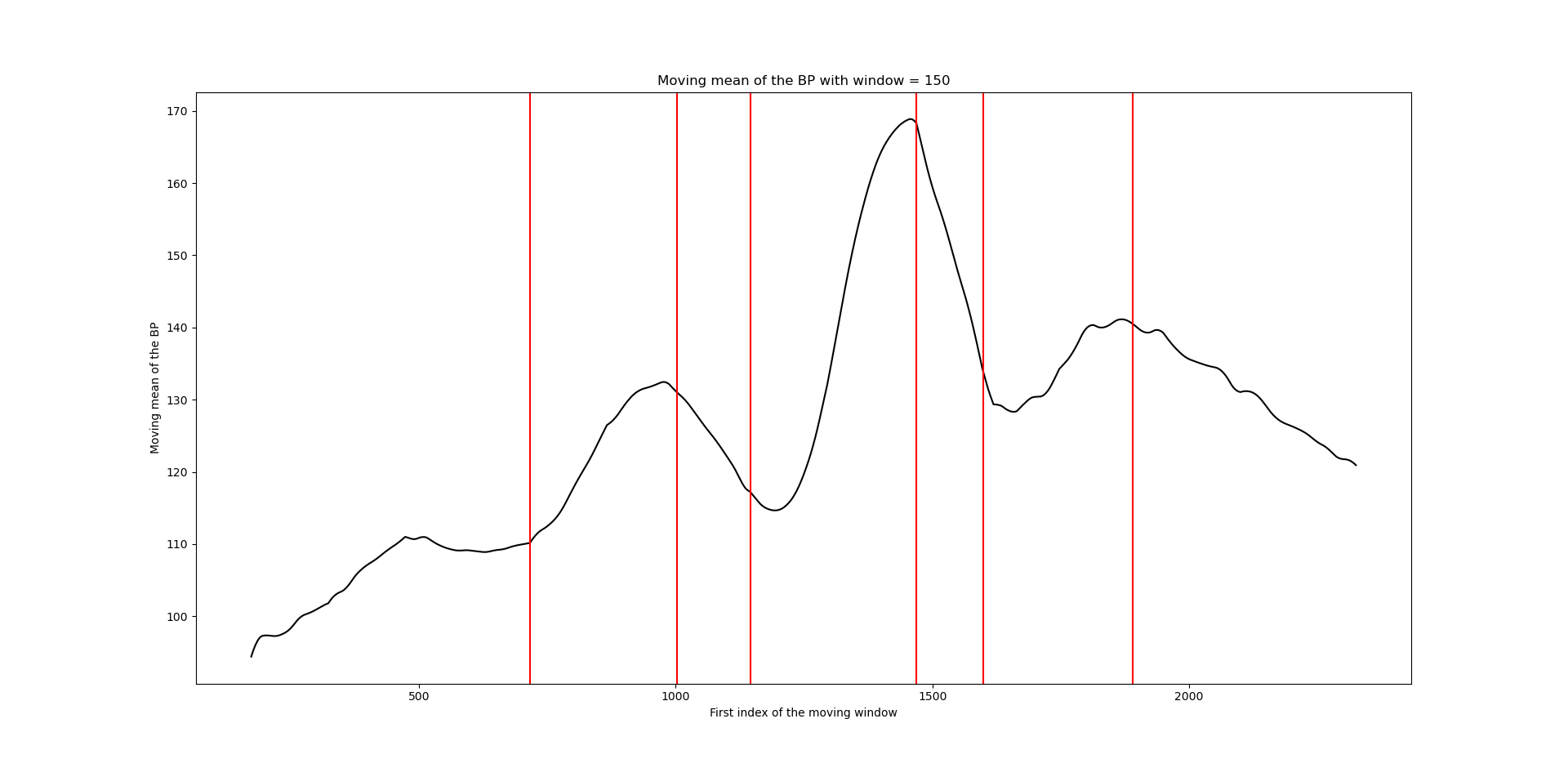
*Figure 1: Systolic blood pressure and PAT filtered (the data from the original excel file) plotted as function of time.*

### Stationarity

When working with time series data, stationarity analysis is crucial, as many ML models assume stationarity to make accurate predictions. In addition, for our particular interest, this analysis is an important part of achieving a better understanding of the data. Stationarity means that the statistical properties of a process generating time series, such as the mean, variance, and autocorrelation of the data, do not change over time. Thus, if the time series has trends or seasonality, it is not stationary. There are several ways to assess the stationarity of the data, by for example visual interpretation of the series in order to look for trend or seasonality, by splitting the series into several portions and comparing the statistical inference, or by specific statistical tests for stationarity. Below we explore these methods for assessing the stationarity of the blood pressure and the PAT filtered data.

#### Moving statistics

Moving statistics are used to analyze the long-term trends of a series. To investigate the moving statistics in our data, we choose an interval named “window” and calculate statistics on this interval to obtain the first point of the curve. This process is then repeated by shifting the window by one on the x-axis. Firstly, we try two different window sizes : W\_{size,1} = 150 and W\_{siz\_2} = 300. The choice of 150 is because this is the smallest time interval in the protocol, and 300 because it is close to the mean of time intervals. Both these choices yield the same results, and thus the following analysis is carried out for W\_{size,1}= 150.

Moving mean and median:

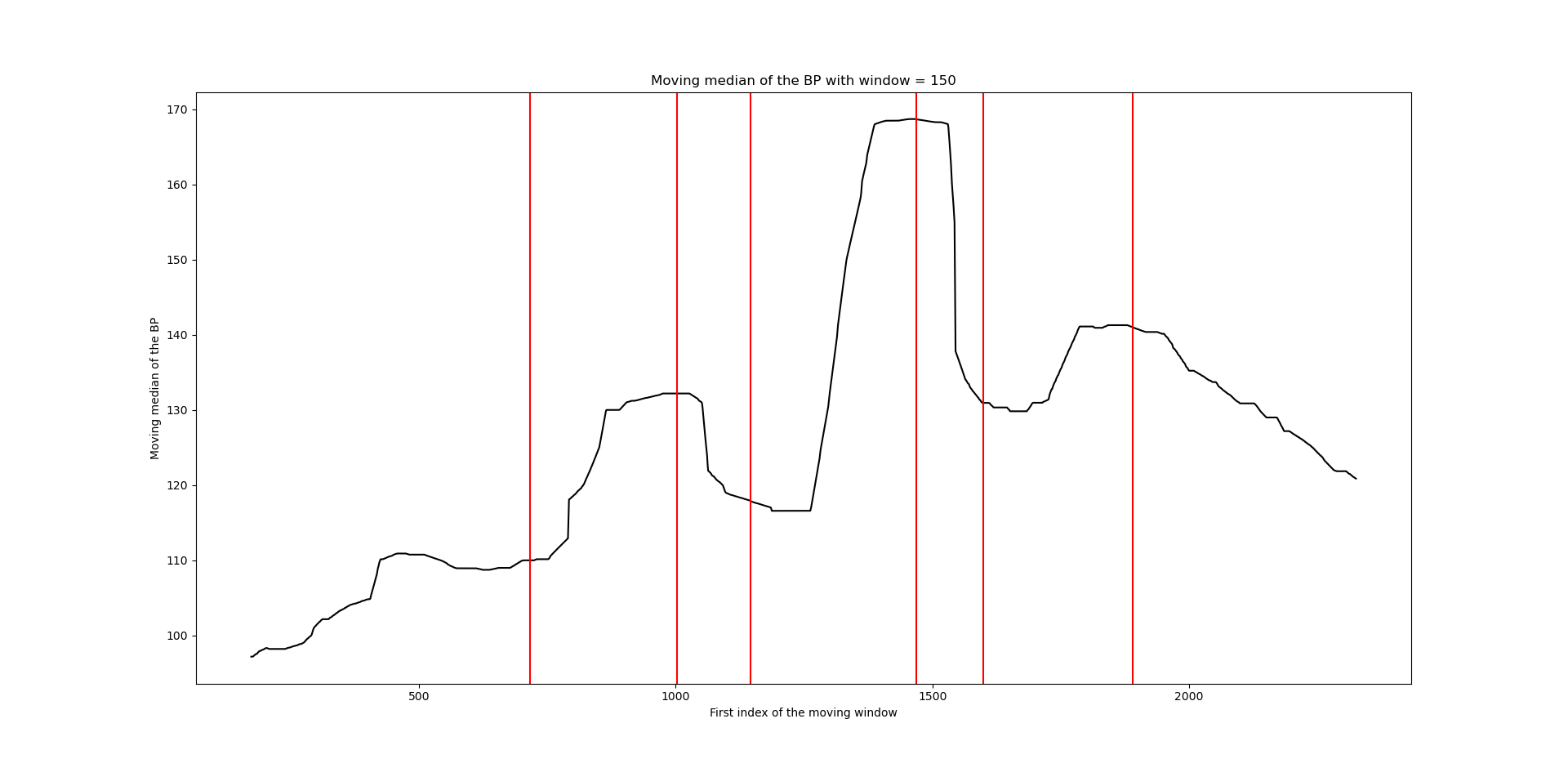
*Figure 2: The moving mean of the blood pressure with window size 150, as function of index of the moving window*.

The moving mean of the blood pressure is shown in Figure 2, with the different periods of activity or rest given by the protocol are labelled with red, vertical lines in the figure, and starting from the left this corresponds to the protocol by the following:

* 1° period: rest: moving mean increase slowly: influence of the second period
* 2° period: pilates statics: increase more quickly
* 3°period: rest: decrease
* 4° period: pilates dynamic: increase quickly
* 5° period: rest: decrease
* 6° period: mental exercise: increase but with variations
* 7°period: rest: decrease

From this figure one can observe that the points where there is a change in the slope correspond quite well with the transition between two different exercises in the protocol. This is interesting because….

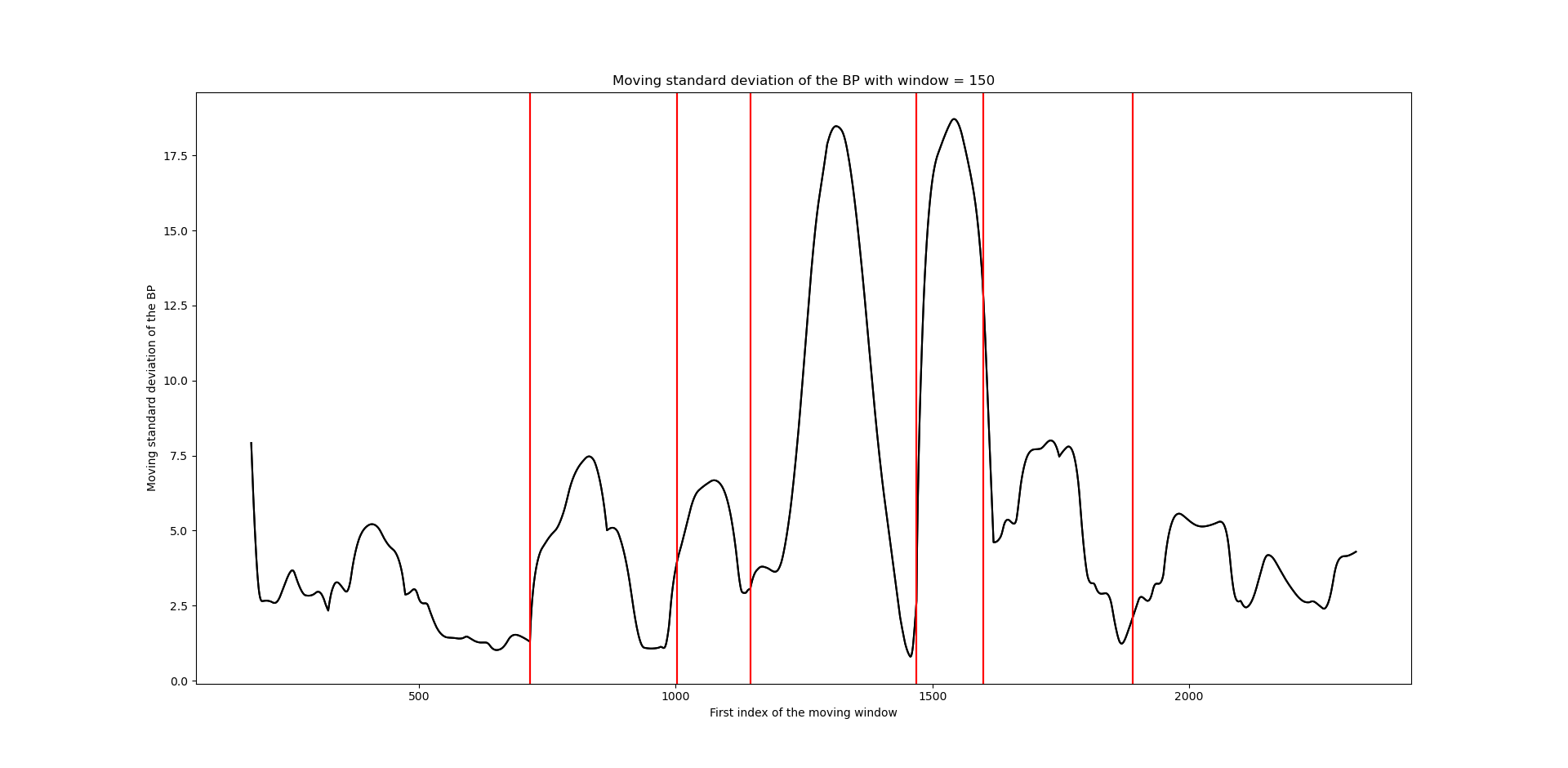
We proceed by plotting the moving median, shown in Figure 3. The analysis of this curve is similar to the analysis of the mean. We observe that the median evolves according to the exercise periods. There is a peak for the dynamic pilates and then decreases during resting periods. What is also observed is that we have periods of stationarity or nearly stationarity, as the median seems to be nearly constant in some regions of the measurement.



*Figure 3: The moving median of the blood pressure with window size 150 as function of window index.*

Moving standard deviation:

To further improve the understanding of the data, we investigate the moving standard deviation of the blood pressure. This is displayed in Figure 4. It can be seen that there are strong variations from the mean for each period of time. The greatest differences are for the periods corresponding to the dynamic pilates and to the rest period just after. This seems logical, as the dynamic period is preceded and followed by a resting period, which implies greater variations in the blood pressure of the subject. This is the same for the second rest-period which is preceded by dynamic pilates and followed by exercises.



*Figure 4: Moving standard deviation blood pressure with window size 150 as function of window index*.

The visual inspection of the moving statistics shows that there is change in the blood pressure according to the type of exercise. The blood pressure decreases at rest and increases during physical or mental exercise. It increases more rapidly when the exercise is more physical (for example during dynamic pilates). Moreover, it can be seen that the blood pressure level during resting time is not the same as in the previous activity. For further development of the research, it could be interesting to compare this with other patients and longer resting times.

#### ADF and KPSS test

For further assessment of the stationarity of our data, we carry out two statistical tests, known as the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) testsKwiatkowski–Phillips–Schmidt–Shin (KPSS) test. These tests are commonly used as a tool to distinguish series that appear to be stationary, to have a unit root and series for which the data(or the tests) are not sufficiently informative to be able to classify the series as stationary or not.

For the ADF test, the null hypothesis that a unit root is present in a time series sample [1]. The unit root is a characteristic of a time series which makes it non-stationary. The statistic of the ADF test is a negative number, which the more negative implies the stronger rejection of the hypothesis that there is a unit root at some level of confidence. Opposed to the ADF,the KPSS test is testing a null hypothesis that an observable time series is stationary around a deterministic trend [2]. Thus, the alternative hypothesis is the presence of a unit root. Additionally, in the KPSS test, the absence of a unit root is not a proof of stationarity but, by design, of trend-stationarity. This is an important distinction since it is possible for a time series to be non-stationary, have no unit root and yet be trend-stationary.

From the tests, it was found that the filtered PAT was difference-stationary, meaning that by performing a differentiation of the data, one could obtain transformed values of the data which can be treated as stationary. However, the PAT matrix used for these computations was obtained by filtering the data to get rid of the empty cases (From 2342 points to 1490). In the process, some measurements that were in reality made at a few heartbeats intervals, were considered to be following each other in time. This could lead to a source of imprecision in the computations of the difference, where the new value for a time t is obtained by subtracting the original one with the value from t-1, such that in regard to the matrix the t-1 value is the previous index although in reality this value can be a few heartbeat older.

A way to resolve this issue could be to first compute an interpolation to find the missing values and approximate a more precise differentiated PAT signal. However, as we were still not so sure of this way to modify the data and as the analysis of the moving statistics showed stationarity in some phases, we decided to proceed working with the non-differentiated data. The preprocessing of this data is described in the next section.

### Correlation coefficient

For further analysis of the relation between blood pressure and PAT, we compute the correlation coefficient between the two features, which is a useful statistic to assess the relationship between two variables. Two common types of correlation coefficients are the Pearson and the Spearman coefficients [3]. A Pearson correlation coefficient measures a linear association between two variables, which is assumed to be normally distributed. A Spearman correlation coefficient is a nonparametric measure of rank correlation and assesses how well the relationship between two variables can be described using a monotonic function. The Spearman correlation coefficient is relatively robust to outliers. Regarding the assessment of the correlation between PAT and BP, the most useful to use is thus considered to be the Spearman coefficient.

Table 1 below displays the calculated Pearson and Spearman coefficients for the BP and PAT. In our case, the Spearman coefficient is of the highest interest. With a value of -0,81, we have an indication of a nearly monotonic relationship between the BP and PAT. Moreover, as the Pearson coefficient has a value of -0,29, we can precise this by saying that there is a non-affine monotonic relationship between the two variables.

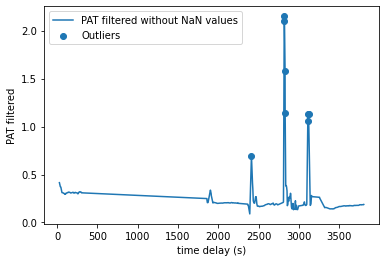
|  |  |
| --- | --- |
| **Coefficient** | **Calculated result** |
| Pearson coefficient | -0,29 |
| Spearman coefficient | -0,81 |

*Table 2: Obtained results of Pearson and Spearman correlation coefficien*t.

## Preprocessing of data

### Outliers and NaN values

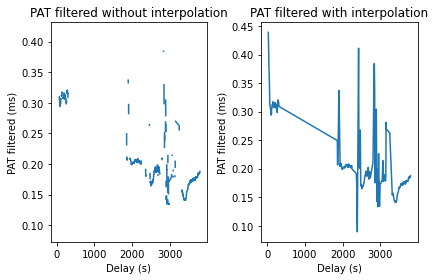
If the data being used for the prediction of a certain parameter contains many outliers, this may contribute to a loss of robustness and accuracy in the predictions. Although the filtered PAT data provided in this project already have been filtered of some outliers, we assess the possibility of some outliers still being present by calculating the z-score on the PAT. The z-score is a statistical measure that indicates how many standard deviations a data point is away from the mean. For our particular case, we set a threshold of z = 3, which is a standard threshold of the z-score, and classify an outlier of the PAT as any point having a larger z-score than the threshold. Points in our data containing NaN values, we replace by the mean PAT, which implies that the z-score of these data points will be calculated to be 0. From the calculation of the z-score, eight points were detected as outliers, shown in Figure 5, which displays the filtered PAT plotted as a function of the time. The detected outliers corresponded to PAT values longer than 1 second. As the PAT is shorter than a heartbeat, and the heartbeat is between 60 and 120 bpm, any PAT longer than 1s is likely to be an outlier, which matches our findings. The found outliers were then removed from the original dataframe.

*Figure 5: PAT filtered plotted as function of the real time, with the outliers marked with points.* 

### Interpolation

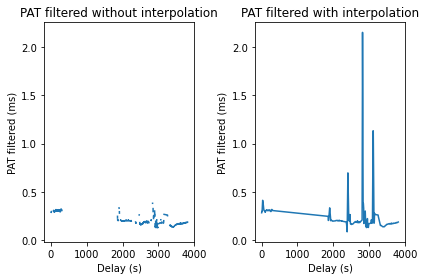
The second pre-processing step is the data interpolation. Some data are missing, particularly for the PAT filtered, so we need to complete the set in order to be able to do a correct temporal analysis on a sufficient number of points that follow each other. To do that, we realize a linear interpolation with the function interpolate.interp1d from the scipy module. This function replaces the NaN values of the PAT filtered by values coming from an interpolation of the other values.

The results are the following :



*Figure 6: PAT filtered after interpolation and without outliers*.

It is important to do the interpolation after removing the outliers to avoid interpolating data containing wrong points. Indeed, as the curve below illustrates, outliers completely modify the interpolation, and the error produced by these points is multiplied.

  
*Figure 7: PAT filtered after interpolation but containing the outliers.*

### PAT inversion

We have seen in the section concerning correlation coefficient that the Spearman and the Pearson coefficients are negative for the PAT and the blood presure. A negative correlation coefficient indicates an inverse correlation between the two variables, which means that the relationship between the variables is reversed: when PAT increases, the blood pressure tends to decrease. This observation gave us the idea of inverting data PAT filtered before applying the model.

PATfiltered = 1/PATfiltered

Advanced machine learning models can of course catch the inverse relationship without this preprocessing step, but it could have an impact on more simple models. For this reason, we try each model with and without inverted PAT. This step is done just after the data interpolation.

### Normalization

Finally, the last step of the data pre-processing is the normalization. We use the standard scaler from the module sklearn.preprocessing over the PAT filtered. This function do the following operation :

PATfiltered = (PATfiltered – mean) / standard deviation

It allows all variables to be scaled to the same scale. In our case, this may improve only slightly algorithm performance, as this is our only input feature. It could become more significant if the model is trained on different patients, to normalize all data between patients in the same way.

## Machine Learning and Deep Learning models

In this project, the following main classes of models were tried out: linear, polynomial, Support Vector Machine model and two deep learning models, beeing the Multilayer Perceptron (MLP) regressor and the Long Short-Term Memory (LSTM) neural network. All these models are regression models, as the goal is to predict the blood pressure from the PAT and not to do a classification. In order to compare the effect of the different preprocessing steps on the prediction accuracy of blood pressure, all models were tried out with the PAT data with no preprocessing, the inverted PAT, with normalization and with both normalization and inverted PAT. In addition, for model selection by cross validation, different approaches of cross validation were applied to each model, beeing train-test split, rolling train test split and time-series-split. The results of all the models were compared by the Root Mean Square Error (RMSE) on test samples and the accuracy score of prediction, R^2. The summary of the results from all the models is provided in the Appendix*.*

### Regression models

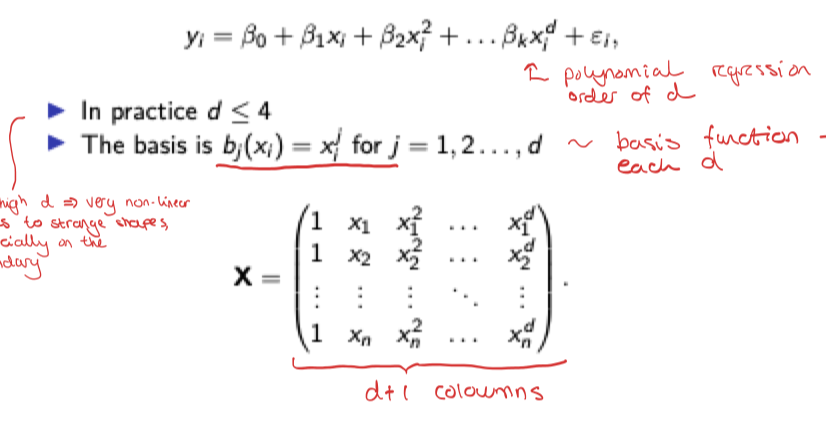
The simplest form for regression is the linear regression model, where the response variable Y is related to the explanatory variable X by

Y =X\*beta + e (error term)

It is a parametric model, with a quantitative response vs one or several explanatory variables, and is a very simple approach for supervised learning. It can be regarded as a too simple model, but at the same time it can be useful to understand this in order to implement and understand the behavior of more complex models, as many other models can be seen as a generalization of the linear model.

**Polynomial regression model**

The polynomial regression model is an extension of the linear regression model, which includes powers of X in the regression. Mathematically, the model is described by:



*(write the formula in latex)*

For this project, we choose a model of degree 3, as this yields the smallest r^2 and rmse compared to degrees of 2, 4 and 5.

**Support Vector Regression (SVRR)** is a regression function that is generalized by support vector machines (SVM), which is a machine learning model used for classification on continuous data *(source)*. Unlike the linear regression, where the goal is to fit a regression line to the data such that the deviation of the data points from the regression line is minimized, the SVR model sets a threshold error allowance around the regression line such that all the data points within this threshold is not penalized for their error. In this project, we use the SVR model from the python machine learning library scikit.

For the deep learning models, we study the performance of two neural networks, being the Multi-layer Perceptron regressor- (MLP) and the Long Short-Term Memory (LSTM) neural network. In general, a neural network is a set of functions inspired by a biological brain. It is composed of different layers, the input layers, with the aim to predict a non-linear process *(source)*.

The MLP regressor implements a multi-layer perceptron (MLP) that trains using backpropagation, using the square error as the loss function and the output set of continuous values *(source)*. An LSTM neural network, is a recurrent neural network (RNN) which has the ability to handle sequential data, such as time series *(source)*. LSTMs address this problem of learning long-term dependencies by introducing a memory cell, which is a container that can hold information for an extended period of time. This memory cell is composed of three gates; the input gate, the forget gate, and the output gate. The input gate controls what information is added to the memory cell. The forget gate controls what information is removed from the memory cell. And the output gate controls what information is output from the memory cell. This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies.

For the purpose of this project, LSTM is expected to be the most interesting, as this is a temporal model, really taking the time dependency of the data into consideration, opposed to the other models. The application of the simpler ML models can be seen as more of an investigation part of the project, useful for studying particularly the importance of taking the time dependency of the data into account when modeling the blood pressure based on the PAT. Especially for further development of the study, with more data and the possibility of adding more features in the prediction of the blood pressure, the LSTM model (or this kind of model), will probably be the most interesting.

\*\* Comparing the MLP (not time into account) - make small paragraph about taking time into account vs not, if it improves the prediction.

### Cross-validation methods

To validate the models, we test its predictions on part of the dataset. To do this, we perform cross-validation by separating the dataset into a training set and a validation or testing set. This separation can be done in different ways and has an impact on the model training [4]. Indeed, the model will perform differently depending on the sets on which it has been trained. This part of the report presents the different methods used and their impacts.

The first cross-validation method used is the train-test-split method, which is a general cross validation method, not taking the time series into consideration. To perform this method, we use a sklearn function named sklearn.model\_selection which shuffles the data set and separates it into two sets of the desired size. Usually, the training set is around 70% of the dataset and the testing set is around 30%. This is the simplest separation method :

Data set

Training set

Testing set

The problem with this method is that it doesn't take into account the order of the temporal data, since all the data points are mixed.

There are also more complex separation methods for testing the model on several sets of test data. These cross-validation techniques can be used on small datasets to improve the model. The method we have used in this work is the expanding window validation: we divide our dataset into smaller sets. At first, we train the model on the first set and test it on the second, then we train the model on the first and second sets and test it on the third set etc... (see figure...).

Data set

After 5 steps

Training set

Testing set

This method seems to be the most appropriate for cross-validating a time series since it allows the model to be trained while maintaining the temporal order of the series.

### Metrics

There are different metrics to evaluate and compare different machine learning models. As we are talking about regression models, we have to choose metrics that compare analytically the prediction value and the real value (and not the class prediction to the real class as we do for classification algorithms). To do that, different metrics can be used [5].

One of the best-known is the coefficient of determination defined as :

r2 =1- = 1-

It allows us to compare the variance explained by the model to the total variance of the dataset. It is between 0 and 1 and the closer it is to 1 the best is the regression. It could be negative if the error of the machine learning model is greater than the error produced when comparing the true value and the average.

This indicator is controversial, as it is highly sensitive to outliers and it is sensitive to the number of parameters, indeed, when we improve the number of variables, r2 is closer to 1 [6]. In this way, r2 is not always a good indicator to evaluate a model, it is important to use another indicator or graphics to complete this first assessment.

Another indicator is the root mean square error defined as :

RMSE =

It compares the real values Y and the predicted values for each point of the dataset. To have a good regression model, we want to have a RMSE as close to 0 as possible.

In our work, we choose to use this metric rather than the Mean Absolute Error (MAE) in order to highlight large discrepancies between predictions and actual values because we believe that in our scope of application, a big mistake could lead to a bad diagnosis. Moreover, we use it and not the Mean Square Error (MSE) because as it is the same unit as Y, it is easier to understand and interpret.

# Interpretation of the results

## Machine Learning models

### Impact of pre-processing

We have tested the various machine-learning models mentioned in the previous section. To do this, we tested each model by varying certain pre-processing steps and cross-validation methods in order to see the influence of each parameter on the results. All the results are given in the Appendix.

This experiment gives us information on the impact of the various pre-processing steps have on the data. To analyze the results of applying different pre-processing, we choose to use only data obtained from the SVM model with a linear kernel to be more consistent, but the results of the other models are following a similar trend. Moreover, these results correspond to the train test split cross-validation method, and are shown in Table 2.

|  |  |  |
| --- | --- | --- |
| Pre-processing step | RMSE (mmHg) | Accuracy score (r2) |
| No | 17,666 | 0,109 |
| Standard Scaler | 15,351 | 0,327 |
| Inverted PAT | 16,226 | 0,249 |
| Standard Scaler + Inverted PAT | 15,344 | 0,328 |

*Table 3: The RMSE and accuracy score (R^2) of the SVM model with different pre-processing of data performed.*

It can be seen from this table that the use of a standard scaler improves our model performance, the RMSE decreases and the accuracy score increases. The use of inverted PAT also allows us to improve our model performance. Thus, the best results are obtained using a standard scaler and the inverted PAT, which is what we expected.

However, the improvement is not significant, which can be explained by the fact that the project is only dealing with one feature (the PAT) and one patient so far, for prediction of the BP. Therefore, normalization does not have a huge impact, but this would be expected to change if more data and features were to be added to the experiment.

From the analysis of the results it can be seen that sometimes (for example with SVM model and Gaussian kernel), the PAT inversion has no impact on the output of the model. This is probably due to the fact that the model is powerful enough to detect this inversion relationship.

### Impact of cross-validation

The second kind of analysis we can perform is the impact of the cross-validation method used. We study the SVM Regressor model with a Gaussian kernel and with the same pre-processing steps (standard scaler and inversed PAT) but we use different cross-validation methods . The results from this is summarized in Table 4.

|  |  |  |
| --- | --- | --- |
| Cross-validation method | RMSE (mmHg) | Accuracy score (r2) |
| Train Test Split | 14,430 | 0,406 |
| Times Series Split (6-folds) | 4,007 | 0,508 |

*Table 4: RMSE and accuracy score (R^2) of the SVM regressor model obtained with the normal train-test split CV and time series split with 6 folds.*

It can be seen that the results are better with the times series split cross-validation method. This difference in results can be explained by the fact that this method allows us to take into account the temporal order of the data during the training, which has a significant impact since PAT varies according to its previous values.

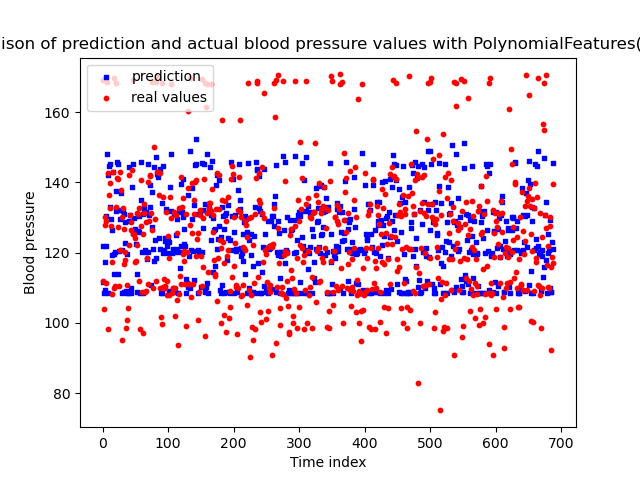
Except for SVM models, we tend to obtain an inverse relationship between RMSE and R2 when using train test split and time serie split, for examples for the polynomial regressor of degree 3 we obtain :

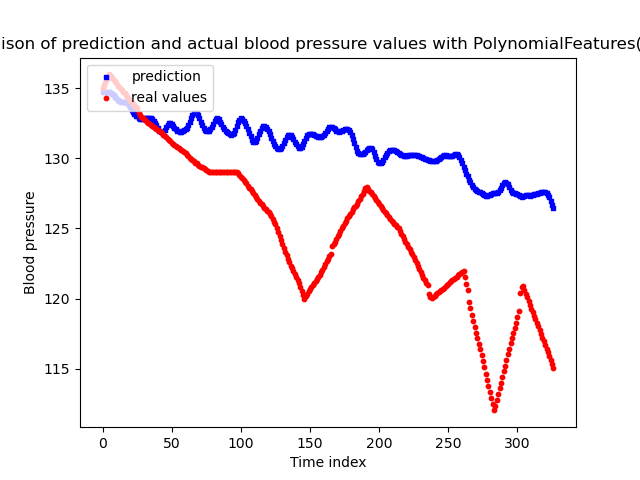
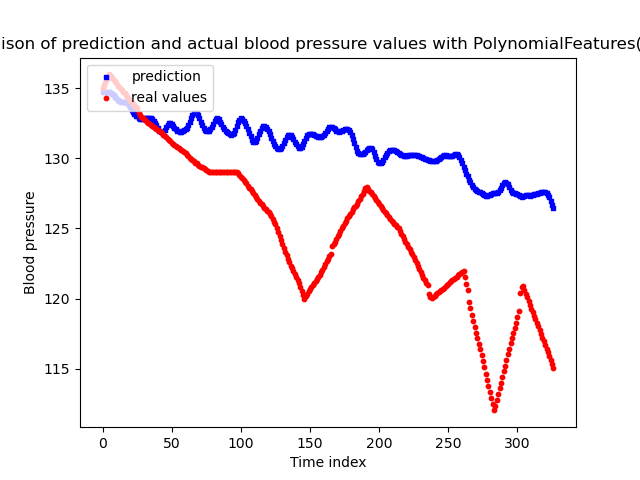
|  |  |  |
| --- | --- | --- |
| Cross-validation method | RMSE (mmHg) | Accuracy score (r2) |
| Train Test Split | 14,96031 | 0,361163 |
| Times Series Split (4-folds) | 4,729809 | -0,69941 |

*Table 5: RMSE and accuracy score (R^2) of the polynomial regression model(degree 3) obtained with the normal train-test split CV and time series split with 6 folds.*

The observed differences in performance metrics between the train-test split and time series split methods are notable. With the train-test split, despite yielding an unfavorable RMSE, we achieve a more favorable R2, signifying the model's relatively better ability to generalize results. This occurs because the random data distribution in the train-test split enables the regressor to grasp broader patterns. However, while predictions may not be highly accurate, the randomness of the data splitting allows to keep the variance between data and thus it allows to generalize.

Conversely, the time series split provides a favorable RMSE but exhibits a notably low R2. This outcome is influenced by considering the temporal nature of the data. The split ensures continuity in time, evaluating the model on data corresponding to a time period that the model never saw for the test phase. This approach challenges generalization as the test occurs on previously unseen data, leading to a lower R2 score. The model faces the difficulty of predicting on unfamiliar temporal patterns, contributing to the lower R2 score despite a favorable RMSE.





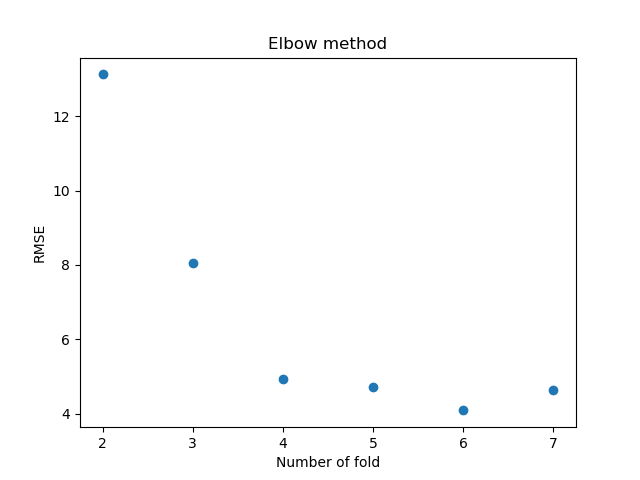
While acknowledging that R2 serves as supplemental rather than sole efficiency assessment, the lower R2 obtained is attributable to specific factors. Despite this, the time series split remains the preferred cross-validation method due to its demonstrated efficacy.

### Choice and design of the best model

One of the final aims of our analysis is to design a good model to predict the blood pressure from the PAT. In order to do that, and according to the theoretical part, we will first compare the RMSE and the accuracy score which are among the best metrics to evaluate a machine learning model. We compare models using the time series split and with all the pre-processing steps (Inverted PAT and standard scaler).

We can see in the table (Appendix n°…) that the best model according to these metrics is the SVM Regressor with a Gaussian kernel. Indeed this model, using times series split, has the best RMSE score (4,007) combined with a good accuracy score (0,508).

To improve the optimization of our model, we want to choose the best number of folds used for the cross-validation method. Indeed, according to the number of folds used, the results are not the same and it’s possible to find a number of folds balancing results and calculation efficiency. To do that, we use the elbow method.

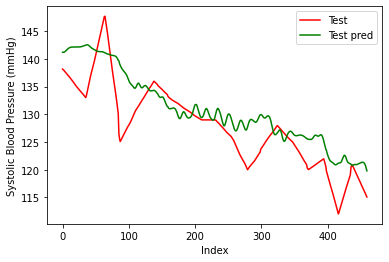


*Figure 8 : Elbow method representation*

By applying the elbow method, we can see that the optimum number of folds is 4. Indeed, for a higher number, the RMSE is slightly better for a much higher calculation cost. Thus, we can conclude that the best machine learning model obtained by comparing RMSE and accuracy score is the SVM with a Gaussian kernel and the use of Times Series Split with 4 folds.

### Going deeper into the models outputs (predictions vs expectations):

Even if the use of RMSE and accuracy score can be interesting to evaluate the quality of the model it can also be misleading regarding the accuracy of the predictions made. A way to verify such accuracy is to make sure to plot the comparison between the expected and the predicted values. For example, the machine learning model that yielded the best RMSE is the SVR model with a Time Series Split cross-validation but plotting the graph of the comparison between the expected Blood pressure values and the one predicted by the model shows that the prediction is far from perfect.



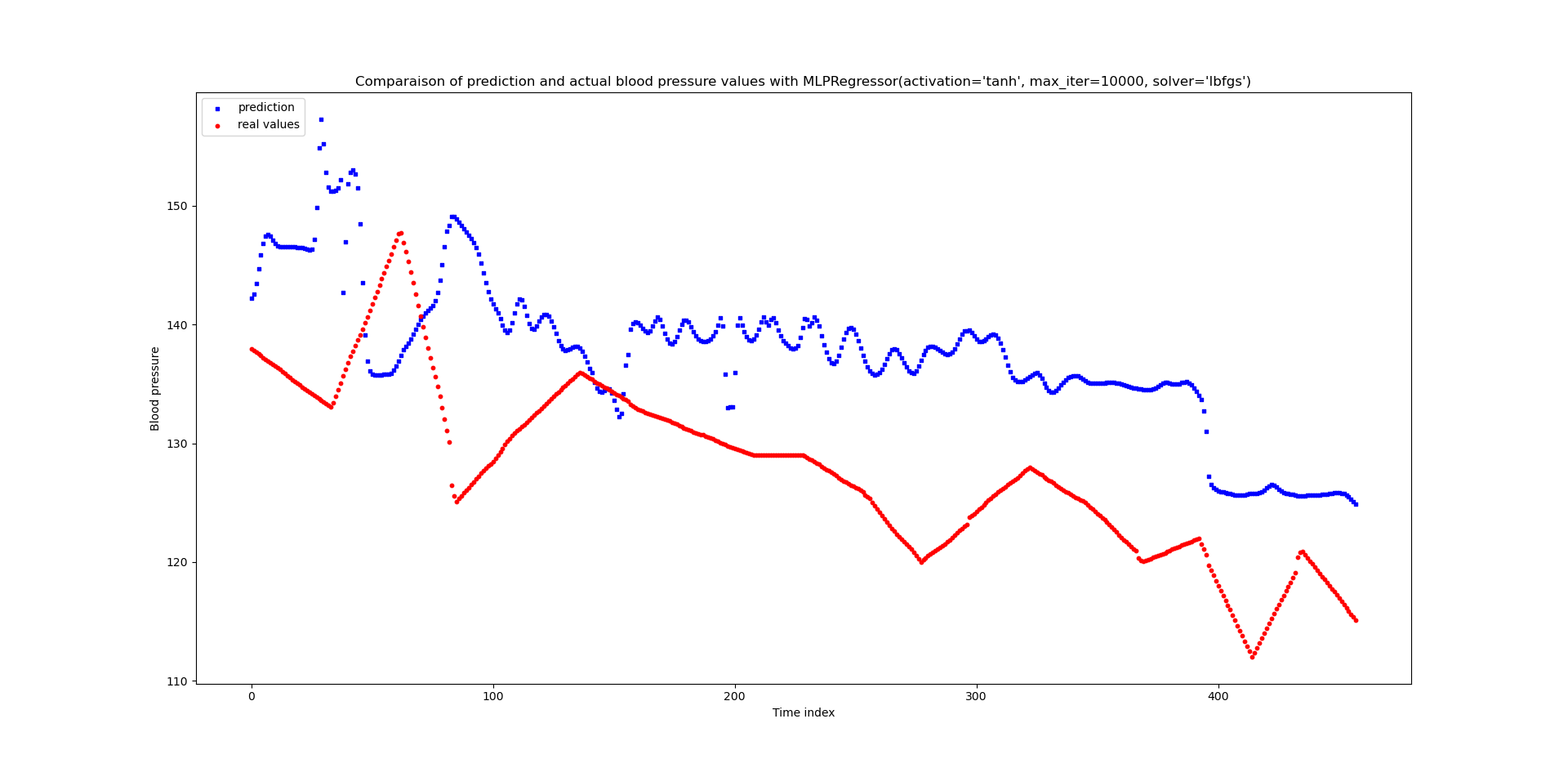
*Figure 9: Comparison between ypredict (in green) and y (in red) for the SVM model*

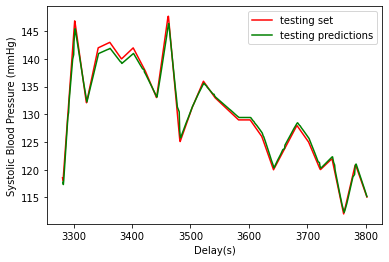
\*\* plot with the time

It can be seen that the global trend of the curve is somehow respected but the general shape is far off. In fact, the model seems to adapt poorly when there are BP peaks. However, it is still unknown if such results would be sufficient for the application they are designed for because although it does not accurately predict blood pressure, it does give an idea of the patient's average blood pressure and can therefore be used to predict a case of hypertension.

## Deep Learning models

As explained in part XXX, two deep learning models were used for investigation, a pre-built MLP regressor and a model built with a layer of LSTM neurons. The main difference between them is that the model using an LSTM layer is an autoregressive model, meaning it uses the time and the previous values of Systolic Blood pressure as features during training.

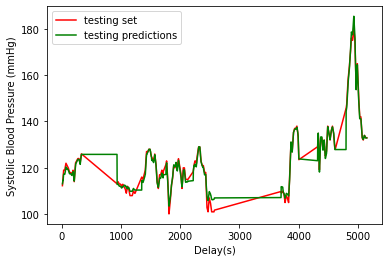


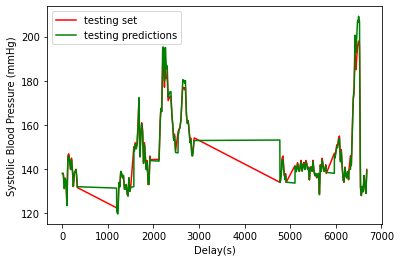
This difference results in a model that is more accurate than the previous one while not being much more ressource expensive. Figure.. shows the comparison between the output of the MLP regressor model and the expected value, Figure … showcase the same comparison but for the LSTM model. Although the output of the MLP regressor appears to follow the trend of the expected values, the output of the LSTM regressor are almost matching them perfectly.

*Figure 11: Comparison between ypredict (in green) and y (in red) for the LSTM model*

This output of the model seemed to be concerning at first as it looked like the model was overfitting. To address that issue another dataset that the model had never seen was needed to evaluate it further. If the model is overfitting, the performance on this brand new data should be really bad. Inversely measuring a good performance is the sign that the model’s output is relevant and that it can be used as a stepping stone for the model development for the final device.

Figures … and … shows the output of the model on 2 new datasets the results are encouraging with excellent predictions on changing value but on straight values the predictions are not as accurate, although that could be corrected by training the model with more data to take such patterns into account.



*Figure 12: Comparison between ypredict (in green) and y (in red) for the LSTM model on brand new dataset n°10*

# 

*Figure 13: Comparison between ypredict (in green) and y (in red) for the LSTM model on brand new dataset n°5*

# Conclusion and recommendations for future development of study

## Conclusions

* Taking the time dependency of the PAT and blood pressure into account when modeling really is important, both regarding the use of cross validation method and the use of a model (autoregressive).
* The results from the LSTM shows promising potential for the ability of predicting blood pressure based on PAT.
* There is still need for deeper study of performance of the models, with more data.
* Have to be aware of limitations of our study:
* Limitation in amount of time
* Only one feature for prediction is limiting
* Always possible to have more knowledge about the models we are using. To have someone with more machine learning background to validate could be useful.
* Amount of data - need to develop models on
* With more data, the capacity of computers running the models could be a problem

## For further development

1. Predict mean BP from mean of PAT instead of for each heartbeat. Could be a better way to start
2. Possibility of different approaches for model: population model vs individual volume
3. Including other features and how this affects the implementation of ML models:

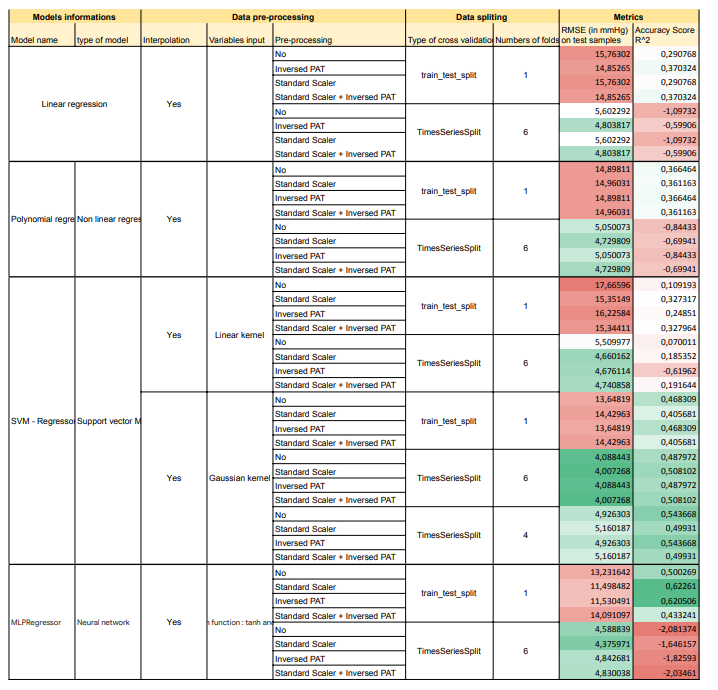
* volume of artery(in the future with ultrasound)
* Heart beat variability

The problem with this method is that it doesn't take into account the order of the temporal data, since all the data points are mixed.

There are also more complex separation methods for testing the model on several sets of test data. These cross-validation techniques can be used on small datasets to improve the model. The method we have used in this work is the expanding window validation: we divide our dataset into smaller sets. At first, we train the model on the first set and test it on the second, then we train the model on the first and second sets and test it on the third set etc... (see figure...).

# Appendix

## Appendix 1 : Preprocess and computation of the accuracy (models.py):

**

## Appendix 2 : Preprocess and computation of the accuracy (models.py):

|  |
| --- |
| import pandas as pd import numpy as np import matplotlib.pyplot as plt from scipy.interpolate import interp1d from scipy import stats from sklearn.preprocessing import StandardScaler from sklearn.linear\_model import LinearRegression from sklearn.preprocessing import PolynomialFeatures from sklearn.model\_selection import train\_test\_split, TimeSeriesSplit from sklearn.metrics import mean\_squared\_error, r2\_score   class Preprocess():  def \_\_init\_\_(self, path):  self.df = pd.read\_csv(path)  self.df = self.df.dropna(subset=['blood pressure\_systolic'])    def interpol(self):  time\_continuous = self.df['wrist@(9mm,809nm)\_delay\_s']  time\_not\_continuous = self.df.loc[self.df['wrist@(9mm,809nm)\_filtered\_pat\_bottomTI'].notna(), 'wrist@(9mm,809nm)\_delay\_s']    interp\_func = interp1d(time\_not\_continuous.array, self.df.loc[self.df['wrist@(9mm,809nm)\_filtered\_pat\_bottomTI'].notna(), 'wrist@(9mm,809nm)\_filtered\_pat\_bottomTI'].array, kind='linear', fill\_value="extrapolate")  continuous\_values = interp\_func(time\_continuous)    self.df.insert(loc=3, column='pat\_filtred\_continuous', value=continuous\_values)     def outliers(self):  df\_outlier = self.df.copy()  df\_outlier['wrist@(9mm,809nm)\_filtered\_pat\_bottomTI'] = df\_outlier['wrist@(9mm,809nm)\_filtered\_pat\_bottomTI'].fillna(df\_outlier['wrist@(9mm,809nm)\_filtered\_pat\_bottomTI'].mean())    # Commpute the z-statistic  z = np.abs(stats.zscore(df\_outlier['wrist@(9mm,809nm)\_filtered\_pat\_bottomTI']))  # Identify outliers as the pat\_filtered with a z-score greater than 3  threshold = 3  outliers = df\_outlier['wrist@(9mm,809nm)\_filtered\_pat\_bottomTI'][z > threshold]   # Remove outliers  self.df = self.df.drop(outliers.index)    def process\_data(self, interpolation, invert, outlier):  """"  Process the data with multiple possibilities  interpolation : interpolate the missing PAT data using scipy function  invert : the correlation between PAT and BP is negative so we invert the PAT  outlier : remove the outliered values of the PAT data   """    if outlier == True :  self.outliers()    if interpolation == True :   self.interpol()  else :  self.df.dropna(subset=['wrist@(9mm,809nm)\_filtered\_pat\_bottomTI'])    X, y = self.df.iloc[:, 3].to\_numpy().reshape(-1, 1), self.df['blood pressure\_systolic'].to\_numpy().reshape(-1, 1)    if invert == True:   X = 1/X    return X, y    class Model():  def \_\_init\_\_(self, path, mod, interpolation = True, invert = True, outlier = True, normalize = True):  """   Parameters  ----------  path : access to the data, need to be in the same file.  mod : Used model od deep learning.  interpolation : interpolate the missing PAT data using scipy function  default is True  invert : the correlation between PAT and BP is negative so we invert the PAT  default is True  normalize : normalization of the PAT data   default is True    """  self.mod = mod  self.normalize = normalize   self.X, self.y = Preprocess(path).process\_data(interpolation, invert, outlier)    def normalization(self, X\_train, X\_test):  scaler = StandardScaler()  X\_train = scaler.fit\_transform(X\_train)  X\_test = scaler.transform(X\_test)    return X\_train, X\_test    def data\_splitting(self, test\_size = 0.3, random\_state =42):  if isinstance(self.mod, PolynomialFeatures):  self.X = self.mod.fit\_transform(self.X)  self.mod = LinearRegression() #After adding polynomial feature, the prediction is done using a linear regressor.    X\_train, X\_test, y\_train, y\_test = train\_test\_split(self.X, self.y,   test\_size = test\_size, random\_state = random\_state)    if self.normalize == True :  X\_train, X\_test = self.normalization(X\_train, X\_test)    return X\_train, X\_test, y\_train, y\_test    def n\_split(self):  rmse\_n = []  for n in range (2,10):  rmse\_n.append(self.accuracy(split\_type = 'tscv', n\_splits=n)[0])  plt.plot(range(2,10), rmse\_n)    return rmse\_n.index(min(rmse\_n))    def accuracy(self, split\_type = 'classical', n\_splits=6):  """  Parameters  ----------  split\_type : Type of cross validation used, optional  -- The default is 'classical', corresponding to one training set   containg 70% of the data and one test set containing 30%.   -- 'tscv' : time series cross-validation.    n\_splits : int, optional  Number of folds, training set, when using tscv. The default is 6.   Returns  -------  rmse  Root mean square error between the predicted value and the real value.   In the case of tscv it is the max rmse over all the folds.  r2  R^2    """  if split\_type == 'classical':  X\_train, X\_test, y\_train, y\_test = self.data\_splitting()    self.mod.fit(X\_train, y\_train)    y\_pred = self.mod.predict(X\_test)  self.y\_pred = y\_pred  self.y\_test = y\_test   return mean\_squared\_error(y\_test, y\_pred, squared = False), r2\_score(y\_test, y\_pred)    if split\_type == 'tscv':  tscv = TimeSeriesSplit(n\_splits)  rmse = 1000   r2 = 0  for train\_index, test\_index in tscv.split(self.X, self.y):  if isinstance(self.mod, PolynomialFeatures):  self.X = self.mod.fit\_transform(self.X)  self.mod = LinearRegression() #After adding polynomial feature, the prediction is done using a linear regressor.    X\_train, X\_test = self.X[train\_index], self.X[test\_index]  y\_train, y\_test = self.y[train\_index], self.y[test\_index]    if self.normalize == True :  X\_train, X\_test = self.normalization(X\_train, X\_test)    self.mod.fit(X\_train, y\_train)     y\_pred = self.mod.predict(X\_test)    if mean\_squared\_error(y\_test, y\_pred, squared = False) < rmse:  rmse = mean\_squared\_error(y\_test, y\_pred, squared = False)  r2 = r2\_score(y\_test, y\_pred)  self.y\_pred = y\_pred  self.y\_test = y\_test    return rmse, r2 |

## Appendix 3 : LSTM model for dataset modeling :

\*To be added \*

# 

# Bibliography

[1] Augmented Dickey–Fuller test, 2023. Wikipedia.

[2] KPSS test, 2023. Wikipedia.

[3] Schober, P., Boer, C., Schwarte, L.A., 2018. Correlation Coefficients: Appropriate Use and Interpretation. Anesthesia & Analgesia 126, 1763.<https://doi.org/10.1213/ANE.0000000000002864>

[4] Filho, M., 2022. How To Do Time Series Cross-Validation In Python [WWW Document]. URL<https://forecastegy.com/posts/time-series-cross-validation-python/> (accessed 12.13.23).

[5] Evaluation Metrics for Regression Models [WWW Document], n.d. URL<https://www.enjoyalgorithms.com/blog/evaluation-metrics-regression-models> (accessed 12.13.23).

[6] How To Interpret R-squared in Regression Analysis - Statistics By Jim [WWW Document], n.d. URL<https://statisticsbyjim.com/regression/interpret-r-squared-regression/> (accessed 12.13.23).

[t](https://en.wikipedia.org/wiki/Augmented_Dickey%E2%80%93Fuller_test)

[2] <https://en.wikipedia.org/wiki/KPSS_test>

[3] https://www.statsmodels.org/stable/examples/notebooks/generated/stationarity\_detrending\_adf\_kpss.html#KPSS-test

[4] https://journals.lww.com/anesthesia-analgesia/fulltext/2018/05000/correlation\_coefficients\_\_appropriate\_use\_and.50.aspx#:~:text=Correlation%20coefficients%20describe%20the%20strength,monotonic%20relationship%20between%202%20variables**.**

**[5]** <https://forecastegy.com/posts/time-series-cross-validation-python/>

[6] <https://www.enjoyalgorithms.com/blog/evaluation-metrics-regression-models>

[7] <https://statisticsbyjim.com/regression/interpret-r-squared-regression/>