Data Mining Techniques – Assignment 1 Advanced

TA: Paulo Fijen

Group 7:

Levi Kamsteeg 2584230
Kilian Wessels 2579173
Steven Logger 2595779

Vrije Universiteit Amsterdam

1 Introduction

In this report we will create a model which is able to predict the average mood of a person one day ahead of time. We use a dataset which comes from the mental health domain, where a patient scores his current mood in an app on his/her smartphone. The dataset also contains additional sensory data about the mobile phone behaviour and application usages of the patient, which might impact the mood of the patient the following day. We made use of Python to conduct our analysis.

Predicting the average mood of a patient a day in advance can be useful when this patient face depression issues. Increasingly more mobile phone applications are developed to help persons who are facing a depression. Practical implications of this report could therefore be useful when developing these anti-depression applications.

We find that the instance and temporal based models outperform the benchmark model. In addition, the random forest is best capable of predicting the average mood the next day. Moreover, the lagged mood is an important feature when predicting the average mood the next day.

The rest of this report is structured as follows: first we will perform some exploratory data analysis and explain our general setup for feature engineering, next we will start to build are first models and evaluate them, finally we will analyse our results and conclude our findings.

2 Pre-processing the dataset

2.1 Exploratory data analysis

The dataset contains patient ID numbers (27 unique patients), timestamps (the exact time the patient reported or the phone stored a variable), the variable name and the corresponding variable value. Our target variable is named mood, which is scored by the user on a scale of 1-10. The user can score their mood multiple times a day, so we want to take the average to get an aggregation of the mood of the user. Other variables are phone call and SMS dummies, which are equal to one if the user made a phone call or sent a SMS.

Before we can build our models we will need to re-arrange the data format. We want our dataframe to look like Figure 1.

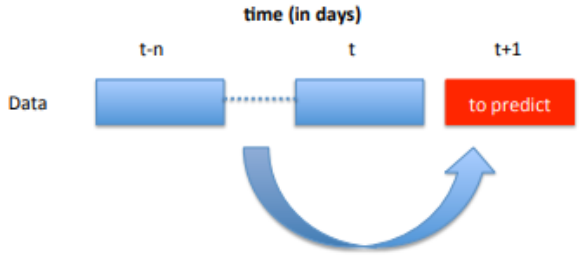

Figure 1: Predictive model

After performing some transformations to the raw data, we got the dataframe as in Figure 1. Now there are two ways how we can build a predictive model based on this dataframe; using machine learning models which are able to cope with temporal data (ARIMA or recurrent neural network) or aggregate the history in such a way which can be inputs for regular machine learning models (random forest or support vector machines). In this report we will look into both approaches.

We decided to use a few simple methods to aggregate the history. We consider data five days prior to when the target variable takes place. So, when we want to predict the average mood at t=6, we use the data from t=1 to t=5. First we used the mean variable from the past five days to aggregate the history. We used this method for all variables except the dummy variables call and SMS. For the call and SMS variables we summed them up, since taking the average of a dummy is not very insightful.

We did not find any missing or incorrect values in the dataset. Although there were a few outliers considering some variables. We decided to use winsorizing, at 99%, for a few variables which contained outliers. In addition, the dataset does not contain any redundant or equivalent variables.

Now let us take a look at the average mood during the sample period of each patient, which can be found in Figure 2. We can see that there is some fluctuation in the average mood of each patient. Due to most of the patient's' mood lay between the values 6 and 8 we decided not to create individual models for each patient but created general models instead.

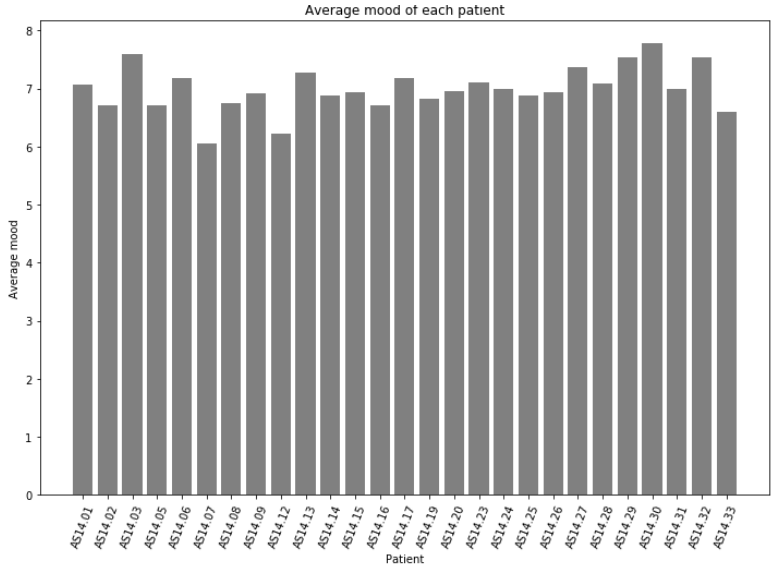

Figure 2: Average daily mood of each patient.

We can also look at the mood of the patients over time to see if there is some sort of temporary trend. In Figure 3 the average daily mood over time for a few patients are depicted. From the figure we can conclude that the average daily mood for each patient differs. In addition, we see that patient 2 and 5 have a significant decrease in their daily moods a few days after they started reporting his/her mood. While Figure 2 shows that the average daily mood of the patients are somewhat the same, Figure 3 shows that individual models might also be taken into consideration when creating a predictive model.

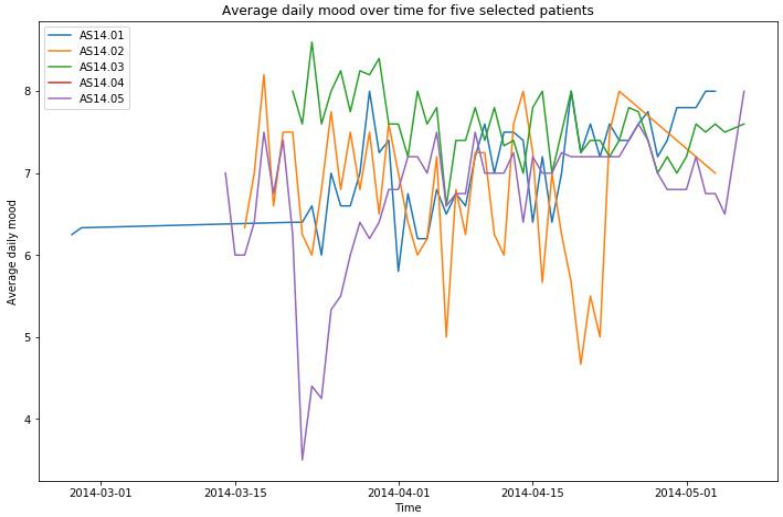

Figure 3: Average daily mood of the first five patients over time.

2.2 Feature Engineering

Moreover, we created some additional features. We based our additional features on a psychology field article from Egloff et al. (1995). In their paper they look at how the time of the day and the day of the week impact the mood of a patient. They find that in the afternoon (after work) and during the weekend (free time) the patient's mood is the highest. In addition, they find that the mood increases when the day goes by; from morning to evening.

First, we made dummy variables which correspond with the time of the day of when the variable was created. We made morning (from 06:00 am to 12:00 am), midday (from 12:00 am to 12:00 pm) and night dummies (from 12:00 pm to 06:00 am). Next we multiplied these dummies with other features, so we get interaction variables. So, we might have a morning call or evening SMS. We created the day of the week dummy the same way as described above. We also created the average daily lagged values of all variables, including our target variable mood. If there is some autocorrelation which might impact the target variable, some of this should be captured by taking the daily lagged values. Our target variable is created by taking the average mood of the next corresponding day.

2.3 Final attributes

In this section we will look at the final attributes which we will use for our models. After feature engineering we are left with a lot of features, using too much features in our models will lead to unnecessary complexity and an increase in noise. We start by looking at features which have high correlations with each other and remove these features, reducing complexity and noise. We do this by looking at the correlation matrix of the features, which can be found in Figure 4. There is a somewhat high correlation between the morning call, day call, night call and call categories. This makes sense since the sum of the morning, day and night is equal to the overall call value (multi correlation issues). So, we dropped all the original features since now we have them split up in morning, day and night. From the correlation plot we can see that there are not very bright or dark correlations (above 0.8), which indicate high positive or negative correlation, so the issue of multicollinearity should be solved.

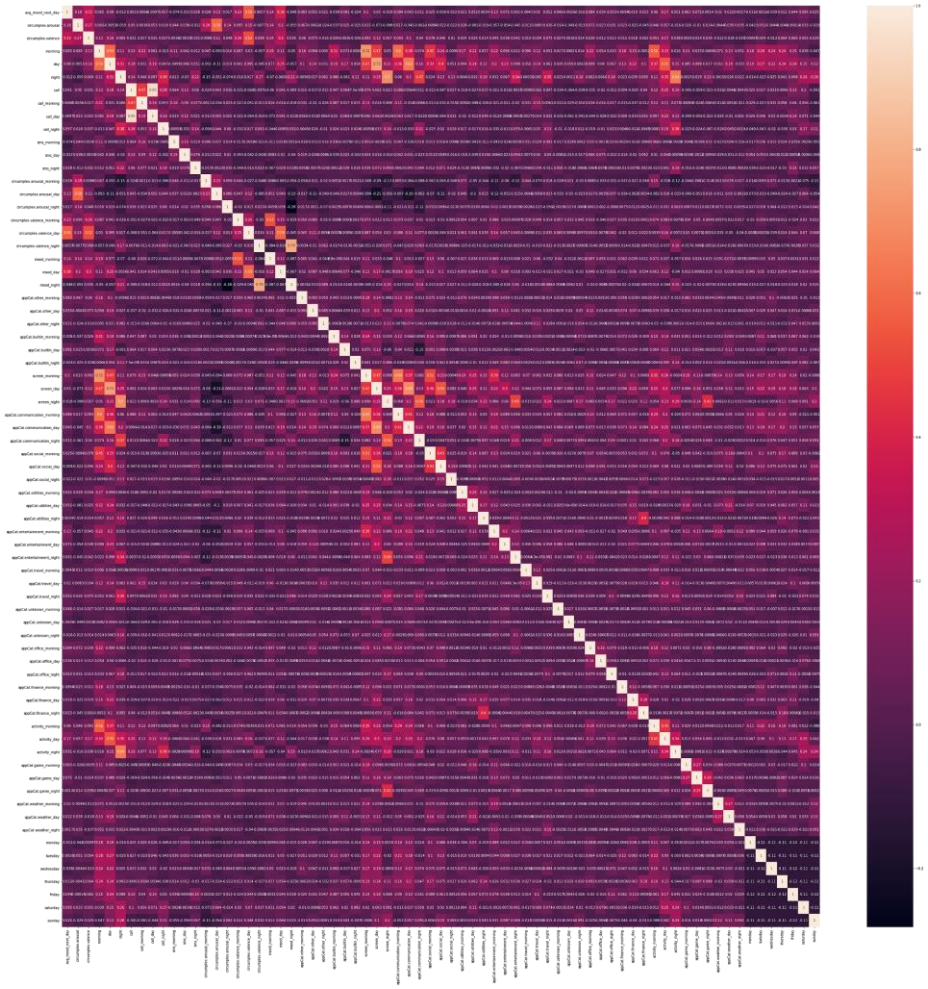

Figure 4: Correlation matrix of the features (dark areas mean negative correlation up to -0.25, while light areas indicate high correlation up to 1.00).

In addition, we make use of Principle Component Analysis (PCA) to decrease the dimensionality but maintain the explained variance. As lot of our features have strong factor structure, PCA can help to reduce overfitting (Kozak, 2020). In total we created 13 PCA features, that are named 1 to 13.

3 Results

In this section we will analyse the results obtained from our analysis. As discussed before, two types of datasets are considered in this research. One with the focus on all instances, thereby ignoring the movement over time, and a temporal dataset that does

take time into consideration. At the end we will compare both approaches using regression analysis performance metrics such as Mean Squared Error, Absolute Mean Error and the Coefficient of Determination.

Before we begin analysing the models we set up a benchmark model so we can compare models with each other. This benchmark model simply consists of the observation of a patient's mood today as a prediction for the patient's mood tomorrow. This benchmark gives us the following results:

Table 1: The performance of the benchmark model. Described are the Mean Squared Error, Mean Absolute Error and R Squared.

	Benchmark
MSE	0,52
MAE	0,54
R-Squared	0

Now we can start with the introduction of the other two models. Firstly, for the instance-based approach we will show that no linear relationship can be found, which is why non-linear machine learning models are considered such as Random Forest and Multilayer Perceptron (MLP). When conducting an OLS regression we find that for the most important features their coefficient is very small and non-significant. The negative coefficient of determination also suggests that no clear linear relationship has been found that can explain the average mood for the next day.

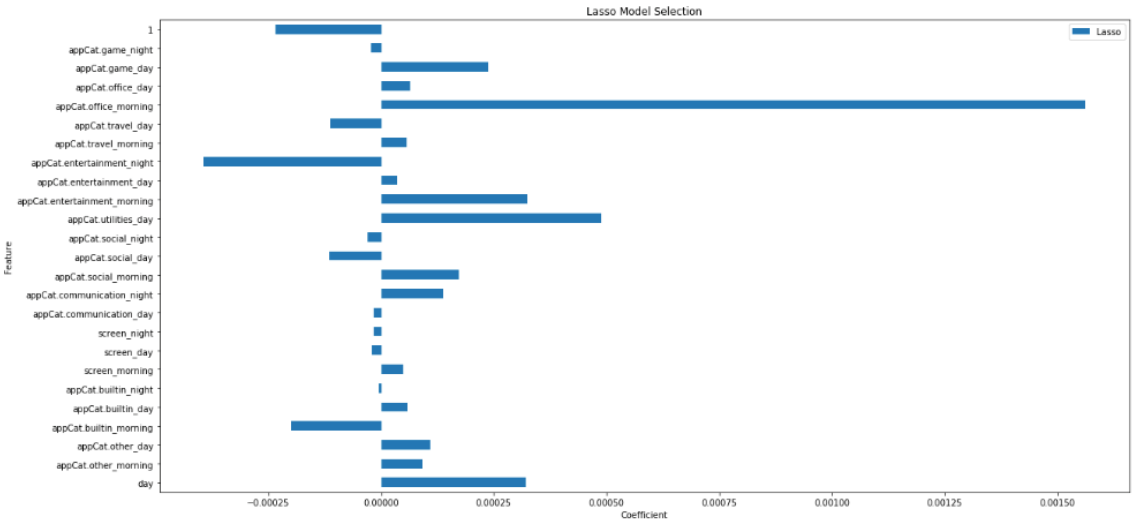

Figure 5: The coefficient of the most important features after setting non-important features to zero with Lasso regularization.

Considering the Random Forest model which can map X to y in a non-linear fashion (Lidia, 2012), we might find a better model for our predictions. To fully optimize the model an exhaustive GridSearch has been conducted in order to find the optimal hyper parameters. This has been done using cross-validation in order to create more robust results (James, 2013).

The hyperparameter max depth is used to decide on the depth of each tree. On default this parameter will continue to grow the tree until a leaf is pure, which means that all the data points on the leaf will contain the same class. From theory we know that the deeper the tree, the more prone it is to overfitting as we are narrowing the pureness of a leaf. The GridSearch showed that a depth of 10 was enough to create robust decision tree. Another optimized parameter is the number of trees that are build. Here we saw that 200 trees create the most optimal performance. Note that the more trees we build in a forest will not degrade the model but will only increase the computational time. As we are dealing with a smaller dataset this does not create any problems.

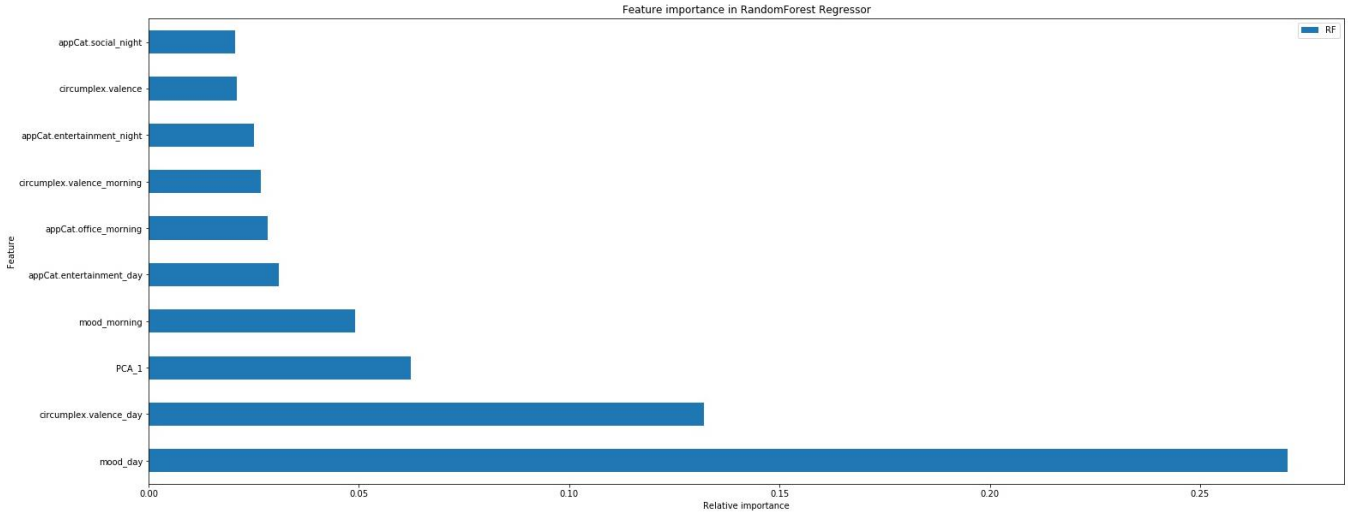

Figure 6: Top 10 important features of the random forest model.

The feature importance of the random forest is presented in figure 6 above. This graph tells us the relative level of information contribution of the features when training the model. The most important feature here is mood_day, which can be seen as the average mood that the patients filled in during the day. Note that we also included 'mood_morning' and 'mood_night', which do not seem to contribute to the explaining ability of the average mood next day. This is mainly due to the fact that most of the entries are done during the day. The Valance during the day and the first principle component are adding value to the model. It is hard to interpret the principle component, due to the fact that is consists of the explained variance of multiple features, however we can conclude that the explained variance of a series of grouped features

does contribute in the creation of a better model, hence we keep this component in our data set. The use of entertainment and social apps during the night also seems to influence the average mood the next day.

Next, we consider the MLP model. For the optimization of this model we also conducted an exhaustive GridSearch, especially focusing on the numbers of hidden layers and neurons within those layers as here the depth of the neural network is determined. The number of layers determines how complex input information can be retrieved. Often 1 layer will suffice for simple problems, whereas in deep learning multiple layers will suit the model better but will be computationally costly. Jeff Heaton (2008), proposes a rule of thumb that the number of layers should be between the input and output size. Taking this into account we test for 3 possible layers, namely (10,8,6,4,2) (100,) and (100,100,100,100,100). The results show that extending the number of neurons did not increase the performance of the model, but also 1 layer is not enough to capture all information. Therefore, we settled on the first option, were a decreasing number of neurons are chosen to cut down in computation time.

The outcome of the model is somewhat different from what we saw in the Random Forest model. We can see that the dummy denoting 'Thursday' is relevant, as well as the number of SMSs send out during the day. See figure 7 below for the feature importance of the 10 most relevant features in the MLP. As these are both different

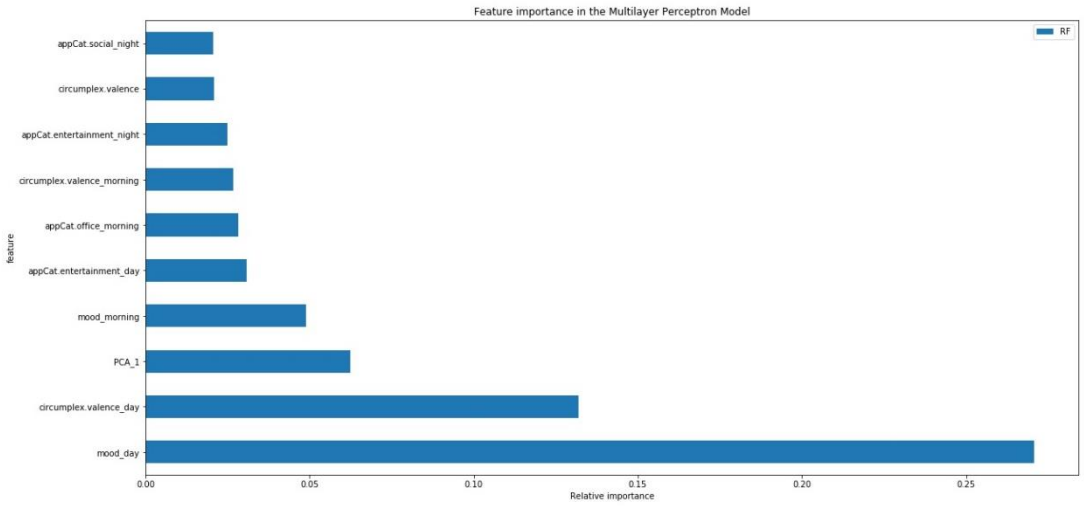

Figure 7: Top 10 important features of MLP.

models using different input to output mappings it is common to that both models use different features. However, as we only observe the first layer in the MLP we will consider the feature importance of the random forest model more relevant, as these are averages taken over two hundred decision trees.

Naturally, we also want to know how well the forecasts of the mood worked on the training set. As we worked with k-fold cross validation, we will consider the average score of the folds as a performance metric. Please note that for the MLP to perform properly, the data must be scaled at each fold of the training set separately to prevent information leakage into the test set. This has been done using the pipeline, provided by Sklearn (Pedregosa et al., 2011)

Table 2: The prediction performance of both models; the Mean Squared Error, Mean Absolute Error and R Squared.

	Random Forrest
MSE	0,07
MAE	0,47
R-Squared	0,64
	Multilayer Perceptron
MSE	0,33
MAE	0,41
R-Squared	-0,62

As we can see in table 2 the Random Forest performed significantly better in terms of the mean squared error, but slightly worse in means of the mean absolute error. Due to the fact that the MSE first squares the errors before they get averaged it gives a higher weight to large errors. We therefore can conclude that the Random forest does not contain that many high errors, which is desirable when considering that a large mood change of the patient could be the difference of being healthy and depressed.

Next, we will consider the temporal approach consisting of the classical model for time-series: ARIMA. As it is very difficult to say something about the implications of the model when all patients are grouped together, it can be more insightful to compare individual patients with each other, providing multiple time-series to model. By considering all 27 patients and their time-series, only four patients had significant autocorrelation that could be modelled, creating ARIMA models for the other patients would result in bad quality of predictions. By choosing appropriate orders for the Autoregressive as the Moving Average part we modelled the patients' mood separately. The outcomes for the patients can be considered in the following table 3.

Table 3: The prediction performance of multiple ARIMA models on the temporal dataset. Described are the Mean Squared Error, Mean Absolute Error and R Squared.

Patient	*MSE*	*MAE*	*R-Squared*
AS 14,01	0,16	0,36	-1,61
AS 14,05	0,21	0,28	-0,36
AS 14,13	0,65	0,66	-0,14
AS 14,16	0,17	0,31	0
Average	0,2975	0,4025	-0,5275

As we can see in terms of measurement errors, this temporal based model performed quite reasonable. The mean squared error is on average higher than the MLP forecasts, but the mean absolute error is slightly lower. The drawback of this method is however that the modelling has to be done for each patient separately, whereas the instance-based approach models can be used for each new observation that has been made on a new day.

4 Conclusion

Overall, both the instance based and temporal based models performed better than the benchmark. Comparing the ARIMA models with the Random Forest and Multilayer Perceptron models there is not a very clear conclusion, as we had to model each patient separately for the temporal based approach. The lowest error was found in the random forest model, which was the only model to show a positive R squared of 0.64. This indicates that the predictions followed the actual observations in a linear fashion, whereas the negative R-Squared suggests that such a relationship could not be found. We saw in the ARIMA models as well as in the feature importance of the random forest model that the previous day mood is important in predicting the next day's mood. The principle components also seemed to contribute to both the instance-based models.

5 References

Egloff, B., Tausch, A., Kohlmann, C. W., & Krohne, H. W. (1995). Relationships between time of day, day of the week, and positive mood: Exploring the role of the mood measure. *Motivation and emotion*, 19(2), 99-110.

Kozak, S., Nagel, S., & Santosh, S. (2020). Shrinking the cross-section. *Journal of Financial Economics*, *135*(2), 271-292.

Auret, L., & Aldrich, C. (2012). Interpretation of nonlinear relationships between process variables by use of random forests. *Minerals Engineering*, *35*, 27-42.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning*. New York: springer.

Heaton, J. (2008). *Introduction to neural networks with Java*. Heaton Research, Inc..

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, *12*(Oct), 2825-2830.