DS- Project Group 23 submission

Edition: 2023 2A
Project: AF
Primary Topic: DM
Secondary Topic:
Members: Gijs de Smit, Ararat Khachadurian
Last update: 17/04/2024, 14:45:02

Motivation

Atrial fibrillation (AF) is a medical condition in which the heartbeat becomes abnormal. It is a condition that can develop in the postoperative period after cardiac surgery and can lead to serious complications in the patient. Early detection of AF is crucial to prevent these complications from developing. However, since AF may only occur occasionally, it requires extensive monitoring of ECG data and thus the analysis of a large amount of data to be able to diagnose the condition.

A technique that can automate this process could have a significant clinical impact, result in a large reduction of costs and allow for more valuable data for research efforts in understanding and managing AF.

It would have a clinical impact because it could detect the condition at system runtime, so that immediately after its occurrence is measured it can be detected and signalled. Furthermore, it has the potential to have a higher detection accuracy than that of the current process of diagnosis.

The cost reduction would logically follow because the manual analysis of large ECG records is very time-consuming and thus costs a significant amount of labour from the medical staff.

If the technique is able to automatically detect more amounts of occurrences of AF than is currently possible, more data on AF would easily be retrievable which can be used for further research in this domain.

These three reasons make it clear why an automated technique for the diagnosis of AF is relevant and important.

(Business/Research) questions

The main research question is: "To what extent can one automatically detect episodes of Atrial Fibrillation?". We divide this problem into the following sub-questions:

- What characteristics of ECG data determine whether a patient is having an AF episode?
- Which Machine Learning models are suitable for the provided data to perform binary classification of AF?
- What is the best performing Machine Learning model based on F1-score?
- How does the automatic detection of AF by the selected model compare with existing detection techniques in terms of accuracy?

Source data

The ECG data is provided to us in two formats, a preprocessed and a raw version.

The raw data consists of 804 files where each file contains roughly 24 hours of R-R interval durations in milliseconds extracted from a patient's ECG data with timestamps. Each raw data file is paired with a control file containing the ground truth label for the timestamps (not AF, AF or invalid). We manually transformed these 804 file pairs into a single .csv file where each row represents a window of 40

consecutive R-R interval values, each in its own column with an additional column for the class label (AF/no AF). This gives a total of 41 columns. Processing the raw data to make it suitable for model input required a lot of time and precise transformations.

The preprocessed data provided to us is a table from a single .csv file. A row in the table represents a single sample of a 30 second ECG measurement. Each row is essentially a histogram, where each feature is a bin (with a width of 50ms) with the (normalised) count of R-R interval duration values within a 30 second window.

For our initial approach we use the preprocessed data. The preprocessed data contains a class imbalance, meaning that the count of R-R intervals with 'No AF' (~113k) is much greater than with 'AF' (~37k). We simply detected this by a count of the labels with value 0 and 1. We balance the dataset by a random sample of the majority class with the same sample size as the size of the minority class. Before we train models we separate the dataset into a train- and a test-set with an 80/20 ratio.

For our second approach we use the raw data to make it suitable for neural network input. We remove invalid sequences (label -1), duplicates and extreme outliers with R-R interval durations greater than 1800ms. We have to balance the set once again using the same technique as before leaving us with ~55k rows in each class for a total of ~110k samples. We separate the data into a train and test set and scale each feature column to mean 0 and standard deviation 1.

Method

We use two approaches:

1. Use the preprocessed data and experiment with suitable simple ML models to find an optimal result
2. Use the raw data, process it and experiment with neural networks

For our first approach we:

1. Observe the data formatting. Since each column (except for the last) represents a bin (with a width of 50ms) and the values a frequency of occurring (count), we use a bar plot to show a row (=sample). We use the class label as a title and to indicate ground truth.
2. Check for a class imbalance. We create a plot of the class distribution of the data by counting the class labels in the datasets to get a number of 'No AF' and 'AF' rows. We find an imbalance and solve it through reducing the majority class size to be equal to the minority class size by a random sample. Afterwards we verify the balancing by another count.
3. Separate the data into a train and test set with a 80/20 ratio.
4. Train a Decision Tree, Support Vector Machine and Logistic Regression model with default parameters and evaluate the accuracies and F1-scores on the testing set. Additionally we tweak some hyperparameters for each model to investigate improvements.
5. Train a Random Forest model with settings [max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=200] and evaluate the testing set performance.
6. Rank the features. We extract feature_importances_ from the RandomForestClassifier, sort them in descending order and show the importance of each feature. This is based on a mean impurity decrease (e.g. Gini impurity) [2], so the mean decrease in uncertainty of all trees after a split using a specific feature.
7. Optimise the Random Forest hyperparameters. We use grid search cross validation (5-fold) to fit the parameters and evaluate the testing set performance and the feature ranking.
8. Compare performances between the models using a bar plot of the testing set performance metrics found.

For our second approach we:

1. Reformat the raw data into samples of 40 consecutive R-R interval durations (skipping values with indecisive label) with the majority class as its label. Then we drop duplicates, remove outliers

and balance the dataset as done before. Due to the size of the raw dataset the processing is quite time consuming.
2. We design several Neural Network models with some fine-tuning of their parameters:
3. Long short-term memory (LSTM) neural network, characterised by its ability to retain long-term dependencies and handle sequences of data, making it suitable for time series prediction of longer sequences.
4. Fully connected (FC) neural network, characterised by each neuron in one layer being connected to every neuron in the subsequent layer, enabling it to learn complex relationships in data.
5. Convolutional neural network (CNN) neural network, characterised by convolutional and pooling layers, which enable it to effectively capture spatial hierarchies.
6. Convolutional Long Short-Term Memory (CNN-LSTM) neural network, characterised by combining the strengths of CNNs in capturing spatial features and LSTMs in handling sequential data.
7. We tweak the common training parameters including batch-size and epochs and train the models.
8. We evaluate the performances of the models.

Results

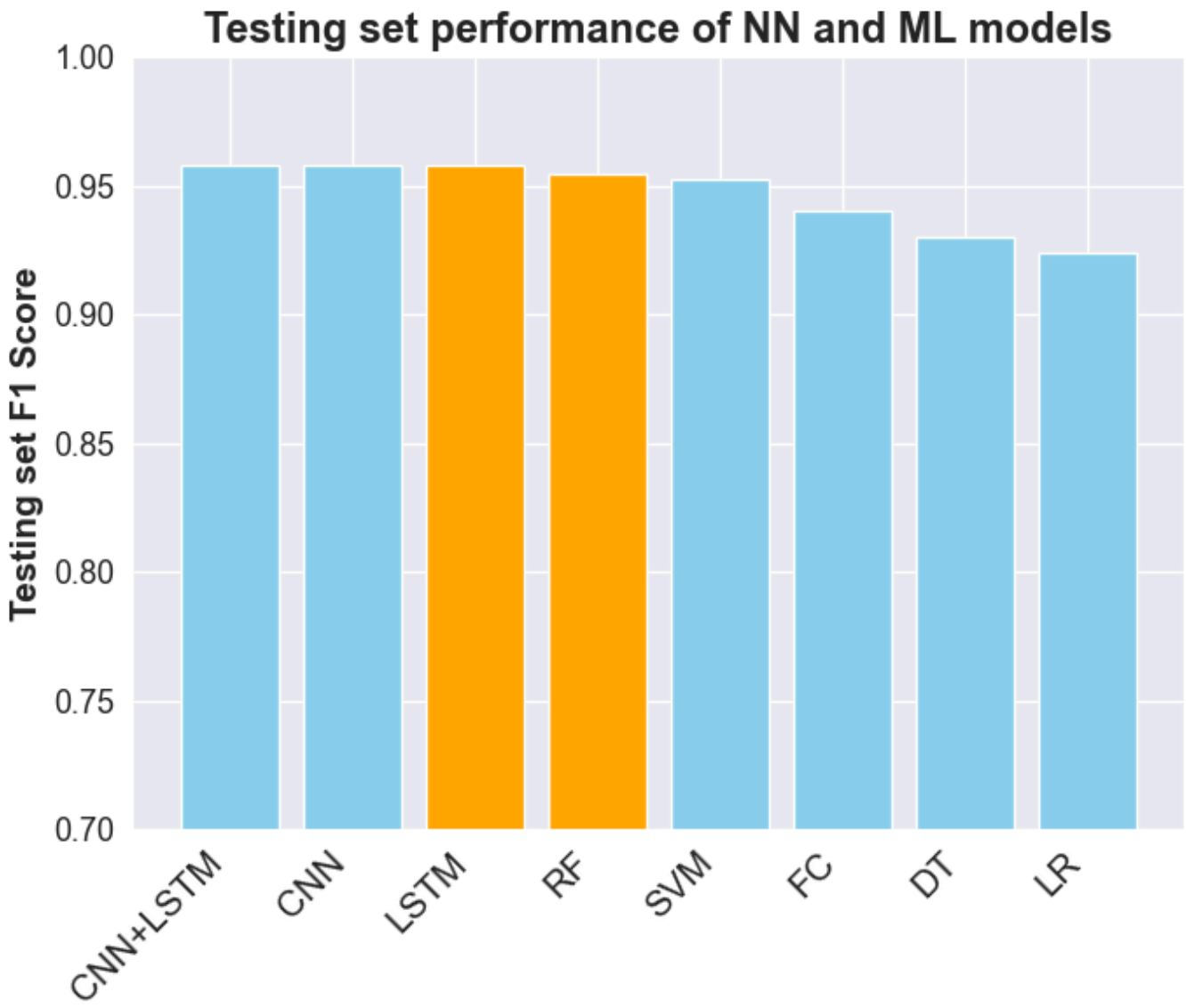

Figure 1: Testing set performance of all models

- CNN = Convolutional Neural Network
- LSTM = Long-Short-term Memory Neural Network
- CNN+LSTM = Combining both an LSTM, CNN and FC layers as well as Dropout
- RF = Random Forest

- SVM = Support vector machine
- FC = Fully connected neural network
- DT = Decision Tree
- LR = Logistic Regression

From Figure 1 we can see that all Machine Learning models (including NNs) achieve a higher than 0.92 testing set F1-score. It also seems that the more complex a model is (meaning more parameters the model has), the better the performance on unseen data. Combining spatial feature extraction of a CNN and temporal feature extraction from LSTM into a single model does not improve over their individual parts. We suggest using the Random Forest or LSTM model as they achieve a relatively high testing performance and are not complex, which makes them efficient.

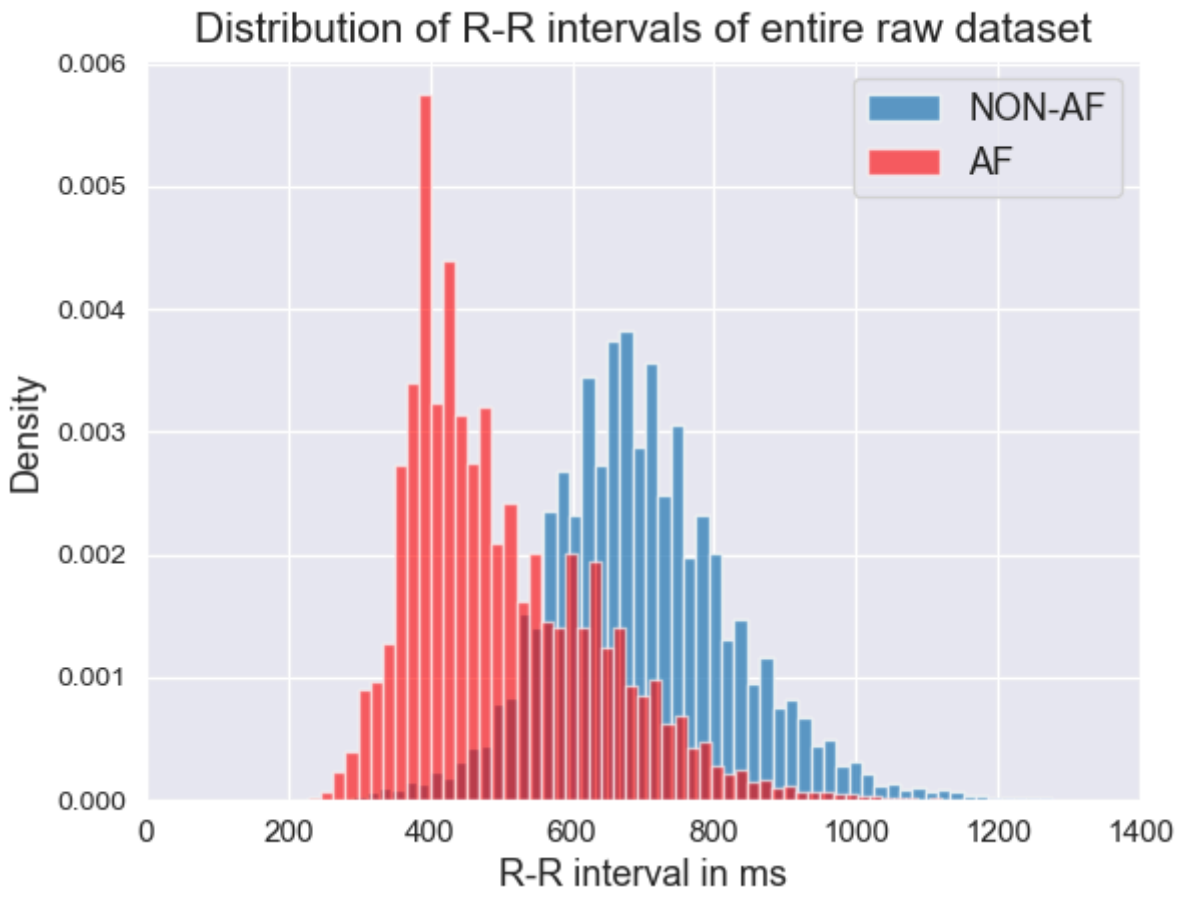

Figure 2: Raw data feature distribution of each class

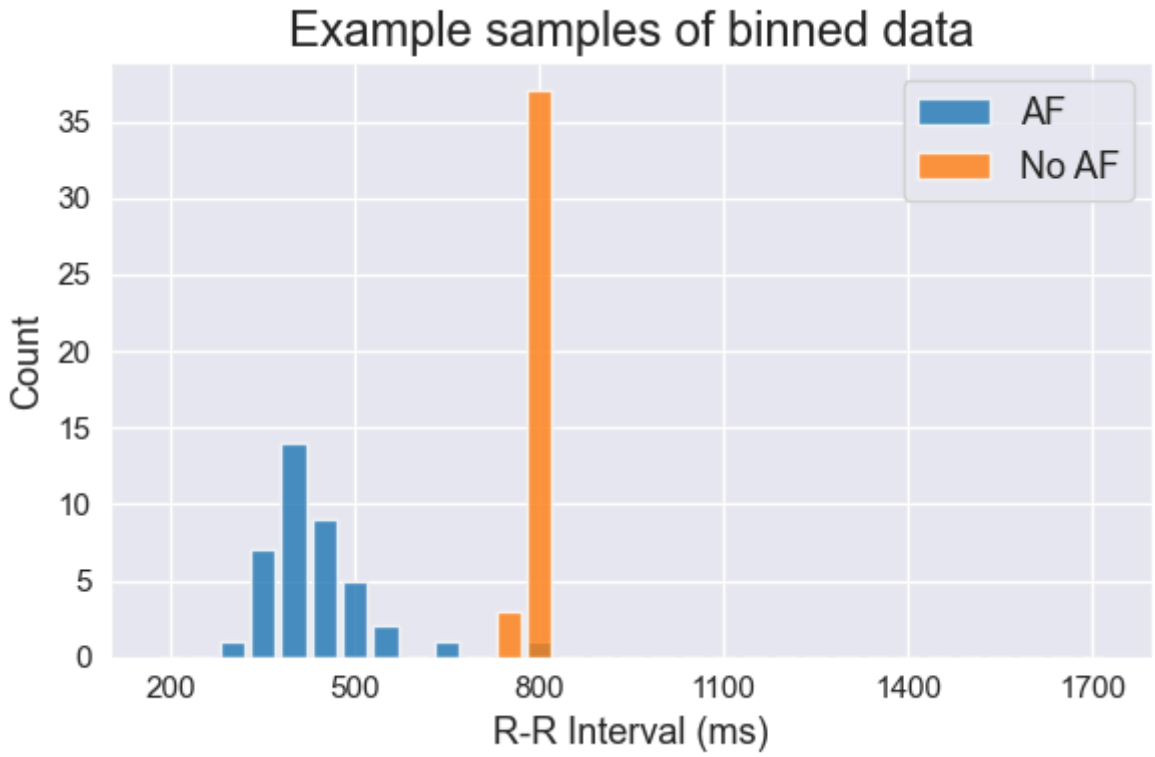

Figure 3: One example sample from each class, on binned data (basically a histogram of Fig. 4)

Figure 4: One example sample from each class, on raw data. Raw data version of Figure 3.

Examining the Figures 2, 3 and 4 we can see what features determine AF or No AF. From the distribution of R-R intervals we see a difference in skew and mean in the two class distributions, indicating which values are telling for identifying AF. Moreover, in Figure 3 and 4 we see that a sample labelled 'No AF' has a stable heartbeat with consistent R-R interval values. On the contrary, when AF occurs the R-R interval values are more widely distributed and are less consistent. These conclusions align with the theory, where patients undergoing an AF episode have more irregularity in R-R intervals.

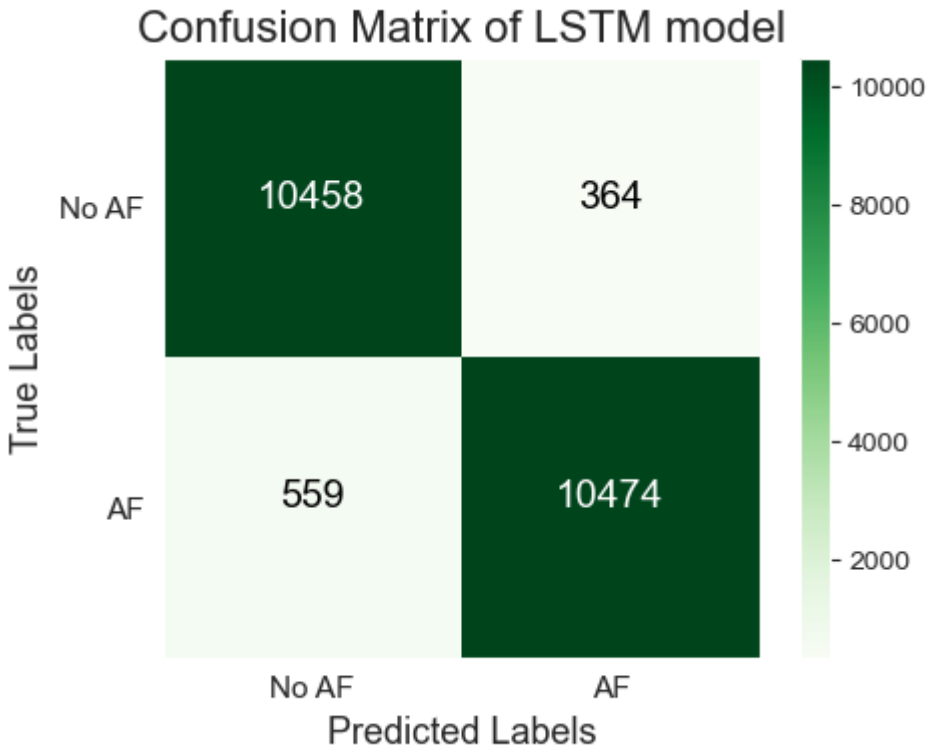

Figure 5: Confusion matrix of (one of the) best performing model on the test set, F1-score = 0.96.

Figure 5 shows the predictions the LSTM model has made versus the true labels. Such a confusion matrix can be summed up with F1-score. We see that very rarely the model wrongly predicts a time window as 'No AF', while the ground truth for that sample is 'AF'. However it does so more often that it predicts 'AF' for ground truth 'No AF'. In healthcare wrong diagnosis can be costly and therefore a certain 'cost' can be assigned to this misclassification.

Reliability of results

First of all the data was provided to us without any patient identifiers, which resulted in a train/test split based on selecting random samples. However, a better split could be made based on patient ids. That way the testing set shows more 'realistic results' because such a model would be deployed on unseen patients and not on patients already seen in the training set. Even better would be to collect data using a different ECG measuring device with patient data from a different location for the test set to show generalisability.

Moreover, we don't know how accurately the ground-truth was (manually) labelled. This could explain why our most complex models are limited by 0.96 F1-score. There might have been entire AF episodes, or pauses between AF episodes, in the data that were missed and wrongfully labelled.

Furthermore, balancing the dataset based on removing samples most likely did not introduce any problems for training the model. Meaning, the number of samples after balancing was significant enough to have reliable training and allowed for a 80/20 training and testing split.

After obtaining our results, we found a source that also implemented similar NN and ML models to AF classification [3]. Their best model achieves roughly 86% testing accuracy. We beat their results by roughly 10% in testing set performance.

Technical depth

Methods that were used for this project that go beyond the scope of the primary topic include: use of various neural network architectures and data transformation pipeline in Python.

Firstly we used Deep Learning techniques such as Convolutional, LSTM and Dropout layers to build four different Neural Network architectures in Python using Pytorch. We learned these techniques from projects we did before and the course "Deep Learning: From Theory to Practice". Convolution is used to analyse spatial dependencies and see patterns and recurrent networks (like LSTM) are used to find temporal dependencies. Because we have data over time, we found both of these Neural Network layers to be applicable to our data. We also incorporated Dropout layers, to prevent overfitting during training and we performed hyperparameter tuning to find optimal values for parameters such as batch size and number of epochs. This includes inspecting the loss and accuracy of the model for each epoch.

Secondly, since the data was provided both in a preprocessed and raw manner, we opted for the challenge and tried to extract more features using the raw data. This feature extraction is of course done by the neural network, but we had to build a pipeline to transform the data from 804 file pairs into a single .csv file while also cleaning the data. Doing so involves the used of a nested for loop and dataframe join operations. Running the data transformation pipeline takes roughly 3 hours on jupyter.utwente.nl.

Conclusions & recommendations

The target stakeholder is the medical institution responsible for the treatment of patients that require cardiac surgery and may also need post operation care. This stakeholder is interested in providing the highest quality care for the patient and providing it in the most efficient way to allow for more capacity of medical care and a reduction of costs. We know that the automatic detection of AF can address all of these concerns.

We have found the LSTM neural network to provide one of the best accuracies and recommend its usage due to its relatively low complexity and computational efficiency. It can detect an occurrence of AF with a 96% accuracy from ECG data. For every 30 second window of ECG data, the model can correctly classify whether it belongs to an AF episode 96% of the time. Furthermore, this model is suitable for runtime monitoring of the patient to immediately detect the AF episode during its occurrence.

While the 4% failure to detect seems to be unreliable, it is the fact that for this use case it is not essential that all time windows that belong to an AF episode are recorded. It is only essential to detect whether the patient suffers from AF episodes, or not. The more time windows occur, the higher the accurate classification rate of the patient's condition. For example, if a patient suffers from 5 minutes of AF episodes in a day, the probability of failing to detect the condition once is almost nonexistent. Therefore, more research is required into the probability of AF episodes post cardiac operation to determine how long to monitor the patient before reaching a satisfiable certainty.

Furthermore due to the false positive rate, we recommend an implementation that only reports detections of AF episodes to the doctor for confirmation.

Reflection

We have addressed all project challenges that we have set up for ourselves, including all challenges provided in the project description on canvas. Initially we had difficulty with using and transforming the raw data, but we successfully processed it and were able to achieve better results by using the raw data. In addition, we think that the skills learned in the topic DM were very useful for this project, but we still needed to research and study additional material in order to realize the project. We think that DM gives a very limited introduction to Machine Learning.

We have used ChatGPT and GitHub Co-pilot to help us code. However, in our opinion these only sped up the process of programming and these AI models did not provide any additional insights we hadn't

thought of ourselves. We didn't use any AI model for helping us write the report, because we think we can better describe our thought process and intentions.

DM issues

For our first approach we:

- analysed the features used by extracting their importances in classification based on the mean impurity decrease (e.g. Gini impurity)
- addressed class imbalance through decreasing the size of the majority class by taking a random sample of the same size as the minority class.
- used a confusion matrix to analyse the accuracy of the models in more depth, to understand the rate of its type of errors.

Four our second approach we:

- used various neural networks and other models to perform classifications from our processed dataset, including: Convolutional neural network, long-short-term memory neural network, combining both an LSTM, CNN and FC layers as well as Dropout, random forest, support vector machine, fully connected neural network, decision tree and logistic regression

More detailed descriptions are covered in the Method section.

References

[1] https://www.hopkinsmedicine.org/health/conditions-and-diseases/atrial-fibrillation
[2] https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html
[3] https://www.mdpi.com/2078-2489/11/12/549

Appendix