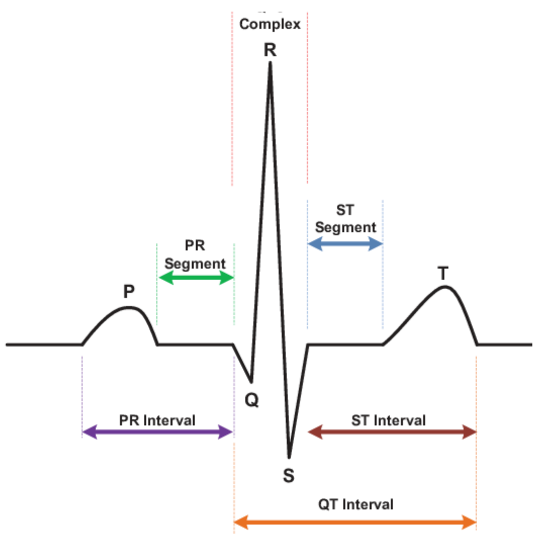
Pre-processing of ECG data for arrythmia detection

This part describes how to read and pre-process **electrocardiogram (ECG)** data. This data will be used in the next part to detect **arrhythmia**. The electrocardiogram is a medical test that detects cardiac abnormalities by measuring the electrical signals generated by the heart during contraction. This test is the most accessible and inexpensive tool for diagnosing conditions like arrhythmia. Arrhythmia is a cardiac abnormality related to the rate and rhythm of the heartbeat. Despite being the most frequently used diagnosing tool, the rates of ECGs misdiagnosis are still too high.

A study from 2016 showed that approximately one in three patients out of 550,000 had their ECGs misdiagnosed and misinterpreted (Wu et al. 2018). This study provides an insight that ECG misrepresentations may have far reaching health diagnostic ramifications. Hence, algorithms for accurate ECG classification should be developed to avoid misdiagnoses before affecting a patient. Traditionally, the analysis of these signals can take time as it is performed by cardiologists. Therefore, automation through **Machine Learning (ML)** classification is being increasingly proposed which would allow ML models to learn the features of a heartbeat and detect abnormalities.

The models however fail to recognize unseen ECG data from new datasets while also facing interpretability and explanation problems, which is a topic we will explore more in our next module. These gaps in interpretability of models occur due to the 'black-box' nature of ML models which render these techniques untrustworthy by clinicians and in-turn limit their application in patient driven healthcare industry.

An ECG test consist of collecting data through the electrical activity of the human cardiovascular system from various angles by placing electrodes at different points on the skin. This non-invasive method of analysing the heartbeat consist of three key features which represent distinct stages of the heartbeat, i.e the P-wave, which reflects the depolarization of the atria; the QRS complex, which shows the depolarization of the ventricles; and the T-wave, which represents the re-polarization of the ventricles. This allows us to detect abnormalities by equating each phase to the normal cardiac cycle. The figure below shows the ECG signal representation of a normal beat. These ECG signals are extremely susceptible to high and low frequency noise which usually occur from baseline wander, misplaced electrode contact, motion artifacts, or power line interference (Velayudhan and Peter 2016).



1. Download the MIT-BIH Dataset

Here, we will work with the MIT-BIH dataset. The MIT-BIH dataset is a public database consisting of many beats and is frequently used for time-series classification research. The MIT-BIH Arrhythmia Database contains sections of ambulatory ECG recordings, from 47 subjects, digitized at 360 samples per second per channel with 11-bit resolution at 10-mV range on two channels, studied by the BIH Laboratory. Here 23 recordings were picked at random from a set of 4000 24-hour ECG recordings collected from a population 60% of inpatients and 40% outpatients (Moody and Mark 2001).

The dataset can be obtained from https://physionet.org/content/mitdb/1.0.0/ and should be stored in a folder called 'mitbih'.

2. Read the data and generate the arrhythmia classifications

The data has been pre-annotated and labelled by cardiologists. These different annotations refer to various normal and abnormal ECG signals which represent different types of arrhythmias. The dataset consists of ECG signals of various classes, but the eight classes used for this investigation are 'N', 'L', 'R', 'V', 'A', 'F', 'f', '/'.

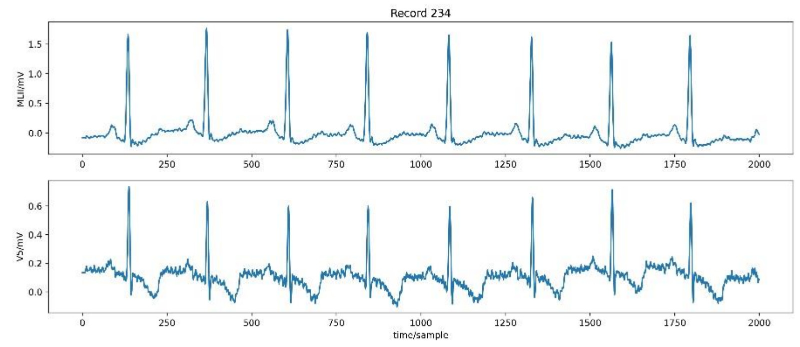
For reading the MIT-BIH dataset, the native python waveform-database (WFDB) package is used, which is a library of tools for reading, writing, and processing WFDB signals and annotations. This allows us to obtain all the annotations for all normal and abnormal ECGs. More details about the package can be found here: https://www.physionet.org/content/wfdb-python/3.3.0/. This also only shows data from one of two channels of the MIT-BIH database. To obtain all the beats, ECG signals from both the channels need to be extracted and stacked. After extracting the 8 important classes that we will be working with and removing all the other classes, you apply data clean-up processes. The first step will be assigning each of the 8 classes a number which would allow us to make the data purely numerical and easy to work with. This is shown in the table below.

Annotation	**Class ID**

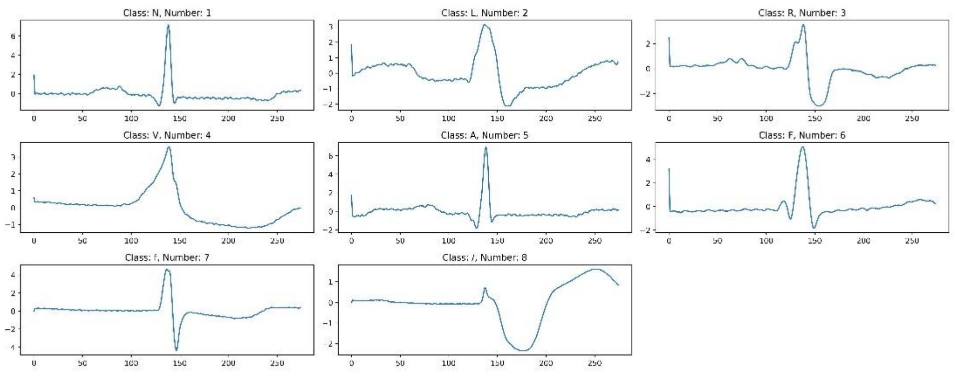
Undetermined	0
Normal	1
LBBBB	2
RBBBB	3
Premature ventricular contraction	4
Atrial premature beat	5
Fusion ventricular normal beat	6
Fusion of paced and normal beat	7
Paced beat	8

3. Pre-processing: centering R-peaks, standardization, and get the patient record number

Next, the data needs to be pre-processed. The figure below visualizes the ECG data before pre-processing.

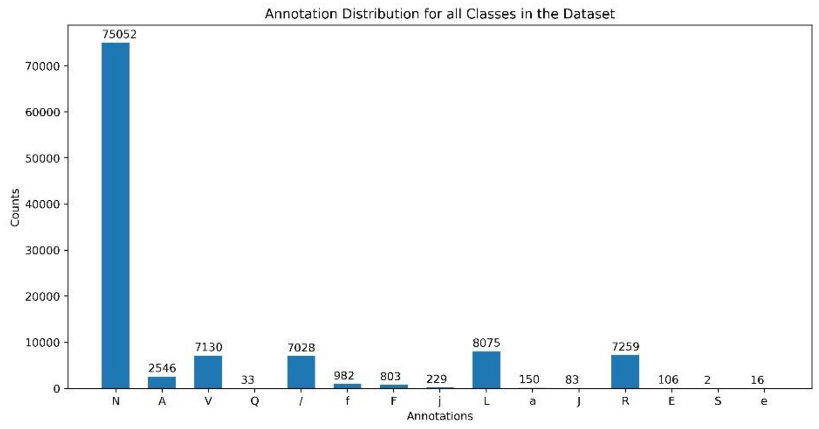


Each patient record included a complete set of heartbeats, so each individual beat has to be extracted from all the records by matching the R-peaks of the ECG with their respective annotation class and appending the class's numerical value at the end of the beat. To make every beat contain equal amount of data points, the R-peaks needs to be centered, and equal amount of data points should be selected on both sides of the peak, this allows us to maintain consistency for each beat. To avoid irregular amplitudes between signals, all the beats need to be standardized. The patient record number needs to be also appended at the end of the beat along with the annotation class number. The figures below visualizes individual ECG beats for each of the eight classes after these pre-processing steps have been performed.



4. Visualize the data per class and save as a CSV file

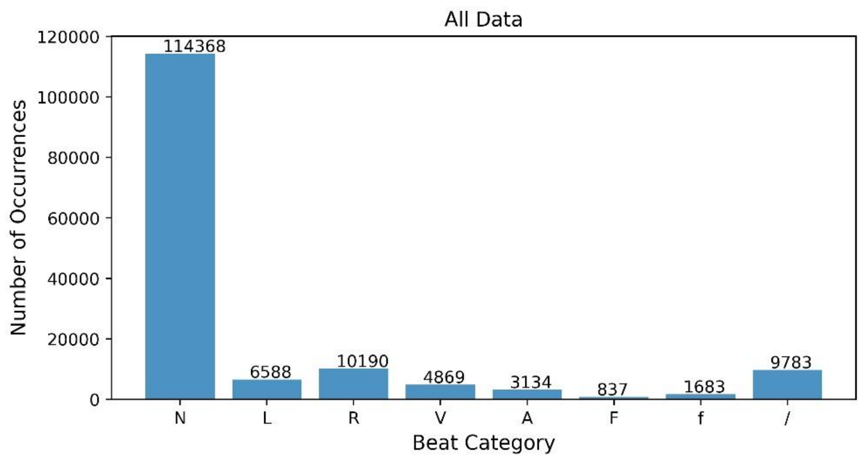
We are interested in the distribution of beats across the different classes. Therefore, we can visualize the data by means of a histogram. The figure below shows the distribution of the beats in the pre-processed data.



The imbalance in the ECG dataset is clearly visible, as we see an abundance of 'N' beats, while all the other beat classes do not even pass the 10,000 thresholds. In the next part, we will use the pre-processed data in deep learning. Therefore, all the cleaned-up data has to be stored into one .csv file containing all beats of all records.

5. ECG data splitting and resampling for machine learning

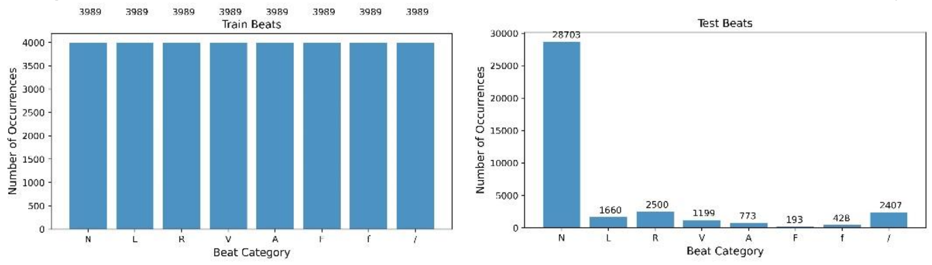
The figure below visualizes the data that we extracted in the previous part. A large class imbalance can be observed. In this part, we will address this problem so we can use the data for deep learning afterwards. We will focus on both beat resampling and patient hold-out validation protocols.



6. Resample training data based on beat classes

To address the imbalance between the classes in the MIT-BIH dataset, we use the resample technique by Sci-kit Learn (Pedregosa et al. 2011). This resampling method uses bootstrap method which estimates statistics on a data population by sampling a dataset with replacement through iteration, using a sample size and the number of repeats. For up-sampling and down-sampling, the value is calculated by taking the mean values of the total number of beats of the abnormal classes. The beat hold-out validation protocol used for training and testing splits the data into 75-25%.

After resampling the training data, you could again visualize the distribution of beats across the different classes. The results for the training and test dataset are visualized below. We observe that, after resampling, all the 8 classes in the training dataset have 3990 samples. It is advised to save the newly generated training and test datasets as .csv files, because we will need them in the next part.

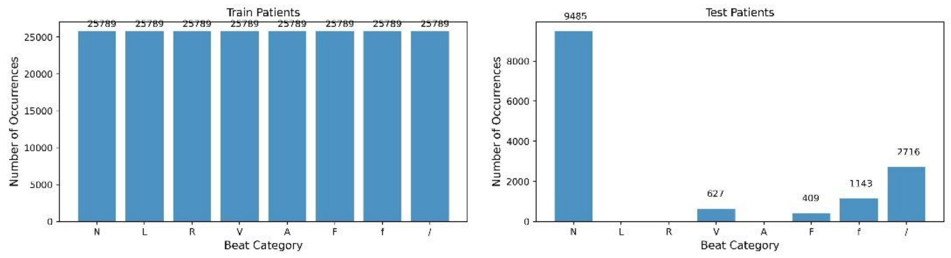


7. Extract training and testing data using the patient holdout validation protocol

The Leave Groups Out validation protocol allows us to test on unseen patients as patient number 104, 113, 119, 208, and 210 are removed from the dataset to be used for testing, while the remaining 43 patients are used for training. Holding out patients from the dataset instead of beats leads to the risk of also removing classes of ECG beats, since a few abnormalities may be limited to one or two patients. The same method has also been used by Jones et al. (2020), where they also chose the same patient numbers to hold out.

The selected patients however do not contain any beats from classes L, R, and A. Patient holdout protocol allowed us to test on beats from unseen patients, whereas the beat holdout protocol shuffled beats of all patients for training and testing purposes. Although testing using the beat holdout method is considered the most common method, it can result in flawed results due to data leakage. Separating ECG recordings by beats and randomly splitting a shuffled set of these beats for testing results in having beats in both the training and testing dataset from the 24 same patient. Due to similar characteristics of patients like electrode placement, pre-existing conditions, and medications, there is a chance of common morphology between beats. In simpler terms, a beat from one patient will probably look similar to another beat (of the same class) from the same patient. This results in data bleeding between the training and dataset, which should be avoided (Jones et al. 2020).

The resulting distribution across beat classes can be seen below for the training and testing data. As can be seen, there are 25789 samples or beats for every category in the training data. However, for the testing data there are some classes missing. It is advised to save the newly generated training and test datasets as .csv files, because we will need them in the next part.



Requirements for assessed exercise:

1. Load and inspect the data, plot few examples from different arrythmia types

First of all, you can use Glob and Numpy libraries to load the holdout beats .csv file into an empty Numpy array. The data should have the following dimensions:

Training data: (number of samples, time) → (for example: 31920, 275)
Testing data: (number of samples, time) → (for example: 37863, 275)

The labels of the heartbeat classes should be one-hot encoded. The table below shows how to encode them.

Annotation	Class	Class ID
Undetermined	-	0
Normal	N	1
LBBBB	L	2
RBBBB	R	3
Premature ventricular contraction	V	4
Atrial premature beat	A	5
Fusion ventricular normal beat	F	6
Fusion of paced and normal beat	f	7
Paced beat	/	8

2. Classification of ECG beats based on the holdout splitting method

In this part, we will aim to classify beats based on the support vector machine. We will apply the previously discussed beat holdout validation protocol on the training data to handle the large class imbalances. In this part, we are using the .csv files of the beat holdout training and testing data that we created in the previous part.

Moreover, we want to create a function which outputs accuracy, precision, recall, f1score and a confusion matrix of the model results.

3. Classification of ECG beats based on the leave out patients-hold out validation protocol

In this part, we will aim to classify beats based on the support vector machine. We will apply the previously discussed patients-hold out validation protocol. Moreover, we want to create a function which outputs accuracy, precision, recall, f1score and a confusion matrix of the model results.

4. For each of the models developed above use permutation feature importance to explain the model's function

Permutation feature importance works by permuting one feature of the dataset at a time. Then, this new dataset and the SVC are used to make new predictions, and the importance of the permuted feature is computed by comparing the change in a performance metric that is caused by the permutation. In this practical exercise, we will be using a version of permutation feature importance that is also using K-fold cross validation. The algorithm is shown below.

Algorithm 1

```
Algorithms for PermFIT
1: Randomly divide the data into $K$ folds.
2: for $k = 1$ to $K$ do.
3:     Denote the data in $k^{\text{th}}$ fold as $V_k$ and the rest of the data as $\overline{V}_k$.
4:     Build the machine learning model with $\overline{V}_k$, denoted as $\widehat{\mu}_k(\cdot)$.
5:     for $j = 1$ to $p$ do
6:         Calculate $\widehat{M}_{ij}^{(P,CV)}$ for subjects in $\mathcal{D}_k$.
7:     end for
8: end for
9: for $j = 1$ to $p$ do
10:     Calculate $\widehat{M}_j^{(P,CV)}$ and estimate $\widehat{\text{Var}}\left[\widehat{M}_j^{(P,CV)}\right]$. Calculate p-value by assuming nomality.
11: end for
```

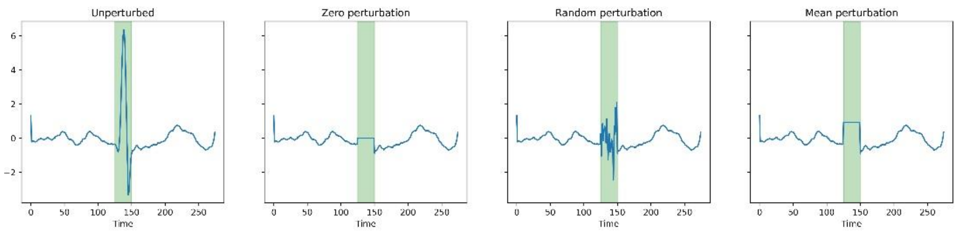
In the algorithm, j denotes the current feature, i denotes the sample, and matrix M denotes the feature importance matrix. The feature importance matrix is computed as follows:

$$\widehat{M}_{ij}^{(P,CV)} = \sum_{k=1}^{K} \mathrm{I}\left(i \in V_k\right)\left[\left\{Y_i - \widehat{\mu}_T\left(X_{i\cdot}^{(j)}\right)\right\}^2 - \left\{Y_i - \widehat{\mu}_k(X_{i\cdot})\right\}^2\right],$$

Where Xi(j) is the input with feature j permuted, and Xi is the original input. Yi are the true labels for sample i, and µT and µk are the (trained) models. We will not estimate the variance and the p-value in this exercise, but we will take the average over all the samples within a slice.

Next, we need to implement the permutation feature importance algorithm with K-fold cross validation for the SVM. The cross-validation should be stratified on the heartbeat class. We have used 5 folds. Within every fold, we use the training data to fit a new SVM and the test data to get predictions. Then, we take all the data that falls within a slice, permute the corresponding columns, and make a new prediction. These two predictions, together with the true labels, are used to compute the importance score Mij. Afterwards, we can average over the samples, so that we get the mean importance score per feature.

One important decision for permutation feature importance is how to permute the data. The figure below visualizes three different approaches: zero perturbation (set every sample within a slice to zero), mean perturbation (set every sample within a slice to the average of the slice), and random perturbation. For this practical exercise, we would like you to randomly shuffle the data points within a slice. Note that we have 11 slices, each containing 25 samples.



6. Apply the same explainability technique with different type of classifiers and discuss the differences

Minimum number of experiments in the following:

- Inspect your data and plot different types of arrythmia. Run the python notebooks provided and plot also the distribution of samples across classes

4-6 members per group: (**At least two** different classifiers)

- Classification of ECG beats based on the holdout splitting method
- Classification of ECG beats based on the leave-out, patients-hold out validation protocol
- For each of the models developed above use permutation feature importance to explain the model's function
- Apply the same explainability technique with different type of classifiers and discuss the differences

6 members per group: (In addition to the above task):

- Use **at least one** clustering technique to visualize the data and understand better their structure and how well classes are separated. Perform classification based on the cluster features and discuss the results.

References:

Jones, Y., Deligianni, F. and Dalton, J. (2020), Improving ecg classification interpretability using saliency maps,in'2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)', pp. 675–682.

Mi, X., Zou, B., Zou, F., & Hu, J. (2021). Permutation-based identification of important biomarkers for complex diseases via machine learning models. Nature communications.

Wu, J., Gale, C. P., Hall, M., Dondo, T. B., Metcalfe, E., Oliver, G., Batin, P. D., Hemingway,H., Timmis, A. and West, R. M. (2018), 'Editor's choice - impact of initial hospital diagnosison mortality for acute myocardial infarction: A national cohort study', European Heart Journal: Acute Cardiovascular Care7(2), 139–148.

Velayudhan, A. and Peter, S. (2016), Noise analysis and different denoising techniques of ECG signal - a survey, in 'Noise Analysis and Different Denoising Techniques of ECG Signal – A Survey'.

Moody, G. and Mark, R. (2001), 'The impact of the mit-bih arrhythmia database', IEEE Engineering in Medicine and Biology Magazine20(3), 45–50.