ML4H Project 1

April 18, 2023

Aloïs Thomas
alothomas@student.ethz.ch

Jaro Meyer
jarmeyer@student.ethz.ch

Part 1: Heart Disease Prediction Dataset

Q1: Exploratory Data Analysis

The classes in the training dataset are well balanced (398 positive samples, 336 negative samples) so we can skip techniques to counter class imbalance.

We looked at the minimum and maximum value of every column in both training and test datatset and found out that in the training dataset there are 141 rows with `Cholesterol == 0` and one row with `RestingBP == 0`. These are clearly impossible values; thus, we have to manually imputate the data. We chose to just use the mean value over the whole training dataset for both `Cholesterol` and `RestingBP`. The same is done for all rows with `Cholesterol == 0` in the test dataset using the mean calculated on the training dataset.

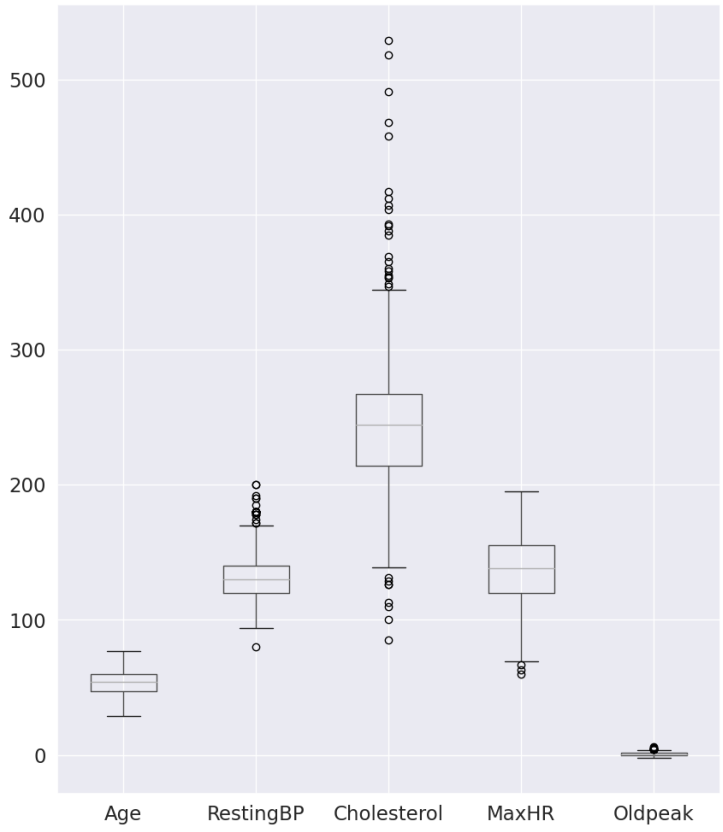

Figure 1: Boxplot of numerical features

To get a rough estimate of the distribution and possible outliers we first created a boxplot (depicted in Figure 1) showing all numerical features. This reveals that `Cholesterol` has some pretty extreme outliers, but otherwise everything looks good.

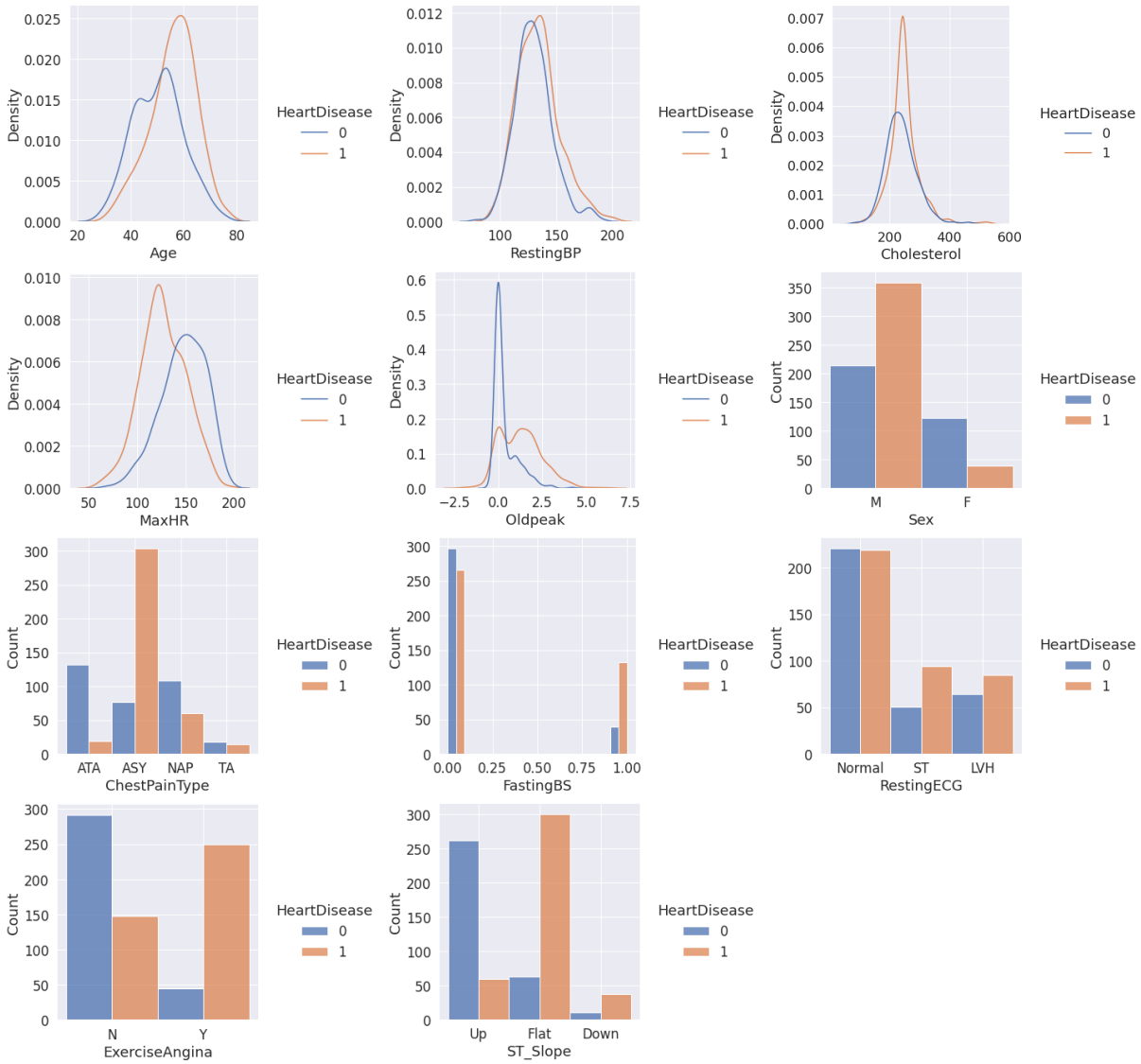

Figure 2: Feature distributions by class

As the boxplot may hide many of the details of the distributions, we also plotted the distribution by class for every feature seperately (depicted in Figure 2). For numerical features kernel density estimation (KDE) was used to represent the data using a continuous probability density curve. From just looking at the curves there were no obvious abnormalities that would require further preprocessing. Notable were `ST_Slope`, `ChestPainType`, `ExerciseAngine`, `OldPeak`, and `MaxHR`. Those features show a drastically different distribution depending on the class, suggesting their important for predicting `HeartDisease`.

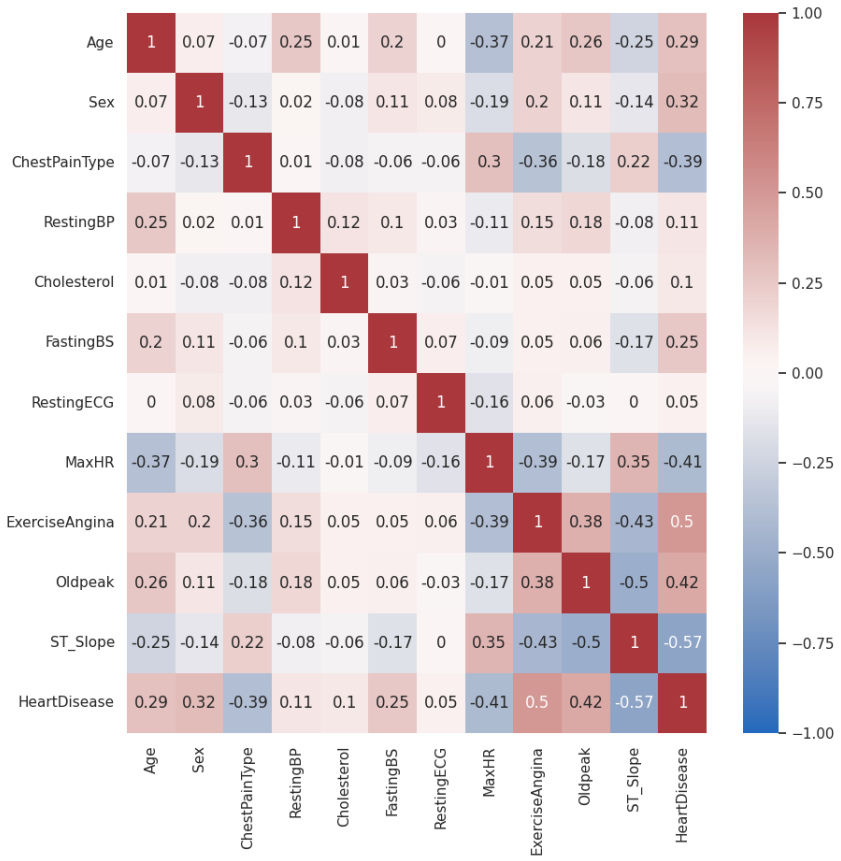

Figure 3: Correlation matrix

Next we explored pairwise correlation. Looking at the correlation matrix as depicted in Figure 3 one can see that `HeartDisease`, `ST_Slope`, `ExerciseAngine`, `OldPeak`, and `MaxHR` are all highly correlated. This confirms that `ST_Slope`, `ExerciseAngine`, `OldPeak`, and `MaxHR` will likely be important for predicting our label `HeartDisease`. However no two of these features are perfectly correlated (>0.8) so we decided not to delete any features.

Because some features are categorical we have to convert them to numerical values. The strategy we chose to accomplish this is basic label encoding.

Q2: Logistic Lasso Regression

Because the coefficients directly depend on the scale of the input features, it is very important to standardize the data. Otherwise features with a small scale will result in large coefficients and thus appear more important.

Using scikit-learn we fitted a logistic regression model with L1 regularization. With $\alpha = 0.1$ only 6 coefficients are non zero after training (see Table 1), indicating that the corresponding features `ST_Slope`, `ExerciseAngina`, `ChestPainType`, `Sex`, `MaxHR`, and `Oldpeak` are relevant for the regresion. It is important to note that the same features have already been shown previously to all have a high correlation coefficient relative to `HeartDisease`. To select more or less features we could choose another α value. The F1-score of the model on the test dataset is 0.862.

To further confirm the importance of these features revealed by the Lasso regression we also fitted a standard logistic regression using only the 6 selected features mentioned above, leading to a F1-score of 0.837. This overall good F1-score clearly proves the importance of the 6 features since they are sufficient to achieve decent classification performance. The slightly lower F1-score compared to Lasso

regression is probably the result of overfitting because we removed the L1 penalty. Therefore it is still sensible to use Lasso over standard logistic regression to reduce overfitting.

ST_Slope	-0.136763
ExerciseAngina	0.069196
ChestPainType	-0.031884
Sex	0.018499
MaxHR	-0.013677
Oldpeak	0.004876
Age	0.000000
RestingBP	0.000000
Cholesterol	0.000000
FastingBS	0.000000
RestingECG	0.000000

Table 1: Coefficients of Lasso regression with $\alpha = 0.1$ (sorted by magnitude)

Q3: Decision Trees

For this subtask, we trained a decision tree classifier on the dataset using scikit-learn and visualized feature importance according to the Gini coefficients. Our trained model achieved a F1-score of 0.787 on the test set, which is notably worse than the Logistic regression classifier from the previous subtask. From the Gini importances listed in Table 2 we can see that for this model `ST_Slope`, `ChestPainType`, `Cholesterol`, and `Oldpeak` belong to the most important features, since they have the highest Gini coefficients. These results generally agree with previous observations, however here `ExerciseAngina` seems to be of less and `Cholesterol` of more importance compared to the results from Q2.

ST_Slope	0.405451
ChestPainType	0.113993
Cholesterol	0.095950
Oldpeak	0.091048
MaxHR	0.083790
RestingBP	0.082555
Sex	0.049404
Age	0.046498
RestingECG	0.021871
FastingBS	0.005871
ExerciseAngina	0.003569

Table 2: Gini importances (sorted by magnitude)

Q4: Multi-Layer Perceptrons

For this subtask we used TensorFlow & Keras to create a simple neural network with a total of 1,321 trainable parameters. It consists of two dense layers with ReLU activation and 22 respectively 44 outputs and a final layer with a single output and sigmoid activation function. To train the model we first take 20% from the training dataset to create a validation dataset and standardize all datasets. The model was then trained on the remaining 80% of the training dataset for 20 epochs using the Adam

optimizer with a learning rate of 10^{-3}. The classification report of the model on the test set can be viewed in Table 3. As one can see, we get fairly good performance metrics with an F1-score of 0.81 and accuracy of 0.82. We then used the SHAP library to calculate the shapley values for all samples from the test set and plotted them for 4 positive and 4 negative samples in Figure 4 and Figure 5. Finally we also visualized the average feature importance over the whole test dataset in Figure 6.

Accuracy	0.82
Precision	0.81
Recall	0.81
F1-score	0.81

Table 3: Classification performance on test dataset

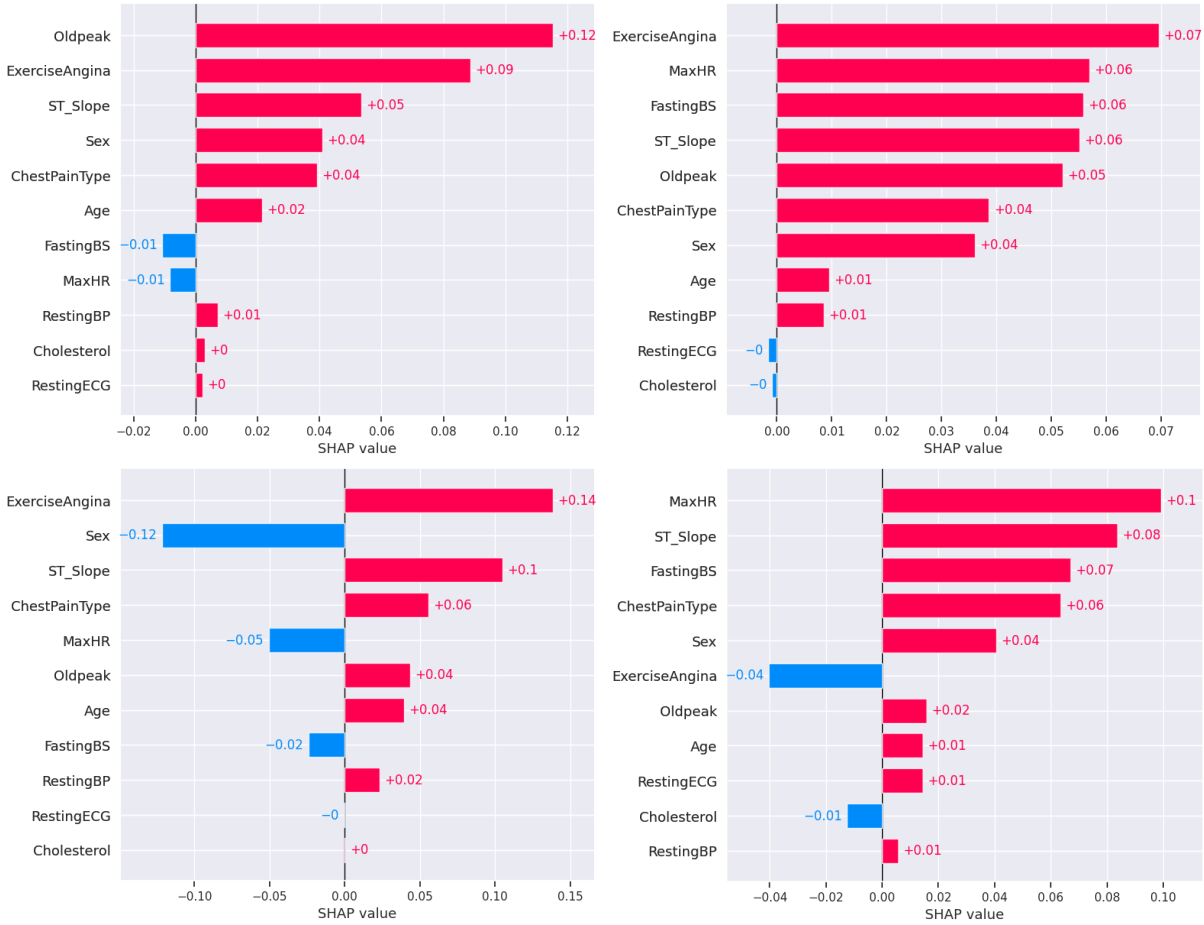

Figure 4: SHAP values for 4 positive samples

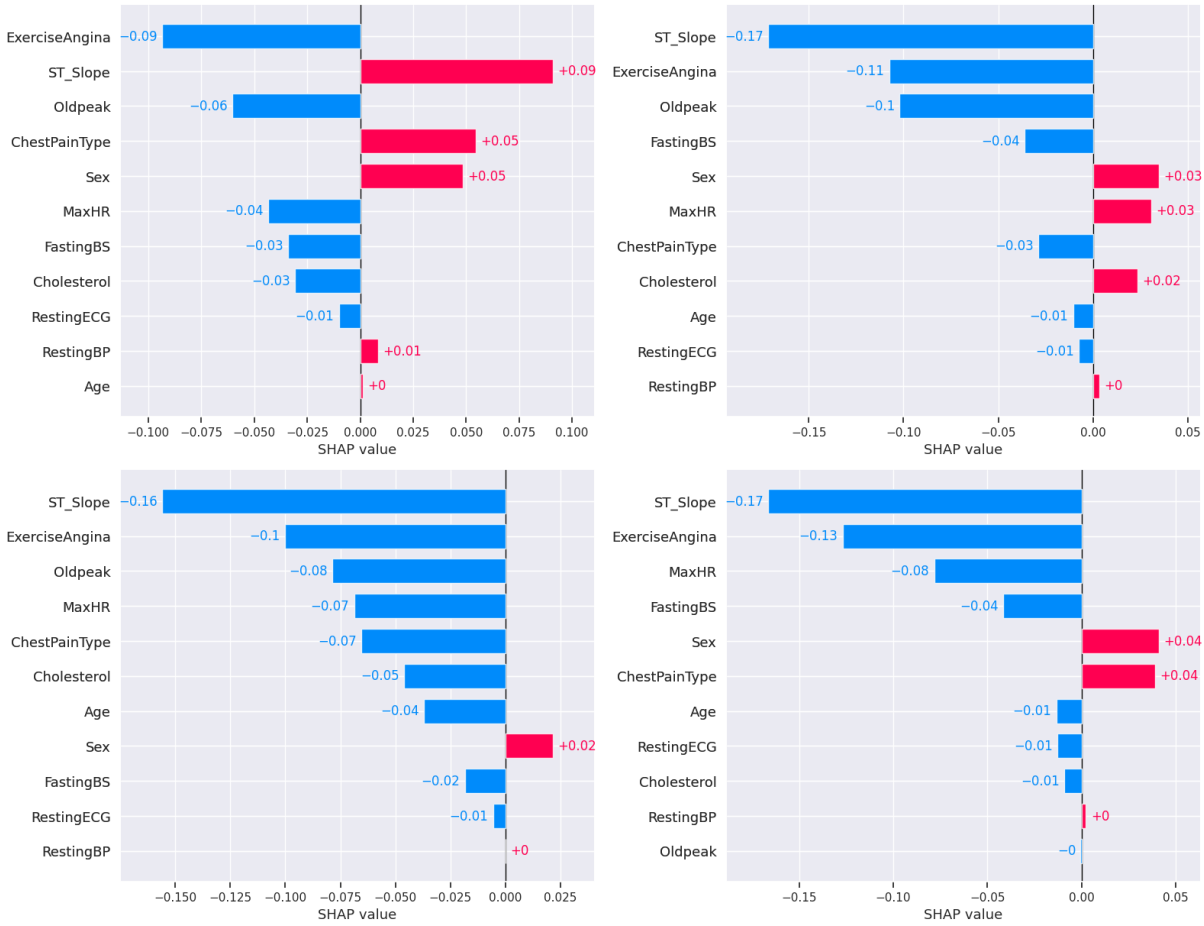

Figure 5: SHAP values for 4 negative samples

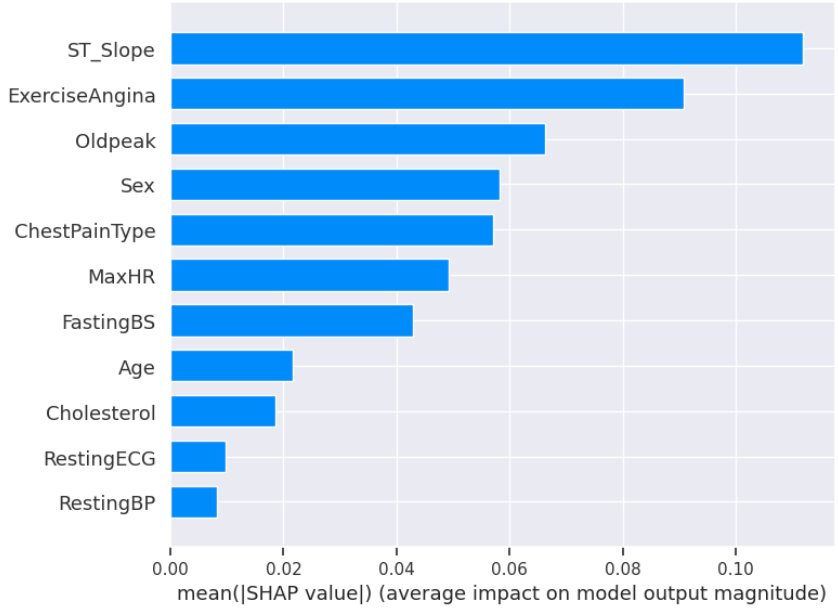

Figure 6: Overall feature importance

While for all negative samples `ST_Slope` and `ExerciseAngina` seems to have the largest impact, there is a bit more variance for the positive samples. `MaxHR` and `FastingBS` seem to be more important for predicting positive samples. Both `ChestPainType` and `Oldpeak` also consistently rank high across all shown samples. Looking at Figure 6 we can see that overall `ST_Slope` clearly has the most impact

on the output, closely followed by `ExerciseAngina`, `OldPeak`, `Sex`, and `ChestPainType`. `RestingBP` and `RestingECG` have close to no impact on the predicted output and could thus be removed without negatively impacting classification performance. These findings also agree with the general theme from Figure 4 and Figure 5.

Challenge 1: Neural Additive Models

We again used Keras to build a simple NAM style neural network. For every feature there is a small MLP conisting of two ReLU activated dense layers with 4 respectively 8 outputs and a final dense layer with a single output and sigmoid activation. To train the model we connect the outputs of the 11 "subnets" using an averaging layer. Then we fitted the completed model to optimize the binary crossentropy loss using Adam (lr = 10^{-3}, 10 epochs). The trained classifier reached a F1-score of 0.82 on the test dataset. Standardized training, test, and validation dataset were reused from Q4.

To get the feature importances the averaging layer was removed so that we get 11 (one per feature) independent predictions between 0 and 1 for every sample. By calculating the absolute distance from 0.5 (high uncertainty) we obtain 11 values representing how much impact a certain feature had. Figure 7 and Figure 8 show the plotted importances for the same 8 samples as in Q4.

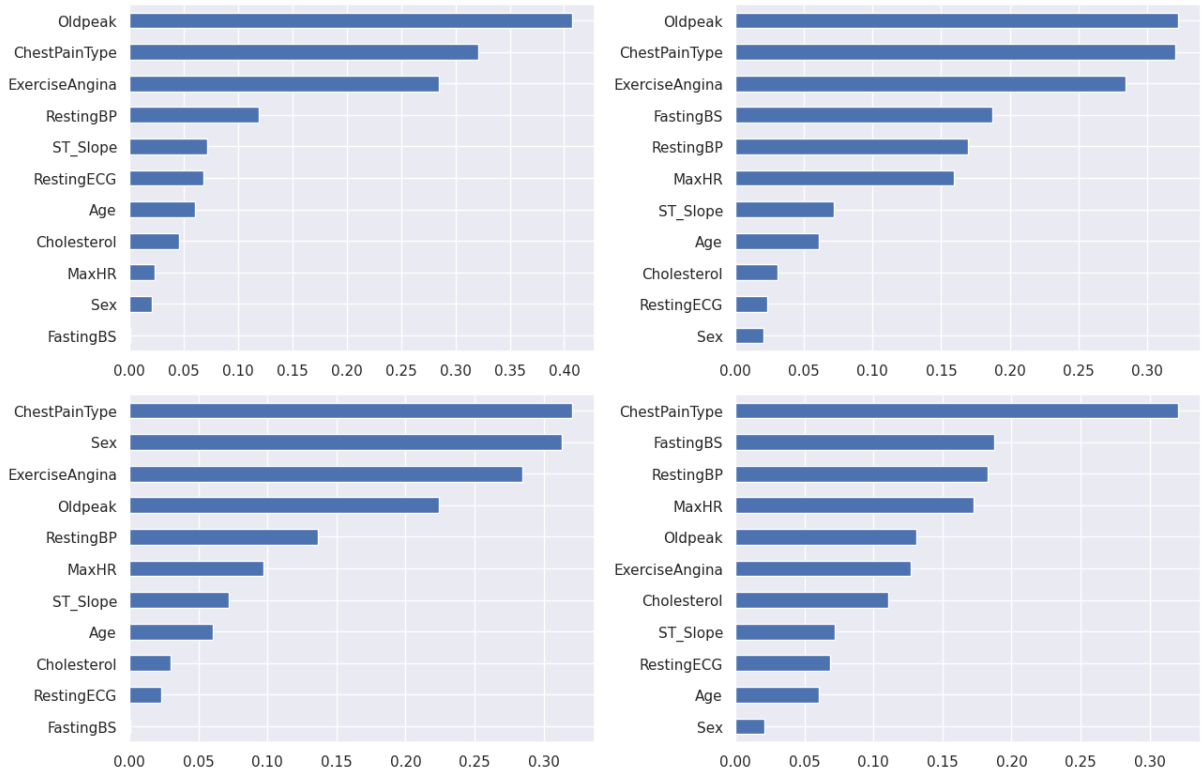

Figure 7: NAM feature importances for 4 positive samples

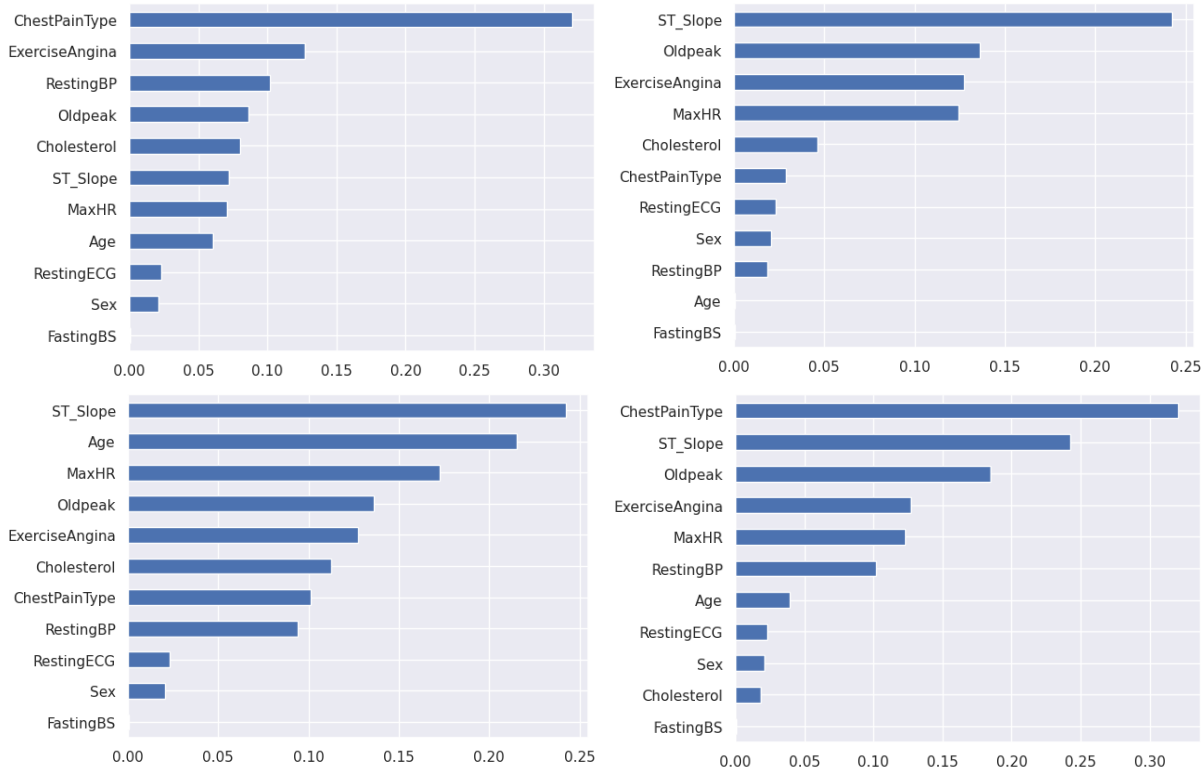

Figure 8: NAM feature importances for 4 negative samples

Importances between samples vary quite a lot. For example `FastingBS` had a high impact on the predictions of both left samples in Figure 7 but for all other samples it had no impact at all. However apart from some exceptions we can still observe that overall the same features as in Q2-4 (especially `ST_Slope`, `ChestPainType`, and `Oldpeak`) consistently rank high.

While the functions learnt per feature by the subnets are non linear, the way they are combined to form the final prediction is. Therefore the inputs to the averaging layer can directly be interpreted as some measurement of importance, similar to what we did in Q2. It would also allows to seperately visualize the learnt single input functions to get an idea of how a feature affects the final prediction.

Because we only have to propagate every sample through the neural network exactly once to obtain these values compared to thousands of perturbations per sample in Q4, this method is orders of magnitude faster than SHAP (<1s vs 370s for the whole test set).

Part 2: Pneumonia Prediction Dataset

Q1: Exploratory Data Analysis

The Pneumonia dataset contains chest X-ray images labeled either `Pneumonia` or `Normal`, split into three disjoint datasets: training dataset, validation dataset, and testing dataset. We will sometimes also call the classes `positive` and `negative`. The training dataset contains 3875 images labeled `Pneumonia` and 1341 `Normal` images, thus we are facing a quite drastic class imbalance (~3:1) which may lead to bias in the model for predicting `Pneumonia` for a given input image. The solution we chose to combat this bias will be explained in the CNN part. The test dataset is better balanced with an approximately equal number of images of `Pneumonia` (390) and `Normal` (234). Finally, the validation set only contains 8 images per label (total of 16 images) which makes it useless for proper validation due to the low number of images. For this reason we ignore it and we created our own validation dataset by randomly taking 20% from the training dataset.

By comparing random images we can also see that not all the images have the same format in terms of resolution, sharpness, brightness, contrast, and number of channels. Some preprocessing to normalize the images is thus required. Prior to training we rescaled all images to 224x224 pixels.

From Figure 9 one can also see that it appears that images labeled `Pneumonia` presents lungs with asymetries, blurry shadows, and changes in contrast. However some images labeled `Pneumonia` are very hard to distinguish manually from the `Normal` ones.

We also notice that most images have the letter 'R' written somewhere to mark the right side. There could be a possibility of overfitting where the model just 'remembers' the position of the 'R' for all training images instead of actualy looking at the lungs. Many images also contain foreign (metal) objects like implants or wires.

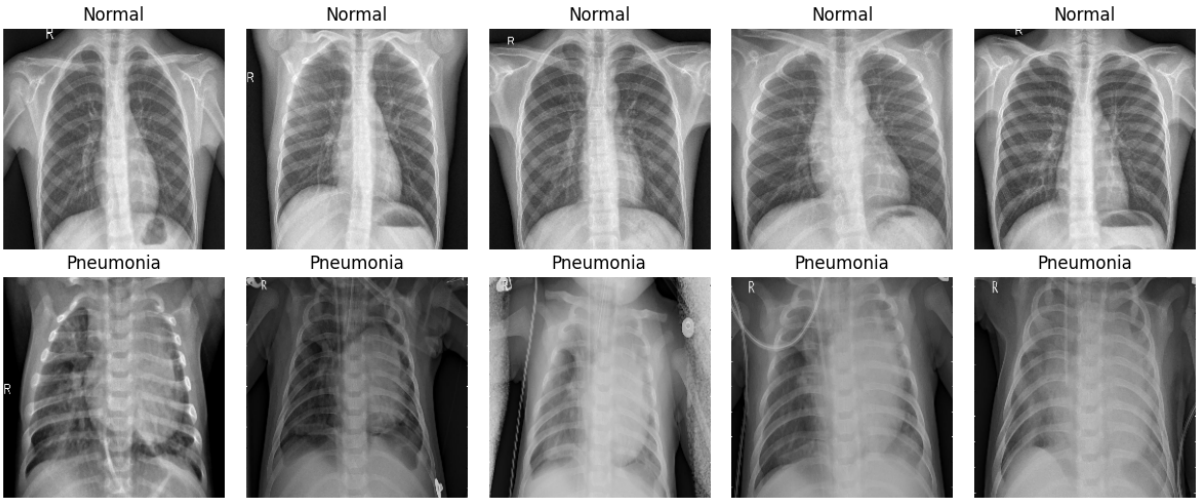

Figure 9: 5 positive and 5 negative randomly selected samples

Q2: CNN Classifier

We used Tensorflow and Keras to build a simple CNN binary classification model with 5 (Conv2D -> ReLU -> MaxPool)-blocks where each block doubles the number of channels while halving the spacial resolution in both dimensions. The output of the last block is directly connected to a single dense layer with softmax which predicts the probabilities for both classes. Thus the CNN is composed of 110,786 total parameters which all are trainable. The complete structure of our CNN can be found in the appendix. In order to counter class imbalance, we used class weights computed on the training set. These weights scale the loss depending on the class of a training sample.

The model was trained using the Adam optimizer (lr=10^{-5}, batchsize=64) for 30 epochs.

To evaluate the performance, we print precision and recall on the validation dataset after every epoch and after training we report accuracy, precision, recall, and F1 score on the testing dataset and show the corresponding confusion matrix. The performance metrics of the test set can be found in Table 4.

Accuracy	0.880
Precision	0.889
Recall	0.923
F1-score	0.906

Table 4: Performance metrics on the test set

From this table, we can see that we get a fairly good accuracy and a very high recall probably mostly due to class imbalance and thus a bias towards predicting `Pneumonia`. This is confirmed by looking at the confusion matrix depicted in Figure 10 where we have 45 false positives and only 30 false negatives.

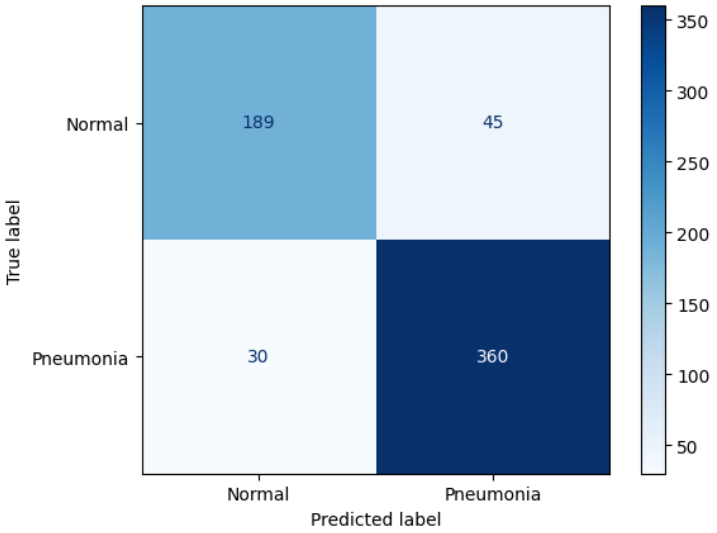

Figure 10: Confusion matrix of predictions on the test set

Q3: Integrated Gradients

In this part we used an implementation of the integrated gradients algorithm for Keras from the TensorFlow website [1] to try to visualize the important regions of the images on which the CNN bases its predictions for the positive class. We then plotted the attribution mask overlayed over the original images for 5 positive and 5 negative test samples in Figure 11. Most notable is the strong attribution of the overlayed 'R' across all samples. Apart from that integrated gradients seem to highlight pixels with high intensity from the and ribs spine. Additionaly the overall attribution strength is higher for negative samples than for positive ones.

Unfortunately we were not able to see a meaningful concept which means that the integrated gradients method is unable to properly explain to a human how a prediction was made.

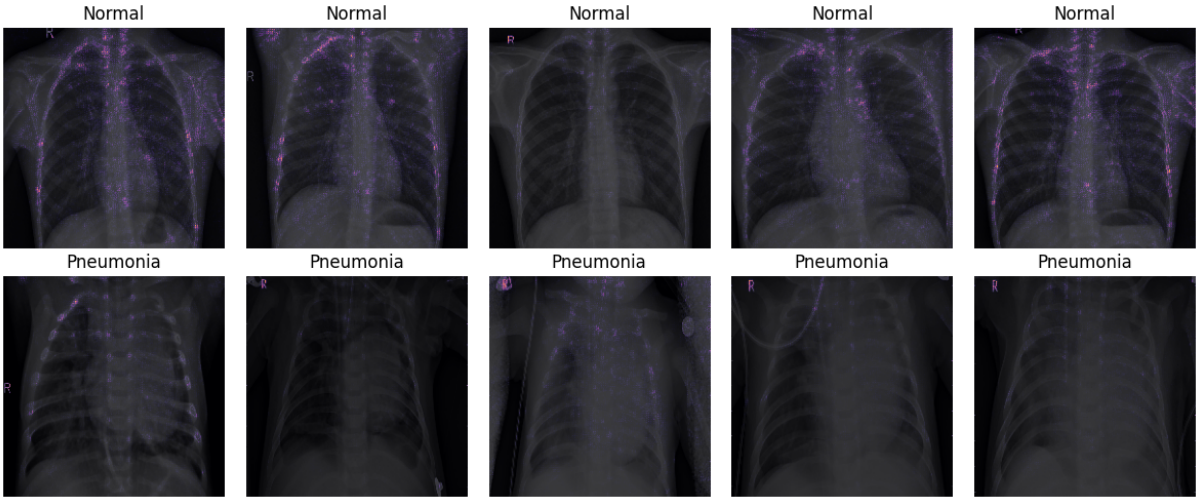

Figure 11: Integrated gradients visualization of 5 positive and 5 negative samples

Q4: Grad-CAM

Here we used a Grad-CAM implementation from a Keras example [2] to visualize the activation maps for the positive class on 5 positive and 5 negative samples from the test dataset. The result can be viewed in Figure 12. These visualizations show that our model has actually learnt, that pneumonia has to do with the lungs and it correctly focuses on the lungs in all images while largely ignoring other regions. This ability to show that our model has learnt a correct concept and to show which parts of the lungs look suspicious makes this method more suited for interpreting the predictions than the integrated gradients results in **Q3**. The Grad-Cam visulaizations also confirm our observation from **Q1** that pneumonia often presents itself as an asymmetry of the lungs. While in the negative samples the hotspot is quite small and centered around the spine and part of the heart, in the last three positive samples the hotspot region is way larger and extends to the right border of the ribcage. Also according to Grad-Cam the 'R' had no significant impact on the predictions.

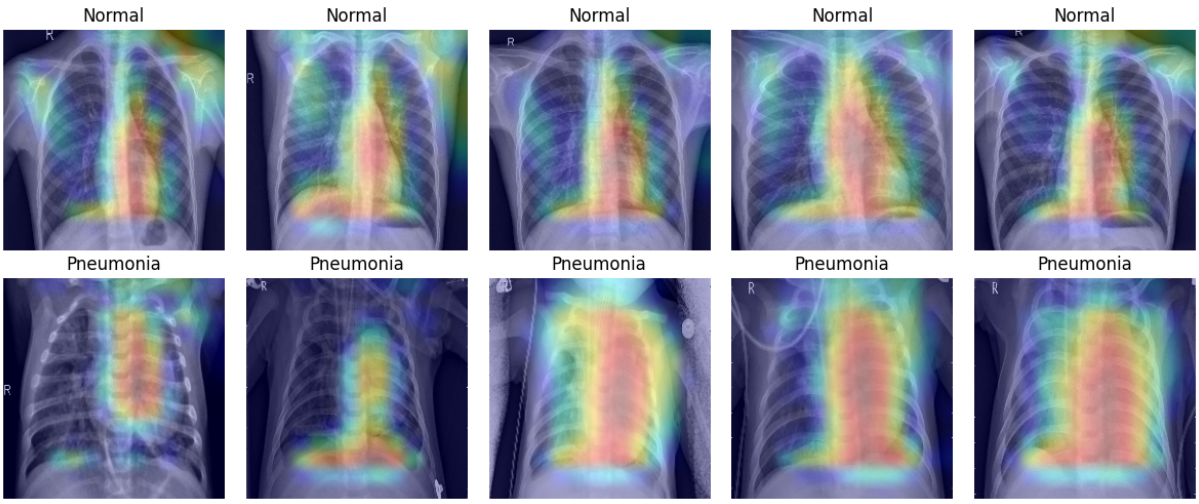

Figure 12: Grad-CAM visualization of 5 positive and 5 negative samples

Q5: Data Randomization Test

To investigate the trustworthyness of the activation maps from **Q3** and **Q4** we retrained our model on the training dataset but this time with randomized labels. Then we again visualized the same 5 positive and 5 negative samples using both integrated gradients (Figure 13) and Grad-CAM (Figure 14).

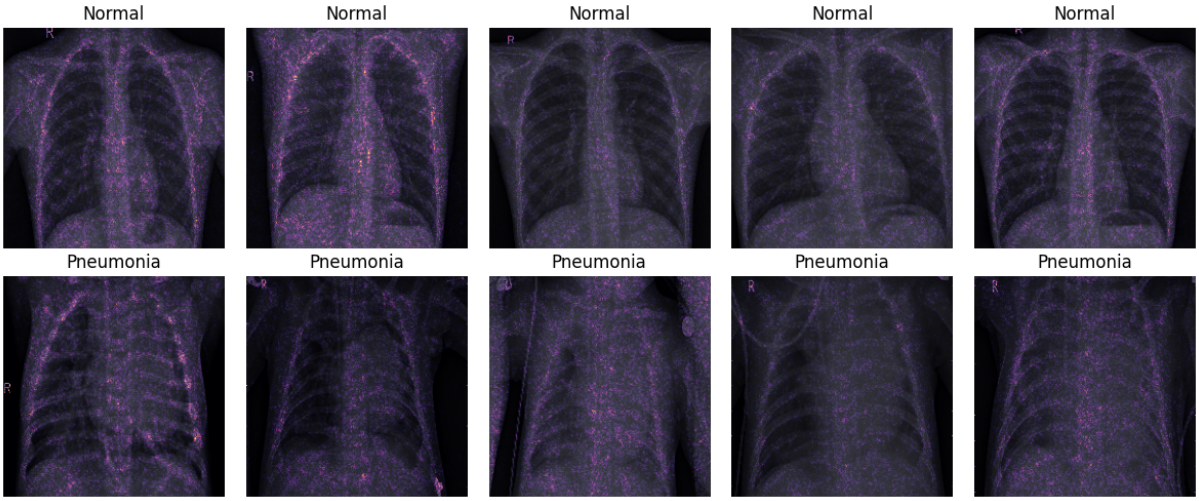

Figure 13: Integrated gradients visualization of model trained on randomized labels

As suspected in Q3 the results confirm, that integrated gradients partly visualize the edge detection part.

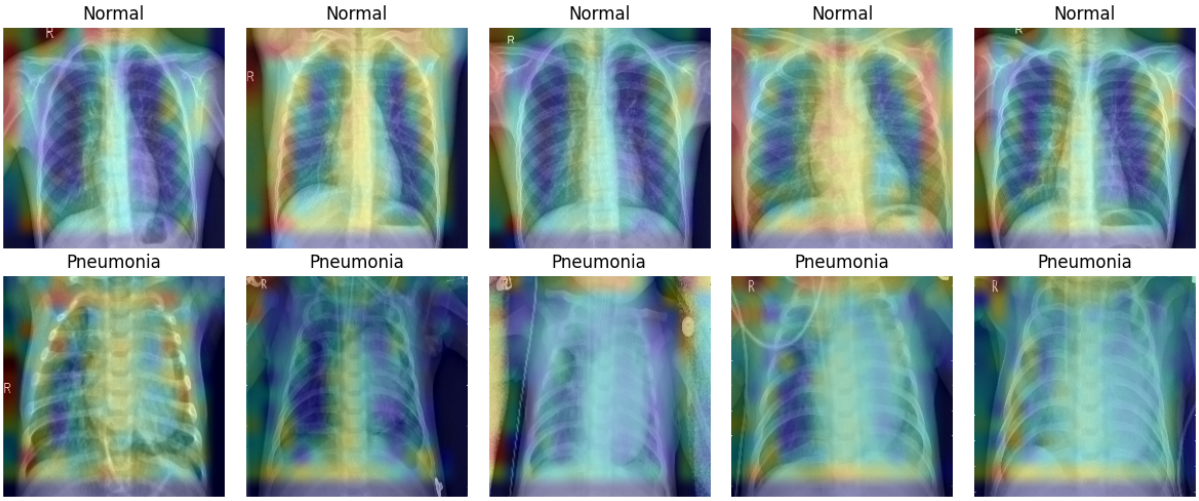

Figure 14: Grad-CAM visualization of model trained on randomized labels

The integrated gradients maps all only contain noise with an intensity proportional to the instensity of the original pixel. The white 'R' and the section where the ribs intersect the image plane thus show the strongest activations. This confirms that the maps produced in **Q3** actually show some learnt concept that was used to differentiate between the classes, but still we werent able to completely understand what this concept is.

The heatmap generated by Grad-CAM are completely different to what we saw in **Q4** as they don't show any hotspots on the lungs. Rather the attribution is pseudorandomly distributed over the whole image. One consistent observation is the strong region around the top and left edges where many images contain the overlayed 'R'. This would suggest that Grad-CAM managed to successfully explain the learnt concepts and what influenced a prediction.

Part 3: General Questions

Q1: How consistent were the different interpretable/explainable methods? Did they find similar patterns?

Part 1:
When comparing the logistic Lasso regression, decision tree, and MLP with Shapley values, we can observe a certain amount of similarity in the interpretability of the models. Indeed, all of these models show a generally high importance for `ST_Slope`, `ChestPainType`, `Oldpeak` and `Sex`. However the exact order of those features fluctuate between the models and between different samples for the MLP and NAM. For both neural networks the initial weight initialization also affects the final importances. We can also note that `ExerciseAgina` has a fairly high lasso coefficient of 0.069 but is almost totally absent in the deceision tree in term of importance with a gini coefficient of 0.004.

Part 2:
When comparing integrated gradients and Grad-CAM for the CNN designed in **Part 2 Q2**, the maps generated by Grad-CAM are clearly better for interpreting the results as they correctly show the inflammated parts of the lungs. The integrated gradients look completely different as they mainly highlight areas of the skeleton which intuitively are insignificant for predicting pneumonia.

Q2: Given the "interpretable" or "explainable" results of one of the models, how would you explain and present them to a broad audience? Pick one example per part of the project.

Part 1:
If one the models had to be presented to a broad audience, we would select the logistic lasso regression as the math behind how the coefficients affect the output is pretty easy. We would first start to explain what a lasso regression is and how it works and present how we used this technique to elaborate a model to predict Heart Disease. Showing and explaining how accurate the model is in its predictions and clearly showing which features have been used and comparing their lasso coefficient with the correlation matrix would be an important part of the presentation in order to clearly show how the model makes the predictions.

Part 2:
Here Grad-Cam is the clear winner. The generated heatmaps show inflammated tissue and asymmetry of the lungs which is easy to understand for someone with no computer science or medical background. In order to present it to an audience, explaining the architecture of the CNN as well as how the Grad-CAN results were produced would be required.

Q3: Did you encounter a tradeoff between accuracy and interpretability/ explainability?

Part 1:
In this case we didn't encounter such a tradeoff since logistic lasso regression, which is an inherently interpretable model, achieved the highest F1-score of all models that we tested. Also if there was some neural network with a higher score we could still use SHAP to accurately explain the predictions.

Part 2:
We tried more complex CNNs and residual networks which had a slightly higher F1-score than our simple CNN, but as the network was able to learn more complex concepts, the Grad-CAM maps

looked more arbitrary and less easily understandable by humans. Overall we can say that for neural networks increasing the complexity of the model lead to less interpretable.

Q4: Do your findings from the interpretability/explainability methods align with the current medical knowledge about these diseases? You may take inspiration from the references of the project presentation.

Part 1:
All models assigned `ST_Slope` a high importance and indeed irregular cardiac rythm which can be detected with an ECG by analyzing heart rythm and ST intervals [3]. Furthermore chest pain, which can be a sign of potential upcoming heart failure, is also given great importance by all models. Overall, we can say that the interpretability methods do align well with current medical knowledge about heart failure.

Part 2:
Current diagnosis of pneumonia is done partly with lung X-ray imagery [4]. An X-ray image with lung asymetry, different contrasts in the lungs region is usually attributed to pneumonia. According to Grad-CAM the same areas used in current medicine seems to be used by our CNN to classify an image as pneumonia and thus matches partly current practices in medicine.

Q5: If you had to deploy one of the methods in practice, which one would you choose and why?

Part 1:
Using only F1-scores as a performance metric, logistic lasso regression should be the model selected to be implemented in practice. Furthermore, it is also a fairly simple model which can be easely understood by clinicians and the lasso coefficients inherently allow interpretation. We can also mention the relatively low computational cost required to train the model compared to neural networks and since the coefficients are used there is no additional computational cost for interpretability like with SHAP for example.

Part 2
Grad-CAM would be the method of choice to deploy in practice as it highlights suspicious regions of the lungs which could help a doctor to further verify the diagnosis while integrated gradients failed to visualize an understandable concept. Additionaly it is a widely used method to provide interpretability and it's computationally fast and easy to implement thanks to great libraries.

Bibliography

[1] “Integrated gradients.” ((https://www.tensorflow.org/tutorials/interpretability/integrated_gradients))

[2] F. Chollet, “Grad-cam class activation visualization.” (https://keras.io/examples/vision/grad_cam/)

[3] “Heart failure - symptoms and causes.” (https://www.mayoclinic.org/diseases-conditions/heart-failure/symptoms-causes/syc-20373142)

[4] National Heart Lung, and B. Institute, “Diagnosis of pneumonia.” (https://www.nhlbi.nih.gov/health/pneumonia/diagnosis)

Appendix

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_2 (Rescaling)     (None, 224, 224, 1)       0

 conv2d_8 (Conv2D)           (None, 224, 224, 8)       80

 re_lu_10 (ReLU)             (None, 224, 224, 8)       0

 max_pooling2d_10 (MaxPoolin  (None, 112, 112, 8)      0
 g2D)

 conv2d_9 (Conv2D)           (None, 112, 112, 16)      1168

 re_lu_11 (ReLU)             (None, 112, 112, 16)      0

 max_pooling2d_11 (MaxPoolin  (None, 56, 56, 16)       0
 g2D)

 conv2d_10 (Conv2D)          (None, 56, 56, 32)        4640

 re_lu_12 (ReLU)             (None, 56, 56, 32)        0

 max_pooling2d_12 (MaxPoolin  (None, 28, 28, 32)       0
 g2D)

 conv2d_11 (Conv2D)          (None, 28, 28, 64)        18496

 re_lu_13 (ReLU)             (None, 28, 28, 64)        0

 max_pooling2d_13 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 last_conv (Conv2D)          (None, 14, 14, 128)       73856

 re_lu_14 (ReLU)             (None, 14, 14, 128)       0

 max_pooling2d_14 (MaxPoolin  (None, 7, 7, 128)        0
 g2D)

 flatten_2 (Flatten)         (None, 6272)              0

 dense_2 (Dense)             (None, 2)                 12546

=================================================================
Total params: 110,786
Trainable params: 110,786
Non-trainable params: 0
_________________________________________________________________
```

Figure 15: Summary of the architecture of the CNN