RESEARCH

Exploaratory data analysis on the Cleaveland heart disease dataset and training 4 classifiers to detect coronary artery disease

Yunus Emre Kurnaz, Aakash Nepal, Cuong Gia Pham and Evelina Gudauskayte — Group 10

Full list of author information is available at the end of the article

Abstract

The goal of this first project was to get a general overview of the large field of data science. Important terminologies, plots as well as methods from statistics should be explained and brought closer.
However, the main focus was on the selection and application of a suitable classifiers based on the research question.
Linear regression, Decision trees, k-Nearest-Neighbor, Gaussian Naive Bayes and Random Forest are a set of methods, that can be used to identify and assign unknown variables (observations) to a given dataset.
In this project we have learned, that it is not possible to use one classifier for all problems, but that there are multiple approaches, depending on the given data set and the research question. In addition, we learned the ability to deal critically with a confusion matrix and its concepts such as the importance of specificity, accuracy and sensitivity, as well as the interpretation of ROC curves and their implementation.
A total of 8-10 hours were needed to determine the final results.

1 Scientific Backround

When using a classifier, data automatically gets categorized into one of many classes. These classifications are created by algorithms. This procedure is called machine learning. Predictive classification models approximate a function (f) that relates input variables (X) to discrete output variables (Y). These methods are also used in many biochemical and diagnostic medical applications, as they can be used to make fairly accurate predictions. However, this requires quite a lot of clinical data to achieve a suitable prediction. Since this cannot always be guaranteed, the current incentive is to categorize new patient data based on old existing data. The Cleveland database offers one possibility to get familiar with machine learning. In this study, the collected data were used to determine coronary artery disease (CAD) using a logistic regression. CAD is characterized by narrowing of the arteries that supply oxygen to the heart muscle. CAD can be acute or chronic. In the acute form, a heart attack occurs because a blood clot blocks one or more coronary arteries. As a result, part of the heart muscle does not receive oxygen. In chronic CAD, a coronary vessel is permanently narrowed. As a result, less blood flows to the heart muscle than normal. During physical exertion, the heart is then unable to beat more forcefully because it receives less oxygen. This can lead to symptoms such as shortness of breath and a feeling of tightness in the chest.

2 Goal of the project

The goal of this first project is to become familiar with statistical terminology, to perform an exploratory data analysis for a the Cleveland dataset and to train at least three classifiers to diagnose heart diseases. This includes standard procedures such as creating and interpreting boxplots, pairplots, etc. to get a general overview of the data. Here, we take a practical approach to get a basic overview of machine learning, rather than a theoretical (mathematical) approach.
It is important to familiarize with the topic of classifiers. It is needed for the prediction of a qualitative response (class) for an observation. However, not every classifier is equally applicable to every question; they may have different limitations based on the research question.
In this first project, we practiced and programmed different approaches to classifiers and their interpretation on a "real life problem" in order to identify possible differences within these classes, while also performing a simple machine learning algorithm to train the dataset, which can determine whether or not a heart disease can be diagnosed.

3 Data and Preprocessing

The multivariate dataset used here, named "Cleveland heart disease," contains clinical and biomedical features of more than 300 patients. It was published on the Donald Bren School of Information and Computer Sciences website as a primary source of machine learning datasets. Of the original 76 attributes, only 14 attributes were used for further analysis in most cases. However, for this project, only the "goal" field was important, because it refers to the presence of a heart disease in a patient. An integer value of 0 indicated no presence, whereas integers of 1 to 4 represented the respective severity of an existing heart disease.
The missing values were removed either in python or in R. Another option was to impute the missing values for example with the scikit function: impute.SimpleImputer() or impute.IterativeImputer(). Because the column of a presence or an absence of a heart disease is important, care was taken in the creation of a training and testing data set to ensure that the division of the data sets took this into account. However, the "target" column shows an imbalance between patients who do not have heart disease and those who have been categorized by severity. In further steps, this column is restructured so that instead of the severity of heart disease, a distinction is made only between existing heart disease (1) and no heart disease (0). Thus, this problem has been changed into a binary problem to compensate for the inequality of the test and training data sizes.

4 Methods

As described in the previous chapters, the analysis steps were performed using two python libraries. The first one is seaborn, a data visualization library based on matplotlib and the second is scikit-learn, an efficient tools for predictive data analysis. The missing values in the dataset were manually removed as decribed in the previous chapter and prepared for further analysis.
The seaborn library offers a number of functions to get a general overview of the data, in order to do a explorative data analysis. From pairplots and correlation matrices to heatmaps; all necessary statistical functions are included. For this purpose,

simply concatenate the pandas.dataframe with the respective functions.
However, the actual analysis was performed twice. First, the dataset and the respective features are split into a training and a test dataset, which can be executed with the provided function of the scikit-learn package ("train_test_split())". Finding a suitable test size is not always guaranteed, so it is often necessary to split the data again with other test sizes.
Afterwards, the generated test and training sets are passed to one of the many classifiers. In this project, Linear Regression, K-nearest Neighbor, Gaussian Naive Bayes, Decision trees and Random Forest algorithms were used as classifiers. The respective confusion matrices can be plottet with the function: "confusion_matrix()". It is possible to determine the sensitivity, specificity and accuracy of the selected model and compare it with previously calculated matrices.
Second, the same data set was executed again, but this time with the difference, that instead of the severity of a heart disease, only a binary distinction is made whether a heart disease is present (1) or not (0). This procedure was performed because the size of the "goal" column is unbalanced. To compensate for this imbalance, we changed the "goal" column into a binary classification problem, to overcome the imbalance. On the basis of this confusion matrix, again, sensitivity, specificity and accuracy could also be printed.
Finally, the results of the second run were generated using the function "roc_curve()", a function to generate a resulting ROC-curve for a given model. The individual results of the classifiers were plotted as bar plots in a separate file.

4.1 Linear Regression
Linear Regression fits a linear model with coefficients x = (x1, . . . , xp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

4.2 K-Nearest Neighbor
The goal of the K-nearest neighbor algorithm is to determine the nearest neighbors of a given query point, so that we can assign a class label to a point.

4.3 Gaussian Naive Bayes
Naive Bayes classifiers are based on the Bayes theorem. One of the assumptions made is the belief of strong independence between features. These classifiers assume that the value of a particular feature is independent of the value of another feature according to a gaussian distribution.

4.4 Decision Trees
It is a mathematical model that can be used to determine decisions. The hierarchical diagram has a tree-like structure and represents a directed decision path. It consists of root, nodes, branches and leaves. The number of decision levels can vary from decision tree to decision tree.

4.5 Random Forest
It combines the results of many different decision trees to make the best possible decision.

5 Results

To make this section more reader-friendly, only the final results are presented below, as well as the ROC curves of the best classifier.
For this, we divide the analysis results into a multi-class and binary part.

5.1 Multiclass metrics

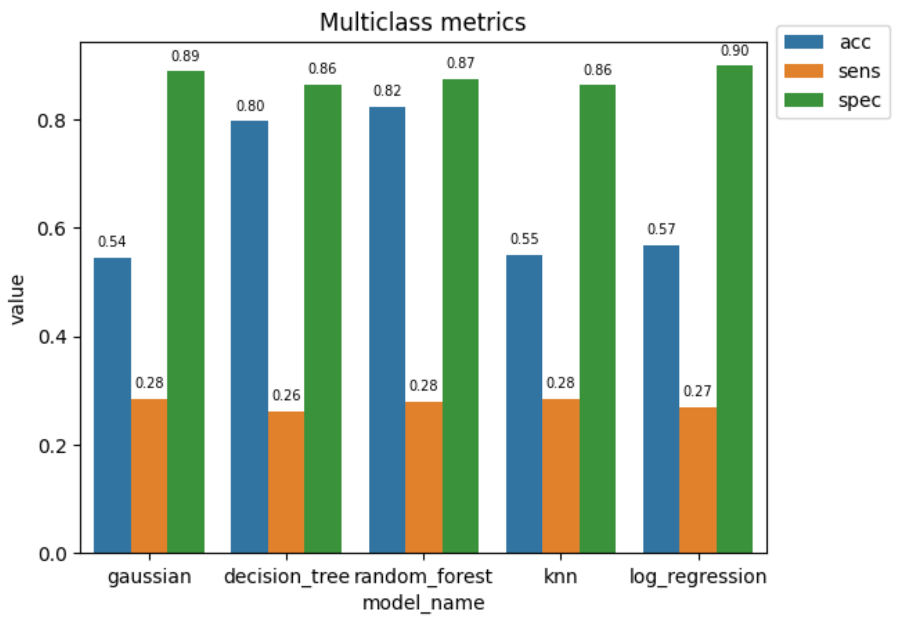

Figure 1 A grouped bar chart of used classifiers (multiclass) regards their accuracy (blue), sensitivity (orange) and specificity (green).

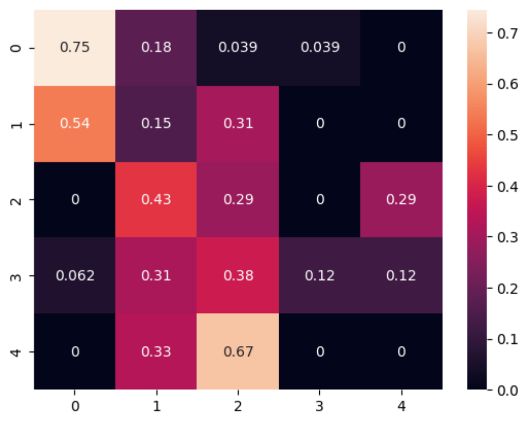

Figure 2 Confusion matrix of the Random Forest classifier (multi-class).

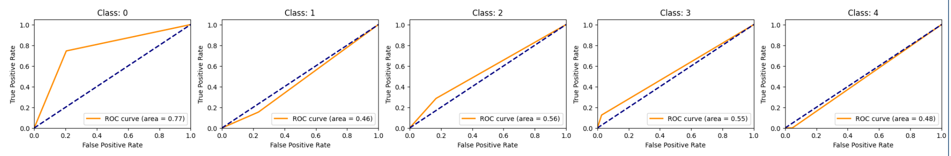

Figure 3 ROC curve of the Random Forest classifier.

Figure 1 shows the different classifiers (x-axis) and their respective values in percent on the y-axis. It is clear that the respective classifiers for the multiclass metric, that distinguish between the severity of heart disease, all produce very similar results. In all cases we have an approximate sensitivity (orange) of 27%. Looking at the results of the accuracy (blue) of the individual classifiers, we get a rather ambiguous result. Decision Tree and Random Forest achieve the highest values of around 80-82% accuracy, this value stagnates at about 55% for the remaining models. A more unambiguous and unanimous result is provided by the specificity (green). All models show an approximately equal result of about 88%.
The ROC curves (Fig. 3) are the results obtained from the confusion matrix (Fig. 2). The confusion matrix (Fig. 2) is a 4x4 matrix that represents the presence of heart disease by its severity or no heart disease. Each row of the matrix represents the instances of an actual class, while each column represents the instances of a predicted class, or vice versa. Looking at the individual ROC curves, which are intended to represent the informative value of this model, it becomes clear that here too, depending on the "class", a different result is achieved. In it, the false positive rate and the true positive rate are compared and a curve is drawn using the confusion matrix. In 4 out of 5 cases (Class 1-4), this curve is a diagonal straight line with area under the curve (AUC) scores of 0.46 to 0.56. Only the class "0" achieved a better score of 0.77, which means, that its curve is located above the original straight line (blue dashed line).

5.2 Binary metrics

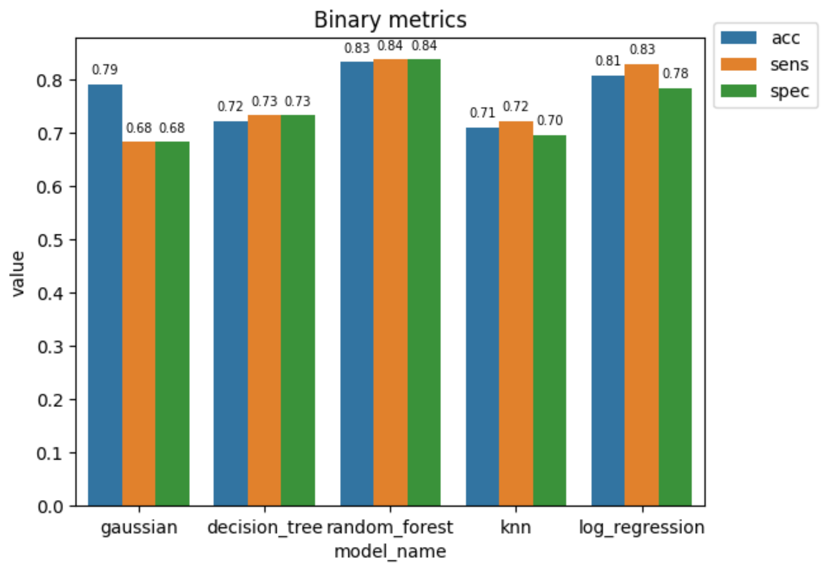

Figure 4 A grouped bar chart used classifiers (binary) regards their accuracy (blue), sensitivity (orange) and specificity (green).

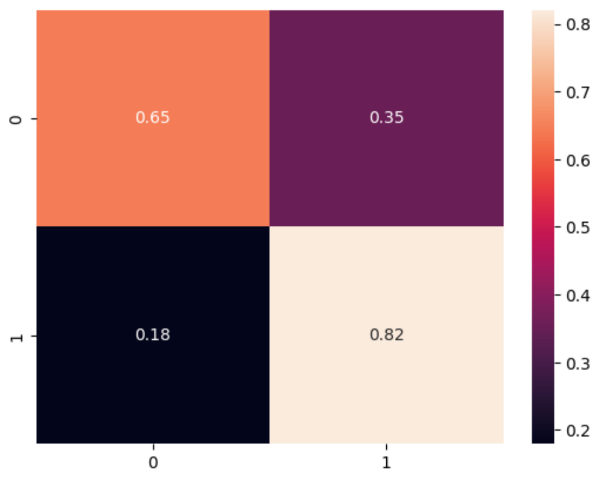

Figure 5 Confusion matrix of the Random Forest classifier (binary).

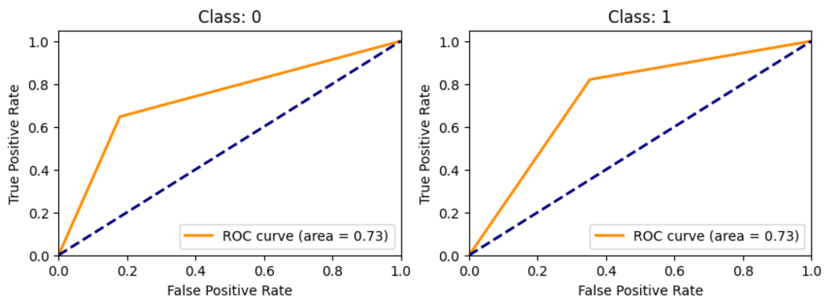

Figure 6 ROC curve of the Random Forest classifier.

By converting the metric into a binary metric ("0", "1"), so that only patients with a heart disease and healthy patients can be distinguished, a significantly different result is obtained than in the previous results (Fig. 4). In the grouped bar chart plot (Fig. 4), it is clear that all classifier models now produce a more consistent result. All have approximately the same accuracy (blue), sensitivity (orange) and specificity (green). In all cases, values above 70% are assumed. The best classifier in this case is the Random Forest, where all values are above 83%. It is also clear from the confusion matrix of the Random Forest (Fig. 5), that this is now only a 2x2 matrix, since the values have been adapted to a binary metric. The results from the confusion matrix, shown as ROC curves (Fig. 6), also show a different result. Here, the false positive rate is plotted against the true positive rate and it is made clear that the ROC curve is clearly above the original straight line (blue dashed line) and that the area under the curve assumes values of 0.73.

6 Discussion

The Random Forest performs best when the outcome is binary. This is because the correlation between the different non-zero levels of the outcome variable and the selected features is weaker than the correlation between the zero outcome and the selected features. If we start considering zero and non-zero outcomes, we can separate the data better than if we have to additionally distinguish each non-zero outcome. This effect is best illustrated in figure 1 and 4. This means, that the Random Forest is best adapted to the data.
It is a simple and but quite important topic in statistics or machine learning in general. It illustrates that one classifier cannot always be applied to many problems and that a a one-vs-rest strategy, which converts a multi-class problem into a set of binary tasks for each class in the objective can achieve a better outcome based on the research question.

7 Appendix

Yunus Emre Kurnaz (Bioinformatics) programmed the linear regression classifier and wrote the report.
Aakash Nepal (Bioinformatics) programmed the K-nearest-Neighbor classifier.
Cuong Gia Pham (Bioinformatics) programmed the Gaussian Naive Bayes classifier.
Evelina Gudauskayte (Data Science) programmed the Decision Tree and Random Forest classifiers.

RESEARCH

Deep Learning algorithms for image classifications of breast cancer histopathology images

Yunus Emre Kurnaz, Aakash Nepal, Cuong Gia Pham and Evelina Gudauskayte — Group 10

Full list of author information is available at the end of the article

Abstract

Deep learning is a subset of machine learning algorithms that use multiple layers of neural networks to process data and compute large amounts of data. Deep learning algorithms are based on the functions and behavior of the human brain. Just like the human brain uses millions of neurons to compute information, deep learning algorithms rely on neural networks to operate, with the help of so-called layers. A distinction is made between supervised and unsupervised learning. However, this project is about classifiers with supervised learning.

Following classifiers were used in this project: Fully connected neural networks (FNN), residual neural networks (RNN), and convolutional neural networks (CNN). In this project, the data ("BreakHis") is classified based on biomedical (microscopic) images, i.e. breast cancer histopathology images.

Therefore three different Deep Learning (DL) topologies are executed and an evaluation is performed using the appropriate techniques.

The main result of this project is that all three deep learning topologies behave differently and thus achieve different results/evaluations. this project has shown that Deep Learning is a very complex area of machine learning.

As in the previous project, it is not possible to use one classifier for all research questions. This means that one has to look critically at the data, because with more layers, the complexity of the model also increases, and more data is needed for the model to perform correctly. However, if the amount of data is less than the complexity of the neural network, a different approach must be taken into account.

With the help of this project, we were able to program three different approaches of an image classification algorithm in google collab with tensorflow and keras, and both refreshed or relearned important terminologies and procedures for dealing critically with the field of deep learning. A total of approx. 1-2 days were required to conclude all of the results.

1 Scientific Backround

Deep learning is a sub-field of machine learning in which computer models perform classification tasks directly from images, texts or acoustic data. A special feature is that deep learning models are able to learn independently. This is done by the systems repeatedly linking what they have learned with new content and thus relearning. This is done with the help of so-called neural networks. They are compared to "natural neural networks", which are a network of neurons in the nervous system of a human, the so-called layers.

In other words; the input feature/data is passed into the "input layer". The resulting output, serves as input for the next layer, the so-called "hidden layer". This is done until the final output is given to the "output layer". Depending on the complexity of the research question, a number of input layers, hidden layers or output

layers can be added.
That's why this learning process is called deep learning, because there are so many computations going on between the input and output layers.
The importance of deep learning algorithms is characterized by the fact that they can achieve higher accuracy and are therefore used in many areas of daily life. Well-known examples are autonomous driving, weather forecasts or predictions of diseases in medical researches. This however, requires a lot of classified data and high computational resources (GPU).
Since large data sets cannot be used for this project, due to lack of computing resources to process these data, the problem was limited to the "BreaKHis" database. This database was created in collaboration with the P&D Laboratory - Pathological Anatomy and Cytopathology, Parana, Brazil, and contains approximately 7909 clinically representative microscopic images of breast tumors provided by 82 patients with different magnification factors (40×, 100×, 200×, and 400×). The database contains 2480 benign and 5429 malignant samples.
This data will be used to classify data based on biomedical (microscopic) images with the help of three different deep learning toplogies. The goal of this project is to become familiar with the specialized subfield of machine learning and to try three different deep learning (DL) topologies and perform an evaluation using the appropriate techniques, where one of the three DL topologies must be a CNN.

2 Goal of the project

As mentioned in the previous chapter, the goal of this project is to become familiar with the complex topic of Deep Learning and to apply the knowledge learned from the lecture to a real-life example. The focus is not only on programming an algorithm to classify breast cancer data based on biomedical (microscopy) images using TensorFlow and Keras, but also on learning important basics and concepts that are mandatory for this topic in order to create a good predictor. Depending on the size and complexity of the data and the research question, different approaches may be needed to provide suitable results. It is also important to understand what a neural network is and how it works, since it is the basis for a deep learning algorithm.

3 Data and Preprocessing

The BreaKHis database contains microscopic biopsy images of benign and malignant breast cancer tumors. These images were collected as part of a clinical study from January 2014 to December 2014. The samples were stained by with hematoxylin and eosin (HE), collected mainly by surgical (open) biopsies, prepared for histologic examination, and processed by P&D laboratory pathologists.
Preservation of the original tissue structure was important for observation of samples under a light microscope. Pathologists identify tumor areas on each slide by visually analyzing tissue sections under a microscope. A definitive diagnosis is made in each case by an experienced pathologist and confirmed by additional investigations such as immunohistochemical analysis.
Digital images of breast tissue sections were acquired using a microscope in combination with a high-resolution camera. To date, the database contains 7909 images classified into benign and malignant tumors. The dataset currently contains four

types of benign breast tumors: adenosis (A), fibroadenoma (F), phyllodes tumor (PT), and tubular adenoma (TA), and four malignancies: ductal carcinoma (DC), lobular carcinoma (LC), mucinous carcinoma (MC), and papillary carcinoma (PC). The resolutions of the biposy images are available in the following resolutions: 40×, 100×, 200× and 400×. However, three different models were programmed, where each model scaled the images as follows: 64x64 (CNN), 224x224 (RNN) and 64x74 (FNN). Thus, the only preprocessing step was suitably scaling the images for further analysis.

4 Methods

The following paragraph explains the three algorithms that were modeled. The focus is therefore on a briefly definition of the used classifier and additionally a description of how the classifier was handled during programming.

4.1 Fully Connected Neural Network

4.1.1 Definition

A Fully Connected Neural Network is a neural network that consists of a series of fully connected layers. Each fully connected layer consists of nodes that can also me called a Perceptron. Perceptron consists of weights and an activation layer. The input of each node is multiplied by weights and the result of multiplication is given to the activation function. Two other techniques that were used to train fully connected neural network are batch normalization and dropout. Batch-normalization layers normalize the contributions to a layer for every mini-batch. This has the impact of settling the learning process and drastically decreasing the number of training epochs required to train deep neural networks. Dropout layers randomly set input units to 0 with a frequency of given rate at each step during training time, which helps prevent overfitting.

4.1.2 Usage

To train fully connected neural network, classes ”benign” and ”malignant” were used, only 400x images from all sub classes were taken. Architecture of network can be described as three blocks with linear layer, batch normalization and dropout and sigmoid function after these three blocks. Exact parameters can be found in the source code. Pytorch and torchvision libraries were used to implement such solution. All images were resized to the size 46x79 and then have been flattened. The training was performed for 10 epochs using Adam optimizer and BCE with logits loss function. Data was split such that 80 percents of data is in used for training and 20 percents used for validation purposes. During training values of f1 metric, sensitivity and specificity was monitored.

4.2 Residual Neural Network

4.2.1 Definition

A Residual Neural Network (ResNet) is an artificial neural network (ANN) based on a collection of connected units, the artificial neurons. A pre-trained version of the network, trained with more than one million images, can be loaded from the ImageNet database. This pre-trained network can be used for tasks such as object detection, image classification and object localisation. In a normal deep model, adding more layers leads to a correspondingly higher training error. To solve this problem, typical ResNet models are implemented in such a way that some layers are skipped by taking shortcuts, which simplifies the network and reduces the error. A residual neural network was used to win the ImageNet2015 competition and became the most cited neural network of the 21st century. ResNet-50 is a convolutional neural network with a depth of 50 layers, i.e. 48 convolutional layers with a MaxPool and an AveragePool layer.

4.2.2 Usage

First, the two classes "benign" and "malignant" were preprocessed based on all images (example: only 400x images from all sub classes were taken). The given images have higher image dimensions, so they were reformatted to a dimension of 240x240 pixels, which means that they are each 3x224x224 input data for the network. Using tensorflow.keras.utils.image_dataset_from_directory(), the image data was split into 80% for training and 20% for validation. A batch size of 32 was taken. The Matplotlib library was used to visualise some images in the training data, and image.shape() was used to check the shape of the image. After that, the ResNet50 model was imported and loaded. The precalculated weights from the Imagenet data were used so that the weights did not have to be recalculated to save computing time and memory. After importing the pretrained model, a fully linked and an outer layer with 2 output neurons were imported in which a softmax activation function was used as there were only two classes in the data. The model was then compiled with Adam optimizer(with default learning rate of 0.001) algorithm and ran for 10 epochs. The model then was evaluated on the basis of validation and training accuracy for each epoch in a plot using the Matplotlib library. Prediction was also performed by taking a random image from the validation data. To be able to determine, if the results are improving when the data is augmented, before running the model, the augmentation step was performed additionally and the model was reran with the same parameters as before. For this augmentation step the training set was applied transformations such as image flipping, rotation and zoom.

4.3 Convolutional Neural Network

4.3.1 Definition

A Convolutional Neural Network (CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing steps required in a CNN model are less, compared to other classification algorithms. Primitive methods require hand-engineered filters. However, with enough training, a CNN is able to learn these filters/characteristics. The architecture of a CNN is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the visual cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. A collection of such fields overlaps to cover the entire visual area.

4.3.2 Usage

First, a CSV file that contains paths to images and each image's label is generated by the script generate_dataset.py, to gather the chosen images. After that, the CSV file is used to split to the train and validation set by using the built-in function of TensorFlow ImageDataGenerator. Here, the train set accounts for 75%, and the validation set accounts for 25%. Besides that, the ImageDataGenerator function normalizes the generated matrix by dividing each pixel by 255. With the built-in function flow_from_dataframe of TensorFlow, the batch size is set to 32, and each image is cropped to 64x64 to improve the training process. The train set is then ready to fit in the model. Our CNN model is built based on the architecture that every two convolution layers come with one MaxPooling or one AveragePooling layer. The model is trained with ten epochs. At each epoch, the model shows the loss value, the AUC score and the accuracy of train set and validation set.

5 Results

5.1 Fully Connected Neural Network

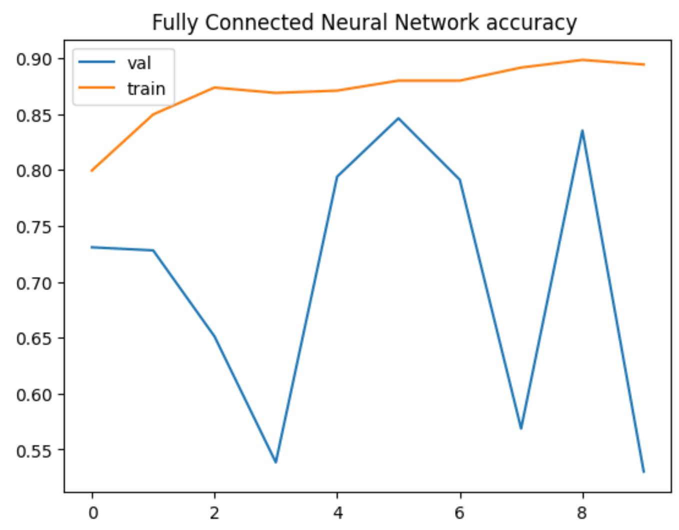

Figure 1 Plot showing train and validation accuracy on the y-axis in relation to each epoch on the x-axis.

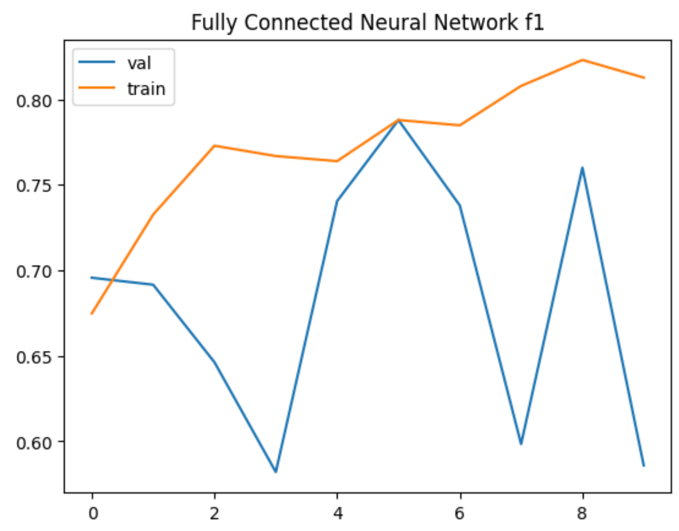

Figure 2 Plot showing train and validation f1 metric on the y-axis in relation to each epoch on the x-axis.

As can be seen in figure 2, validation metrics are the best at epoch 5, where f1 hits value 0.79. Overall metrics in figure 1 and in figure 2 are not very high, as when flattening applied, an image is represented as a vector and a lot of dimensional information is lost. Moreover, it is clearly to see, how the model overfits. Therefore the fully connected neural network is not the best choice for given data as it performs worse than other types of neural networks.

5.2 Residual Neural Network

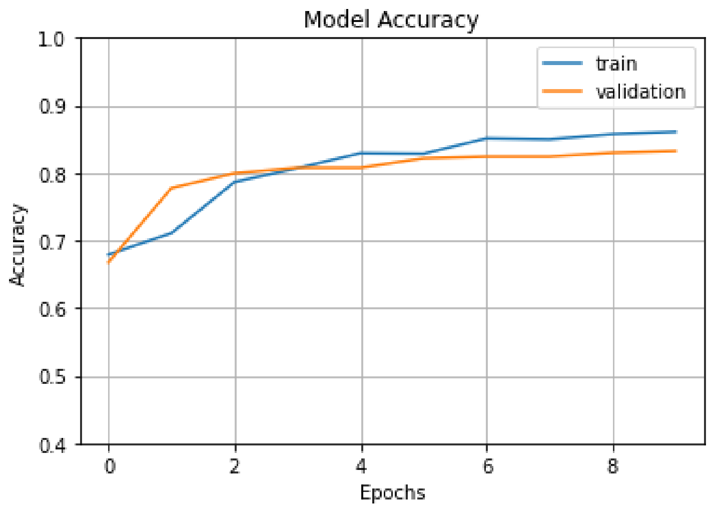

Figure 3 Plot showing train and validation accuracy on the y-axis in relation to each epoch on the x-axis for the initial data without augmentation.

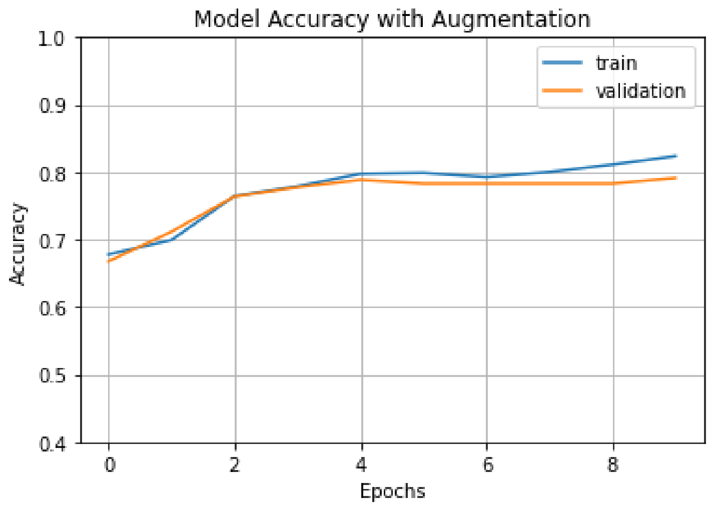

Figure 4 Plot showing train and validation accuracy on the -axis relative to each epoch on the x-axis for augmented data.

In figure 3, the observed accuracy per epoch for the training set goes slightly higher than the accuracy of the model for the validation set, which means that the model seems to be somewhat overfitted. The highest validation accuracy reached is approximately 83%. But after augmentation, which can be seen in figure 4, both training and validation accuracy increase at about the very similar rate close to each other and reaching the highest validation accuracy at epochs 10 to be approximately 80%.

5.3 Convolution Neural Network

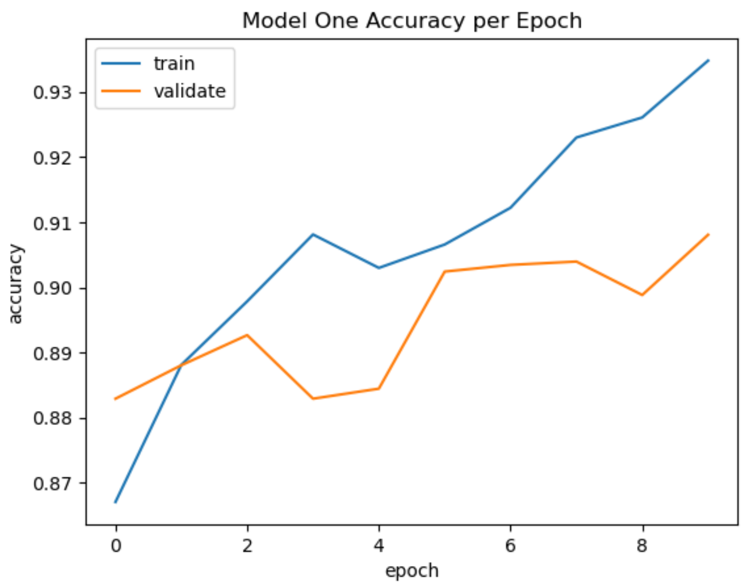

Figure 5 Accuracy of the CNN model per each Epoch

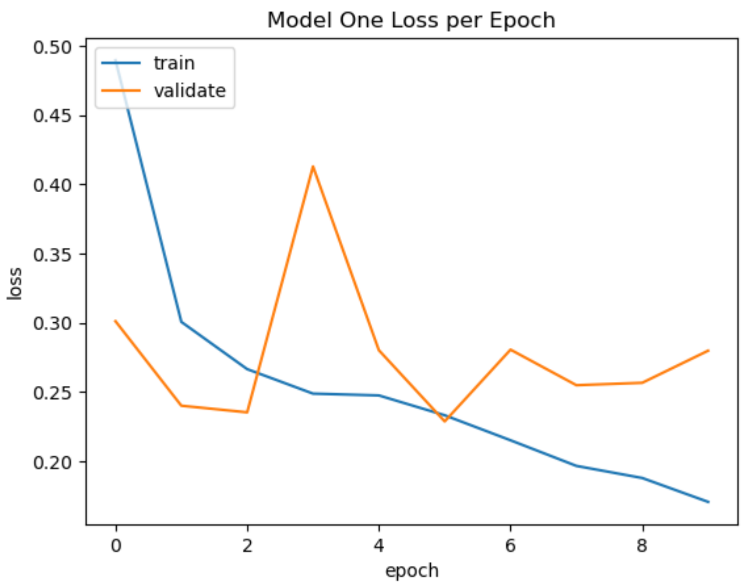

Figure 6 ROC of the CNN model per each Epoch

In figure 5, the accuracy of train and validation sets are fluctuating. It shows that some portion of the data set is classified randomly. Besides, in figure 6, the loss line of validation set tends to go up, which means that the model seems to be overfitted.

6 Discussion

As can be seen from the previous chapter, all of the models used, tend to overfit, which is also evident in the figures 1-6.
Overfitting is not always easy to avoid, since in most cases it is necessary to take into account what kind of data are available and what potential impact, methods to avoid overfitting, may have on the research question.
Probably the simplest approach is to train a classifier with more data. Since this cannot always be guaranteed and there is also the risk of adding additional "noisy data" into the classifier, this approach cannot always guarantee that overfitting can be avoided. There exist numerous methods: "early stopping", "regularization", "ensembling", "feature removal" or "augmentation", which can be used. However, none of them is always the solution to solve every problem. Each of them brings advantages and disadvantages, that must be weighted in order to avoid overfitting while obtaining a suitable result in the same time. This project, is an appropriate example for a data scientist, representing a possible real-world scenario where it is necessary not only to deal more critically with the data, but also to be aware of the consequences of overfitted and underfitted data before adopting a final result.

7 Appendix

Approximately the same amount of work was assigned among all group members. Yunus Emre Kurnaz wrote most of the report and also worked with Cuong Gia Pham on a convolutional neural network (CNN).
Aakash Nepal programmed a deep learning algorithm based on a residual neural network (RNN).
Evelina Gudauskayte programmed a fully connected neural network (FNN) for the above problem.