Chapter 1. Baseline model

Section 1: Data Analysis

First of all, we imported the data from kaggle.com as a Pandas dataframe. When we checked what the data looked like, using info(), we saw that there were some categorical and some numerical values. The outcome variable was 'stroke' and was already one-hot-encoded. We removed the value 'ID', because this does not influence the outcome variable and should thus not be trained on. Subsequently, we replaced the variable columns that contained categorical values to one-hot-encoded columns. There were a few missing values in the 'BMI' column. We chose to remove the cases where BMI was missing as this does not constitute a significant data loss. Finally, we split the data into features and labels and into training, testing and validation data; we used a 60% - 20% - 20% split. The validation data will be used for parameter optimization, while the testing data will be used for final model evaluation.

One important thing we noted was that the stroke and non-stroke groups were highly unbalanced; there were much more subjects who would not get a stroke compared to the subjects with a stroke. For our baseline models we have not yet implemented any methods to counteract this imbalance. However, we noticed that we should emphasize this imbalance in our model evaluation (see model evaluation).

In order to have a better grasp of our quantitative data, we visualized the age, average glucose level and BMI with respect to the chances of a stroke occurring (Figure 1). In total, we made six plots. Three histograms with age, average glucose level and BMI on the x-axis, and the count of strokes and non-strokes on the y-axis. These plots clarified the kinds of distributions. The other three plots are cumulative histograms, plotted against density. These plots better display the distribution of the volume, since the heavy imbalance in strokes and non-strokes makes it difficult to see the distribution of the strokes.
From the age plots, we can conclude that the higher the age, the bigger the chances are of having a stroke. The stroke distribution strongly rises around the age of 70, whilst being almost non-existent in around the 10 to 20 age range.
The average glucose level plots tell us that most strokes occur at an average glucose level of 70 mg/DL. This is also the most occurring level for people where a stroke did not occur. At 200 mg/DL, there is also a significant change of having a stroke. The amount of people having a level of 200 mg/DL is significantly lower than that of 70 mg/DL.
The plots on BMI tell us that the BMI of a person has very little to do with the probability of having a stroke. The density distribution of stroke vs non-stroke is more or less the same.

Section 2: Data Pipeline

To begin with, we made 3 separate base models to see how well they would be able to classify the patients that were at risk for a stroke. We created a k-Nearest Neighbor classifier model, a decision tree and a neural network. Our idea is that these models can first be optimized separately. If several models are predicting strokes significantly above chance level on validation data, these models can be stacked.
The k-Nearest Neighbor algorithm classifies unknown data points based on similarity to other, known data points. We did not necessarily expect this model to perform very well, but it is a very simple model which might do a decent job at separating the stroke patients from the healthy patients.

The decision tree is highly interpretable and is well fit to deal with both categorical and numerical data. Besides, its implementation is relatively trivial, which makes it a good choice as a classification method. The model can be further extended to a random forest in later models and be tweaked to deal with overfitting which easily occurs in decision trees.

Finally, a deep neural network is used for classification which can learn more complex non-linearity from the input features. The number of parameters to add is virtually limitless, but especially with unbalanced data the challenge will be to overcome overfitting of the model to the training data.

K-Nearest Neighbors

For K-Nearest Neighbors, we used the sklearn KNeighborsClassifier as our base model.
The main steps for this model are choosing the right hyperparameters, fitting the classifier to the training data and letting the model predict the classes of the testing data. As we have learned, k-NN classifiers typically use numerical data, for which a distance metric can be calculated such as the 'Euclidean distance' or 'Manhattan distance'. However, there are also several metrics to use categorical data in a kNN classifier, for example the Jaccard distance metric. We chose to split our dataset in two columns based on the type of feature (numerical data and categorical data). Each model is used in a separate model with a different distance metric. If any model predicts stroke, the final classification is stroke. In this way, the numerical data and categorical data are stacked to get more sensitivity to the stroke predictions.

Decision Tree

A basic decision model is built using the sklearn toolkit and fit using the training data.

Neural Network

For our basic neural network, we used TensorFlow to build a sequential model. The model has an input layer with one node for each input feature in the model. The input layer is connected to a hidden layer with 25 nodes and a ReLU activation function. The hidden layer is connected to a sigmoid output layer with 1 node. The output of the final sigmoid function is a float between 0 and 1 which can be split at 0.5 for a prediction of 0 for no stroke and 1 for stroke.

Section 3: Model Training

K-Nearest Neighbors

The nearest neighbors model is relatively simple; the only parameters that can be set is the amount of neighbors that should be considered for classification, the metric used for distance and the weights function to determine how a point in a 'neighborhood' should be classified. For the starting model, we used a k of 5 and the weight function "distance", which meant that a point was classified as one class or another by taking the distance to all k neighbors into account, instead of just picking the majority.

Decision Tree

No distinction was made between the categorical and continuous data, so the categorical data was actually treated as continuous in this basic model. This is something that might need to be improved during model optimization. For our first training of the model, no limitations in the depth of the tree or other optimization methods were added so far.

Neural Network

Cost is defined by the sparse categorical cross-entropy. 5 epochs are used for training (since there are little parameters to optimize so far.) For predictions, the index of the max for the output vector of the SoftMax output layer is used.

Section 4: Model Evaluation

Our data labels were highly unbalanced. This can lead to very high accuracies when the label with the highest prevalence is always predicted. However, in such a case there are almost no false negatives, but many false positives. For all three models we have calculated the confusion matrix to inspect this, and found that indeed after training each of these models almost no validation samples were predicted positive for stroke. Since accuracy is not a good metric for unbalanced data, we also added the balanced accuracy score as a metric to evaluate the model.

K-Nearest Neighbors

The accuracy on the test data of the k-NN algorithm with, k = 5 nearest neighbours, was 0.939. However, as stated in the paragraph above, the data was unbalanced: looking at the confusion matrix for only numerical data, there were 920 true negatives and 2 true positives, but 11 false positives and 49 dangerous false negatives! The accuracy might be high, but this is only because the stroke cases are so rare; if every sample was classified as no risk at stroke, you would still get about 95% accuracy, as the group that suffered from a stroke is very small. Therefore, the calculated balanced accuracy was 0.514; this is very low. Combining the categorical and numerical data had some small positive effect on the balanced accuracy, which was 52.03% with 48 false negatives left.

Decision Tree

For the Decision Tree we implemented an analysis to find the optimal depth of the tree to reach the highest balanced accuracy on the validation data. So far, this accuracy did not reach higher than 53% with a depth of 9 splits.

Neural Network

The balanced accuracy of our basic neural network is about 50%, which is exactly at chance level. However, there are many parameters to add such as the addition of more layers to the model and hyperparameters to change such as the cost and activation functions. Furthermore, the accuracy on the testing data does not get much higher than 0.95, which is as it were a local minimum; the model just classifies every datapoint as not having a stroke. So, there is much room for improvement here.