Kaggles stroke dataset

Ciaran Welsh

August 14, 2019

1 Introduction

I have trained a feed forward neural network to classify people into stroke victims or not-stroke victims based on their age, their average glucose level, hypertension, heart disease, gender, bmi, marital status and work type.

1.1 Exploration

I produced a selection of histograms and PCA plots before deciding that the best way to visualise this data is by using scatter matrix plots with kernel density estimators along the diagonal. An example is shown in figure fig. 1 which the rest can be found in ./data/Plots/scatter$_matrix$.

2 Preprocessing

The biggest problem with this dataset is that the number of stroke samples was much smaller than the number of non-stroke samples (98% vs 2%). To deal with this problem, I chose to under sample the non-stroke data, ensuring the number of stroke and non-stroke samples were equal in both the train and validation data. This turned out to be a naive method because it doesn't factor in the impact that age has on stroke incidence. When training with this naive method, the model validation score was very unstable between runs and dependent on the age distribution of stroke victims. Stratifying the under-sampling such that the age distribution within the stroke and non-stroke in both the train and validation data were the same seemed to greatly help with this problem. This idea was based on

Other ways I cleaned this dataset include:

1. Discard young data. There are very few people under 30 in this dataset that have had a stroke. Therefore these were removed from the analysis.

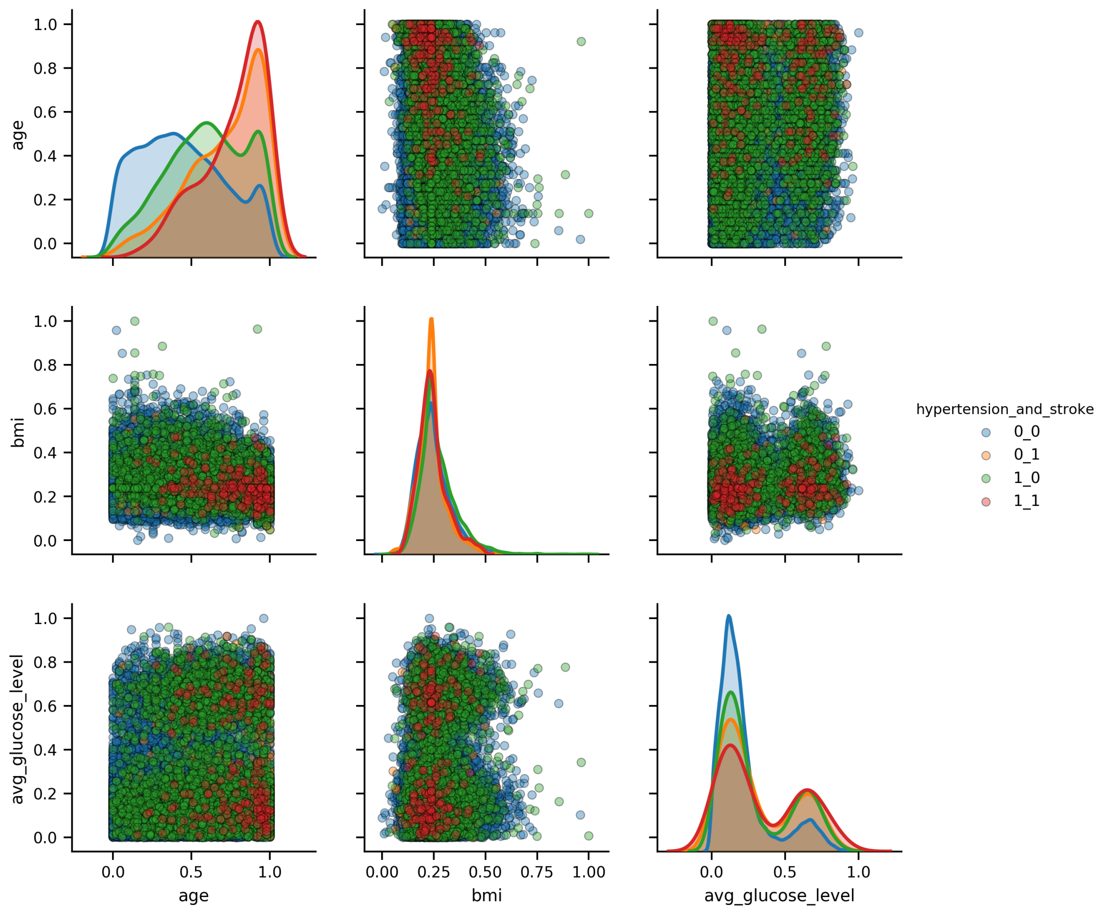

Figure 1: Scatter matrix showing relationships between bmi, average glucose levels. This plot is coloured by hypertension and stroke

2. Drop the other gender. There are very few 'Other' values and none who have had strokes.
3. Impute the bmi column using the median, since only around 3% of data were missing this is a reasonable thing to do instead of deleting the sample.
4. Scale continuous data between 0 and 1 so they exist on a similar scale for fitting
5. One hot encode categorical variables
6. convert boolean variables to 1's and 0's
7. Remove the residence category: exploratory data analysis seems to suggest its not predictor of stroke incidence.

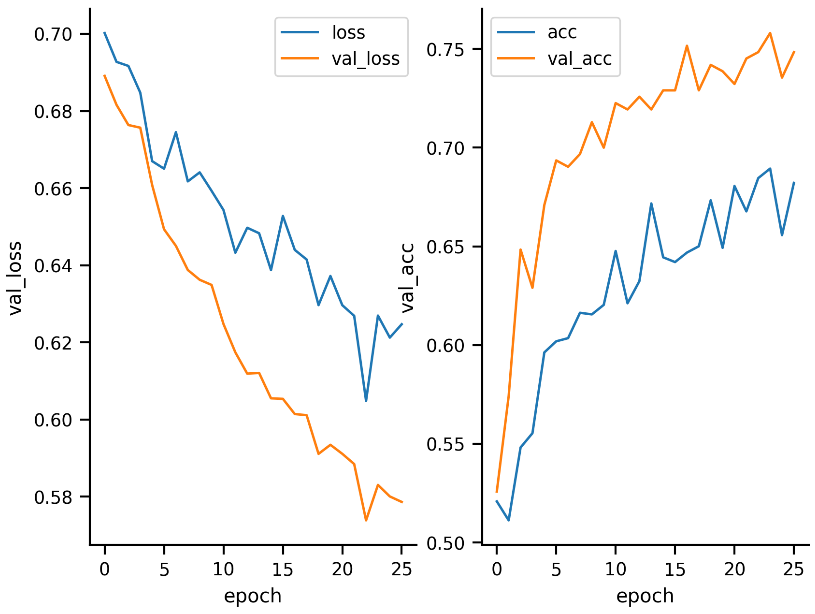

Figure 2: Training history

8. Remove smoking status category. It has too many missing values.

9. remove any remaining samples with nan values.

Note that model performance may still be improved by modifying the preprocessing strategy. Of note there is a package called Imbalanced-learn that would be of interest.

2.1 Model architecture

A simple feedforward network was implemented using the tensorflow interface to keras. Dropout layers were used after each dense layer for regularisation and the relu activation funciton was used. The output layer has a single neural with a sigmoid activation function and the model was trained by minimizing the binary crossentropy objective function using the ADAM optimizer. An early stopping callback was used to prevent overfitting by stopping training when the validation accuracy begins to decline. A plot of training history can be found in fig. 2.

2.2 Model performance

Since an under sampling strategy was used to deal with the imbalanced data problem, the model only trains with a fraction of the data. To assess the models predictive robustness, a bootstrapping strategy was used. Model train and validation scores were monitored while repeating the model fitting process with a different sample from the not-stroke population with the same stroke population. Its possible this decision may have a negative impact on model generality due to over fitting the stroke population. As before, sampling was stratified by age group in addition to ensuring equal proportions of stroke and not-stroke victim.

Usually the separate test dataset would be useful in assessing

2.3 Things I have run out of time for

Time constrains - could not tune hyperparameters well - many factors can influence this model including groups used for stratification. inclusion of not needed features. - all the hyper parameters, model architecture, drop out rates, loss and algorithm choice.

2.4 Room for improvement