Prediction of Brain Hemorrhages from Stroke Patient MRIs

Uday Alla
Chuyue(Camille) Zhang

I. Introduction

When treating patients with ischemic stroke, recanalization therapy comes with certain risks of developing hemorrhagic transformation. The goal of this project was to train a set of machine learning models on stroke patients MRIs in order to predict the development of hemorrhagic transformation, so as to better assist doctors with their decision making in terms of undergoing recanalization treatment.

Specifically, features such as pre-treatment diffusion-weighted image(DWI), perfusion-weighted image(PWI) and arterial input function (AIF) are used as input to the various machine learning models, the presence of HT observed in follow-up gradient recalled echo (GRE) images are used as groundtruth labels during training. The binary result of whether is patient is likely to develop HT - represented by 1, or no - represented by 0, is then produced by feeding training labels to our machine learning models. In our project, we used 618 features and 50,000 MRIs in total for our training and testing.

II. Methods

Our input data comes in the form of a csv file with 50k entries, where each entry representing feature data derived from each MRI. There are 618 features in total, with the first 18 associated with DWI, the next 540 associated with PWI, then the last 60 associated with AIF. The last column represents the ground truth which is either 1 or 0.

We used Tensorflow and Scikit-learn machine learning frameworks to train the data. With Tensorflow framework, we implemented logistic regression and multi-layer perceptron network models. With the Scikit-learn framework, we implemented NerestCentroid, logisticRegression, MLPClassifier, SGDClassifer, KNeighborsRegression, KNeighborsClassifier, RadiusNeighborClassifier, RandomeForestClassifier, DecisionTreeClassifier and SVC.

In order to train and test these various models we had to split the data into training and testing sets. For the sci-kit learn framework we essentially used cross-validation. In a k-fold cross-validation approach, we essentially randomly partitioned the original sample into k almost equal size subsamples. Of the k subsamples, a single subsample was retained as the validation data for testing the model, and the remaining k-1 subsamples were used as training data. The cross-validation process was then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds were then averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. In our case, for each sci-kit learn model we used 3-fold, 5-fold and 7-fold cross validation for testing accuracy. We also ensured that when we split the data into subsamples each patients data was grouped together so as to ensure that each patients data was in either the testing subsample or the training subsample and not both at the same time.

For tensorflow models we split the data into 80% training set and 20% testing set. Again we ensured that no patients data was present in both the testing and training set at the same time. We partitioned the data such that for each patient all data was either in training set or the testing set and not both.

III. Evaluation

To evaluate the performance of our classifiers, we essentially measure the test accuracy for each model. For TensorFlow classifiers, we train our model with 80% of our 50,000*618 input numpy array as the training examples, and 80% of our 50,000*1 ground truth binary numpy array as the training results. We feed the data into both Linear Regression model and MLP Neural Network models to calculate the accuracy of our test results against ground truth binary data. For Scikit-learn classifiers, we ran the same models multiple times with varying parameters, such as different types of activation function, loss function, to obtain the highest accuracy of each model. Each model is also run with different number of folds to represent different ways of splitting the data for cross-validation. The test accuracy results are represented in the table and graph on the next page.

IV. Discussion

From this study we found that the occurrence of hemorrhagic transformations in Acute Ischemic Stroke patients can be predicted with decent accuracy from MRI scans (PWI, DWI and AIF) using various machine learning models. Solely based on testing accuracy, the Sci-kit Learn's logistic regression, MLPClassifier(with identity activation function), SVC(with rbf kernel) and SGDClassifier(with loss=perceptron) predict the formation of hemorrhagic transformation in AIS patients the best, with accuracy for each being around 80%. On the other hand KNeighbourClassifiers, DecisionTreeClassifier, DecisionTreeRegressor, AdaboostClassifier, NearestCentroid, ExtraTreesClassifier and RadiusNeighbourClassifier underperformed with an accuracy ranging from 40%-60%.

The Tenserflow's logistic regression also performed well with a testing accuracy of about 80%, while it's multilayer perceptron neural network underperformed with an accuracy of about 52%. This may be attributed to the fact that since Tensorflow is a low level library we had to manually configure the parameters which again might not have been as optimal as possible.

Some of the prediction error can also be attributed to the noise present in the data, specifically with the PWI images. Various advanced filtering techniques could be used to reduce and even eliminate some of the noise present in the data. Noise, especially for models such as the decision tree classifiers and the k-Nearest Neighbours algorithm, can have a significant effect. This could be one of the reasons as to why KNeighbourClassifiers, DecisionTreeClassifier, DecisionTreeRegressor, AdaboostClassifier, NearestCentroid, ExtraTreesClassifier and RadiusNeighbourClassifier underperformed. Since we have a large number of features and noisy data, KNN is one algorithm which seems to make less sense as distances become more uniform with the growth in dimensionality and this makes clustering difficult.

One way we can possibly improve our accuracy is to use a larger data set. Currently, for this study, we used 50,000 data points from a total of 60 patients. Furthermore, for each patient there were a varying number of points and therefore when using cross-validation it was hard to split the data into k equal subsample. MLPClassifiers and other neural network models which employ deep learning require a large amount of data for the most accurate results.

Lastly, from research we know that different regions of the brain are prone to varying degrees for developing hemorrhagic transformations. Therefore in the future if we are able to incorporate this knowledge into our data and machine learning models we can possibly gain better accuracy at predicting hemorrhagic transformations in AIS patients.

V. Conclusion

For greater accuracy along with simpler implementation we found sci-kit learn machine learning framework to be significantly better than tenserflow's. We also found that regardless of which framework we used, logistic regression, multi layer neural networks and SVM performed the best with about 80% testing accuracy. We also noted that nearest neighbours and decision trees classifiers underperformed due to noisy data and the presence of a large set of features. In the future we can possibly improve accuracy by filtering out the noise in the data and train the machine learning models on a larger data set. Therefore to conclude it is possible to use machine learning to predict the likelihood of AIS patients developing hemorrhages.

VI. References

Y.Yu, F. Scalzo, et al. Prediction of Hemorrhagic Transformation Severity in Acute Stroke from Source Perfusion MRI. 2013:31:961-969

Thornhill RE, Chen S, Rammo W, Mikulis DJ, Kassner A. Contrast-enhanced MR imaging in acute ischemic stroke: T2* measures of blood-brain barrier permeability and their relationship to T1

estimates and hemorrhagic transformation. AJNR
Am J Neuroradiol. 2010;31;1015-1022.

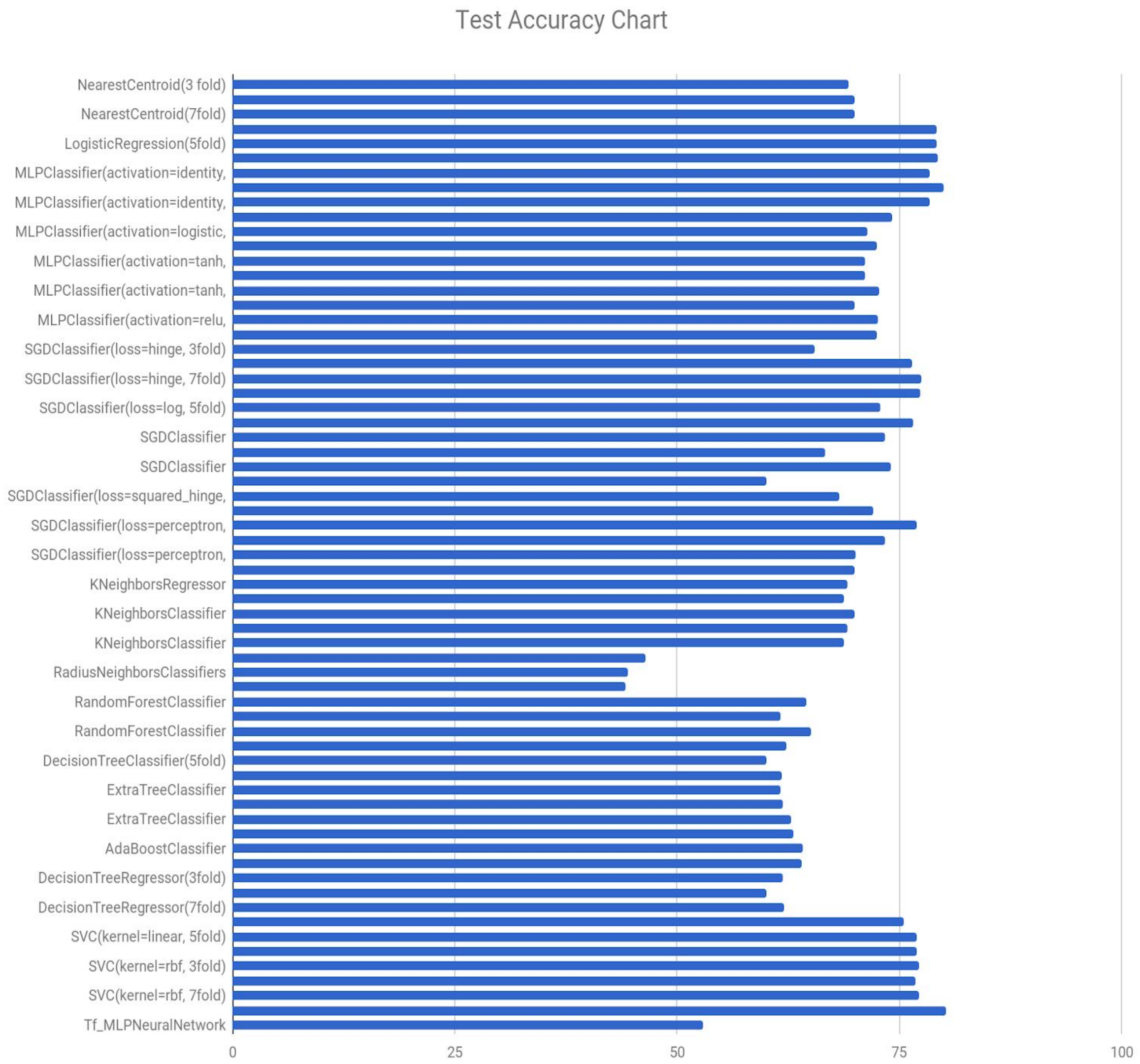
TensorFlow Classfiers

Classifier	Training/Testing Accuracy
Logistic Regressions	96.5% | 80.3%
MLP Neural Network	70.2% | 52.9%

Scikit-Learn Classifers

Classifier	#Folds	Testing Accuracy
NearestCentroid	3|5|7	69.3%|70.0%|70.0%
LogisticRegression	3|5|7	79.2%|79.2%|79.3%
SVC (kernel=linear)	3|5|7	75.5%|76.9%|76.8%
SVC (kernel=rbf)	3|5|7	77.2%|76.8%|77.2%
MLPClassifier (activation=identity)	3|5|7	78.4%|80.0%|78.4%
MLPClassifer (activation=logistic)	3|5|7	74.2%|71.4%|72.4%
MLPClassifer (activation=tanh)	3|5|7	71.2%|71.1%|72,7%
MLPClassifer (activation=relu)	3|5|7	70.0%|72.6%|72.4%

SGDClassifer (loss=hinge)	3|5|7	65.5%|76.4%|77.5%
SGDClassifier (loss=log)	3|5|7	77.3%|72.9%|76.5%
SGDClassifier (loss=modified_huber)	3|5|7	73.4%|66.7%|74.0%
SGDClassifier (loss=squared_hinge)	3|5|7	60.0%|68.2%|72.1%
SGDClassifier (loss=perceptron)	3|5|7	77.0%|73.4%|70.1%
KNeighborsRegressor (n_neighbor=5)	3|5|7	70.0%|69.2%|68.7%
KNeighborsClassifier (n_neighbor=5)	3|5|7	70.0%|69.2%|68.7%
RadiusNeighborClassifier (n_neighbor=5)	3|5|7	46.4%|44.5%|44.2%
RandomForestClassifier (n-estimators=35)	3|5|7	64.6%|61.6%|65.1%
DecisionTreeClassifer	3|5|7	62.3%|60.0%|61.8%
ExtraTreesClassifier (n_estimators=35)	3|5|7	61.6%|61.9%|62.8%
AdaBoostClassifier (n_estimators=35& max_depth=3)	3|5|7	63.1%|64.1%|63.9%
DecisionTreeRegressor	3|5|7	61.9%|60.0%|62.0%


Test Accuracy Chart
NearestCentroid(3 fold)
NearestCentroid(7fold)
LogisticRegression(5fold)
MLPClassifier(activation=identity,
MLPClassifier(activation=identity,
MLPClassifier(activation=logistic,
MLPClassifier(activation=tanh,
MLPClassifier(activation=tanh,
MLPClassifier(activation=relu,
SGDClassifier(loss=hinge, 3fold)
SGDClassifier(loss=hinge, 7fold)
SGDClassifier(loss=log, 5fold)
SGDClassifier
SGDClassifier
SGDClassifier(loss=squared_hinge,
SGDClassifier(loss=perceptron,
SGDClassifier(loss=perceptron,
KNeighborsRegressor
KNeighborsClassifier
KNeighborsClassifier
RadiusNeighborsClassifiers
RandomForestClassifier
RandomForestClassifier
DecisionTreeClassifier(5fold)
ExtraTreeClassifier
ExtraTreeClassifier
AdaBoostClassifier
DecisionTreeRegressor(3fold)
DecisionTreeRegressor(7fold)
SVC(kernel=linear, 5fold)
SVC(kernel=rbf, 3fold)
SVC(kernel=rbf, 7fold)
Tf_MLPNeuralNetwork
0
25
50
75
100