Brain Stroke Prediction using Machine Learning

Siddhant Agarwal
2020247
siddhant20247@iiitd.ac.in

Rahul Sehgal
2020232
rahul20232@iiitd.ac.in

Deeptanshu
20202371
deeptanshu20371@iiitd.ac.in

Abstract

Stroke is the fourth most significant cause of death in India. Thus, predicting the onset of a stroke can help provide the immediate medical support needed, potentially saving countless lives. While there is extensive research regarding strokes, their prediction using machine learning is still improving, and a new set of features and models can potentially boost its performance. Trying to find information regarding this topic, we came across various studies and datasets related to how various factors affect the occurrence of a brain stroke. We extended our interest on the topic and reviewed some research papers to get a better understanding about what work has been done on the issue and how we can direct our research in this domain.

https://github.com/siddhant20247/Brain_Stroke_Detection_ML_Project

1. Introduction - Problem Statement

Stroke is one of our country's leading causes of death. Early detection of this disease is crucial in decreasing its mortality rate. Current prediction systems are based on the judgment of symptoms, and machine learning models are not very prevalent.

Our problem statements cover this project's critical points of motivation and relevance. Current machine learning models for stroke predictions are not accurate enough to be used extensively in healthcare. Stroke being one of the leading causes of death, gives this project enough opportunity to potentially save countless lives.

Healthcare is a developing field with a high scope of technological intervention. Deployment of technologies like Machine Learning can significantly impact the industry as they can help provide accurate predictions about various illnesses.

2. Related Work

[2] use the same dataset from Kaggle as ours. The data is highly imbalanced, so the authors used undersampling to balance the positive and negative samples. Oversampling could also have been used by repeating the positive samples, which was not discussed in this paper. The authors use an 80-20 train-test split, combined with classification algorithms such as logistic regression, decision tree, random forest, k-nearest neighbours, support vector machines, and naive Bayes classifier. The naive Bayes classifier had the best performance with an f1 score of 82.3.

[1] use the extensive CHS dataset containing many features. Hence, it discusses Data Imputation and Feature selection techniques such as 'forward feature selection, 'L1 regularised logistic regression', and 'Conservative mean feature selection' for the dataset. The research paper considers the use of Support Vector Machines and Margin-based Censored Regression as the learning algorithms for predicting the risk of stroke. The use of performance metrics, such as the area under the ROC curve and concordance index, can be seen in the paper. The metrics are reported for all combinations and compared with the results of the Cox model. One of the drawbacks is that the Conservative mean feature selection algorithm may not work well with other datasets, so L1 regularisation is suggested in this case.

3. Dataset

The dataset we selected includes various features contributing to Stroke prediction. This particular dataset consists of 4981 samples and 10 features. The features are 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married', 'work_type', Residence_type', 'avg_glucose_level', 'bmi', and 'smoking_status'. A column 'stroke' includes the output values 0 or 1, where '0' signifies no stroke predicted and '1' represents stroke predicted. There were no null values in the data.

3.1. Pre-processing

This dataset requires pre-processing as it has inadequacies. As the algorithm doesn't take strings as input and works with numbers only, the dataset needs to be label-encoded as it contains features containing 'Yes' and 'No'. These inputs cannot be worked upon further during Model training, so we have used one hot encoding using the

Attribute Name	Type (Values)
1. gender	String literal (Male, Female)
2. age	Integer
3. hypertension	Integer (1, 0)
4. heart_disease	Integer (1, 0)
5. ever_married	String literal (Yes, No)
6. work_type	String literal (children, Govt_job, Never_worked, Private, Selfemployed)
7. Residence_type	String literal (Urban, Rural)
8. avg_glucose_level	Floating point number
9. bmi	Floating point number
10. smoking_status	String literal (formerly smoked, never smoked, smokes, unknown)

Figure 1. Data Features

get_dummies() function. Since the data consisted of features containing String inputs, we were required to use One hot encoding as it replaces the string literals in the dataset into integer values. This was applied to 5 features having type as string literals, which helped make the dataset a combination of integers that could be worked upon by the machine learning algorithms.

3.2. Data Imbalance

This dataset is also highly imbalanced as the possibility of '0' in the output column ('stroke') outweighs that of '1' in the same column. Out of 4981 samples, 4733 samples have '0' as their 'stroke' output, and only 248 samples have '1' as their 'stroke' output. The dataset selected for the study is imbalanced. So this hinders the model's prediction since it would give high accuracy, but precision and recall would lack. To combat the issue of imbalanced data, the data processing approach that we have taken involves undersampling the data and reducing the data points containing a stroke value of '0' to the same as that of '1' as well as oversampling the data with a stroke value of '1' to the same of that of '0'. This helps create a balanced dataset for the model's training data and is achieved using modules from the imblearn library.

3.3. Exploratory Data Analysis

Using the heatmap function in seaborn, we are able to plot a heatmap that gives us the correlation between our features. Looking at the correlation between the stroke class and other features, we can get a rough idea regarding the importance of features. Taking some of these features and plotting their boxplots with respect to their labels, we were able to see which features have a high correlation with the output label. Age and average glucose level have a high correlation with the output label, so both their boxplots are

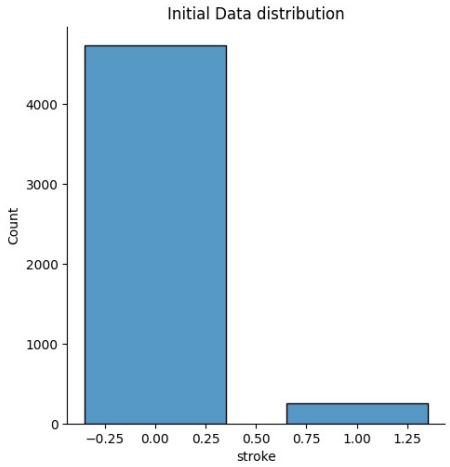

Figure 2. Initial Data Distribution

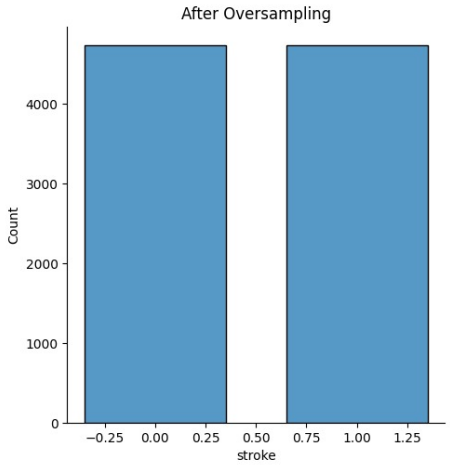

Figure 3. Data after Oversampling

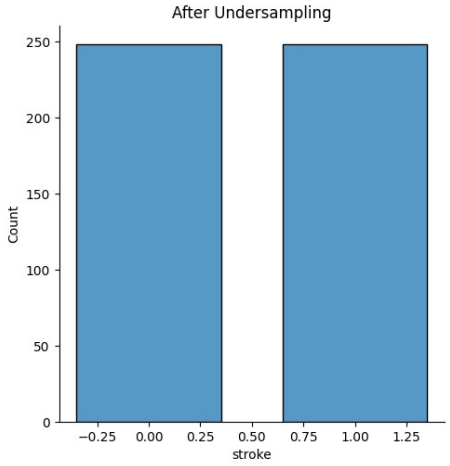

Figure 4. Data after Undersampling

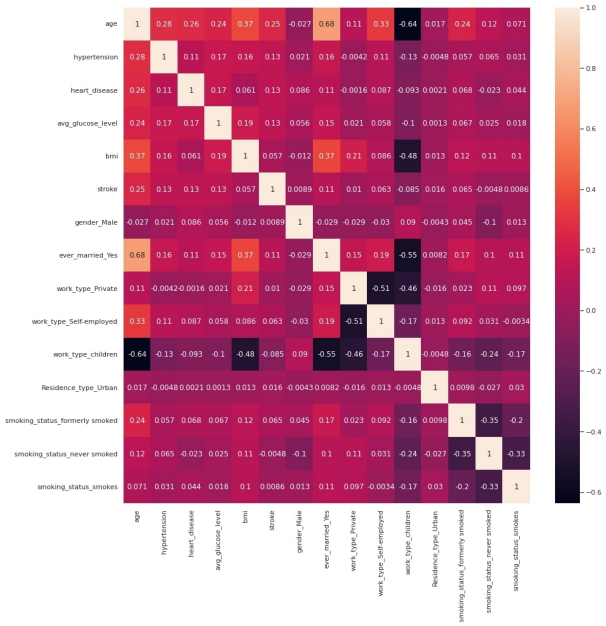

Figure 5. Correlation Heatmap

Attribute	Score
age	322.045639
heart_disease	91.884086
avg_glucose_level	89.971820
hypertension	88.245189
ever_married	59.199605
bmi	16.187059
work_type	8.729184
smoking_status	3.758131
Residence_type	1.354940
gender	0.391752

Figure 6. Sklearn Feature Selection Scores

different, while bmi having a low correlation, has similar boxplots.

Using sklearn's features_selection module, we use the SelectKBest classifier to obtain the scores based on the correlation of variables. We obtain the following scores in descending order.

Using these methods, we were able to conclude that age, hypertension, heart disease, and average glucose level were the most important features.

4. Methodology

4.1. Hyper-parameter Selection

We followed a well-structured and iterative process for fine-tuning our models and algorithms. We started by manually trying out different values of various parameters, such as activation functions, solvers and regularization parameters. Then after filtering these parameters out, we conducted a grid search using the sklearn library implementation upon the filter values, which gave us the set of parameters which delivered the best performance. We took these parameters as a base and then tried tweaking them to obtain the best possible performance.

4.2. Undersampling

The data available is imbalanced and contains only 248 samples having '1' as the class label. The method of undersampling the data points can help obtain data that is not skewed. Undersampling aids in balancing the data as less number of samples of the majority class are taken in the data to match the samples of the minority class. Hence in our case, the majority class(class '0') is undersampled for the minority class(class '1') using the centroid of clusters of classes from the available data. So after undersampling, the resulting dataset will have 248 rows with the value '0' and 248 rows with the value '1'.

4.3. Oversampling

We use the Synthetic Minority Oversampling Technique (SMOTE) using which we synthesis examples for the minority class (class 1 in our case) to re-balance the data. Using SMOTE, points are sampled on the line segment joining a random k-nearest neighbour of the point and the point itself. We apply this technique on the to balance the data which can be further used to train classifiers. Hence in our case, after undersampling, the resulting dataset will have 4733 rows with the value '0' and 4733 rows with the value '1'.

4.4. Modifying the decision threshold

We move the decision threshold to predict class '0' only when we are more confident, that is, the probability of class '0' is very high. This helps us avoid False Negatives for class '1' as we predict class 1 more often than we would with a simple decision threshold of 0.5. This method initially gave us better performance, however, we did not observe improvements using this method in the following analysis, and this method was dropped in favour of oversampling and undersampling techniques.

4.5. Baseline Models

We use the following models to form our baselines in our analysis. The hyperparameters for each model are selected

	Balanced Accuracy:	Macro Precision:	Macro Recall:	Macro F1:
AdaBoost	0.5031	0.5502	0.5031	0.4945
Random Forest	**0.5034**	**0.5752**	**0.5034**	**0.4947**
Logistic Regression	0.5018	0.5251	0.5018	0.4911
MLP	0.4998	0.4751	0.4998	0.4871
SVM	0.5000	0.4751	0.5000	0.4872

Figure 7. Results for Initial Data

using sklearn grid search implementation.

- **Random Forest** - We used sklearn implementation with the following hyper-parameters: n_estimators = 500, max_depth = None and criterion = "gini"
- **Logistic Regression** - We used sklearn implementation with the following hyper-parameters: random_state = 0, C = 1, max_iter = 100 and solver = 'liblinear'
- **Multi Layer Perceptron** - We used sklearn implementation with the following hyper-parameters: activation = 'identity', batch_size = 128, hidden_layer_sizes = [256, 32] and solver = 'adam'
- **Support Vector Machine** - We used sklearn implementation with the following hyper-parameters: C = 0.01 and kernel = 'linear'

4.6. Proposed Model - AdaBoost

The proposed model is an AdaBoost classifier with the hyper-parameters base_estimator = DecisionTreeClassifier(max_depth = 2), learning_rate = 0.1, n_estimators = 150. The hyper-parameters were found using sklearn grid search and then fine-tuned manually considering the cross-validation results.

5. Results and Analysis

We perform K-Fold cross-validation with K = 5 for all our models. We select macro-F1 as the most relevant metric to our task as it captures both precision and recall information. It is observed that for skewed data, either precision or recall may suffer for a class so the macro-F1 score is a fair metric for comparison.

5.1. Baseline Models

- **Random Forest** - The model is the best model performer on the initial data without using any data balancing techniques. With the use of data balancing techniques, the random forest classifier is a consistent second-best performer in our testing.

	Balanced Accuracy:	Macro Precision:	Macro Recall:	Macro F1:
AdaBoost	**0.9557**	**0.9575**	**0.9557**	**0.9556**
Random Forest	0.8731	0.8752	0.8731	0.8728
Logistic Regression	0.6817	0.6853	0.6817	0.6799
MLP	0.6838	0.6897	0.6838	0.6812
SVM	0.685	0.7207	0.6855	0.6729

Figure 8. Results for Under-Sampled Data

	Balanced Accuracy:	Macro Precision:	Macro Recall:	Macro F1:
AdaBoost	**0.9641**	**0.9697**	**0.9641**	**0.9637**
Random Forest	0.9564	0.9575	0.9564	0.9564
Logistic Regression	0.7878	0.7907	0.7878	0.7873
MLP	0.7887	0.7922	0.7887	0.7880
SVM	0.7858	0.7908	0.7858	0.7849

Figure 9. Results for Over-Sampled Data

- **Logistic Regression** - Logistic consistently ranks as the worst model in our tests. The model may be too simple for our data and does not perform well.
- **Multi Layer Perceptron** - Multilayer Perceptron is expected to perform among the best due to its ability to form complex decision boundaries using many neurons. However, we do not observe such an effect and MLP performs well below the best model in our experiments.
- **Support Vector Machine** - Support Vector Machine performs well below the best model in our experiments. The model performance is comparable to MLP classifier.

5.2. Proposed Model - AdaBoost

The proposed model performs significantly better in our tests based on the undersampled data with a macro F-1 score of 0.9556. The model is comparable but slightly better than the random forest classifier with a macro F-1 of 0.9637. An ROC-AUC curve of its performance is given. The Confusion matrix for this classifier is also given.

The overall best model is the AdaBoost classifier on oversampled data with the state-of-the-art performance of a mean cross-validation (K = 5) macro F-1 score of 0.9637. This model is highly effective in the prediction of brain stroke.

6. Conclusion

Among initial baseline results, we got the best results using a Random Forest Classifier. This indicated to us that a

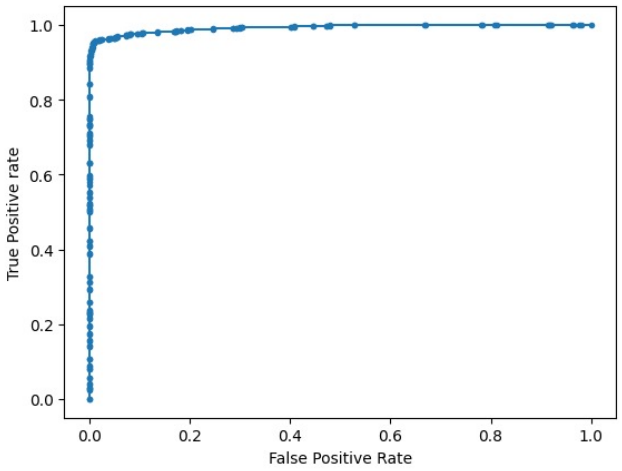

Figure 10. ROC-AUC Curve for AdaBoost Classifier

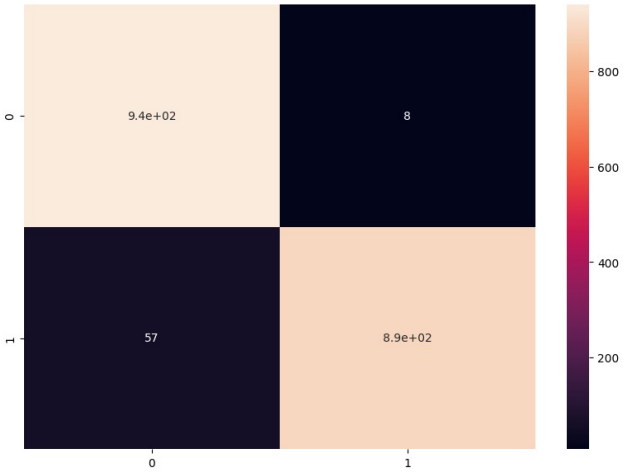

Figure 11. Confusion Matrix for AdaBoost Classifier

tree-based structure may be the most suited for our classification task and we followed through with the concept of boosting using trees with AdaBoost which gave us our final best model. In our analysis that a more complex classifier fails to perform better than the decision tree-based AdaBoost. A classifier like MLP is expected to perform better than AdaBoost for the classification task. However, neural networks require huge amounts of data for training which we believe is why our models for MLP classifier fail to perform. Our dataset is both a small one as well as a skewed one which presents many problems for these classifiers.

We are confident that our performance for the AdaBoost classifier in terms of macro F-1 scores presents it as a reliable model for predicting brain strokes. The exhaustive study of these models gives us confidence that we are close to the peak of performance possible with the dataset at hand.

References

[1] Aditya Khosla, Yu Cao, Cliff Chiung-Yu Lin, Hsu-Kuang Chiu, Junling Hu, and Honglak Lee. An integrated machine learning approach to stroke prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2010.

[2] Gangavarapu Sailasya and Gorli L Aruna Kumari. Analyzing the performance of stroke prediction using ml classification algorithms. *International Journal of Advanced Computer Science and Applications*, 12(6), 2021.