Performance Analysis of Machine Learning Algorithms for Prediction of Cerebral Attack (Stroke)

# 1. INTRODUCTION

A stroke results when our brain can’t get abundant oxygen and required supplements, commonly most strokes (87%) are ischemic strokes. An ischemic stroke take place when blood flow by the artery that supplies oxygen-rich blood cells to the brain emerge as blocked. Each year, approximately 795,000 people in the United States have suffer a stroke, around 610,000 of these are first or new strokes. The after effects of stroke might involve failure of muscular coordination, transient or lasting paralysis, vision problem in one or both eyes and desperate straits in eating or speaking [1].

We can minimize the chance of having a stroke by safekeeping our blood pressure, blood glucose, and cholesterol on mark with nutrients enrich eating, physical activity, and, if vital, medicine. And if smoke, quit. Every step we take will help us move closer to our targets, the better our outlook of preventing a stroke [2].

Machine Learning via pattern perceiving at present is the most trusted way to cure and predict the oncoming disease. Cost of curing strokes in U.S. could soar to $180B annually by 2030 [3]. We can try to reduce 14-18% of this cost by building a well-trained machine learning model which can predict precisely whether a particular patient will suffer a stoke or not. In this research, we have analysed different machine learning models such as KNN, SVM, Random Forest, LGBM classifier and many more, out of these we selected some models for STACKING which will gives us the final result. The detailed work of this research has been mentioned in “WORKFLOW” section and subsection.

# 2. WORKFLOW

A pictorial representation of considered scheme has been depicted below in Fig. 1, which consists of various data pre-processing techniques, data splitting, model development and testing stage and performance analysis of trained models. The procedure for this task is divided into following stages –

a. Data Acquisition

b. Data Study and Feature Engineering

c. Machine Learning for Classification

d. Selecting best ML models for Stacking

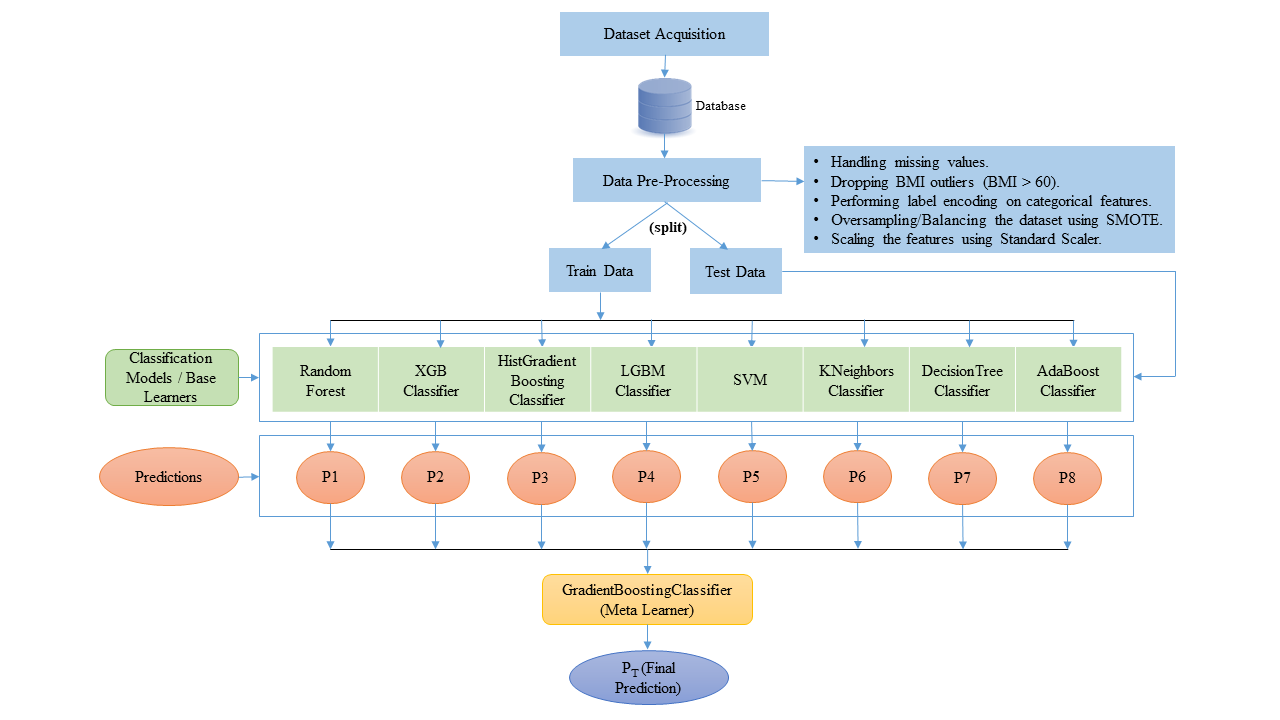


Fig. 1. The architecture for considered system.

## A. Data Acquisition and Feature Engineering

The dataset analysed in this paper is an open-source Stroke prediction dataset downloaded from Kaggle. This dataset formed by conducting the study on 5110 patients and judged them on the basis of 11 factors – “gender”, “age”, “avg\_glucose\_level”, “bmi”, “hypertension”, “heart\_disease”, “ever\_married”, “residence\_type”, “work\_type”, “smoking\_status” and “stroke”. The detailed overview of columns has been shown in Table 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute/Column name | Data-Type | Column Nature | Attribute/Column Description |
| gender | string | Categorical | Tells gender of patient “Male”, “Female” & “Other” |
| age | float64 | Continuous | Tells the age of the patient |
| hypertension | int64 | Binary | Tells whether the patient is hypertensive (1) or not (0). |
| heart\_disease | int64 | Binary | Tells whether a patient has any heart disease (1) or not (0). |
| ever\_married | string | Categorical | Tells the martial status of a patient (“Yes”) or (“No”). |
| work\_type | string | Categorical | "children", "Govt\_jov", "Never\_worked", "Private" or "Self-employed" |
| residence\_type | string | Categorical | “Urban” or “Rular” |
| avg\_glucose\_level | float64 | Continuous | Tells the average glucose level of the patient. |
| bmi | float64 | Continuous | Tells the BMI of the patient. |
| smoking\_status | string | Categorical | "formerly smoked", "never smoked", "smokes" or "Unknown" |
| stoke | int64 | Binary | Tells whether a patient had a stroke (1) or not (0). |

Table 1

## B. Feature Engineering

Machine learning is about guiding a computer system to operate peculiar work placed on inferences drawn from past data. We don’t need to grant explicit training. Yet, we do ought to provide satisfactory data to train the algorithm. Initially the data is in a raw form. At first, we have to extract features from this data before providing it to the ML algorithm. This process of cleaning and extracting important features is called feature engineering.

After analysing the entire Stroke Prediction dataset, we concluded that this dataset contains following issues: -

* 201 missing values found in bmi column.
  + NaN/null is one of the popular practices to represent the absent value in the data, here we filled all null values with their gender specific median.
* Outliers in bmi column.
  + An outlier is a data point which is seemingly different from the rest, here we examined outliers by dropping all the rows where bmi is greater or equals to 60 and we did drop 1 additional row where gender equals “Other”, since "Other" gender is only one instance in the data. Thereafter cleaning, now our dataset contains 5096 instances previously it had 5110 instances.
* Encoding string type categorical columns.
  + Encoding categorical data is a technique of transforming categorical data into integer scheme, such that the data with transformed values can be served to the models. Here we managed string type categorical columns by applying one hot encoding (provided in sklearn.preprocessing module) without going into dummy variable trap.
* Balancing imbalanced data
  + Imbalanced data points to that class of datasets where the target column has an odd distribution of observations, here we used the famous technique SMOTE (provided in imblearn.over\_sampling module) for balancing the imbalanced data. After oversampling our dataset contains 9694 instances.

|  |  |  |
| --- | --- | --- |
|  | Before SMOTE | After SMOTE |
| Label 0 count from stroke column | 4847 | 4847 |
| Label 1 count from stroke column | 249 | 4847 |

* Scaling features to common range.
  + Feature Scaling is a technique to standardize the range of independent variables in a fixed range. Here we used standard scalar method (from sklearn.preprocessing library) in our study to scale the numerical features into common scale.

Thereafter pre-processing, we split the both old (Before SMOTE) and new (After SMOTE) dataset into train and test separately with ratio of ***0.80:0.20*** respectively.

|  |  |  |
| --- | --- | --- |
|  | Dataset split before SMOTE | Dataset split after SMOTE |
| Count of training instances | 4076 | 7755 |
| Count of test instances | 1020 | 1939 |

## C. Machine Learning Classification

In this research, we analysed our both balanced & imbalanced processed test data on Logistic Regression, SVM, K-nearest neighbours Classifier, Gaussian NB, Bernoulli NB, Decision Tree Classifier, Random Forest Classifier, XG Boost Classifier, Ada Boost Classifier, Gradient Boosting Classifier, Hist Gradient Boosting Classifier & LGBM Classifier respectively.

After investigating all the above trained ML model results, we concluded that K-nearest neighbours’ classifier performed well for imbalanced test data with an accuracy of 95.29% and on the other side Random Forest classifier delivered the best result for balanced test data with an accuracy of 97.31.

## D. Stacking

Ensemble learning is the technique via which numerous models, united to determine a specific computational intelligence problem. In stacking, an algorithm receives the outputs of sub-models as input and try to grasp how to best bind these input predictions in order to accomplish better prediction results.

In our scheme, we trained and evaluated the prepared balanced data on various machine learning algorithms viz. – *Logistic Regression, SVM, K-nearest neighbours Classifier, Gaussian NB, Bernoulli NB, Decision Tree Classifier, Random Forest Classifier, XG Boost Classifier, Ada Boost Classifier, Gradient Boosting Classifier, Hist Gradient Boosting Classifier & LGBM Classifier*. At the beginning we practised all these models without hyperparameter tuning, so here we used RandomizedSearchCV (provided in sklearn.model\_selection module) for tuning some models before applying stacking. The list of tuned models with their parameters given in Table 2.

|  |  |
| --- | --- |
| Model Name | Tuned parameters |
| Logistic Regression | solver='newton-cg', penalty='l2', C=0.01 |
| SVM | kernel='rbf', gamma=1, C=10, probability=True |
| KNeigbors | weights='distance', n\_neighbors=5, metric='manhattan' |
| DecisionTree | min\_samples\_leaf=5, max\_depth=20, criterion='entropy' |
| RandomForest | n\_estimators=90, min\_samples\_split=5, min\_samples\_leaf=1, max\_features='log2', max\_depth=18, criterion='gini' |
| XGBClassifier | min\_child\_weight=1, max\_depth=12, learning\_rate=0.1, gamma=0.1, colsample\_bytree=0.5 |
| AdaBoostClassifier | random\_state=0, learning\_rate=0.1, n\_estimators=200 |
| GradientBoostingClassifier | subsample=0.5, n\_estimators=1000, max\_depth=9, learning\_rate=0.01 |

Table 2

Later we performed stacking with the help of StackingClassifier api given in sklearn.ensemble module. For base model we have selected 8 models, where we choose XGB Classifier, Random Forest, SVC, K-Nearest Neighbors, Decision Tree, AdaBoost Classifier from Table 2 and we did take HistGradientBoostingClassifier, LGBMClassifier with their default parameters as these two performed well without any parameter tuning. Next, we picked Gradient Boosting Classifier as meta learner. Thereafter we again study our balanced test data with the recently created stacked model and we achievd an Accuracy of 98.85% with Precision, Recall, F1-score, Cohen-Kappa score and Roc\_Auc score as 100.00%, 98.00%, 99.00%, 97.00% and 99.00% respectively.