Stroke Prediction

Dev Goel (B20CS090), Ravi Ramavat (B20CS053), Sarvesh Kulkarni (B20EE059)

GitHub Repo: PRML-Project, Deployed at PRML-Project-Frontend

Abstract — **This paper reports our experience with building a Stroke Prediction classifier. We have been given dataset that can be used to predict whether a patient is likely to get a stroke based on the input parameters like gender, age, various diseases, and smoking status. This report analyses various Machine Learning models and discusses the comparison of them.**

Keywords — Neural Network, Random Forest Classifier, kNN, LightGBM, XGBoost, Support Vector Machine, Radial Basis function kernel

I. Introduction

The World Health Organization (WHO) identifies strokes as the second leading cause of death globally. A stroke happens when a person's blood supply to their brain is interrupted or reduced, causing brain cells to die within minutes. It prevents the brain tissue from getting the oxygen and nutrients that it needs and is responsible for approximately 11% of total deaths. The objective is to examine the use of various machine learning classification models on the given dataset that can aid in identifying the chance of stroke. The project aims at classifying the stroke based on the input parameters like gender, age, various diseases, and smoking status. Since, the project is related to medical domain multiple models were trained and their performance was compared considering the sensitivity, accuracy, as well as specificity scores.

II. Data Description And Preprocessing

This dataset is used to predict whether a patient is likely to get stroke based on the input parameters. Each row in the data provides relevant information about the patient.

Preprocessing

The id feature was unnecessary and was hence, dropped. Given that the dataset contains only 1556 rows and 201 of them were null valued, it was decided not to simply drop them, but to replace them with class-average values. Label encoding was applied in the categorical features, and appropriate features were scaled using the Standard Scaler provided by the sklearn library. The dataset was then split into training and testing sets in the ratio of 80: 20 stratifying with respect to the classes present.

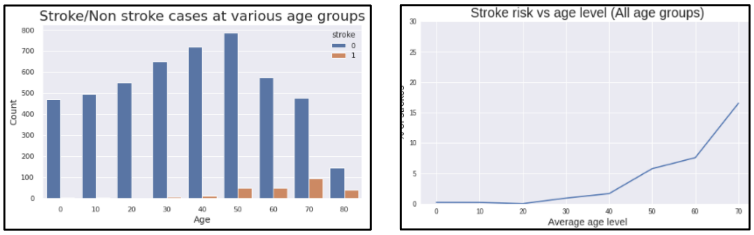
The model could be well trained for predicting no stroke, but due to a lack of training data for positive cases, it will not function well in predicting positive stroke, which of course is the most crucial point of the model. Random Oversampling was done with minorities as a sampling strategy to increase the occurrences of stroke positive cases in the training dataset. This helped the model train well for positive stroke cases too. This was done using RandomOverSampler provided by imblearn library.

III. Exploratory Data Analysis

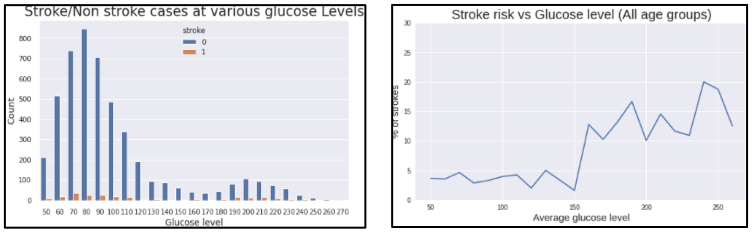
Exploratory Data Analysis is primarily used to see what the data can reveal beyond the formal modeling or hypothesis testing task and to provide a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate or not. Since there were a lot of features like (age, hypertension, heart disease, ever married, BMI, average glucose level, etc.), it was important to understand the key features and the relationship between them.

Stroke was visualized against the features to determine the comparative contribution of each feature. BMI, Age, and Glucose level were found to be more impacting (contrary to anticipated heart disease and hypertension).

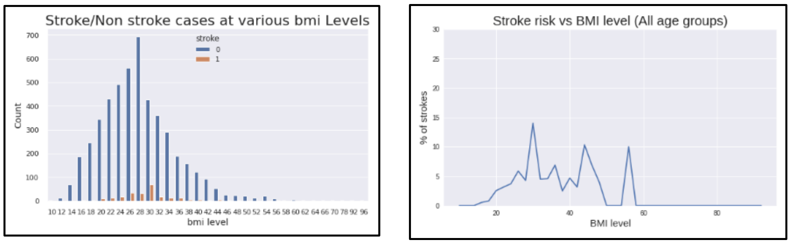
Plot: Stroke against Age



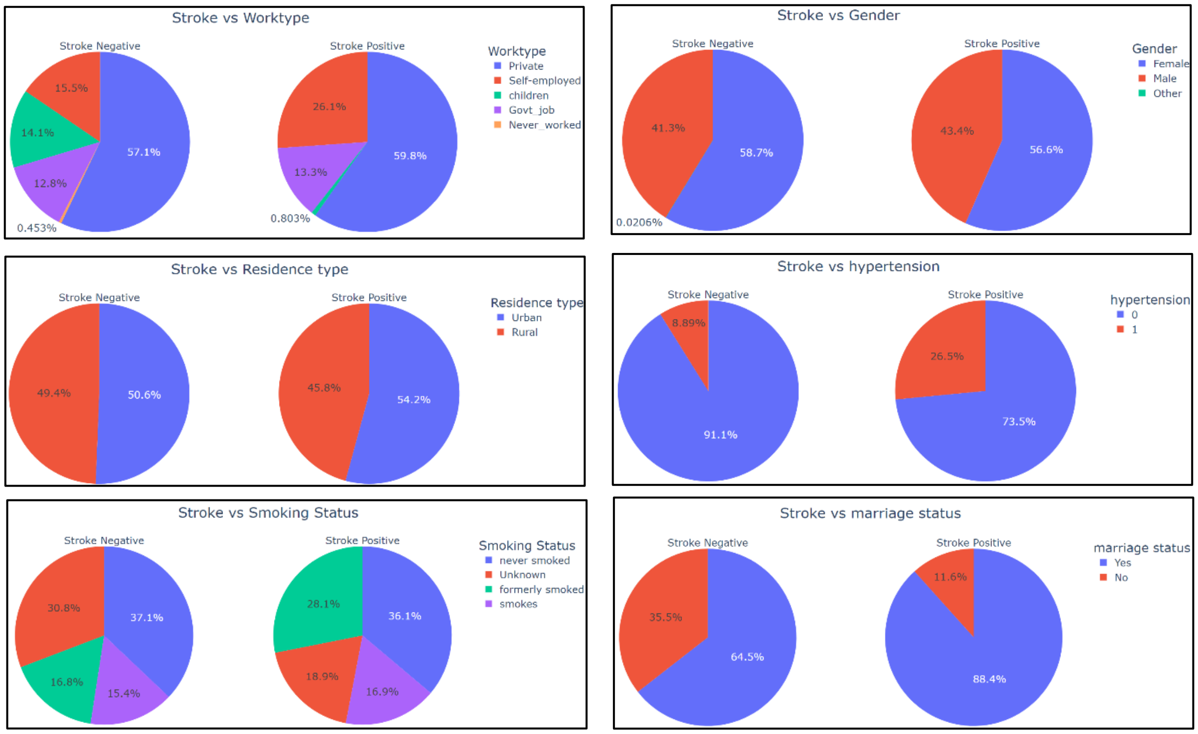
Plot: Stroke against Glucose Level



Plot: Stroke against BMI



Plot: Stroke against all categorical columns



IV. MACHINE LEARNING MODELS & ANALYSIS

Note: Since, the problem was binary classification and dataset constraints was minimal so we tried to apply as many classifiers as possible. To analyze how they are performing with/without hyperparameters tuning.

A. *Decision Tree Classifier*

Decision tree classifiers are used successfully in many diverse areas. It is generated with nodes as decision units, penetrating multiple layers. **Default:** The Decision tree classifier was trained on *default hyperparameters* and got *sensitivity as 0.14, specificity as 0.96 and accuracy score as 0.92*. As we can see accuracy score is very good because model was not able to predict positive samples. Hence, sensitivity score is too low so the model is performing very poor. Then, grid search was applied on decision trees. **With Grid Search:** The Decision tree classifier was trained with *criterion = 'gini', max_depth = 5, min_samples_leaf = 1* and got *sensitivity as 0.74, specificity as 0.78 and accuracy score 0.78*. This implies that performance of decision tree was improved drastically after grid search.

B. *Extreme Gradient Boosting Classifier*

XGB classifier speed and performance are unparalleled and it consistently outperforms any other algorithms aimed at supervised learning tasks. **Default:** The XGBoost classifier was trained on *default hyperparameters* and got *sensitivity as 0.74, specificity as 0.80 and accuracy score as 0.799*. As we can see accuracy score is decent because model was able to predict some positive samples accurately. Therefore, sensitivity is also good. Then, grid search was applied on XGBoost classifier. **With Grid Search:** The XGBoost classifier was trained with *colsample_bytree = 0.8, gamma = 0.5, max_depth = 5, min_child_weight = 1, subsample = 0.6* and got *sensitivity as 0.56, specificity as 0.91 and accuracy score 0.89*. This implies that performance of XGBoost classifier tree was improved drastically for specificity and decreased slightly for sensitivity.

C. *Random Forest Classifier*

Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It then aggregates the votes from different decision trees to decide the final class of the test object. **Default:** The Random Forest classifier was trained on *default hyperparameters* and got *sensitivity as 0.02, specificity as 0.99 and accuracy score as 0.94*. As we can see accuracy score is very good because model was not able to predict positive samples at all. Hence, sensitivity score is nearly zero so the model is performing very poor. Then, grid search was applied on the model. **With Grid Search:** The Random Forest classifier was trained with *max_depth = 7, max_features = "sqrt", min_samples_leaf = 1, min_samples_split = 2, n_estimators = 50* and got *sensitivity as 0.72, specificity as 0.77 and accuracy score 0.77*. This implies that performance of random forest classifier was improved drastically for sensitivity after grid search.

D. *Light Gradient Boosting Machine*

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages faster training speed and higher efficiency, lower memory usage, better accuracy, support of parallel, distributed, and GPU learning, capable of handling large-scale data. **Default:** The LightGBM classifier was trained on default hyperparameters and got sensitivity as 0.22, specificity as 0.96 and accuracy score as 0.92. As we can see accuracy score is very good because model was not able to predict positive samples. Hence, sensitivity score is too low so the model is performing very poor. Then, grid search was applied on model. **With Grid Search:** The Light GBM classifier was trained with bagging_fraction = 0.8, max_depth = 11, n_estimators = 300, num_leaves = 200 and got sensitivity as 0.08, specificity as 0.99 and accuracy score 0.94. This implies that model was overfitted on training data as training score was 0.99.

E. *Linear Support Vector Classification*

The objective of a Linear SVC (Support Vector Classifier) is to fit the data you provide, returning a "best fit" hyperplane that divides or categorizes your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is. **Default:** The Linear SVC was trained on default hyperparameters and got sensitivity as 0.80, specificity as 0.72 and accuracy score as 0.73. It is observed that this model gave the best sensitivity score as well the accuracy and specificity score as compared to previous ML models.

F. *Radial Basis Function SVC*

Default: The Radial Basis Function SVC was trained on default hyperparameters and got sensitivity as 0.68, specificity as 0.72 and accuracy score as 0.72. The performance was similar to Linear SVC model. Then, grid search was applied on random basis function SVC. **With Grid Search:** The Radial Basis Function SVC was trained with C = 10, gamma = 1 and got sensitivity as 0.08, specificity as 0.94 and accuracy score 0.90. This implies that model was overfitted on training data as training score was 0.97.

G. *Linear Discriminant Analysis*

Linear Discriminant Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modeling differences in groups i.e., separating two or more classes. It is used to project the features in a higher dimension space into a lower dimension space. **Default:** The LDA classifier was trained on default hyperparameters and got sensitivity as 0.82, specificity as 0.73 and accuracy score as 0.73. It is observed that this model gave the best sensitivity score as well the accuracy and specificity score among all other ML models.

H. *Quadratic Discriminant Analysis*

Default: The QDA classifier was trained on default hyperparameters and got sensitivity as 0.80, specificity as 0.75 and accuracy score as 0.75. It is observed that this model gave the best sensitivity score as well the accuracy and specificity score among all other ML models.

I. *K-Nearest Neighbors*

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. **Default:** The kNN was trained on default hyperparameters and got sensitivity as 0.22, specificity as 0.88 and accuracy score as 0.85. It is observed that this model gave very poor sensitivity score.

J. *Multi-Layer Perceptron Classifier*

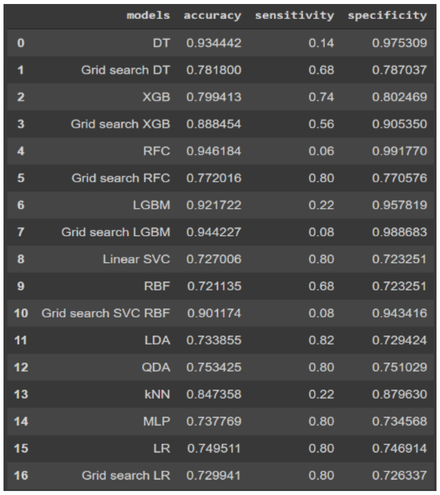
Default: The MLP classifier was trained on default hyperparameters and got sensitivity as 0.78, specificity as 0.74 and accuracy score as 0.74. It is observed that this model gave the best sensitivity score as well the accuracy and specificity score among other ML models.

K. *Logistic Regression*

Default: The LR classifier was trained on default hyperparameters and got sensitivity as 0.80, specificity as 0.75 and accuracy score as 0.75. It is observed that this model gave the best overall score. Then, grid search was applied on model. **With Grid Search:** The Light GBM classifier was trained with C = 0.01, penalty = 'l2', solver = 'newton-cg' and got sensitivity as 0.80, specificity as 0.73 and accuracy score 0.73. This implies that model performed good.

V. Overall Analysis

The image in the right shows the analysis of all the machine learning models on the basis of accuracy, sensitivity and specificity as evaluation metrics.



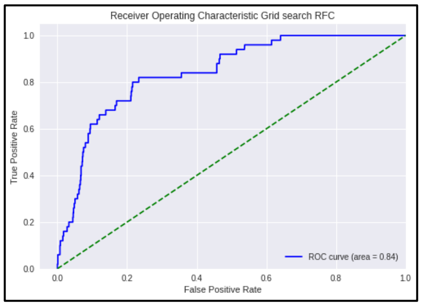
	models	accuracy	sensitivity	specificity
0	DT	0.934442	0.14	0.975309
1	Grid search DT	0.781800	0.68	0.787037
2	XGB	0.799413	0.74	0.802469
3	Grid search XGB	0.888454	0.56	0.905350
4	RFC	0.946184	0.06	0.991770
5	Grid search RFC	0.772016	0.80	0.770576
6	LGBM	0.921722	0.22	0.957819
7	Grid search LGBM	0.944227	0.08	0.988683
8	Linear SVC	0.727006	0.80	0.723251
9	RBF	0.721135	0.68	0.723251
10	Grid search SVC RBF	0.901174	0.08	0.943416
11	LDA	0.733855	0.82	0.729424
12	QDA	0.753425	0.80	0.751029
13	kNN	0.847358	0.22	0.879630
14	MLP	0.737769	0.80	0.734568
15	LR	0.749511	0.80	0.746914
16	Grid search LR	0.729941	0.80	0.726337

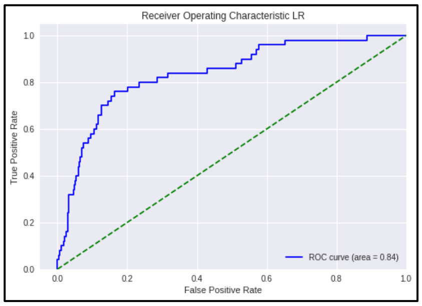
It is observed that on setting the default parameters the models Linear Support Vector Classifier, Logistic Regression, Multi-Layer Perceptron, and Quadratic Discriminant Analysis classifier gave the best sensitivity, specificity and accuracy. (Here sensitivity is considered in priority with respect to specificity and accuracy)

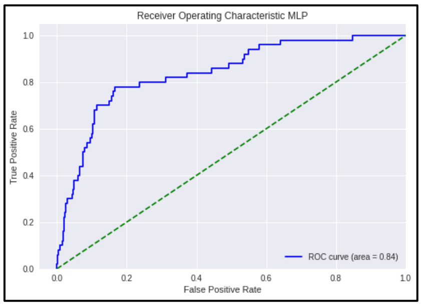
Then, on applying grid search on some machine learning models like Decision Tree Classifier, XGBoost Classifier, Random Forest Classifier shown improvement in the sensitivity, specificity and accuracy scores. Whereas other models like LGBM Classifier, SVC RBF were overfitting on the dataset and hence their scores on testing dataset were very poor.

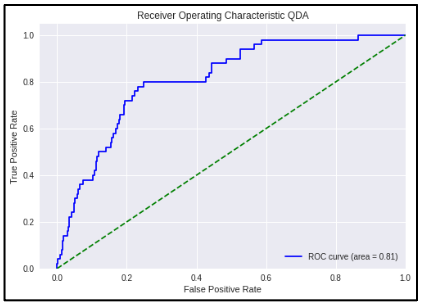
So, on an overall analysis it is observed that the best 5 machine learning models were Random Forest Classifier (after hyperparameter tuning with Grid Search), Linear Support Vector Classifier, Logistic Regression, Multi-Layer Perceptron, and Quadratic Discriminant Analysis that gave the best evaluation with respect to sensitivity, specificity and accuracy scores (in priority order).

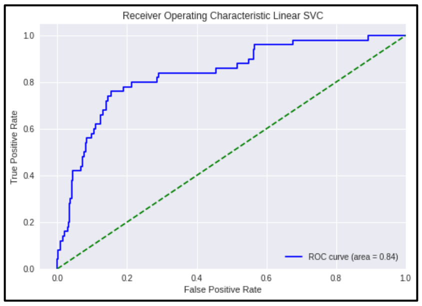
We have plotted the ROC-AUC curve for every mentioned model above in over code file. The best 6 model's ROC-AUC curves are reported below.

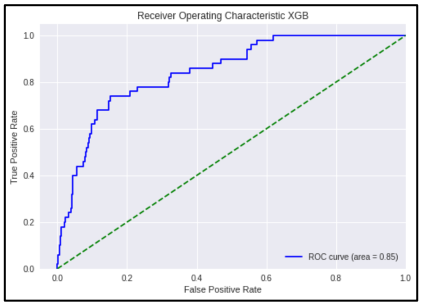












VI. Conclusion

The Linear Models (such as Logistic Regression, Linear SVC, LDA) performed very well even without the hyperparameter tuning. Tree models such as random forest classifier and decision tress performed poor on default parameters but performed well after hyperparameter tuning. Other models like neural networks and discriminative models (such as QDA, LDA) also performed well with default parameters. Hence, we were able to build a machine learning classifier that it was able to predict the chance of strokes.

References

[1] https://www.hindawi.com/journals/jhe/2021/7633381/
[2] https://www.who.int/southeastasia/news/speeches/detail/world-stroke-day-2019
[3] https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/
[4] https://thesai.org/Downloads/Volume12No6/Paper_62-Analyzing_the_Performance_of_Stroke_Prediction.pdf

Contribution

- Dev Goel (B20CS090) - Worked on implementing different machine learning classifiers and done analysis of them through evaluation metrics. Also, contributed in dataset preprocessing. He is responsible for deploying the frontend of the project.
- Ravi Ramavat (B20CS053) - Worked on iterating different preprocessing aspects and best in-depth exploratory data analysis. He is also responsible for implementing and hyperparameter tuning of different machine learning classifiers.
- Sarvesh Kulkarni (B20EE059) - Worked on implementing the machine learning classifiers and contributed in different preprocessing aspects.