Stroke Prediction Dataset

https://github.com/KiraShen33/2_Final_1030/

Ruiqi Shen

Dec 06th, 2022

Introduction

Background of Data

According to the World Health Organization (WHO), we could know that stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. The death rate is a large percentage of actual death worldwide, encouraging my interest in predicting the underlying risk factors of stroke. Meanwhile, even though some patients suffer from stroke and so luckily survive, they might also suffer from expensive medical bills and even disability. Thus, this foreseeing becomes extremely valuable to stroke screening and prevention.

This dataset is from a Kaggle csv file (unknown source) containing 5,110 points and 12 columns as features to analyze which features play the most crucial role. It is used to predict whether a patient will likely get a stroke based on input features like age, gender, heart disease, and various features. We could find the relevant information about the patient in each row and corresponds to the attributes that are id, gender, age, hypertension, heart_disease, ever_married, work_type, Residence_type, avg_glucose, bmi, smoking_status, stroke. Meanwhile, based on features, we could conclude that this is a classification problem.

Feature Definition

1) id: unique identifier
2) gender: "Male", "Female" or "Other"
3) age: age of the patient
4) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
5) heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
6) ever_married: "No" or "Yes"
7) work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
8) Residence_type: "Rural" or "Urban"
9) avg_glucose_level: average glucose level in blood
10) bmi: body mass index
11) smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*
12) stroke: 1 if the patient had a stroke or 0 if not

***Note: "Unknown" in smoking_status means that the information is unavailable for this patient**

Data Cleaning

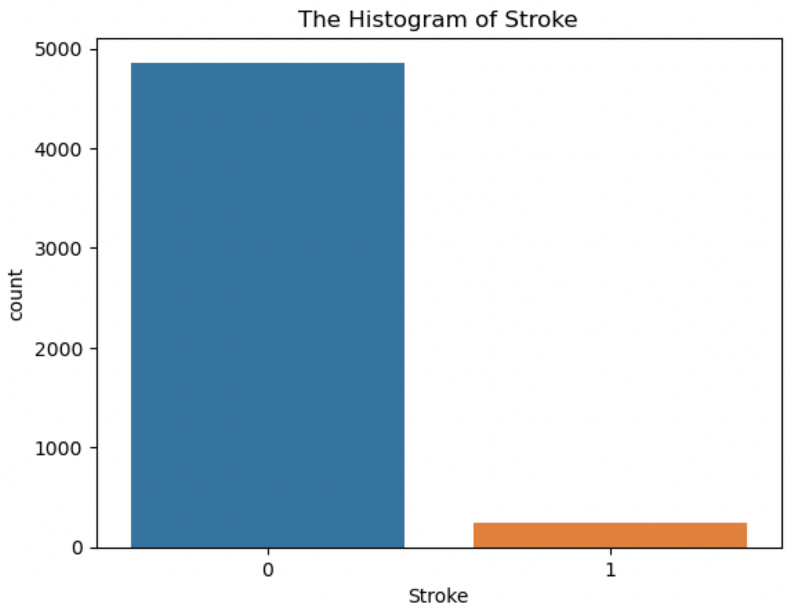
The data was cleaned by keeping columns and rows deemed relevant in our models and dealing with missing values. Firstly, we decided to remove columns with values unrelated to our question, like id. We chose to keep columns 'gender,' 'hypertension,' 'heart_disease,' 'ever_married,' 'work_type,' 'Residence_type,' 'smoking_status,' 'bmi,' 'avg_glucose_level' and 'age' since believed the following columns contained information that could be the potential features causing the stroke. Then, for the missing value, we decided to use the KNN imputation, a better filling method than using mean or median or deleting those points to fill in those lost points. The KNN imputation uses these ‘k’ samples to estimate the value of the missing data points. Each sample’s missing values will be imputed using the mean value of the ‘k’-neighbors found in the dataset. If filled in by the mean value, it might present some bias because it considers every data point, including outliers. Meanwhile, deleting missing values also does not work for this dataset since it is imbalanced. After counting, we could notice that out of 249; we have 40 missing values for the stroke = Yes. Thus, handling the missing values is better than dropping them.

Previous Project

1. Research posted by Alex in 2022 on Kaggle tried to foresee which features affect stroke. He utilized machine learning models, such as logistic regression, decision tree, random forest, and gradient boosting. Eventually, he gains the result that the logistic regression model performs the best in predicting the patient with a stroke, compared with other models. It has twice as much False Positive than False Negative.
2. Another research posted by Bilal Bora in 2022 on Kaggle also mentioned that logistic regression is the best model to predict which features affect stroke. He did a total of seven models: LR, KNN, SVC, CART, Adaboost, and GBM. Linear regression is the one with the highest AUC (0.8470871).

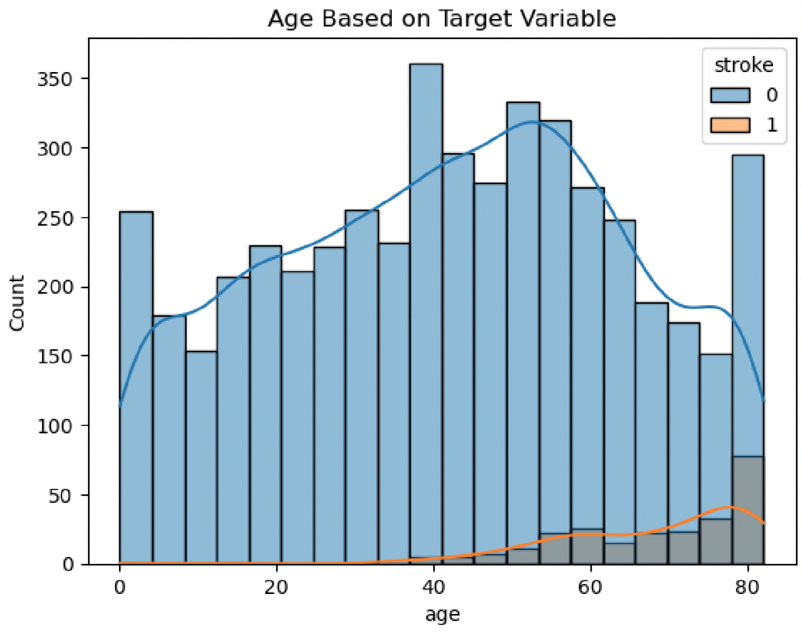
EDA

Target Variable (Stroke)



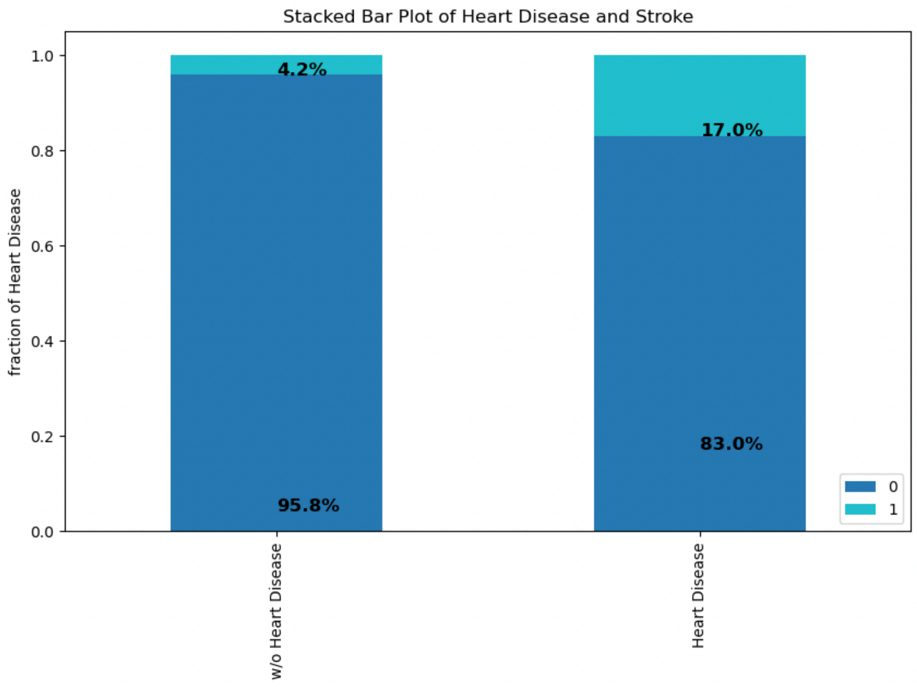
As we can see that from the above plot, its a class imbalancing problem. The number of people who actually had a stroke are very less in our dataset. We'll use oversampling technique to deal with this. About 95.13% of the data is about people who didn't get stroke (class - 0) and only 4.87% of the data is about people who got stroke (class - 1)

Age



From the above plot, we can conclude that there exists a pattern in the distribution of Age. Older people have a much higher chance of getting a stroke as compared to younger individuals.

Heart Disease



According to the above stacked histogram, we can see that 4.2% of patients who don’t have heart disease are with stroke, and 17% of patients with heart disease are also with stroke. We could conclude that heart disease might be an essential potential feature causing stroke. Once patients already have heart disease, they are more likely to get a stroke.

Preprocessing

Since the dataset is not identically distributed, we could know that it is not an iid dataset. Every patient is unique in this dataset, so the group structure will not exist. Furthermore, it won't be time series data since no time variables exist in this dataset. We are only curious about which plays a significant role in causing a stroke.

To better deploy the model in the future, We use the random oversampling method to deal with the dataset. According to both distribution and histogram plots, we could know that the dataset is imbalanced. Thus, the random oversampling process becomes the best due to it randomly selecting examples from the minority class, with replacement, and adding them to the training dataset, which perfectly fits the situation of the dataset and slight variation while oversampling. After applying this method to the training dataset, we could notice that the training point became balanced, from 135 to 2930. In this way, we will be more ready for the model-building part.

OneHotEncoder is applied on the variables 'gender', 'hypertension', 'heart_disease', 'ever_married', 'work_type', 'Residence_type', 'smoking_status', since all of them have unique values. This means that they are categorical variables. Further, OneHotEncoder could help prepare for the model parts by converting categorical features into dummy arrays. StandardScaler is applied on the variables 'bmi', 'avg_glucose_level', 'age' because each of them are continuous variables. Through this encoder, we could standardize them by removing the mean and scaling them to unit variance.

After processing all the features, we could get that the dataset has 5109 data and 11 features.

Model-Building

Due to the large dataset, we could follow the basic splitting rules by assigning 60% of the data as train data. Others will be divided as half-half, separately assigned to validation and test data.Since our problem deals with classification (predicting the important features causing the stroke), we decided on some classification models like Random Forest Classifier, SVC, XGBoost, and Linear Regression as a baseline. Meanwhile, although PR curves are specifically tailored while classifying most or all minority classes, we would choose the f1 score as a measurement. F1 score is the harmonic mean of Precision and Recall, and it captures the contribution of both of them.

Logistic Regression (Benchmark Model)

In the benchmark model, we built a simple linear regression model. We used the LinearRegression() function to fit and transform the data and get the accuracy score of both the validation and test dataset, then leveraged RandomizedSearchCV() to obtain model accuracy in terms of the f1-score. Our linear regression model returned f1-score for test dataset, which is about 75%.

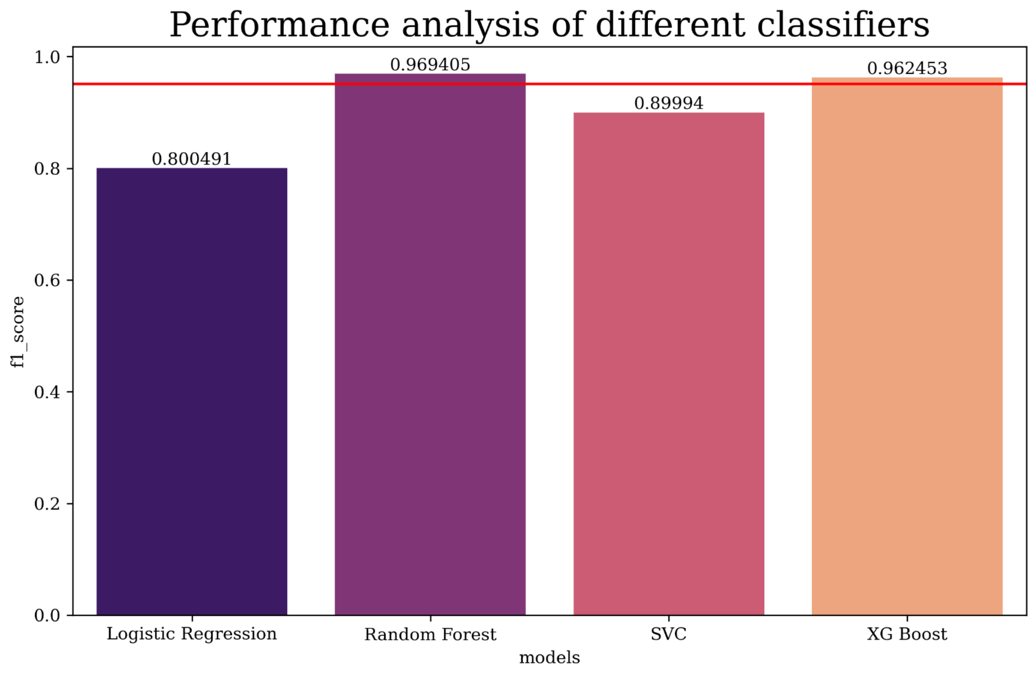
Additional Models
We also trained a random forest model using RandomForestRegressor() and again used RandomizedSearchCV() to obtain a test f1 score of 92%, which is better than our linear regression model. Furthermore, the final two models we trained were the SVC using SVC() and XGBoost model using XGBClassifier(). The SVC model got a f1 score of 93%. The XGBoost model got a f1 score of 89%.

Hyperparameter Tuning
Playing around with the hyperparameters of the model figure out the best parameter for each model and improved most of the models. The f1 score for linear regression model, random forest model, and XGBoost model are improved a lot from its original f1 score; however, for the SVC model, the f1 score decreased instead. Overall, through tuning the parameter of each model, we do make progress on models.

Model	Best Parameters	F1 score (Before tuning)	F1 score (After tuning)
Logistic Regression	'C': 0.3, 'max_iter': 50	75.05%	80.04%
Random Forest	'criterion': 'gini', 'n_estimators': 100	92.37%	96.94%
SVC	'C': 1, 'kernel': 'rbf'	92.86%	89.99%
XGBoost	'max_depth': 5, 'min_child_weight': 1	89.33%	96.24%

Model Comparison
In order to determine the champion model, we considered the f1 score of each. According to the Wikipedia, the F-score is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive.

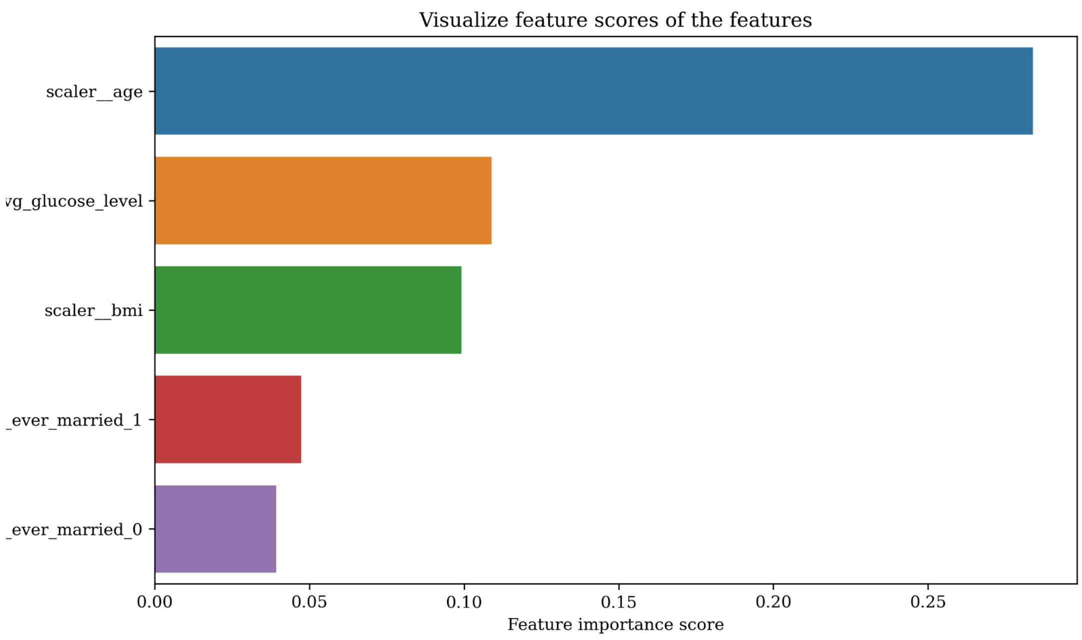
Due to, it is difficult to compare two models with low precision and high recall or vice versa. So to make them comparable, we choose to use F-Score. F-score helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more. The higher the f1 score is (maximum = 1), the better the model predicted.



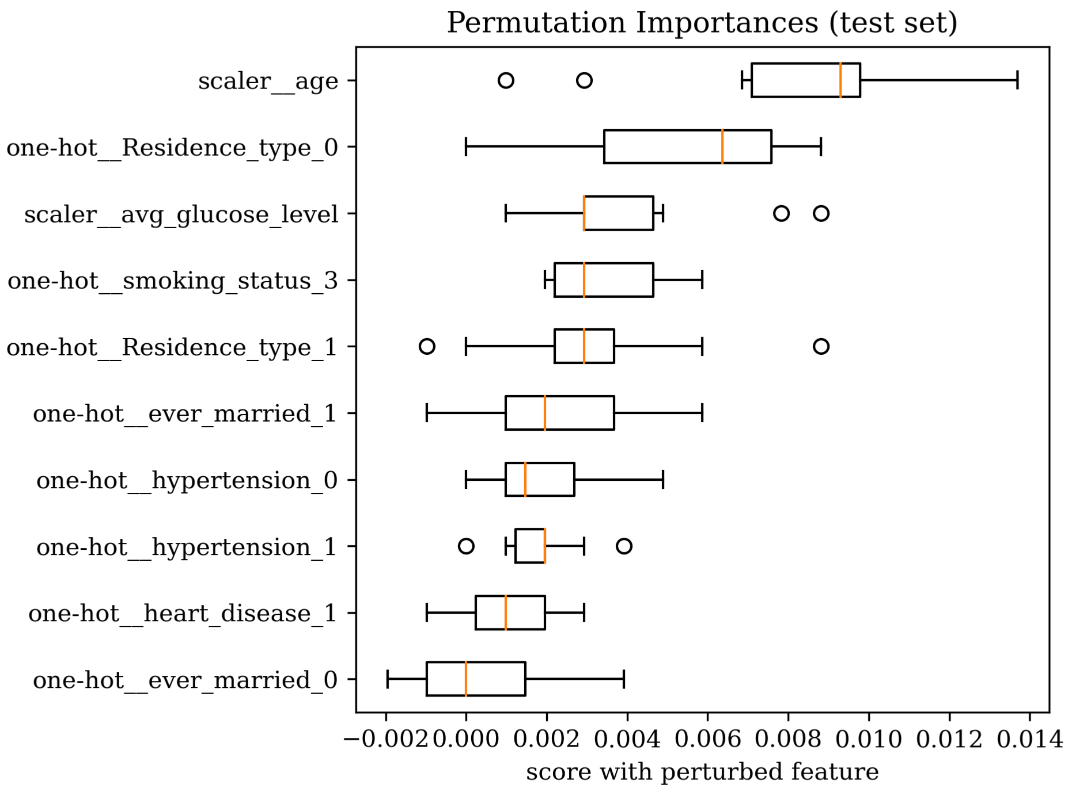
The random forest model has the highest f1 score, which makes it our champion model. We could notice that there are only a little difference between the f1 score of random forest model and XGBoost model. On potential reason might be that the XGBoost always gives more importance to functional space when reducing the cost of a model, while Random Forest tries to give more preferences to hyperparameters to optimize the model.

Feature Importance

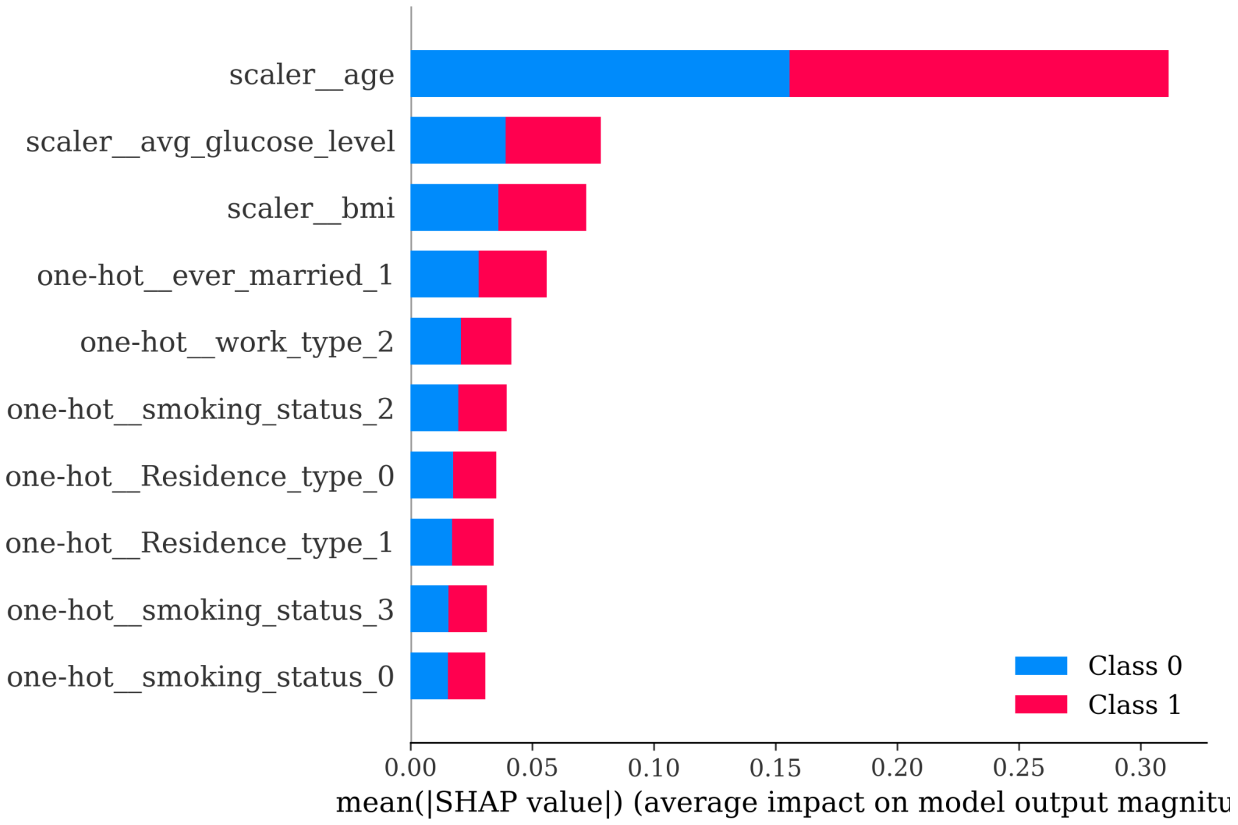
Random Forest Importance

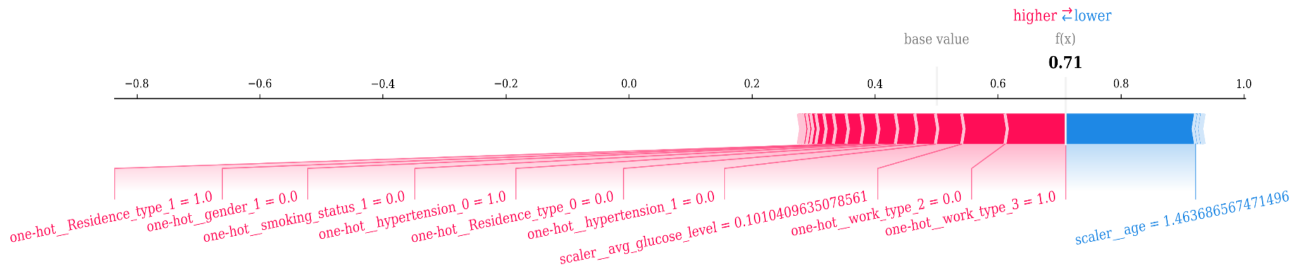


Permutation Importance



SHAP





For all three method to explore feature importance, random forest importance, permutation importance, and SHAP, the feature 'age' is the top one features among all methods of importance. Meanwhile, the feature avg_glucose_level' and 'bmi' tend to be among the top 3 features.

Conclusion

Despite having pretty good f1 scores across all of our models, including our champion model at 96.94%, a decreased f1 score in the SVC model and almost the same f1 score in the XGBoost model indicates that the model has room for improvement and could better fit the data. Firstly, the SVC model is usually used on relatively large data, but this dataset might not be large enough for it. Furthermore, in many situations, XGBoost model usually works better than the random forest model, since XGBoost is an excellent option for unbalanced datasets. In order to account for this, we could try to further adjust the parameters, like increasing its max depth. Therefore, we conclude that the feature 'age' is the most important feature that leads to stroke, and then the feature 'avg_glucose_level' and 'bmi' are secondary factors that influence people to have a stroke. Suppose it is possible to collect more data by gathering it in the hospital or launching an online questionnaire. In that case, we believe more data would help models present a better performance in predicting the potential reasons causing the stroke.

Reference

Alex (Oct 2022). "Stroke Prediction".
https://www.kaggle.com/code/alexandrepetit881234/stroke-prediction

BILAL, BORA (Aug 2022). "Stroke Pred - Plotly EDA - %95 Acc."
https://www.kaggle.com/code/bilalbora/stroke-pred-plotly-eda-95-acc

Wikipedia
https://en.wikipedia.org/wiki/F-score