Final Thesis of Data 698

Title: The association of C-reactive protein and incidence of stroke in the African American population of Jackson Heart Study

Authors: Jun Pan, Ritesh Lohiya, Brian Liles Data: 12/09/2019 Final Version

Abstract

Roughly 700,000 strokes occur each year in the United States. Stroke is the third leading cause of death and the leading cause of neurologic disability. Despite the incidence of stroke increase with age. One third of strokes occurs in people younger than 65 years. The Jackson Heart Study (JHS) is a single-site, prospective cohort study of the risk factors and causes of heart disease in adult African Americans. JHS has 5,301 African Americans participants, aged 21–94 years, residing in a three-county area surrounding the city of Jackson, MS, USA. In this project, we used JHS trans-data which including roughly ~50% random sample of participants in JHS. In part I, we investigated the association of C-reactive protein (CRP) with stroke incidence. The quartiles of hs-CRP in JHS were: minimum (0 mg/dL), 1st quartile (0.1050 mg/dL), medium (0.2610 mg/dL), 3rd quartile (0.553 mg/dL), maximum (10.7060 mg/dL). hs-CRP had big variation with mean (0.4880 mg/dL) in JHS. Using quartile and mean±sd for survival analysis, we failed to conclude the hs-CRP played a role in developing stroke incidence. However, using 1mg/dL of hs-CRP as a cut-off point, we found that hs-CRP had impact on developing of stroke incidence by Cox and Weibull survival analysis ($p<0.05$). In part II, we evaluated the risk factors of stroke using 5 machine learning-based algorithms for selecting features automatically: (1) logistic regression, (2) logistic regression with SGD, (3) random forest, (4) KNN and (5) XGBoost. The evaluation of the performance of our prediction algorithms was based on the area under the ROC curve, accuracy, false negative and false positive. We found logistic regression with SGD rank the best of ROC AUC test. In addition, logistic regression with SGD model had 0 case of false positive and false negative. The accuracy rate for all 5 models were very similar (around 97%). We concluded that logistic regression with SGD were the best model among the 5 machine learning models to predict stroke on JHS trans-data.

Keywords: CRP, stroke, machine learning, ROC, AUC

Review of Literature

CRP is an acute-phase protein featuring a homopentameric structure and Ca-binding specificity for phosphocholine (PCh) (Black S et al, 2004). CRP is mainly produced by hepatocytes in liver which affects plasma concentration in a minute. Any form of tissue injury, infection, inflammation and stress are associated with increasing of plasma CRP values (Kushner et al, 2006). CRP is obviously cleared from the circulation and catabolized exclusively by hepatocytes. The function of CRP is felt to be related to its role in the innate immune system. Similar to immunoglobulin (IgG), it activates complement, binds to Fc receptors and acts as an opsonin for various pathogens. Interaction of CRP with Fc receptors leads to the generation of pro-inflammatory cytokines that enhance the inflammatory response. It has been proven that CRP is useful in predicting a prognosis for stroke patients (Kocaturk 2019). Inflammation has been proposed to contribute to cerebral vascular disease in several ways, which including: (1) the lifelong process of atherogenesis; (2) the acute atherothrombotic event, which causes ischemic necrosis in acute cerebral infarction and the brain damage following ischemic stroke; (3) delayed brain injury after intracerebral hemorrhage; (4) vasospasm after subarachnoid hemorrhage (SAH). CRP, the classical acute phase protein, is among the most extensively studies systemic maker of inflammation.

Serving as a primary defense function of the human body, CRP levels are detected within the limits of 6 to 10 mg/L using a technique known as nephelometry. Of all ethnic backgrounds, the Asian population tend to have the healthiest levels of <0.15 mg/L. Compared to those residing in the Western region of the world, a collective report including over 20,000 able-bodied U.S. partakers had the following results; the 10th, 50th, and 90th percentile values for CRP were 0.40, 1.50, and 6.05 mg/L for men and 0.29, 1.52, and 6.61 mg/L for women, respectively. Participants from Japan had the following outcomes; with the 10th, 50th, and 90th percentile values being <0.03, 0.16, and 0.78 mg/L for Japanese men and <0.03, 0.09, and 0.57 mg/L for Japanese women, respectively (Rifai and Ridker, 2003). Following the trends of most health-related issues, CRP levels are boosted by employing bad habits and curtailed with a healthy diet (Knoops et al, 2004).

Stroke has been a significant health burden for individuals and society. Early identification and intervention can improve the prognosis of this disease. In January 2003, the Centers for Disease Control and Prevention (CDC) and the American Heart Association (AHA) announced a statement for health care providers concerning inflammation markers in CVD and their application to clinical and public health practice (Pearson et al 2003). CRP is among the most studied inflammation biomarkers. An elevated CRP level may predict future ischemic stroke (Arvin et al, 1996). In addition, raised CRP levels may reflect the clinical course of the condition extent of brain infarction and an adverse prognosis (Arvin et al, 1996). High CRP level is associated with stroke severity at admission and is an independent predictor of early seven days mortality after ischemic stroke (Dewan and Rana, 2011).

In whites, the risk was elevated for CRP in the range from 3 to 10 mg/L and even higher for CRP >10 mg/L, whereas in blacks, an association was only seen for CRP >10 mg/L. CRP may not be equally useful in stroke risk assessment in blacks and whites (Evans et al, 2019). Recent findings have demonstrated the important contribution of inflammation to the risk of CVD in individuals with optimally managed low-density lipoprotein cholesterol (LDL-C). In this high-risk population, low hs-CRP (<2 mg/L) appeared to be associated with reduced risk of incident stroke, incident CHD, and CHD death, whereas low LDL-C (<70 mg/dL) was not associated with protective effects (Penson et al, 2018).

Typically, traditional methods for measuring serum CRP are available for use in patients with infectious and inflammatory disorders, which have a detection limit that in the range of 3 to 5 mg/L. That standard is above the concentration observed in most apparently healthy individuals. High sensitivity methods for measurement of CRP (hs-CRP) detect concentrations down to 0.3 mg/L. The assays are necessary for cardiovascular risk stratification, which is based upon discrimination of CRP levels extending below 3 mg/L. This test is sensitized to detect CRP levels less 1 mg/dl (10micgm/ml). Some studies summarize quartiles and tertials of CRP values (lowest <0.55, low-moderate 0.56-1.14, moderate-high 1.15-2.10, highest > 2.11) and the associated probability of a cerebrovascular event. The studies were consistent in their finding of a concentration-dependent relationship between the concentration of CRP and the risk of incident stroke. During the follow-up of 4.9±1.4 years, patients with CRP $<$1 mg/L had 32% reduction compared with that of patients with CRP $\geq$ 1 mg/L. The control of CRP levels appears to be effective for preventing recurrent stroke and TIA in patients with non-cardiogenic ischemic stroke (Kitagawa et al, 2018). CRP level $\geqslant$5mg/l and SNP rs1800947 of the CRP gene were independent risk factors for further adverse CV events among patients with CVD within three years follow-up (Schulz S et al, 2016).

Because stroke risk prediction is based only on conventional risk factors such as blood pressure (BP) it is still not completely reliable, a continued search for predictive markers is of interest. CDC and AHA recommended plasma CRP measurement as an adjunct to use of established risk factors for assessing the risk of coronary heart disease (CHD) in persons with a calculated 10-year CVD risk of 10% to 20% (Pearson et al, 2003). In December 2003, a European study group was formed to review the scientific evidence relating CRP measurement to stroke risk assessment in subjects at risk for cerebrovascular disease and in stroke patients. An evidence-based approach was used to consider the recommendations for applying CRP as a screening tool (Barratt et al, 1999). Several prospective studies have demonstrated that a single, non-fasting measurement of CRP in apparently healthy individuals is a predictor of future fatal and nonfatal cerebrovascular events (Ford et al, 2000; Ridker et al, 2000; Rost et al, 2001).

The relationship between a patient's baseline concentration of CRP and future CVD risk has been consistent in different studies and in most cases has proven to be independent of age, smoking, cholesterol concentrations, BP, and diabetes; the major risk factors evaluated in daily clinical practice. These effects are present in women and men, the elderly and middle aged, smokers and nonsmokers, and those with and without diabetes mellitus. The value of CRP for assessing cerebrovascular risk remains significant after adjustment for the risk factors typically

used in global risk-assessment programs (Ridker et al, 2001). All analyses from these studies provide information about relative risks, we know little or nothing about predictive values and absolute risk for CVD (Horowitz and Beckwith 2000). Those subjects with evidence of inflammation not only had higher BP at study entry but were also more likely to have a greater BP increase over time than those without increased markers of inflammation (Engstrom et al, 2002).

In secondary prevention, the role of CRP is evolving rapidly. Multiple studies demonstrate that CRP concentrations are predictive of future CVD events in stroke patients and are independent of the predictive value of conventional prognostic markers (Di Napoli et al, 2001; Iyigun and Bakirci, 2002). Importantly, plasma CRP concentrations in ischemic stroke patients predict outcome or new vascular events independently of age, stroke severity, and other prognostic factors. However, appropriate clinical cutoff points for CRP in the setting of acute ischemic stroke have not yet been defined, nor has timing of CRP evaluation in relation to the onset of the qualifying event been determined. Although no large study has prospectively assessed the value of CRP for prognostic short-term and long-term stratification of patients with ischemic stroke, many data suggest that CRP might be of value in this group of patients (Muir et al, 1999).

The pathophysiological processes following stroke are quite complex. Understanding of pathophysiology can help improve current clinical conditions of stroke, including diagnosis, assessment, prognosis and therapy. Meanwhile, CRP as a major bio-inflammation marker reflecting relevant events in the ischemic cascade would also be of great use. Though a number of studies related to CRP and stroke have been reported, there is much difficulty in successfully translating this advancement to remarkable application in clinical practice. Many of them are related to the underlying pathophysiology of ischemic stroke. The pathogenesis of stroke is complex, involving multiple mechanisms, in this way, the detection of stroke by use of CRP may require multiple markers to capture simultaneously all processes underlying the ongoing ischemic event. Clearly there is much work needed before promising CRP can be introduced into the clinical practice.

In the following research project, firstly, we are going to investigate whether CRP is a risk factor of stroke using one of the largest epidemiology research in cardiovascular disease in our nation: Jackson Heart Study (JHS) database. Secondly, machine learning is a new cutting-edge tool in medical research. Fewer people used machine learning tools to predict stroke incidence on JHS database. It will be interesting to compare different machine learning models in predicting stroke using JHS database.

Methodology

Dataset

The Jackson Heart Study (JHS) is a single-site, prospective cohort study of the risk factors and causes of heart disease in adult African Americans. 5,301 African Americans, aged 21–94 years, residing in a three-county area surrounding the city of Jackson, MS, USA were recruited and examined (https://www.jacksonheartstudy.org/). Relatives of selected participants were recruited to develop a large, nested family cohort. Participants provided extensive medical and social history, had an array of physical and biochemical measurements and diagnostic procedures, and provided genomic DNA. All the data were collected and stored in a complicated relational database.

The data which we are going to use is the trans-data package of JHS. Briefly, the JHS trans-data package has been constructed (1) to provide educators with tools and resources for training of students and young investigators, and (2) to assist researchers with exploration and discovery of JHS data for submitting a full manuscript proposal. It consists of a de-identified build of the Vanguard Center package released to the collaborative research group partners that has been modified by: creating a set of anonymized IDs, including a ~50% random sample of participants (n=2,653) consenting to all study data usage, windsorizing all continuous variables (1%), truncating low frequency classifications for categorical variables (<5), and setting all dates to the 15th day of the month. Each row of the data includes health related information of each participants.

The transdata set is a very sophisticated relational database which includes 5 major folders and around 25 different tables. The files which we are going to used are stroke incidence file (stroke file) and visit 1 file (v1 file). Visit 1 file includes the data which we had collected from participants at the beginning of this study. From the dictionary of variables, we can see there are demographic, anthropometrics, medications, hypertension, diabetes lipids, biospecimens, renal, respiratory, echocardiogram, electrocardiogram, CT imaging, stroke history, CVD history, healthcare access, psychosocial, life simple 7, nutrition, environmental, genetic and physical activity variables (n=204). Stroke file includes information of the following: date of new incidence of stroke, days or years developing stroke and stroke types. The total number of variables are eight. We merged the v1 data and stroke data by their subject id.

Proposed Methodology

Part I: Survival Analysis

In part I, we investigated whether CRP is a risk factor of stroke incidence using JHS transdata. To reach the above goal, survival analysis techniques were used to determine whether hs-CRP is a risk factor of stroke incidence. Findings from observational studies of time to an event of interest (eg, death, incidence, relapse) were commonly presented in terms of hazard

ratios or rate ratios. These are relative measures of risk (Orsini N et al., 2012). In order to achieve this goal, our approach took the following steps:

1. Applied a systematic method for imputing the missing entries in the dataset.
2. Selected the relevant feature subset based on an automatic procedure.
3. Applied survival models to evaluate the prediction performance.

Kaplan-Meier Survival Analysis

Kaplan-Meier curves and log rank tests - are examples of univariate analysis. They describe the survival according to one factor under investigation, but ignore the impact of any others. Kaplan-Meier curves and log rank tests are useful only when the predictor variable is categorical (e.g.: treatment A vs treatment B; males vs females). They don't work easily for quantitative predictors such as gene expression, weight, or age. The survival probability at any particular time is calculated by the formula given below:

$$S_t = \frac{\text{Number of subjects living at the start} - \text{Number of subjects died}}{\text{Number of subjects living at the start}}$$

Cox Proportional Hazards Regression Analysis

The Cox partial likelihood, shown below, is obtained by using Breslow's estimate of the baseline hazard function, plugging it into the full likelihood and then observing that the result is a product of two factors. The first factor is the partial likelihood shown below, in which the baseline hazard has "canceled out". The second factor is free of the regression coefficients and depends on the data only through the censoring pattern. The effect of covariates estimated by any proportional hazards model can thus be reported as hazard ratios. Cox proportional hazards regression analysis, which works for both quantitative predictor variables and for categorical variables. Furthermore, the Cox regression model extends survival analysis methods to assess simultaneously the effect of several risk factors on survival time. The hazard function for the Cox proportional hazards model has the form:

$$\lambda(t|X_i) = \lambda_0(t)\exp(\beta_1 X_{i1} + \cdots + \beta_p X_{ip}) = \lambda_0(t)\exp(X_i \cdot \beta).$$

This expression gives the hazard function at time *t* for subject *i* with covariate vector (explanatory variables) X_i.

Weibull Survival Analysis

In probability theory and statistics, the Weibull distribution is a continuous probability distribution. It is named after Swedish mathematician Waloddi Weibull, who described it in detail in 1951. The probability density function of a Weibull random variable is:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & x \geq 0, \\ 0 & x < 0, \end{cases}$$

where k > 0 is the shape parameter and λ > 0 is the scale parameter of the distribution. Its complementary cumulative distribution function is a stretched exponential function. The Weibull distribution is related to a number of other probability distributions; in particular, it interpolates between the exponential distribution (k = 1) and the Rayleigh distribution (k = 2 and λ= $\sqrt{2}\sigma$).

Part II: Machine Learning

In part II, we used different models with machine learning techniques to predict stroke incidence by the variables collected in JHS visit 1. Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Within the scientific discipline of machine learning the primary objective was to assemble models that will provide sound predictions. The following modes were used for machine learning.

Logistic regression

Used on target variables that are categorical, a logistic regression is useful in the prediction of probability. The logistic regression model predicts P(Y=1) as a function of X. In order to keep outcomes in the range of 0 and 1 the logistic function is applied.

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

Data in this model were created only with the features and dependent variables while dummy variables were converted for select features. A variety of techniques were employed in order to split the data and handle outliers. The target variable was dropped from the model while the coefficients were obtained and placed within a dataframe. Regression coefficients in this particular model symbolize the transformation in the logit for each unit change in the predictor.

Logistic regression with SGD (stochastic gradient descent)

Stochastic gradient descent is an iterative algorithm that identifies the minimum of a function. In machine learning, this technique is utilized to revise the parameters of models. Considered fast, stochastic gradient descent computes the derivative from training data occurrence and calculates the update. Samples are randomly selected. In this work, 30 fits which were comprised of 5 folds for each of 6 candidates built the model. SGD is an optimization method, while Logistic Regression (LR) is a machine learning algorithm/model. You can think of that a machine learning model defines a loss function, and the optimization method minimizes/maximizes it.

Some machine learning libraries could make users confused about the two concepts. For instance, in *scikit-learn* there is a model called SGD-Classifier which might mislead some user to think that SGD is a classifier. But no, that's a linear classifier optimized by the SGD.

In general, SGD can be used for a wide range of machine learning algorithms, not only LR or linear models. And LR can use other optimizers like L-BFGS, conjugate gradient or Newton-like methods.

Random forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Built on the premise of decision trees, the Random forest algorithm operates as a collaborative. In lay terms, the prediction is based on the tree within the ensemble with the most votes. Within the algorithm, k is randomly selected from the total and then CART (classification and regression trees) is computed. Random forest has processes for balancing errors in data sets where classes are imbalanced.

K-NN

K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions). KNN has been used extensively in statistical estimation and pattern recognition since the beginning of the1970's as a non-parametric technique. Known as a statistical method that doesn't require data to fit a normal distribution. K-NN (k-nearest neighbor algorithm) finds the distances between a query and features in the data. The algorithm sums the distance and the index of the sample to a systematic assembly. It sorts the collection and picks the first k-entries. In case of outcomes, if it is a regression, K-NN returns the mean of the k-labels, if a classification, it returns the mode of the k-labels. A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors. Another approach uses an inverse distance weighted average of the K nearest neighbors. KNN regression uses the same distance functions as KNN classification.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

XGBoost

XGBoost is an open source software library which provides a gradient boosting framework for C++, Java, Python, R, and Julia. It works on Linux, Windows, and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". It runs on a single machine, as well as the distributed processing frameworks Apache Hadoop, Apache Spark, and Apache Flink. It has gained much popularity and attention recently as the algorithm of choice for many winning teams of machine learning competitions.

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

K-folds cross-validation

The K-folds cross-validation procedure evaluates machine learning models based on parameter k. Once a specific value for k is selected, the data is split into as many samples. Used to estimate the skill of a model, this technique divides the data into folds and tested based on the number set for the k parameter. In each iteration, the data is trained on each iteration of the K-Fold process then each score is appended and a mean is obtained to determine the accuracy of the model.

Machine learning Model Comparison using ROC curve

Performance of a classification model is graphically illustrated by the area under curve (AUC) and receiver operating characteristic curve (ROC). The area under the ROC curve (or AUC) is one of the most important metrics for evaluating the performance of classifiers in the medical diagnosis domain (where positive samples are usually small in number) as it considers both sensitivity and specificity, providing a balanced measure for classifier performance. AUC exemplifies degree or measure or separability while the ROC serves as a probability curve. Within this work, the ROC curve is plotted with true positive rate (y-axis) against false negative rate (x-axis).

True Positive Rate:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = 1 - \mathrm{FNR}$$

False Positive Rate:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - \mathrm{TNR}$$

Models considered poor exhibit an AUC near to the 0 while an excellent model is near to 1; considered a good measure of separability. The confusion matrix showcases the predicted and actual class labels from the models. The 2-by-2 matrix from left-to-right are as follows: True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN).

The AUC is used to evaluate the performance of the binary stroke classification task. Essentially, this metric gives an estimate of how accurately the model can answer the question, "is individual A likely to have a stroke in the future?".

Results

Data exploration and cleaning

Visit 1 and stroke data were merged by subject id. Firstly, we checked the missing data of the dataset (figure 1). Then, we excluded those participants with missing information of stroke and hs-CRP. We have 2,472 participants in total where 76 cases experienced a stroke and 2,396 cases were non-stroke (figure 2). Further, we dropped those variables with 90% of missing information.

Figure 1. Missing information in sub-data.

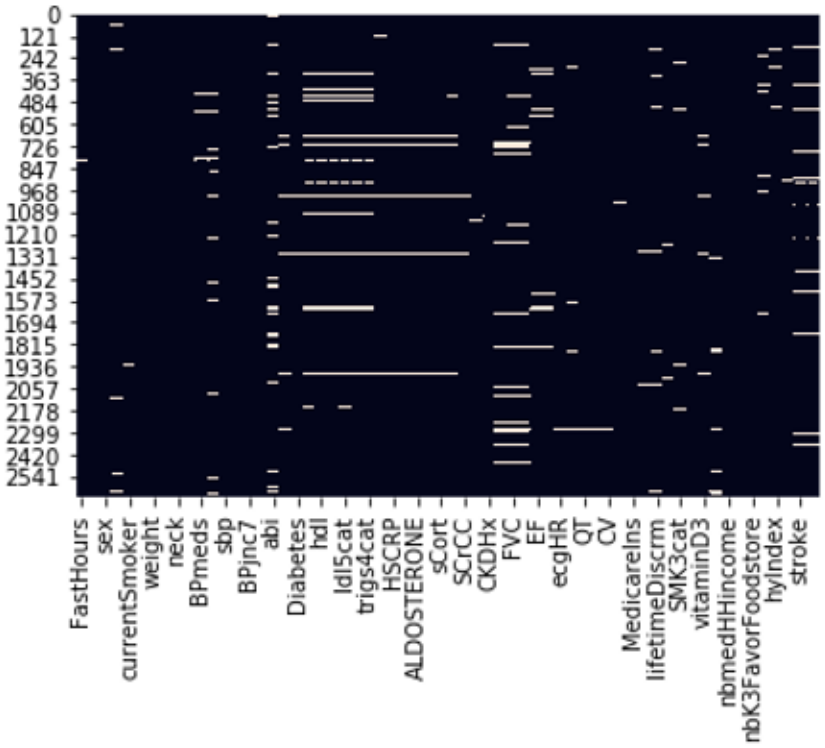
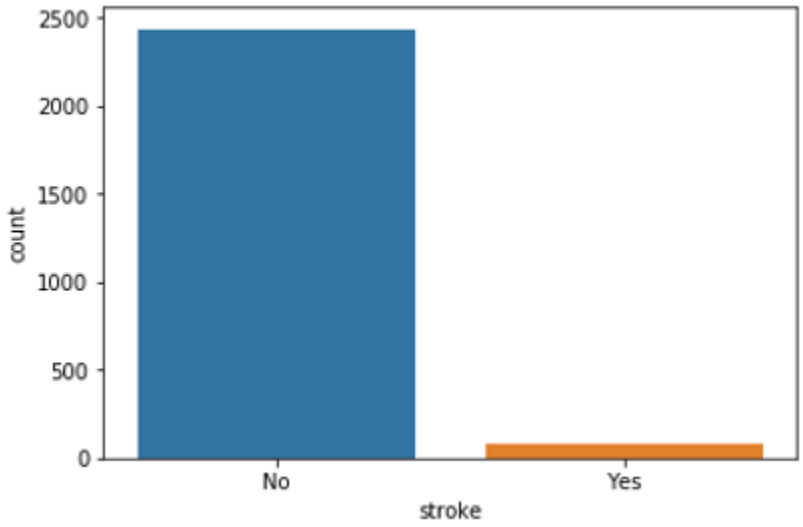
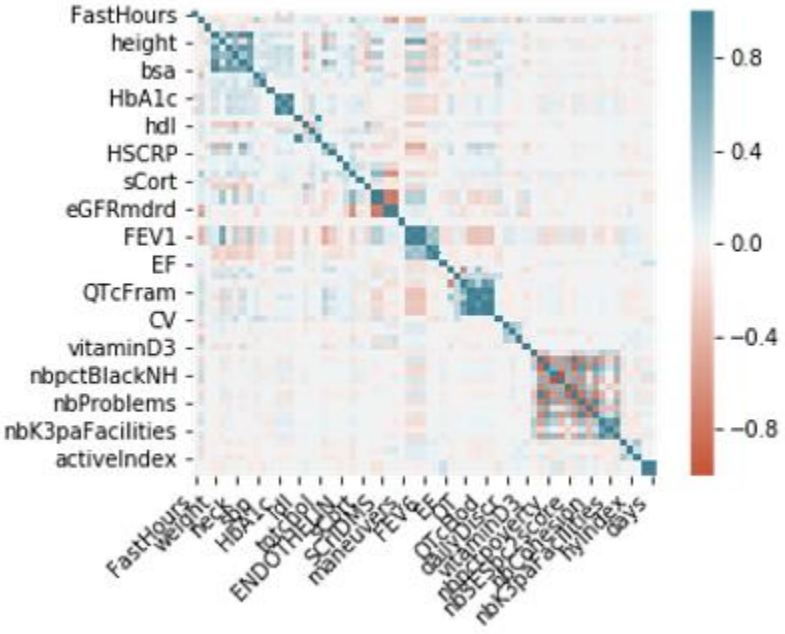


Figure 2. Only around 3% of the study population has stroke incidence



Secondly, correlated matrix and a heat map were created to determine whether or not variables correlated (figure 3). Total number of columns were condensed to 160 after those variables didn't possess predictive power were dropped. Highly correlated variables (> 0.95) were identified and removed from the dataset. In certain models, several issues can arise from variables that are correlated. When independent variables are highly correlated with one or more other independent variables, problems can occur with the interpretation of results. In addition, algorithms such as Random Forests, which is known to detect interaction between features can be misinterpreted when variables are highly correlated. For the distribution of variables, we plotted histograms (figure 4).

Figure 3. Correlated matrix, heat map to check the collinearity of the data



For most of the variables, the distribution looked normal. However, we observed left or right skewness for some variables. Outliers were detected using boxplots with IQR (figure 5). Some imbalanced or disproportional categorical variables were also dropped due to lack of prediction power. For example, there were on 19 cases with anterior major scar variable positive in ECG.

Finally, we used the imputation method where the median replaced missing numeric variables. Categorical data such as sex (male/female) were conerted into numeric variables. For example: 1 for male and 0 female. Stroke status was converted “Yes” to “1”, “No” to “0”.

Figure 4. Histogram showing distribution of the variables

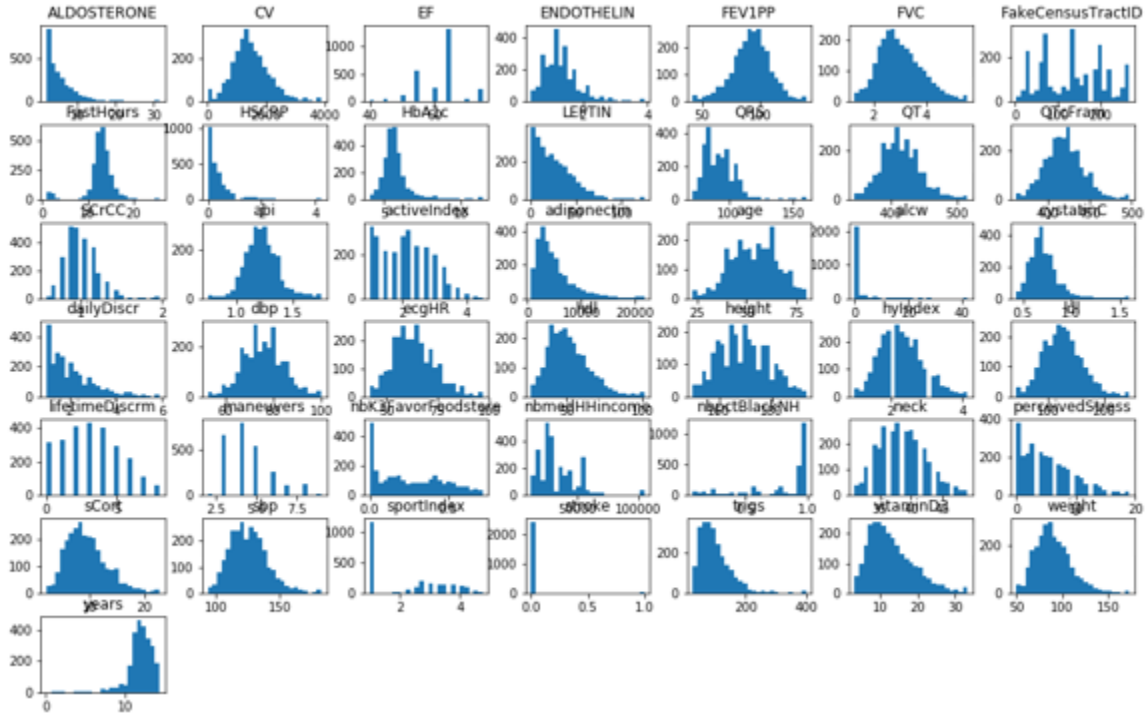
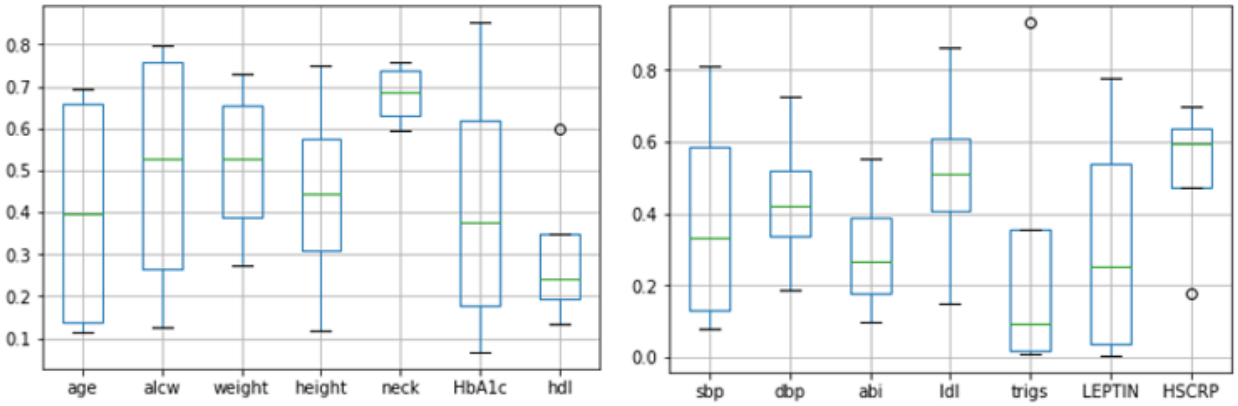


Figure 5. Boxplots with IQR for checking outliers of variables



Demographic information of study population

In JHS, there are 76 incidents of strokes with a mean time to stroke of 11.81 years. Baseline characteristics of the study population were summarized in Table 1. Traditional stroke risk factors such as hypertension, and current smoking were more prevalent in stroke cases compared with non-cases. Individuals with incidents of stroke had significantly higher systolic blood pressure, HbA1c and ldl levels than non-cases. There was no significant difference in hdl level between cases and non-cases. In addition to these differences in traditional risk factors, the weighted mean levels of CRP (0.52 vs 0.49mg/L) has no significant between stroke cases and non-cases

Table 1. Demographic information of study population.

	Stroke (n=76)	**Non-cases (n=2396)**
Age, y	62.37	53.86 ***
Female, %	56.58	62.52***
Current smoker, %	19.74	11.22
Weight, kg	89.36	91.58
Waist circumference, cm	101.99	100.72
BMI	31.92	31.19
Systolic blood pressure, mm Hg	132.68	125.75*
Diastolic blood pressure, mm Hg	75.68	75.95
Hypertension, %	76.32	51.62***
HbA1c,	6.39	5.85***
Total cholesterol, mg/dL	208.59	198.59
Triglycerides, mg/dL	111.96	103.62
HDL-C, mg/d	51.38	51.66
LDL-C, mg/dL	133.72	126.12*
hs-CRP, mg/L	0.52	0.49

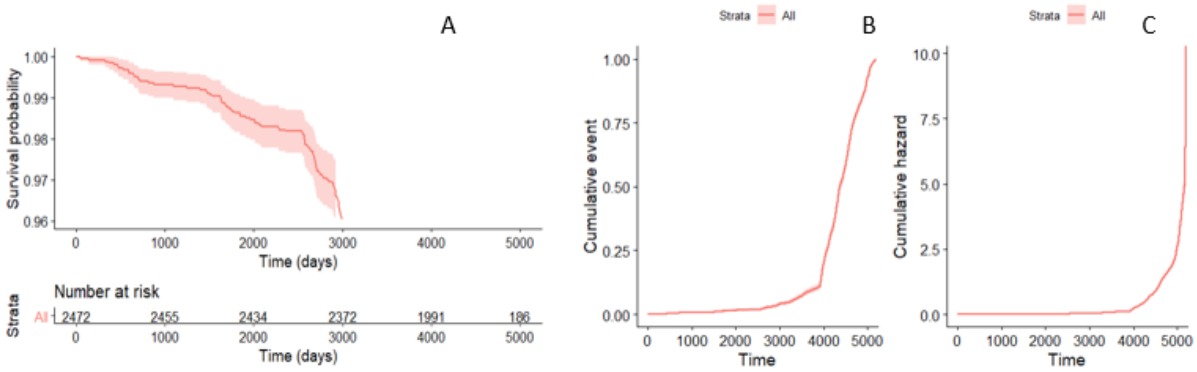
Part I: Survival Analysis

Survival analysis for hs-CRP and stroke incidence

A subset was created with the variables of stroke, years, days, age, sex, current smoker, weight, height, waist, BMI, systolic blood pressure, diastolic blood pressure, hypertension, HbA1c, LDL, HDL, triglycerides, total cholesterol, Afib, and stroke history. We excluded the participants missing information on stroke incidence and hs-CRP. Then, we had 2472 participants. Among them, 76 were stroke and 2,396 were non-cases. Missing information were replaced with the median.

In order to investigate the correlation of time passing and developing stroke under the influence of stroke risk factors, survival analyses were conducted on the sub-data set as we described before. Among the three most popular survival analyses models, Kaplan-Meier curves and log rank tests - were examples of univariate analysis. They describe the survival according to one factor under investigation, but ignore the impact of any others. This is the limitation of the model. Therefore, we just used it as pilot. Running Kaplan-Meier curves (figure 6), we felt that we could use survival analysis models to evaluate some potential new risk factors of stroke.

Figure 6. A: Kaplan-Meier curves of survival probability with days; B: Kaplan-Meier curves of cumulative events; C: Kaplan-Meier curves of cumulative of hazard.

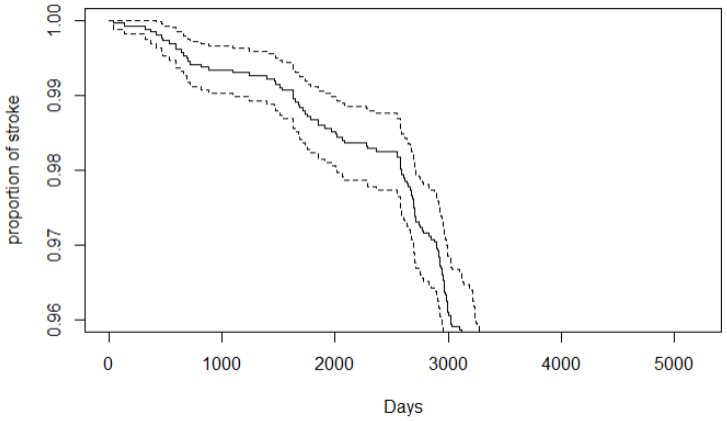


One of our goals in part I is to evaluate whether hs-CRP is a new risk factor to predict stroke. Because hs-CRP has wider variation (sd compared to the mean), we could not find significance using the original values of hs-CRP. So did the quartiles value of hs-CRP. However, using 1 mg/dL as cut point of hs-CRP, hs-CRP had reached statistical significance ($p < 0.05$) in our cox survival analysis model by likelihood ratio test, wald test and score (log rank) test. Summary of Cox model was presenting in table 2. Also, Cox survival curves showed a typical survival curve with 95% confidence interval (figure 6). Similar results were confirmed by Weibull survival analysis model (p=0.05). All statistical analyses were performed using R programming software, version 3.6.

Table 2. Summary of Cox model

	beta	**HR (95% CI for HR)**	**p.value**
CRP, 1mg/dL cut point	0.816	2.261(1.146-4.464)	<0.05
Sex, % of male	0.321	1.378(0.800-2.375)	0.248
Age, y	0.074	1.076(1.047-1.107)	<0.001
Current Smoker, % of yes	0.984	2.676(1.417-5.055)	<0.01
Systolic Bp, mmHg	0.008	1.008(0.992-1.023)	0.331
Hypertension, % of yes	0.288	1.333(0.713-2.494)	0.368
HbA1c, mg/dL	0.195	1.216(1.012-1.459)	<0.05
LDL, mg/dL	0.009	1.010(0.992-1.027)	0.296
Total cholesterol, mg/dL	0.002	0.998(0.981-1.014)	0.769

Figure 6. Cox survival curve



Part II: Machine learning models for predicting stroke incidence

In part II, we built several models to predict stroke using machine learning approaches. In order to perform machine learning on JHS data set, the most important steps are data preprocessing phase. During this phase, data scientist can determine which variables will work better than others during the creation and implementation of the models. As in part I, we merged two datasets (visit 1 and stroke). Variables missing more than 90% of their information were dropped. According to the correlation heatmap, highly correlated variables (>0.95), were removed based on the clinical meaning. Finally, missing information on stroke status (target variable) were moved from data base. The final dimension of the study data frame was 2,543 rows and 62 columns. The 62 variables were fasting hours, age, gender, alcohol drink during past 12 months, current smoker, ever smoker, weight, height, neck circumference, obesity category, BP mediations, diuretic medications, systolic blood pressure, diastolic blood pressure, blood pressure in JNC7, hypertension, ankle brachial index, HbA1c, diabetes, LDL, HDL, triglycerides, simple life 7, leptin, hs-CRP, aldosterone, cystatin C, cortisol, adiponectin, endothelin, serum creatine level, dialysis ever, chronic kidney disease, successful spirometer maneuvers, force vital capacity, force expiatory volume , ejection fraction, ejection fraction categorization, ECG heart rate, QRS interval, QT interval, QT interval adjust by Framingham, Cornel voltages, private insurance, Medicare insurance , daily discrimination, life time discrimination, perceived stress, smoking category, BMI category, D3, Fake Census Tract ID. Neighborhood median house hold income, % of non-Hispanic Black in their neighborhood, favorable food store in neighborhood, sports index, home yard index, activity index, year, stroke. Those variables cover areas of demographic, anthropometrics, medications, clinical condition, lipid profiles, renal function, respiratory function, echocardiogram, ECG, CT, medical history, psychosocial variables, life's simple 7, nutrition, environment, and physical activity.

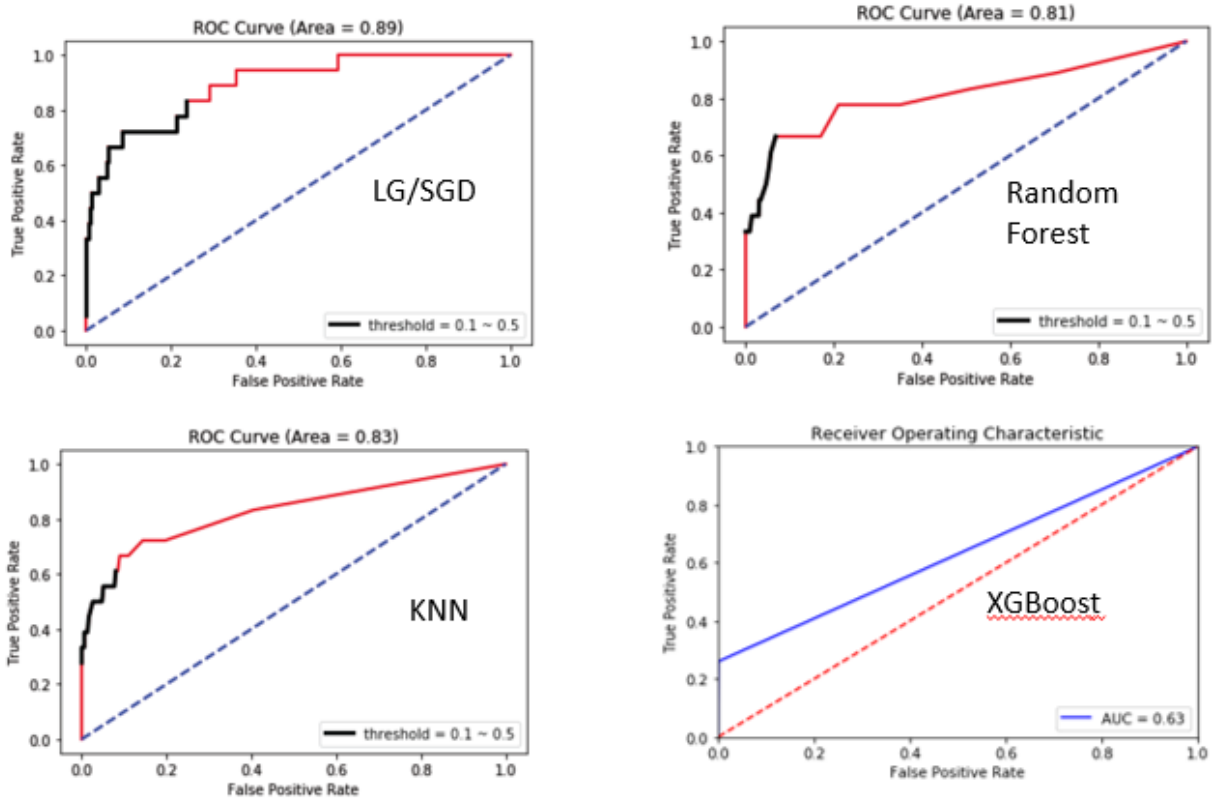
Based on the models' output, table 3 highlighted the comparison of the SGD Logistic Regression, Random Forest, and K-NN models; the model will display the Area Under Curve/Receiver Operating Characteristic scores with the output from the confusion matrix.

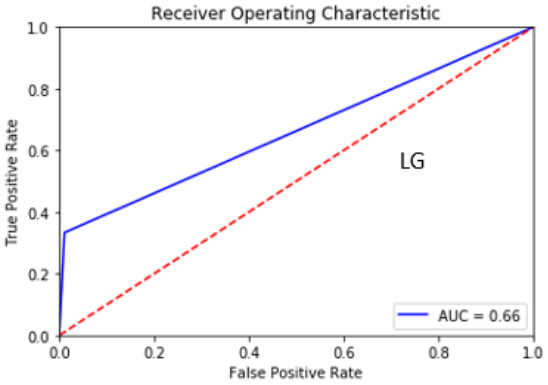
Table 3. Characteristic of machine learning models

Model	ROC AUC Training	ROC AUC Test	TP	FN	TN	FP	Accuracy Training	Accuracy Test
LG w/SGD	0.9202860349636080	0.8931028551771586	646	0	18	0	0.9693313222724987	0.9728915662650602
Radom Forest	1.0	0.8126504987960095	646	0	17	1	1.0	0.9743975903614458
K-NN	0.9573540915583973	0.8279583763329894	646	0	18	0	0.9693313222724987	0.9728915662650602
LG	0.7118180736004354	0.6612487100103198	639	6	12	6	0.9798893916540975	0.9713855421686747
XGBoost	0.9553571428571428	0.6298486161374178	852	6	17	1	0.992	0.9795

TP: true positive, FN: false negative, TN: true negative, FP: false positive; LG: logistic regression.

Figure 7. ROC curves of the models.





Think of XGBoost as gradient boosting on 'steroids' (well it is called 'Extreme Gradient Boosting' for a reason!). It is a perfect combination of software and hardware optimization techniques to yield superior results using least computing resources in the shortest amount of time. But we saw that even though the accuracy was 98%, the ROC AUC was just 62%. According to the findings from the models, the Random Forest algorithm performed best on the training set for the ROC AUC score and the accuracy scores for the test and train sets. On the test set for the ROC AUC test the Logistic Regression w/SGD performed best.

When something is concluded true and it is actually false, we have a false positive or type I error. On the other hand, when something is false and it is actually true, we have a false negative or type II error. All medical tests can be resulted in false positive and false negative errors. Since medical tests can't be absolutely true, false positive and false negative are two problems we have to deal with. A false positive can lead to unnecessary treatment and a false negative can lead to a false diagnostic, which is very serious since a disease has been ignored. So, our team thinks we should choose the model with least false negative rate. For this point of view, Logistic Regression w/SGD are proved again to be the best.

Summary and Future Works

Roughly 700,000 strokes occur each year in the United States. Stroke is the third leading cause of death and the leading cause of neurologic disability. Despite the incidence of stroke increase with age. It is highest in the elderly. One third of strokes occurs in people younger than 65 years. Medication such as aspirin, antihypertension, and statins are used in prevention of stroke. However, there is no national guideline in prevention of stroke. Inflammation has been considered as an important role in cerebral vascular diseases (CVD) as well as coronary heart disease (CHD). CRP is one of the major inflammation markers. Different levels of CRP are associated with increased risk for stroke. However, any form of tissue injury, infection, inflammation and stress are associated with increasing of plasma CRP values (Kushner et al, 2006). A few studies suggest CRP levels are different among different ethnical groups. Asian population has the lowest levels of CRP while African-Americans led the pool of contributors. African-Americans. JHS population is African-American only. The quartiles of hs-CRP in JHS were: minimum (0 mg/dL), 1st quartile (0.1050 mg/dL), medium (0.2610 mg/dL), 3rd quartile (0.553 mg/dL), maximum (10.7060 mg/dL). hs-CRP has big variation with mean (0.4880

mg/dL) Using quartile and mean±sd for survival analysis, we could not conclude the hs-CRP played a role in developing stroke incidence. However, using 1mg/dL as cut off point, we found that hs-CRP had impact on developing of stroke incidence by Cox and Weibull survival analysis ($p<0.05$).

In health care, global interest in the potential of machine learning has increased; for example, a deep learning algorithm has shown high accuracy in predicting stroke (Khosla A, 2010). Most traditional prediction models have adopted features (risk factors) that are verified by clinical trials or selected manually by medical experts. The total variable under investigation are usually less than 20. In contrary, machine learning algorithms are capable of identifying features highly related to stroke occurrence efficiently from the huge set of features; therefore, we believe machine learning can be used to: (i) improve the prediction accuracy of stroke risk and (ii) discover new risk factors.

Selecting relevant features is crucial for building an accurate model of clinical data. For example, the JHS dataset has a large number of attributes, ranging from demographic information and clinical history to biomedical and physical measurements. However, only a small subset of attributes is highly relevant to stroke prediction. The traditional approach to stroke prediction has been to use manually selected features based on risk factors analyzed by medical and clinical studies. Instead of manually selecting features, we evaluated 5 machine learning-based algorithms for selecting features automatically: (1) logistic regression, (2) logistic regression with SGD, (3) random forest, (4) KNN and (5) XGBoost.

First, we evaluated the performance of our prediction algorithms based on the area under the ROC curve. We found logistic regression with SGD rank the best of ROC AUC test. In addition, logistic regression with SGD model had 0 case of false positive and 0 case of false negative. Finally, The accuracy rate for all 5 models were very similar (around 97%). We concluded that logistic regression with SGD was the best model among the 5 machine learning models.

Because we only practice our models on half of the real dataset of JHS, it would be interesting if we can repeat our result in full data of JHS. Also, we like to weight the importance of all select features which can provide a guideline for physician in clinical practice to prevent stroke occurences.

Reference:

1. Arvin B, Neville LF, Barone FC, Feuerstein CZ. Role of inflammation and cytokines in brain injury. Neurosci Biobehav Rev 1996; 20:445-52.
2. Barratt A, Irwig L, Glasziou P, Cumming RG, Raffle A, Hicks N, Gray JA, Guyatt GH. Users' guides to the medical literature: XVII. How to use guidelines and recommendations about screening. Evidence-Based Medicine Working Group. JAMA. 1999; 281:2029–2034.
3. Black S, Kushner I, Samols D. C-reactive Protein. J Biol Chem. 2004 Nov 19;279(47):48487-90
4. Dewan KR, Rana PVS. C–reactive Protein and Early Mortality in Acute Ischemic Stroke. Kathmandu University Medical Journal. 2011;9(4):252-55.
5. Di Napoli M, Papa F. Inflammation, statins, and outcome after ischemic stroke. Stroke. 2001; 32:2446–2447.
6. Di Napoli M, Schwaninger M, Cappelli R, Ceccarelli E, Di Gianfilippo G, Donati C, Emsley HC, Forconi S, Hopkins SJ, Masotti L, Muir KW, Paciucci A, Papa F, Roncacci S, Sander D, Sander K, Smith CJ, Stefanini A, Weber D. Evaluation of C-reactive protein measurement for assessing the risk and prognosis in ischemic stroke: a statement for health care professionals from the CRP Pooling Project members. Stroke. 2005 Jun;36(6):1316-29.
7. Di Napoli M, Papa F, Bocola V. C-reactive protein in ischemic stroke: An independent prognostic factor. Stroke 2001; 32:917-924.
8. Engstrom G, Lind P, Hedblad B, Stavenow L, Janzon L, Lindgarde F. Long-Term effects of inflammation-sensitive plasma proteins and systolic blood pressure on incidence of stroke. Stroke. 2002;33:2744–2749.
9. Evans CR, Long DL, Howard G, McClure LA, Zakai NA, Jenny NS, Kissela BM, Safford MM, Howard VJ, Cushman M. C-reactive protein and stroke risk in blacks and whites: The Reasons for Geographic and Racial Differences in Stroke cohort. Am Heart J. 2019 Aug 12; 217:94-100.
10. Ford ES, Giles WH. Serum C-reactive protein and self-reported stroke: findings from the Third National Health and Nutrition Examination Survey. Arterioscler Thromb Vasc Biol. 2000; 20:1052–1056.
11. Horowitz GL, Beckwith BA. C-reactive protein in the prediction of cardiovascular disease. N Engl J Med. 2000; 343:512–513.
12. Iyigun I, Bakirci Y. Plasma concentrations of C-reactive protein and fibrinogen in ischemic stroke. J Int Med Res. 2002; 30:591–596.
13. Khosla A, Cao Y, Lin CCY, Chiu HK, Hu J. An integrated machine learning approach to stroke prediction. In *KDD*, 2010.
14. Kitagawa K, Hosomi N, Nagai Y, Kagimura T, Ohtsuki T, Maruyama H, Origasa H, Minematsu K, Uchiyama S, Nakamura M, Matsumoto M; J-STARS collaborators. Cumulative Effects of LDL Cholesterol and CRP Levels on Recurrent Stroke and TIA. J Atheroscler Thromb. 2019 May 1;26(5):432-441.

15. Knoops KTB, de Groot LCPGM, Kromhout D, Perrin A-E, Moreiras-Varela O, Menotti A, van Staveren WA. Mediterranean diet, lifestyle factors, and 10-year mortality in elderly European men and women: the HALE project. JAMA 2004; 292:1433–1439.
16. Kocatürk M, Kocatürk Ö. Assessment of relationship between C-reactive protein to albumin ratio and 90-day mortality in patients with acute ischemic stroke.
17. Kushner I, Rzewnicki D, Samols D. What does minor elevation of C-reactive protein signify? Am J Med 2006; 119: 166.
18. Muir KW, Weir CJ, Alwan W, Squire IB, Lees KR. C-reactive protein and outcome after ischemic stroke. Stroke. 1999; 30:981–985.
19. N Orsini, A Wolk, M Bottai. Evaluating percentiles of survival. Epidemiology, 2012:23(5): 770–771.
20. Pearson TA, Mensah GA, Alexander RW, Anderson JL, Cannon RO, III, Criqui M, Fadl YY, Fortmann SP, Hong Y, Myers GL, Rifai N, Smith SC, Jr., Taubert K, Tracy RP, Vinicor F. Markers of inflammation and cardiovascular disease: application to clinical and public health practice: a statement for healthcare professionals from the Centers for Disease Control and Prevention and the Am Heart Association. Circulation. 2003; 107:499 –511.
21. Penson PE, Long DL, Howard G, Toth PP, Muntner P, Howard VJ, Safford MM, Jones SR, Martin SS, Mazidi M, Catapano AL, Banach M. Associations between very low concentrations of low density lipoprotein cholesterol, high sensitivity C-reactive protein, and health outcomes in the Reasons for Geographical and Racial Differences in Stroke (REGARDS) study. Eur Heart J. 2018 Oct 21;39(40):3641-3653.
22. Ridker PM, Hennekens CH, Buring JE, Rifai N. C-reactive protein and other markers of inflammation in the prediction of cardiovascular disease in women. N Engl J Med. 2000; 342:836–843.
23. Ridker PM. High-sensitivity C-reactive protein: potential adjunct for global risk assessment in the primary prevention of cardiovascular disease. Circulation. 2001; 103:1813–1818.
24. Rifai N, Ridker PM. Population distributions of C-reactive protein in apparently healthy men and women in the United States: implication for clinical interpretation. Clin Chem 2003; 49:666–669.
25. Rost NS, Wolf PA, Kase CS, Kelly-Hayes M, Silbershatz H, Massaro JM, D'Agostino RB, Franzblau C, Wilson PW. Plasma concentration of C-reactive protein and risk of ischemic stroke and transient ischemic attack: the Framingham study. Stroke. 2001; 32:2575–2579.
26. Schulz S, Lüdike H, Lierath M, Schlitt A, Werdan K, Hofmann B, Gläser C, Schaller HG, Reichert S. C-reactive protein levels and genetic variants of CRP as prognostic markers for combined cardiovascular endpoint (cardiovascular death, death from stroke, myocardial infarction, and stroke/TIA). Cytokine. 2016 Dec; 88:71-76.

Appendix 1:

````
title: "Is c-reative protein a risk factor of stroke in JHs
transdata?"
       Survival Analysis Part
author: "Jun Pan", "Ritesh Lohiya", "Brian Liles"

date: "11/6/2019"
output: html_document
---

```{r}
#Set working environment, following package may need to be used in
the following data wrangling, cleaning, exploration.
x<-c("broom", "car", "caret", "corrplot", "data.table",  "dplyr",
"geoR", "ggplot2", "grid", "gridExtra", "kableExtra" , "knitr",
"MASS", "naniar", "nortest", "reshape2","psych", "shiny",
"survival", "survminer","survMisc", "reshape2", "testthat",
"tidyverse")
lapply(x, require, character.only = TRUE)
```



##Merge data relational data base: one has stroke incidence, one has
visit 1 variables
```{r}
#import the dataset with stroke incidence
stroke <- read.csv("C:/Users/johnp/Desktop/data/Events/1-
data/CSV/incevtstroke.csv")
```

```{r}
#Overview stroke dataset
head(stroke)
```

```{r}
#import visit 1 data
v1 <- read.csv("C:/Users/johnp/Desktop/data/Analysis Data/1-
data/CSV/analysis1.csv")
```



```{r}
head(v1)
```



```{r}
#Combine two files with primary key"subjid" in both files
stroke_comb <-left_join(stroke, v1, by="subjid")
```
````

##Exclusion, missing info of stroke status and HSCRP
```{r}
#Check missing information on stroke status
summary(stroke_comb$stroke)
```
```{r}
#remove those rows with missing stroke status
stroke_clean <- filter(stroke_comb, (stroke=="Yes"|stroke=="No"))
```

```{r}
#remove missing info of CRP
CRP_clean <- filter(stroke_clean, (HSCRP != "NA"))
```

```{r}
#Check dimension of the cleaned dataset
dim(CRP_clean)
```

##Check data cleaning result of hs-CRP
```{r}
summary(CRP_clean$HSCRP)
```
No missing data of HSCRP

```{r}
summary(CRP_clean$stroke)
```
Ok, no missing info of HSCRP

```{r}
#Get more concised dataset based on literature review.  In here, we picked potential risk factors of CVD and stroke.
ndata <- dplyr::select(CRP_clean, subjid, stroke, years, days, age, sex, currentSmoker, weight, height, waist, BMI,sbp, dbp, HTN, HbA1c, ldl, hdl, trigs,totchol,HSCRP, Afib, strokeHx)
```

```{r}
#Overview the subdataset
glimpse(ndata)
```

```{r}
#overview th missing data
vis_miss(ndata)
```

```{r}
```

```
#check the detailed missing information
sapply(ndata, function(x) sum(is.na(x))) %>% kable() %>%
kable_styling()
```
#Replace missing value with medium
```{r}
ndata$weight[is.na(ndata$weight)] <- median(ndata$weight,
na.rm=TRUE)
ndata$waist[is.na(ndata$waist)] <- median(ndata$waist, na.rm=TRUE)
ndata$BMI[is.na(ndata$BMI)] <- median(ndata$BMI, na.rm=TRUE)
ndata$sbp[is.na(ndata$sbp)] <- median(ndata$sbp, na.rm=TRUE)
ndata$dbp[is.na(ndata$dbp)] <- median(ndata$dbp, na.rm=TRUE)
ndata$HbA1c[is.na(ndata$HbA1c)] <- median(ndata$HbA1c, na.rm=TRUE)
ndata$ldl[is.na(ndata$ldl)] <- median(ndata$ldl, na.rm=TRUE)
ndata$hdl[is.na(ndata$hdl)] <- median(ndata$hdl, na.rm=TRUE)
ndata$trigs[is.na(ndata$trigs)] <- median(ndata$trigs, na.rm=TRUE)
ndata$totchol[is.na(ndata$totchol)] <- median(ndata$totchol,
na.rm=TRUE)
```

```{r}
#double check after replace the missing data
sapply(ndata, function(x) sum(is.na(x))) %>% kable() %>%
kable_styling()
```


```{r}
head(ndata)
```

#Replace yes = 1; no = 0; for stroke status, current smoker, and
Afib
#Actually, we found only 5 answered yes for Afib and 3 missing
information. So later, we drop this columne.
```{r}
ndata <- ndata %>%
      mutate(stroke = ifelse(stroke == "No",0,1))
ndata <- ndata %>%
      mutate(currentSmoker = ifelse(currentSmoker == "No",0,1))
ndata <- ndata %>%
      mutate(Afib = ifelse(Afib == "No",0,1))
```

```{r}
#Overview the dataset after replace the categorical data into number
head(ndata)
```
#Cox model 1
```{r}
cxmod <- coxph(Surv(days, stroke) ~ HSCRP, data = ndata)
coef(cxmod)
```

```{r}
summary(cxmod)
```
```{r}
anova(cxmod)
```
```{r}
plot(survfit(cxmod), ylim=c(0.96, 1),xlab="Days",
ylab="proportion of stroke")
```

#cox model 2
```{r}
cxmod2 <- coxph(Surv(days, stroke) ~ HSCRP + sex + age, data =
ndata)
coef(cxmod1)
```
```{r}
summary(cxmod2)
```

```{r}
Anova(cxmod2)
```
```{r}
plot(survfit(cxmod2), ylim=c(0.96, 1),xlab="Days",
ylab="proportion of stroke")
```

#cox model 3
```{r}
cxmod3 <- coxph(Surv(days, stroke) ~ HSCRP +sex + age +
currentSmoker + weight + height + waist + BMI + sbp + dbp + HTN +
HbA1c + ldl + hdl +trigs + totchol + Afib + strokeHx, data = ndata)
coef(cxmod1)
```
```{r}
summary(cxmod3)
```

```{r}
anova(cxmod3)
```

```{r}
plot(survfit(cxmod3), ylim=c(0.96, 1),xlab="Days",
ylab="proportion of stroke")
```
#Covert HSCRP into quatile
#using 1 mg/dL of hs-CRP as cut point
```{r}
```

```
mdata <- mutate(ndata,
       CRPqt = ifelse(HSCRP > 1, "Yes", "No"))
```

```{r}
head(mdata)
```

```{r}
cxmod4 <- coxph(Surv(days, stroke) ~ CRPqt + sex + age +
currentSmoker +  sbp +
                  HTN + HbA1c + ldl + totchol , data = mdata)
coef(cxmod4)
```
```{r}
summary(cxmod4)
```
```{r}
anova(cxmod4)
```

```{r}
plot(survfit(cxmod4), ylim=c(0.96, 1),xlab="Days",
ylab="proportion of stroke")
```

#Weibull model

```{r}
wbmod <- survreg(Surv(days,stroke) ~ HSCRP,data=mdata)
wbmod
```

```{r}
summary(wbmod)
```
```{r}
anova(wbmod)
```

```{r}
wbmod2 <- survreg(Surv(days,stroke) ~ CRPqt ,data=mdata)
wbmod2
```

```{r}
summary(wbmod2)
```

```
```{r}
anova(wbmod2)
```

```{r}
wbmod3 <- survreg(Surv(years,stroke) ~ CRPqt + sex + age +
currentSmoker + weight + height + waist + BMI + sbp + dbp + HTN +
HbA1c + ldl + hdl +trigs + totchol + Afib + strokeHx,,data=mdata)
wbmod3
```
```{r}
summary(wbmod3)
```

#Regular Kaplan-Meier plot
#reference:https://rpubs.com/alecri/258589
```{r}
fit_km <- survfit(Surv(days, stroke) ~ 1, data = mdata)
print(fit_km, print.rmean = TRUE)
```

```{r}
plot(survfit(cxmod4), ylim=c(0.96, 1),xlab="Days",
ylab="proportion of stroke")
```

```{r}
ggsurvplot(fit_km, risk.table = TRUE, ylim=c(0.96, 1), xlab = "Time
(days)", censor = T)
```

```{r}
glist <- list(
  ggsurvplot(fit_km, fun = "event", main = "Cumulative proportion"),
  ggsurvplot(fit_km, fun = "cumhaz",  main = "Cumulative Hazard")
 )
arrange_ggsurvplots(glist, print = TRUE, ncol = 2, nrow = 1)
```
```

Appendix 2:

Machine learning approach for predicting the stroke

By Jun Pan, Ritesh Lohiya and Brian Liles

```
----
# Data Preprocessing
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from datetime import datetime
%matplotlib inline
from IPython.core.display import HTML
pd.set_option("display.max_columns",210)
import warnings
warnings.filterwarnings('ignore')
import urllib
import json
----
----
analysis1 =
pd.read_csv('C:/Users/rites/Downloads/TRANS/data/Analysis Data/1-
data/CSV/analysis1.csv')
----
----
analysis1.describe()
----
----
#analysis1
----
----
stroke = pd.read_csv('C:/Users/rites/Downloads/TRANS/data/Analysis
Data/1-data/CSV/incevtstroke.csv')
----
----
stroke
----
----
stroke.describe()
----
----
df = pd.merge(analysis1, stroke, on='subjid')
----
----
#df
----
----
df.columns
----
----
```

```
df.dtypes
----
----
df.shape
----
----
df.describe()
----
----
# Now we will drop all the columns that have 90% values as NaN
NaN_per = len(df) * .9
df1 = df.dropna(thresh=NaN_per,axis=1)
# After this we are left with only 179 columns.
----
----
#df1
----




----
#Dropping the cloumns that dont have predictive power or are
correlated with other variables
df1.drop(['subjid', 'visit',  'VisitDate',
'nutrition3cat','darkgrnVeg', 'eggs', 'fish',
'frs_atpiii_tenyrrisk','DaysFromV1', 'YearsFromV1', 'ARIC',
'recruit','male','brthyr', 'brthmo',
'alc','medAcct','occupation','HSgrad','PA3cat','edu3cat','idealHealt
hSMK','idealHealthBMI', 'idealHealthPA', 'idealHealthNutrition',
'totChol3cat', 'idealHealthChol', 'BP3cat', 'idealHealthBP',
'glucose3cat', 'idealHealthDM','selfdiabetes',     'FPG3cat',
'HbA1c3cat', 'HbA1cIFCC3cat', 'fastingInsulin', 'diab3cat','V1date',
'date', 'year', 'Status'] ,1, inplace=True)
----
----
df1.shape
----
----
#Lets see correlation
corr = df1.corr()
----
----
corr
----
----
corr.to_csv('out.csv')
----
----
corr = df1.corr()
ax = sns.heatmap(
    corr,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(20, 220, n=200),
```

```
    square=True
)
ax.set_xticklabels(
    ax.get_xticklabels(),
    rotation=45,
    horizontalalignment='right'
);
----
----
#Drop the variables that are highly correlated
# Create correlation matrix
corr_matrix = df1.corr().abs()
# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(np.bool))
# Find features with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] >
0.7)]
# Drop features
df1.drop(to_drop, axis=1, inplace=True)
----
----
#df1
----
----
#Lets see variables with imbalanced data
----
----
df1.MajorScarAnt.value_counts().head(5)
# Can drop this column as it has mostly Absent
----
----
df1.ConductionDefect.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.MinorScarAnt.value_counts().head(5)
# Can drop this column as it has mostly Absent
----
----
df1.RepolarAnt.value_counts().head(5)
# Can drop this column as it has mostly Absent
----
----
df1.MIant.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.MajorScarPost.value_counts().head(5)
# Can drop this column as it has mostly Absent
----
----
df1.MinorScarPost.value_counts().head(5)
# Can drop this column as it has mostly Absent
```

```
----
----
df1.RepolarPost.value_counts().head(5)
# Can drop this column as it has mostly Absent
----
----
df1.MIpost.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.MajorScarAntLat.value_counts().head(5)
# Can drop this column as it has mostly Absent
----
----
df1.MinorScarAntLat.value_counts().head(5)
# Can drop this column as it has mostly Absent
----
----
df1.RepolarAntLat.value_counts().head(5)
# Can drop this column as it has mostly Absent
----
----
df1.MIAntLat.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.MIecg.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.Afib.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.Aflutter.value_counts().head(5)
# Can drop this column as it has all No
----
----
df1.LVHcv.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.speechLossEver.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.visionLossEver.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.doubleVisionEver.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
```

```
df1.numbnessEver.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.paralysisEver.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.dizzynessEver.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.strokeHx.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.MIHx.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.CardiacProcHx.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.CHDHx.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.CarotidAngioHx.value_counts().head(5)
# Can drop this column as it has all No
----
----
df1.CVDHx.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.PrivateIns.value_counts().head(5)
# Cant drop this column as it looks it has predictive power
----
----
df1.MedicaidIns.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.MedicareIns.value_counts().head(5)
# Cant drop this column as it looks it has predictive power
----
----
df1.VAIns.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.InsuranceType.value_counts().head(5)
# Can drop this column as it does not have predictive power
```

```
---
---
df1.Insured.value_counts().head(5)
# Can drop this column as it has mostly No
---
---
df1.PublicIns.value_counts().head(5)
# Can drop this column as it has mostly No
---
---
df1.PrivatePublicIns.value_counts().head(5)
# Can drop this column as it does not have predictive power
---
---
#    df1.nutrition3cat.value_counts().head(5)
# Cant drop this column as it looks it has predictive power
---
---
df1.SMK3cat.value_counts().head(5)
# Cant drop this column as it looks it has predictive power
---
---
df1.BMI3cat.value_counts().head(5)
# Cant drop this column as it looks it has predictive power
---
---
df1.ageIneligible.value_counts().head(5)
# Can drop this column as it has all No
---
---
df1.DMmedsOral.value_counts().head(5)
# Can drop this column as it has mostly No
---
---
df1.DMmedsIns.value_counts().head(5)
# Can drop this column as it has mostly No
---
---
df1.DMmeds.value_counts().head(5)
# Can drop this column as it has mostly No
---
---
df1.statinMeds.value_counts().head(5)
# Can drop this column as it has mostly No
---
---
df1.hrtMeds.value_counts().head(5)
# Can drop this column as it has mostly No
---
---
df1.betaBlkMeds.value_counts().head(5)
# Can drop this column as it has mostly No
---
---
```

```
df1.calBlkMeds.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.diureticMeds.value_counts().head(5)
# Cant drop this column as it looks like it has predictive power
----
----
df1.antiArythMeds.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.BPmedsSelf.value_counts().head(5)
# Can be dropped as it seems its correlated to BPmeds
----
----
df1.DMMedType.value_counts().head(5)
# Cant drop this column as it looks like it does not predictive
power
----
----
df1.dmMedsSelf.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.statinMedsSelf.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.antiArythMedsSelf.value_counts().head(5)
# Can drop this column as it has mostly No
----
----
df1.BPmeds.value_counts().head(5)
# Cant drop this column as it looks like it has predictive power
----
----
df1.BPjnc7.value_counts().head(5)
# Cant drop this column as it looks like it has predictive power
----
----
df1.asthma.value_counts().head(5)
# Can drop this column as it has mostly Never
----
----
df1.LVdilation.value_counts().head(5)
# Can drop this column as it has mostly None
----
----
df1.EF3cat.value_counts().head(5)
# Cant drop this column as it looks like it has predictive power
----
----
df1.alcw.value_counts().head(5)
```

```
# Can drop this column as it has mostly 0.00
----
----
#Dropping the cloumns from the above analysis
df1.drop(['MajorScarAnt', 'ageIneligible', 'DMmedsOral',
'LVdilation', 'asthma', 'DMmedsIns', 'BPmedsSelf',
'antiArythMedsSelf', 'statinMedsSelf', 'calBlkMeds', 'DMMedType',
'dmMedsSelf', 'antiArythMeds', 'betaBlkMeds', 'DMmeds',
'statinMeds', 'hrtMeds', 'ConductionDefect', 'MinorScarAnt',
'RepolarAnt', 'MIant', 'MajorScarPost', 'MinorScarPost',
'RepolarPost', 'MIpost', 'MajorScarAntLat', 'MinorScarAntLat',
'RepolarAntLat', 'MIAntLat', 'MIecg', 'Afib', 'Aflutter', 'LVHcv',
'speechLossEver', 'visionLossEver', 'doubleVisionEver',
'numbnessEver', 'paralysisEver', 'dizzynessEver', 'strokeHx',
'MIHx', 'CardiacProcHx', 'CHDHx', 'CarotidAngioHx', 'CVDHx',
'MedicaidIns', 'VAIns', 'InsuranceType', 'Insured', 'PublicIns',
'PrivatePublicIns'] ,1, inplace=True)
----
----
#df1
----
----
# Now lets see the target variable
#Lets plot and see
sns.countplot(df1["stroke"]);
#looks like highly imbalanced data
----
----
B_Val = {"No":0, "Yes":1}
df1= df1.replace({"stroke": B_Val})
----
----
#lets see the distributions
x = df1.hist(bins=25, grid=False, figsize=(16,10))
----
----
df1.skew()
----
----
#Checking for missing values
sns.heatmap(df1.isnull(), cbar=False);
----
----
df1.isnull().sum(axis=0)
----
----
#df1
----
----
# Missing value treatment for numeric variables
df1.fillna(df1.median(),inplace = True)
----
----
df1.isnull().sum(axis=0)
```

```
----
----
df1.dtypes
----
----
#Lets see the outliers by plotting box plots
df_box1 = pd.DataFrame(data = np.random.random(size=(4,7)), columns
= ['age', 'alcw','weight', 'height', 'neck', 'HbA1c',

'hdl'])
df_box1.boxplot()
----
----
#Lets see the outliers by plotting box plots
df_box2 = pd.DataFrame(data = np.random.random(size=(4,7)), columns
= ['sbp', 'dbp','abi', 'ldl', 'trigs', 'LEPTIN',

'HSCRP'])
df_box2.boxplot()
----
----
#Lets see the outliers by plotting box plots
df_box3 = pd.DataFrame(data = np.random.random(size=(4,4)), columns
= ['ENDOTHELIN', 'ALDOSTERONE', 'cystatinC', 'SCrCC'])

df_box3.boxplot()
----
----
#Lets see the outliers by plotting box plots
df_box4 = pd.DataFrame(data = np.random.random(size=(4,5)), columns
= ['SCrCC', 'maneuvers', 'FVC', 'FEV1PP', 'EF'])

df_box4.boxplot()
----
----
#Lets see the outliers by plotting box plots
df_box5 = pd.DataFrame(data = np.random.random(size=(4,6)), columns
= ['ecgHR', 'QRS', 'QTcFram', 'QT', 'CV', 'dailyDiscr'])

df_box5.boxplot()
----
----
#Lets see the outliers by plotting box plots
df_box6 = pd.DataFrame(data = np.random.random(size=(4,4)), columns
= ['lifetimeDiscrm', 'perceivedStress', 'vitaminD3',

'hyIndex'])

df_box6.boxplot()
----
----
#Lets see the outliers by plotting box plots
df_box7 = pd.DataFrame(data = np.random.random(size=(4,3)),columns =
['FakeCensusTractID', 'nbmedHHincome', 'nbpctBlackNH'])
```

```
df_box7.boxplot()
----
----
#Lets see the outliers by plotting box plots
df_box8 = pd.DataFrame(data = np.random.random(size=(4,3)),columns =
['nbK3FavorFoodstore', 'sportIndex', 'activeIndex'])

df_box8.boxplot()
----
----
#As stroke is our target variable, dropping the records where stroke
value is missing
df1 = df1[pd.notnull(df1['stroke'])]
----
----
df1.shape
----
```

Logistic Regression

```
----
#import the packages that we need
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
----
----
#Create data with only the features and dependent variables.
feature_cols =
['FastHours','age','sex','alcw','currentSmoker','everSmoker','weight
','height','neck','OBESITY3cat',

'BPmeds','diureticMeds','sbp','dbp','BPjnc7','HTN','abi','HbA1c','Di
abetes','ldl','hdl','trigs',

'ldl5cat','hdl3cat','trigs4cat','LEPTIN','HSCRP','ENDOTHELIN','ALDOS
TERONE','cystatinC','sCort',

'adiponectin','SCrCC','DialysisEver','CKDHx','maneuvers','FVC','FEV1
PP','EF','EF3cat','ecgHR','QRS',

'QT','QTcFram','CV','PrivateIns','MedicareIns','dailyDiscr','lifetim
eDiscrm','perceivedStress',

'SMK3cat','BMI3cat','vitaminD3','FakeCensusTractID','nbmedHHincome',
'nbpctBlackNH',

'nbK3FavorFoodstore','sportIndex','hyIndex','activeIndex','years','s
troke']
model1 =  df1[feature_cols]
# convert selected features do dummies
model1 = pd.get_dummies(model1, columns=["sex",
"currentSmoker","everSmoker", "OBESITY3cat",
                                         "BPmeds",
"diureticMeds", "BPjnc7", "HTN", "Diabetes",
```

```
                                                    "ldl5cat", "hdl3cat",
"trigs4cat", "DialysisEver", "CKDHx",

"EF3cat","PrivateIns","MedicareIns", "SMK3cat", "BMI3cat"],
drop_first=True)
# set x and y
X = model1.drop('stroke', axis =1)
y = model1['stroke']
# train test splot
x_train, x_test, y_train, y_test = train_test_split(X,y,
random_state =123)
#handling missing values
x_train.fillna(x_train.median(),inplace = True)
#handling outliers
from sklearn.preprocessing import StandardScaler
# Create an scaler object
scaler = StandardScaler()
# Train on the training features
scaler.fit(x_train)
# Transform training data
x_train = pd.DataFrame(scaler.transform(x_train),
columns=x_train.columns)
# set the model
LR = LogisticRegression()
# fit model
LR.fit(x_train, y_train)
----
----
#getting the coefficients:
a = model1.columns.drop('stroke')
coefficient = LR.coef_[0]
pd.DataFrame([a,coefficient],index =
['Cols','coefficient']).transpose()
----
----
#Now follow the same steps for x_test data that we did for x_train
data.
#handling missing values
x_test.fillna(x_test.median(),inplace = True)
#handling outliers
from sklearn.preprocessing import StandardScaler
# Create an scaler object
scaler = StandardScaler()
# Train on the training features
scaler.fit(x_test)
# Transform training data
x_test = pd.DataFrame(scaler.transform(x_test),
columns=x_test.columns)
#Use the Model to predict on x_test and evaluate the model using
metric
y_pred_test = LR.predict(x_test)
y_pred_train = LR.predict(x_train)
----
----
```

```
#Lets see ROC for test
metrics.roc_auc_score(y_test,y_pred_test)
----
----
#Lets see ROC for train
metrics.roc_auc_score(y_train,y_pred_train)
----
----
#confusion matrix for test
metrics.confusion_matrix(y_test,y_pred_test)
----
----
#confusion matrix for train
metrics.confusion_matrix(y_train,y_pred_train)
----
----
#Lets see accuracy for test
metrics.accuracy_score(y_test,y_pred_test)
----
----
#Lets see accuracy for train
metrics.accuracy_score(y_train,y_pred_train)
----
----
#Plot the ROC curve
fpr, tpr, threshold = metrics.roc_curve(y_test, y_pred_test)
roc_auc = metrics.auc(fpr, tpr)
# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
----
```

K-Folds cross-validation

```
----
#import the packages that we need
from sklearn.model_selection import cross_val_score,
cross_val_predict
from sklearn import metrics
#handling missing values
X.fillna(X.median(),inplace = True)
#handling outliers
from sklearn.preprocessing import StandardScaler
# Create an scaler object
scaler = StandardScaler()
# Train on the training features
scaler.fit(X)
# Transform training data
```

```
X = pd.DataFrame(scaler.transform(X), columns=X.columns)
# set the model
LR1 = LogisticRegression()
# fit model
cross_val_score(LR1, X, y, cv=5)

----
----
# predictions
preds = cross_val_predict(LR1, X, y, cv=5)
----
----
#lets see ROC
metrics.roc_auc_score(y, preds)
----
----
#confusion matrix for test
metrics.confusion_matrix(y_test,y_pred_test)
----
----
#confusion matrix for train
metrics.confusion_matrix(y_train,y_pred_train)
----
----
#Lets see accuracy for test
metrics.accuracy_score(y_test,y_pred_test)
----
----
#Lets see accuracy for train
metrics.accuracy_score(y_train,y_pred_train)
----
```

Logistic Regression with SGD(stochastic gradient descent)

```
----
from sklearn.linear_model import SGDClassifier
from sklearn.pipeline import Pipeline
pip = Pipeline([('model', SGDClassifier(loss='log', max_iter=500,
tol=1e-3, random_state=123, warm_start=False))])
#Hyper parameters
param  = {
    'model__alpha': [1, 2, 5],
    'model__penalty': ['l1', 'l2']
}
----
----
#Set the model
from sklearn.model_selection import GridSearchCV
sgdlr = GridSearchCV(estimator=pip, param_grid=param,
scoring='roc_auc', n_jobs=-1, pre_dispatch='2*n_jobs', cv=5,
verbose=1, return_train_score=False)
----
----
#fit the model
sgdlr.fit(X, y)
```

```
----
----
sgdlr_estimator = sgdlr.best_estimator_
print('ROC: ', sgdlr.best_score_)
print('For parameters: \n', sgdlr.best_params_)
----
----
y_pred_sgdlr = sgdlr_estimator.predict(x_test)
y_prob_sgdlr = sgdlr_estimator.predict_proba(x_test)[:,1]
----
----
y_train_pred_sgdlr = sgdlr_estimator.predict(x_train)
y_train_prob_sgdlr = sgdlr_estimator.predict_proba(x_train)[:,1]
----
```

Random Forest

```
----
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_jobs=-1, n_estimators=100,
random_state=123,max_features='sqrt')
#Hyper parameters
param_rf = {
    'class_weight': [{0:1, 1:1}]
}
----
----
#Set the model
grid_rf = GridSearchCV(estimator=rf, param_grid=param_rf,
scoring='roc_auc',n_jobs=-1,pre_dispatch='2*n_jobs', cv=5,
verbose=1, return_train_score=False)
----
----
#fit the model
grid_rf.fit(x_train, y_train)
----
----
rf_estimator = grid_rf.best_estimator_
print('ROC: ', grid_rf.best_score_)
print('For parameters: \n', grid_rf.best_params_)
----
----
y_pred_rf = rf_estimator.predict(x_test)
y_prob_rf = rf_estimator.predict_proba(x_test)[:,1]
----
----
y_train_pred_rf = rf_estimator.predict(x_train)
y_train_prob_rf = rf_estimator.predict_proba(x_train)[:,1]
----
```

KNN

```
----
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.feature_selection import RFECV
from sklearn import decomposition
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
----
----
#creating the pipeline
pip_knn = Pipeline([
    ('lda', LinearDiscriminantAnalysis()),
    ('model', KNeighborsClassifier(n_jobs=-1))
])
# hyper parameters
param_knn = {
    'lda__n_components': range(3,10),
    'model__n_neighbors': [5, 30, 100]
}
----
----
# setup the model
grid_knn = GridSearchCV(estimator=pip_knn, param_grid=param_knn,
scoring='roc_auc', n_jobs=-1, pre_dispatch='2*n_jobs', cv=5,
verbose=1, return_train_score=False)
----
----
#fit the model
grid_knn.fit(x_train, y_train)
----
----
knn_estimator = grid_knn.best_estimator_
print('ROC: ', grid_knn.best_score_)
print('For parameters: \n', grid_knn.best_params_)
----
----
y_pred_knn = knn_estimator.predict(x_test)
y_prob_knn = knn_estimator.predict_proba(x_test)[:,1]
----
----
y_train_pred_knn = knn_estimator.predict(x_train)
y_train_prob_knn = knn_estimator.predict_proba(x_train)[:,1]
----
```

Model Comparison using ROC

```
----
best_rocs = [sgdlr.best_score_,
grid_rf.best_score_,grid_knn.best_score_]
roc_tbl = pd.DataFrame({"AUROC":best_rocs,"Algorithm":["SGD Logistic
Regression",
"RandomForest","KNeighboors"]})
roc_tbl
----
----
from sklearn.metrics import confusion_matrix, precision_score,
recall_score, f1_score, make_scorer, roc_auc_score,accuracy_score,
roc_curve
#create a function for evaluation
```

```
def evl(X_train, X_test, Y_train, Y_test, Y_train_pred,
Y_train_prob, Y_pred, Y_prob):
    print("ROC AUC")
    print("Training Set:", roc_auc_score(Y_train, Y_train_prob))
    print("Test Set:", roc_auc_score(Y_test, Y_prob))

    TP, FN, TN, FP = confusion_matrix(Y_test, Y_pred).ravel()
    print("\nConfusion Matrix")
    print("True Positive:", TP)
    print("False Negative:", FN)
    print("True Negative:", TN)
    print("False Positive:", FP)

    print("\nAccuracy")
    print("Training Set:", accuracy_score(Y_train, Y_train_pred))
    print("Test Set:", accuracy_score(Y_test, Y_pred))

#creating a function for ploting ROC curve
def plot_ROC(X_test, Y_test, Y_prob):

    #Y_prob = model.predict_proba(X_test)[:,1]
    fpr, tpr, thresh = roc_curve(Y_test, Y_prob, pos_label=1)
    roc_auc = roc_auc_score(Y_test, Y_prob)
    # threshold = 0.1~0.5
    x1 = fpr[(thresh <= 0.5) & (thresh >= 0.1)]
    x2 = tpr[(thresh <= 0.5) & (thresh >= 0.1)]
    fig = plt.figure()
    plt.plot(fpr, tpr, color='r', lw=2)
    plt.plot([0, 1], [0, 1], color='b', lw=2, linestyle='--')
    plt.plot(x1, x2, color='k', lw=3, label='threshold = 0.1 ~ 0.5')
    plt.xlim([-0.05, 1.05])
    plt.ylim([-0.05, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve (Area = {:.2f})'.format(roc_auc))
    plt.legend(loc="lower right")
    plt.show()
----
----
print('Logistic Regression with SGD')
evl(x_train, x_test, y_train, y_test, y_train_pred_sgdlr,
y_train_prob_sgdlr, y_pred_sgdlr, y_prob_sgdlr)
plot_ROC(x_test, y_test, y_prob_sgdlr)
----
----
print('Random Forest')
evl(x_train, x_test, y_train, y_test, y_train_pred_rf,
y_train_prob_rf, y_pred_rf, y_prob_rf)
plot_ROC(x_test, y_test, y_prob_rf)
----
----
print('KNN')
evl(x_train, x_test, y_train, y_test, y_train_pred_knn,
y_train_prob_knn, y_pred_knn, y_prob_knn)
```

```
plot_ROC(x_test, y_test, y_prob_knn)
```

XGBoost

```
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import re
```


```
#Lets do regex on the feature variables
regex = re.compile(r"\[|\]|<", re.IGNORECASE)
#Create data with only the features and dependent variables.
feature_cols =
['FastHours','age','sex','alcw','currentSmoker','everSmoker','weight
','height','neck','OBESITY3cat',

'BPmeds','diureticMeds','sbp','dbp','BPjnc7','HTN','abi','HbA1c','Di
abetes','ldl','hdl','trigs',

'ldl5cat','hdl3cat','trigs4cat','LEPTIN','HSCRP','ENDOTHELIN','ALDOS
TERONE','cystatinC','sCort',

'adiponectin','SCrCC','DialysisEver','CKDHx','maneuvers','FVC','FEV1
PP','EF','EF3cat','ecgHR','QRS',

'QT','QTcFram','CV','PrivateIns','MedicareIns','dailyDiscr','lifetim
eDiscrm','perceivedStress',

'SMK3cat','BMI3cat','vitaminD3','FakeCensusTractID','nbmedHHincome',
'nbpctBlackNH',

'nbK3FavorFoodstore','sportIndex','hyIndex','activeIndex','years','s
troke']
model1 =  df1[feature_cols]
# convert selected features do dummies
model1 = pd.get_dummies(model1, columns=['sex',
'currentSmoker','everSmoker', 'OBESITY3cat',
                                         'BPmeds',
'diureticMeds', 'BPjnc7', 'HTN', 'Diabetes',
                                       'ldl5cat', 'hdl3cat',
'trigs4cat', 'DialysisEver', 'CKDHx',

'EF3cat','PrivateIns','MedicareIns', 'SMK3cat', 'BMI3cat'],
drop_first=True)
# set x and y
X = model1.drop('stroke', axis =1)
y = model1['stroke']
# split data into train and test sets
seed = 7
test_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=test_size, random_state=seed)
```

```
X_train.columns = [regex.sub("_", col) if any(x in str(col) for x in
set(('[', ']', '<'))) else col for col in
                   X_train.columns.values]
X_test.columns = [regex.sub("_", col) if any(x in str(col) for x in
set(('[', ']', '<'))) else col for col in
                  X_test.columns.values]
# fit model no training data
xg = XGBClassifier()
xg.fit(X_train, y_train)
----
----
#Now follow the same steps for x_test data that we did for x_train
data.
#handling missing values
x_test.fillna(x_test.median(),inplace = True)
#handling outliers
from sklearn.preprocessing import StandardScaler
# Create an scaler object
scaler = StandardScaler()
# Train on the training features
scaler.fit(x_test)
# Transform training data
x_test = pd.DataFrame(scaler.transform(x_test),
columns=x_test.columns)
X_train.columns = [regex.sub("_", col) if any(x in str(col) for x in
set(('[', ']', '<'))) else col for col in
                   X_train.columns.values]
x_test.columns = [regex.sub("_", col) if any(x in str(col) for x in
set(('[', ']', '<'))) else col for col in
                  X_test.columns.values]
#Use the Model to predict on x_test and evaluate the model using
metric
y_pred_train = xg.predict(X_train)
y_pred_test = xg.predict(X_test)
----
----
X_test.shape
----
----
y_pred_test.shape
----
----
# make predictions for test data
y_pred_test = xg.predict(X_test)
predictions_test = [round(value) for value in y_pred_test]
----

----
# evaluate predictions for test
accuracy = accuracy_score(y_test, predictions_test)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
----
----
# make predictions for test data
```

```
y_pred_train = xg.predict(X_train)
predictions_train = [round(value) for value in y_pred_train]
----
----
# evaluate predictions for train
accuracy = accuracy_score(y_train, predictions_train)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
----
----
#confusion matrix for test
metrics.confusion_matrix(y_test,y_pred_test)
----
----
#confusion matrix for train
metrics.confusion_matrix(y_train,y_pred_train)
----
----
#Lets see ROC for test
metrics.roc_auc_score(y_test, y_pred_test)
----
----
#Lets see ROC for train
metrics.roc_auc_score(y_train, y_pred_train)
----
----
#Plot the ROC curve
fpr, tpr, threshold = metrics.roc_curve(y_test, y_pred_test)
roc_auc = metrics.auc(fpr, tpr)
# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
----
```