Stroke Outcomes Prediction Model

## Dataset

For this project, our group is using a dataset related to patient stroke outcomes to train and compare several binary classification machine learning models to predict stroke outcomes based on input variables. We will train a variety of different model types, using several different data preprocessing techniques for each one and compare the accuracy and effectiveness of each model. The best performing ones will be selected for presentation on the project's dashboard including the ability to try out the model’s prediction with user generated input variables, as well as a presentation of some of the findings from the dataset in regards to stroke outcomes broken down by data categories.

The dataset contains a combination of medical data and non-medical lifestyle data. We intend to create a predictive model using the entire dataset, but also compare and contrast predictions when using only medical data, or only non-medical data, as well and identifying which data categories seem to be predictive that we would not have expected.

Of course, this model is not intended for any medical use, but merely high-level analysis of the limited data we have available, to find if there are any interesting correlations for stroke outcomes within the data categories of this set.

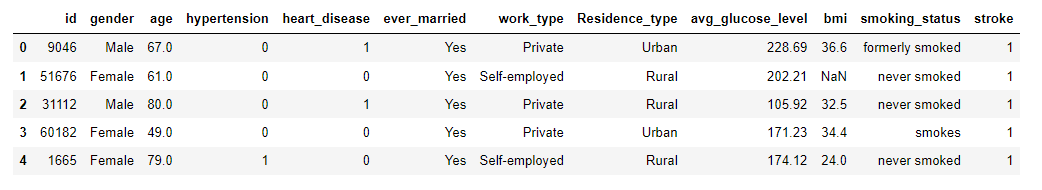
### Questions

1. Can this dataset be used to accurately and correctly predict whether a patient will have a stroke?
2. Does where you live correlate to stroke outcome?
3. Does where you work correlate to stroke outcome?
4. Does your marital status correlate to stroke outcome?
5. How do the non-medical factors impact our model’s accuracy, precision, and recall?
6. What is the likelihood of a smoker having a stroke?
7. How do BMI levels impact stroke predictions?
8. How do hypertension levels affect stroke prognosis?
9. What is the likelihood of heart disease contributing to a stroke prediction?
10. How do glucose levels impact stroke outcome?

### Data Used/Exploration

Kaggle Stroke Prediction dataset: <https://www.kaggle.com/fedesoriano/stroke-prediction-dataset>

A sample of our datatset:



The original dataset has twelve columns of which 11 are features we’re using to build our model:

|  |  |
| --- | --- |
| Field | Datatype |
| id | int64 |
| gender | object |
| age | float64 |
| hypertension | int64 |
| heart\_disease | int64 |
| ever\_married | object |
| work\_type | object |
| Residence\_type | object |
| avg\_glucose\_level | float64 |
| bmi | float64 |
| smoking\_status | object |
| stroke | int64 |

|  |  |
| --- | --- |
| Field | Data contained |
| id | Patient ID - not material to determine stroke outcome so dropped from feature dataset |
| gender | Patient Gender - male, female, other |
| age | Patient age |
| hypertension | Boolean - indicates if patient has hypertension or not |
| heart\_disease | Boolean - indicates if patient has heart disease or not |
| ever\_married | Boolean - indicates if patient has ever been married or not |
| work\_type | Indicates which employment sector patient is in - Private, self-employed, government job, never worked, children |
| Residence\_type | Indicates patient’s residence area - rural/urban |
| avg\_glucose\_level | Patient’s average glucose level (no indication of time span of glucose tests) |
| bmi | Patient BMI (no indication of when BMI last measured) |
| smoking\_status | Patient smoking status - smokes, formerly smoked, never smoked, unknown |
| stroke | Boolean - did patient have a stroke or not |

We identified differences in the data features contained in the dataset. There are medical features, information a doctor would use in a medical diagnosis; Gender, Age, Hypertension, Heart Disease, Average Glucose Level, BMI, and Smoking Status. The other data points; marital status, work type, and residence type are not used to make a medical diagnosis and we classified those features as non-medical or lifestyle choices. We will test many models with and without these lifestyle features to see if they may have an impact on determining stroke outcomes. These features may make for some interesting data visualizations.

### Data Cleaning and Analysis

Pandas will be used to clean the data, split the data (training and testing) and Further data analysis will be completed using Python.

#### Data preprocessing

The initial dataset contained 5,110 rows of patient data. The BMI category had 201 null values. Rows with these null values were removed from the dataset.  The numerical categorical data, hypertension and heart disease, were reclassified as objects. BMI outliners (rows with BMI greater than 50 were removed). We then used the label encoder method to change all categorical data to numerical data. This new dataset was exported to our database to be used in creating four new datasets for testing our four selected machine learning models.

#### Training and testing sets

The cleaned dataset contained 4,830 rows of patient data, of which 208 had strokes and 4,622 did not. We created 4 separate training and testing sets to use in our testing models. We first created a balanced set that contained all 208 stroke patients and 208 random samples from the patients who did not have a stroke. From that balanced sample of 416 records, we split the data into training and testing sets with 80% of the data used for training.

Our second model testing dataset used the new balance dataset with 416 records. We then used the StandardScaler to scale all of the data. This scaled data was then split into another 80/20 training and testing set.

The third and fourth testing dataset used the 4,830 cleaned records. We applied the SMOTEENN method to create a new dataset with 3,491 non-stroke and 4,351 stroke datapoints. The original 4,830 records were split into 80/20 training/testing sets and then the feature training and stroke training sets were joined with the newly created records from the new SMOTEENN data. The new data has 8,113 non-stroke(ns) records and 4,559 stroke(s) records of which 7,189(ns) and 4,517(s) are training points and 924(ns) and 42(s) are testing points. One set was created without any data scaling and the other was created again using the StandardScaler method to scale the data.

### Beginning Analysis

SciKitLearn & Tensowflow are the ML library we'll be using to create a classifier. For the initial approach to select the best model for our project we assigned each team member a machine learning model to build. We chose to try Logistic Regression, Random Forest, Neural Network, and Decision Tree. Other models tried were Support Vector and Gradient Boosting. Each team member would be responsible for creating, training, and testing their model. Once we pulled in the training and testing sets, we were to test 8 models, one with each of the 4 created training and testing and one for each of the training and testing sets with lifestyle(non-medical) features removed. The confusion matrix and classification reports for each model would be compared to determine which model(s) we would use for the final step of our project.

We used an excel spreadsheet where everyone could report the Precision, Accuracy, Recall, and F1 scores of their models for each of the training and testing sets to aid in our decision making.

#### **Model evaluation for Decision Trees and SVM classifiers**

This python script evaluates multiple decision tree models and Support Vector Machine (SVM) incorporating boosting techniques to improve the model performance.

Below models were generated and compared to select the best model:

* Decision Tree
* Support Vector Machine (SVM)
* Boostrap Aggregation (Bagging)
* Adaptative Boosting (AdaBoost)
* Gradient Boosting

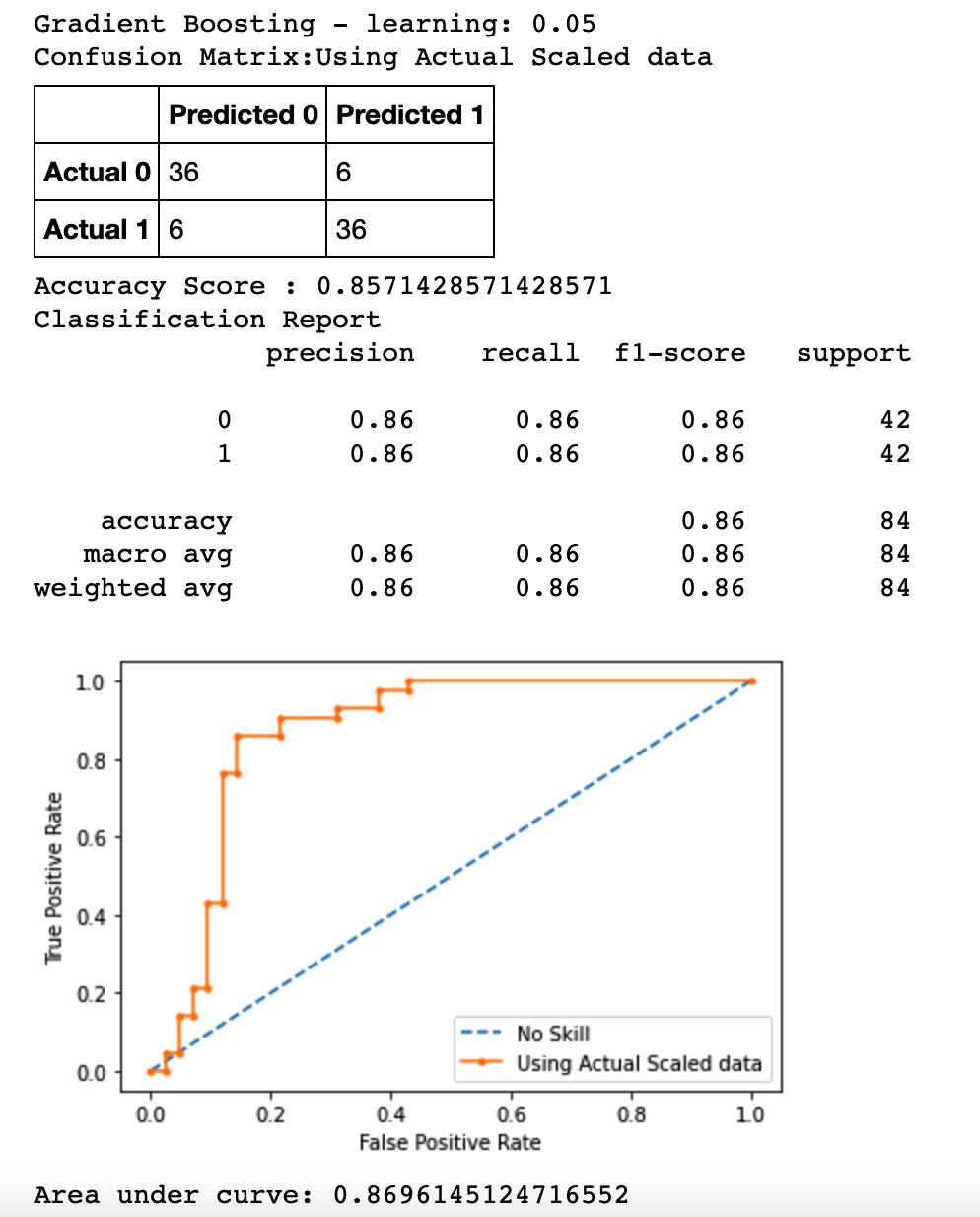
All previous models were generated with their corresponding sampling technique: Actual Data scaled and unscaled and SMOTEENN scaled data.

Generating the following statistics to support the decision of the best performer model:

* Precision, Recall and f1 scores
* Confusion Matrix
* RoC curve and area under the curve (AUC)

**Explanation of model choice, including limitations and benefits**

The wining model from decision trees and SVM was the sequential ensemble method: Gradient Boosting with 200 estimators and pseudo-residuals corrected during each iteration at a 0.05 learning rate. Main benefits of these model is to implement weak learners rather than a strong learner (ensemble method) that focuses on minimizing previous small tree (stumps) errors (pseudo-residuals) however the model is still limited to the small data set for training and testing which prevent it from improving its metrics beyond the number of estimators and learning rates evaluated as part of the analysis.



Model accuracy:

* Accuracy score: 0.857
* F1-score: 0.86/0.86
* RoC, AUC: 0.869
* What would you include if you had more time?

I would explore the sklearn ***GridSearchCV*** class to identify the optimal number of estimator and learning rate to fit the Gradient Boosting Machine Learning model.

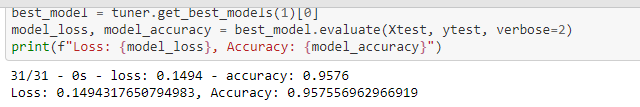
For the current analysis just the learning rate was selected based on training and testing accuracy scores as part of the for loop in ***train\_test\_assess\_boosting\_learning*** function while the number of estimators were determined based on a try and error empiric iterations.

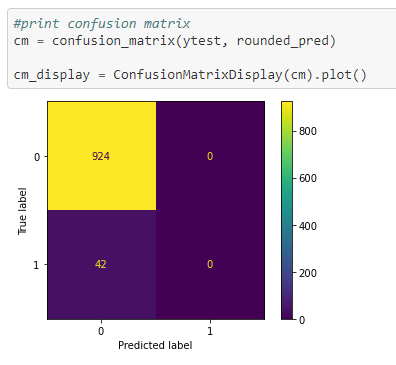
#### Model evaluation for Neural Network

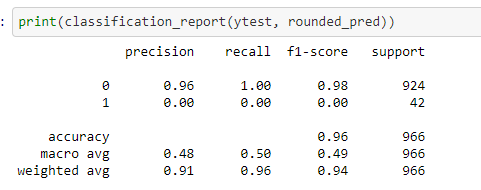
To evaluate the effectiveness of using a neural network as the machine learning model to predict stroke outcomes from this dataset I used the Keras Tuner from the Tensorflow library. A function was built to evaluate a neural network with:

* 1 – 6 hidden layers
* 1 – 10 nodes per layer
* Tuners choice of relu, tanh, and sigmoid activation functions on each layers
* An output layer with sigmoid activation function
* Compiler used binary cross entropy to measure loss, adam optimizer and performance was based on accuracy.
* The tuner would use a max of 50 epochs.

I ran all 8 training and testing sets through the tuner. No strokes were predicted with any of the datasets. The best model from each dataset using SMOTEENN balancing were all pretty good in their accuracy ratings all greater than 95%; however, none of the tests actually predicted any strokes.

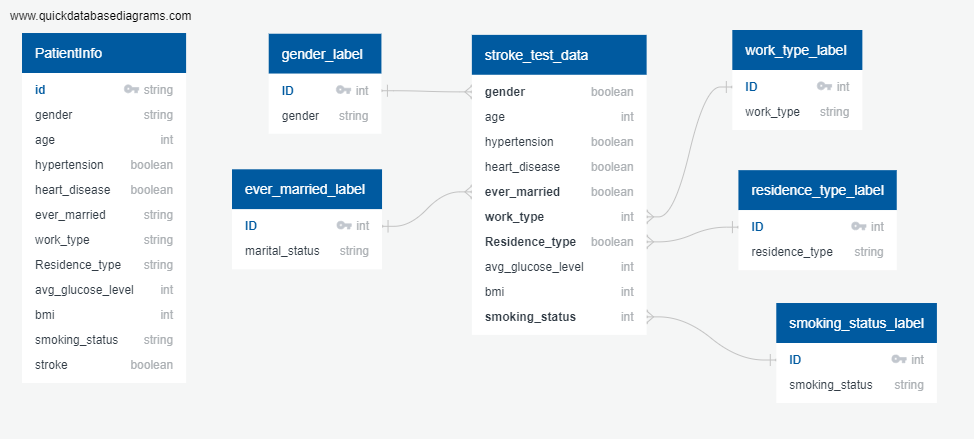






## Database Storage

We’ll be using PostGres SQL database with AWS. Database ERD:

****

## Dashboard

Tableau visuals and story telling functionality will be used to provide user interactivity and identified insights.

## Stretch Goal

Set up a webpage using Javascript forms to allow users to enter their medical, personal and lifestyle data to how likely the would be to have a stroke.