Using Machine Learning to Predict Stroke Risk

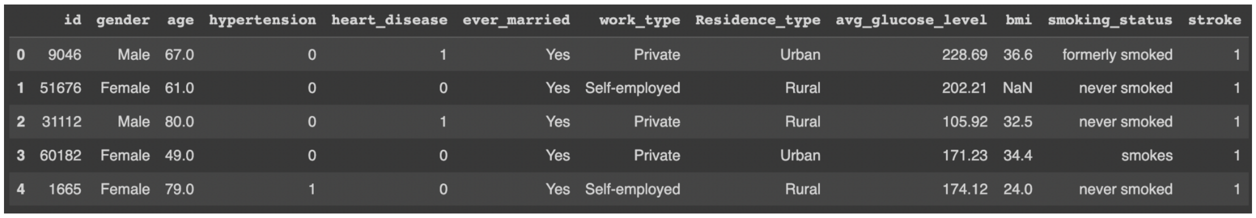
By: Arnav Goel

Many things are believed to cause strokes, but the actual factors that can lead to increased risk of having a stroke can be identified using logistic regression and machine learning. Knowing these factors will allow more insight into stroke prevention.

1. Introduction

Machine Learning, a subfield of the vast artificial intelligence field, describes the development of methods that are capable of learning through datasets. With each new datapoint, the method's "understanding" of the possible outputs for various inputs is enhanced. It uses algorithms to identify patterns in the data and uses those patterns to create a model that makes predictions **[5]**. This has enabled machine learning to greatly improve the data science and analysis field, and it has proven to be useful for countless predictions. One of these uses is in predicting the likelihood of a person to have a stroke based on certain factors. Using an existing dataset consisting of thousands of individuals who either had a stroke or didn't have a stroke coupled with artificial intelligence concepts, we created a model that accurately determines whether any individual is at risk for having a stroke given their gender, age, marital status, work type, residence type, glucose level, BMI (Body Mass Index), smoking status, and whether or not they had hypertension or heart disease (Fig 1). This model addresses an important issue. Strokes are a common and often life altering medical emergency caused by a lack of blood supply to the brain. In order to better distinguish the causes of strokes, these various factors have to be considered.

Figure 1. First Five Rows of Healthcare dataset for stroke patients.



	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

The dataset shows ten of the most commonly considered factors that could lead to a stroke. It consists of 5110 rows, where each row represents a specific person, their information pertaining to the factors, and whether or not they had a stroke. For example, the first person in the dataset was a 67-year-old male. He has never had hypertension, indicated by the 0 in the hypertension column, but has been diagnosed with heart disease, indicated by the 1 in the heart disease column. He is married, works a private job, and resides in an urban setting. His average glucose level is 228.69 and his BMI is 36.6. He has smoked in the past but does not anymore (formerly smoked). This individual **had** a stroke, as marked by the 1 in the final column labeled "stroke".

The problem our model aims to solve belongs to a class of problems known as supervised learning. The data of supervised learning problems consist of information about a collection of examples **[2]**. The examples in our case are the people in each row. The information from each example consists of two types, features and labels. In our case, each example has ten features, the different factors, and 1 label, whether the person had a stroke or not. In general, the label is what we want the model to predict when the features of new examples are fed into the model as input. The strategy of machine learning is to start with a data set where both the features and the labels of the examples are known. In our case, this means that the factors, as well as whether the individual had a stroke, is known for all people in the data set. This data set is called the training set, and the examples in this set are called training examples.

2. Materials and Methods

2.1. Binary Classification

Since our label, whether or not they had a stroke, has two possible outcomes, this is a binary classification problem. The first outcome is that the patient had a stroke, and is denoted by a 1 in the respective column. Likewise, the second outcome is that the patient was healthy and did not have a stroke. This outcome is denoted by a 0 in the respective column. Our model has to take all of the inputs from the features and output a single number between 0 and 1. We denote this predicted value as $\hat{y}$. We also denote each factor as x_1 , x_2 , ... , x_{10}. The predicted value $\hat{y}$ is a function of these factors. If the value of $\hat{y}$ is greater than 0.5, that means the model predicts 1 of the outcomes (assigned to a 1), and if the value of $\hat{y}$ is less than 0.5, the model will predict the other outcome (assigned to a 0). In our case, a value greater than 0.5 means that the model has predicted the person will have a stroke, and a value less than 0.5 means that the model has predicted the person will not have a stroke.

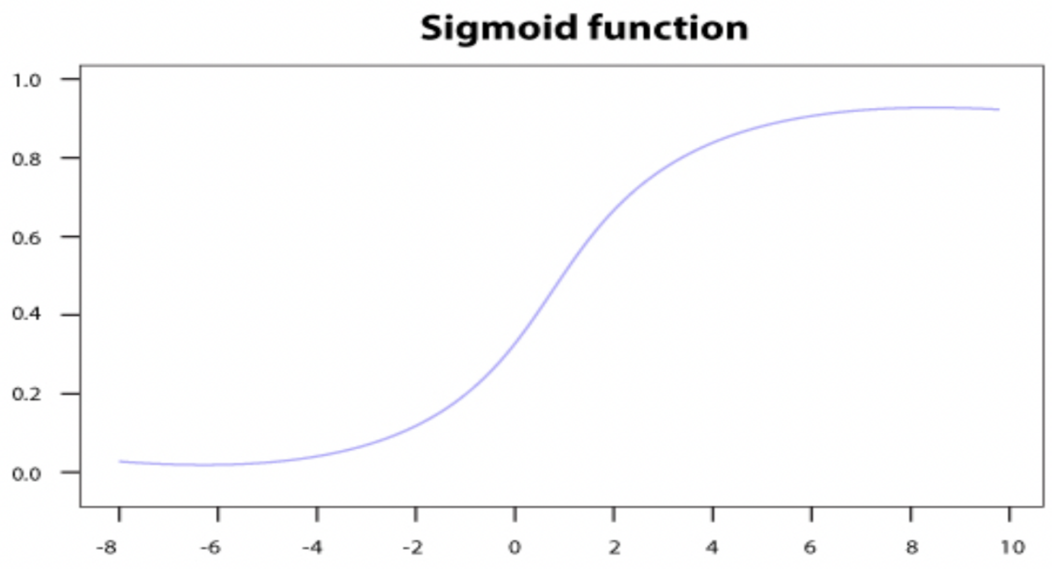
2.2 Logistic Regression

In order to develop an accurate model for this binary classification problem, we use a technique called logistic regression. Logistic regression is used in situations like these where the dependent variable (label) is categorical and constrained to a certain number of values **[4]**. This applies to our model because the outcome is either true or false, which aren't quantitative numbers, and those are the only two possible outcomes. Logistic regression arrives at the best possible solution using the maximum likelihood method, which is a method of estimating the parameters of an assumed probability distribution, given some observed data **[6]**.

A logistic regression equation is of the form $\hat{y} = f(c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 + c_6x_6 + c_7x_7 + c_8x_8 + c_9x_9 + c_{10}x_{10} + b)$. The values x_1, x_2, ... , x_{10} are the features and the values c_1, c_2, ... , c_{10}, b are parameters. The goal of logistic regression is to find the values for the parameters that most accurately model the dataset. In other words, the value outputted by this model should correctly predict whether or not the patient had a stroke most of the time for most of the patients. The result of the expression $c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 + c_6x_6 + c_7x_7 + c_8x_8 + c_9x_9 + c_{10}x_{10} + b$ can be any number. The possibilities that result from the various combinations of features and parameters in this expression are infinite, and don't output the number from 0 to 1 that we desire. In order to rectify this, we use an activation function, which is represented by f.

Activation functions are a crucial part of machine learning algorithms, as they take the infinite output and normalize it. Activation functions also perform a nonlinear transformation on the model to make it more accurate than a simple linear regression **[3]**. The activation function we use in our model is called the sigmoid function.

Figure 2. A graph of the sigmoid function



The sigmoid function takes any number as an input, and always outputs a number between 0 and 1(Fig 2). The sigmoid function is as follows: $\sigma(x) = \frac{1}{1+e^{-x}}$.

As x approaches negative infinity, the value of the sigmoid function approaches 0, because the denominator becomes significantly larger than the numerator. As x approaches positive infinity, the value of the sigmoid function approaches 1, because the value of e^{-x} approaches 0. Finally, when x is equal to 0, the sigmoid function is simply 0.5. Plugging the result of the expression $c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 + c_6x_6 + c_7x_7 + c_8x_8 + c_9x_9 + c_{10}x_{10} + b$ into the sigmoid function will give us our final prediction.

In short,

1. $0 < \sigma(x) < 1$ for all x
2. $\sigma(x)$ is an increasing function
3. $\sigma(x)$ becomes arbitrarily close to 0 as x becomes large in absolute value but negative.
4. $\sigma(x)$ becomes arbitrarily close to 1 as x increases.
5. $\sigma(0) = 0.5$.

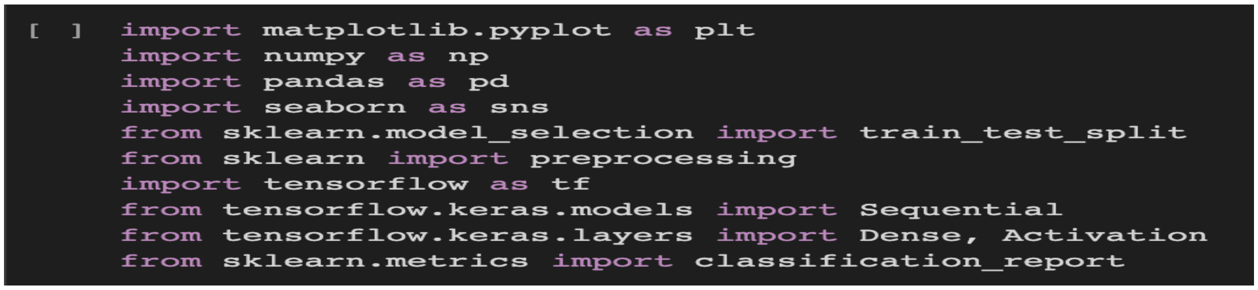
Our final representation of the logistic regression model becomes $\hat{y} = \sigma(c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 + c_6x_6 + c_7x_7 + c_8x_8 + c_9x_9 + c_{10}x_{10} + b) = \frac{1}{1+e^{-(c1x1 + c2x2 + c3x3 + c4x4 + c5x5 + c6x6 + c7x7 + c8x8 + c9x9 + c10x10 + b)}}$

2.3. Coding

In order to apply the aforementioned logistic regression techniques, all of the features have to be converted to numbers, and unnecessary aspects of the dataset should be removed so the model is trained in the best possible manner. In order to run all of the code to analyze the dataset and create the model, we use Google Colab and code in python.

2.3.1. Importing Libraries

Figure 3. Import statements to get access to various models



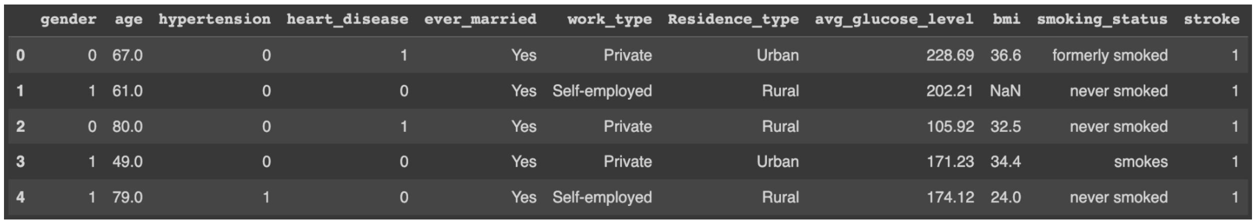
```
[ ]  import matplotlib.pyplot as plt
     import numpy as np
     import pandas as pd
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn import preprocessing
     import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Activation
     from sklearn.metrics import classification_report
```

We start by coding different import statements to perform the necessary functions later (Fig 3). We use pandas in order to easily perform operations such as removing items, viewing portions of the dataset, and adding columns, all of which are required to eventually create the model. We use numpy to perform mathematical operations on arrays and matrices. We use sklearn to handle scaling the data and making it possible to apply the function to the data. Tensorflow and keras handle the actual math and algorithms behind finding the best predictor. Pyplot and seaborn aid with displaying these algorithms.

2.3.2. Data Preprocessing

We begin the streamlining of the data by removing the ‘id’ column. This column’s purpose is just to give a number to each person in the dataset. It does not actually factor into the risk of having a stroke.

Figure 4. Dataframe after gender is converted to numbers



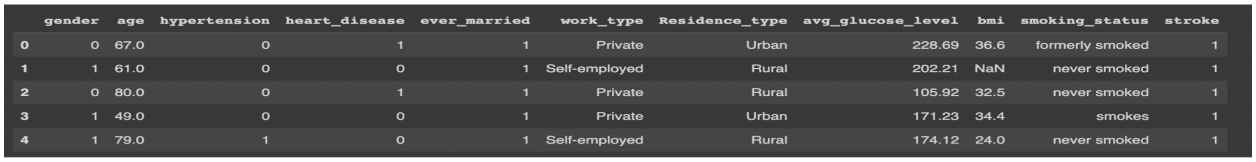
	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	0	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	1	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	0	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	1	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

Next, we must convert our first feature from “words” to numbers. In the gender column, the values present are male(2115), female(2994), and other(1). We begin our operations on this column by removing the row containing the value ‘other’. Since there is only one of these values, there is not enough information to teach the model how to predict a stroke if the gender is classified as ‘other’. Now that there are only two values of gender in the dataset, we can replace them with zeroes and ones. We choose to replace the value ‘male’ with 0 and the value 'female' with 1 (Fig 4).

The next column we work with is the age column, or the second feature. All of these values are already numeric, and there are no unknown or missing values. However, some of the values are obviously impractical and should be removed for better results and conclusions. In our dataset, there are 2026 patients younger than 38 years old, but only three of them had a stroke. Additionally, many of these 2026 patients were children under the age of 18, with some even being less than a year old. These patients are clearly at low risk for stroke, and don’t help the model. Additionally, these patients produce confounding variables when compared with other features. For example, if a patient is a child, their work type is labeled as children because they haven’t joined the workforce yet. Their marital status is definitely no and their smoking status should be never. For all of these reasons, we removed all rows with patients under 38 years old.

The hypertension and heart disease columns are already numerical, and both columns have no missing or unknown values. A zero in the hypertension column means that the patient doesn’t have hypertension, and a one means that the patient has hypertension. A zero in the heart disease column means that the patient doesn’t have heart disease, and a one means that the patient has hypertension.

Figure 5. Dataframe after marital status is converted to numbers

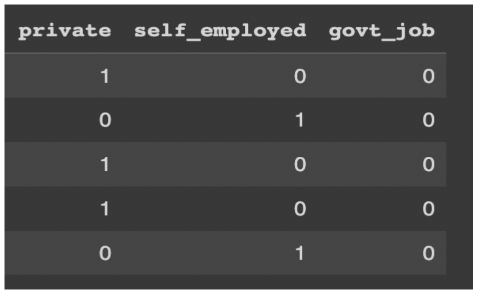


	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	0	67.0	0	1	1	Private	Urban	228.69	36.6	formerly smoked	1
1	1	61.0	0	0	1	Self-employed	Rural	202.21	NaN	never smoked	1
2	0	80.0	0	1	1	Private	Rural	105.92	32.5	never smoked	1
3	1	49.0	0	0	1	Private	Urban	171.23	34.4	smokes	1
4	1	79.0	1	0	1	Self-employed	Rural	174.12	24.0	never smoked	1

Similarly to the gender column, the ‘ever_married’ column contains just two possible values: yes and no. We replace all of the yeses and nos with 1’s and 0’s, respectively (Fig 5).

The work type column requires more transformation before it is ready to be used as a feature for the model. After removing all of the patients aged 38 or less, the three remaining values in the column are private, self-employed, and government jobs. Unlike the gender and marital status columns, replacing each value with a number doesn’t actually make the values quantitative. The 0’s, 1’s, and 2’s would still represent ‘private’, ‘self-employed’ and ‘government job’ instead of true and false. The solution to this is a technique called one-hot-encoding. One-hot-encoding takes the multiple values in a specific column and creates a new column for each value filled with only 0s or 1s. If the row originally had a specific value in it, the new column for that value would have a 1; if not, a 0.

Figure 6. One-hot-encoding for work type



private	self_employed	govt_job
1	0	0
0	1	0
1	0	0
1	0	0
0	1	0

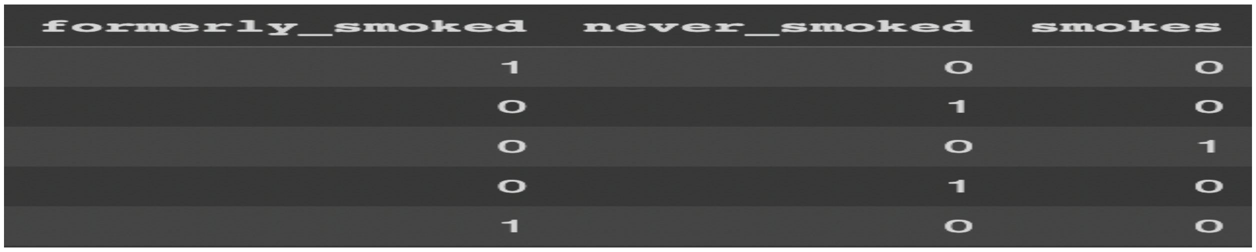
For example, the first five rows of the work-type column were private, self-employed, private, private, self-employed. Now, that original column is removed and three new columns are created for each value. There is a new column called ‘private’, a new column called ‘self_employed’, and a new column called ‘government_job’. The first value of ‘private’ is a 1, because the first row of the work type column was ‘private’. Consequently, the first values of ‘self_employed’ and ‘govt_job’ are 0. The second row of the work type column was ‘self_employed’, so the new ‘self_employed’ column has a 1 as its first value, and the other two columns have 0’s (Fig 6). Doing this allows us to obtain true or false values, in the form of 0’s and 1’s, instead of categorical values.

The next column, residence type, has only two values – urban, and rural. We choose urban to serve as false and rural to serve as true. To make the column usable, we replace all ‘urban’ with zeroes and all ‘rural' with one.

The following two features, glucose levels and BMI, are numerical values. The glucose column has no missing values, so it can be left unaltered. The body mass index column has a few unknown values that are marked by ‘NaN’ in the dataset. We remove all patients with this value so that the model has only numbers to work with.

For each patient, the smoking status feature has four possible values – ‘formerly smoked’, ‘never smoked’, ‘smokes’ and ‘unknown’. The first step in modifying this column is to remove all of the patients with an unknown in that column. Next, left with three possible values, we have to use one-hot-encoding for a second time. We remove the original smoking status column and create three additional columns for each smoking status value. We fill the columns with 1's and 0's as needed (Fig. 7).

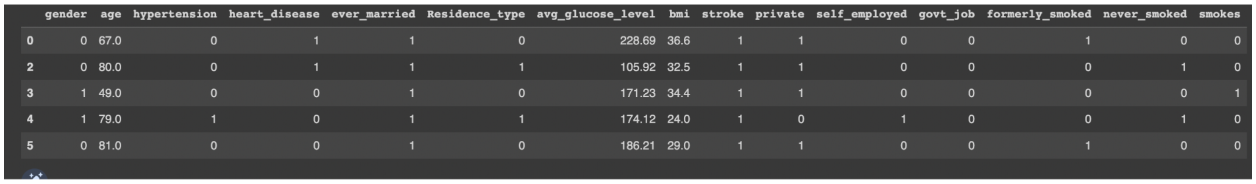
Figure 7. One-hot-encoding for smoking status



formerly_smoked	never_smoked	smokes
1	0	0
0	1	0
0	0	1
0	1	0
1	0	0

After completing all of the steps above, we are left with a dataset that accurately represents the sample. There are no missing values, and all of the features have been transformed into inputs that can be plugged into the algorithm for predicting stroke (Fig 8). We are left with 2389 total patients, of which 179 had a stroke and 2210 did not. The final step before the data is ready for the model is to balance these numbers. We do this so that when the data is split into a training and validation set, there is an equal representation of those who had a stroke and didn't in each group. In order to accomplish this, we duplicate each patient that had a stroke 11 times. The result of this is a dataset with an almost balanced number of both patients, with 2148 patients having a stroke and 2210 patients who did not have a stroke.

Figure 8. First five rows of the final dataset

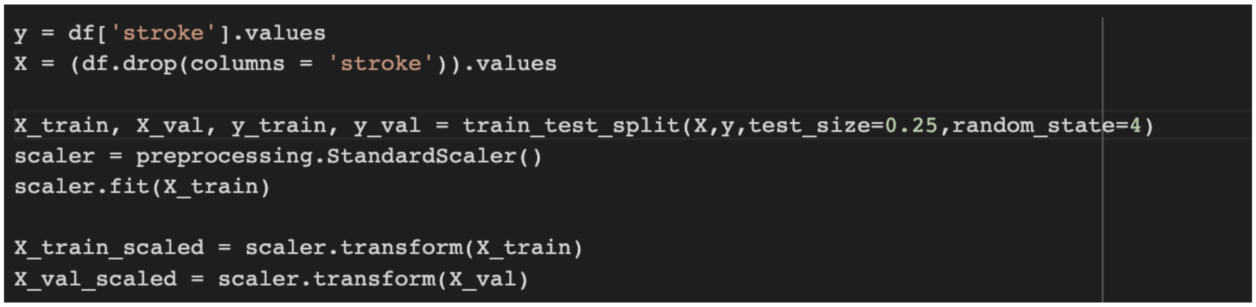


	gender	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_level	bmi	stroke	private	self_employed	govt_job	formerly_smoked	never_smoked	smokes
0	0	67.0	0	1	1	0	228.69	36.6	1	1	0	0	1	0	0
2	0	80.0	0	1	1	1	105.92	32.5	1	1	0	0	0	1	0
3	1	49.0	0	0	1	0	171.23	34.4	1	1	0	0	0	0	1
4	1	79.0	1	0	1	1	174.12	24.0	1	0	1	0	0	1	0
5	0	81.0	0	0	1	0	186.21	29.0	1	1	0	0	1	0	0

2.3.3 Training and validation

The set of examples are randomly split into two sets, the training set and the validation set. The training set contains 75% of the examples, and the validation set the rest 25%. The model will be generated using solely the training set. The accuracy and fit of the model are verified with the validation set. The point of having two different sets is avoiding overfitting. Overfitting describes the concept that a model can determine a curve that encompasses every single datapoint. This model would have 100% accuracy, but in the process of achieving this, it would sacrifice its ability to predict the outcomes of future data points and trends. Introducing a validation set ensures that overfitting is not a problem because a model that overfits based on the training set would have a low accuracy when compared to the validation set.

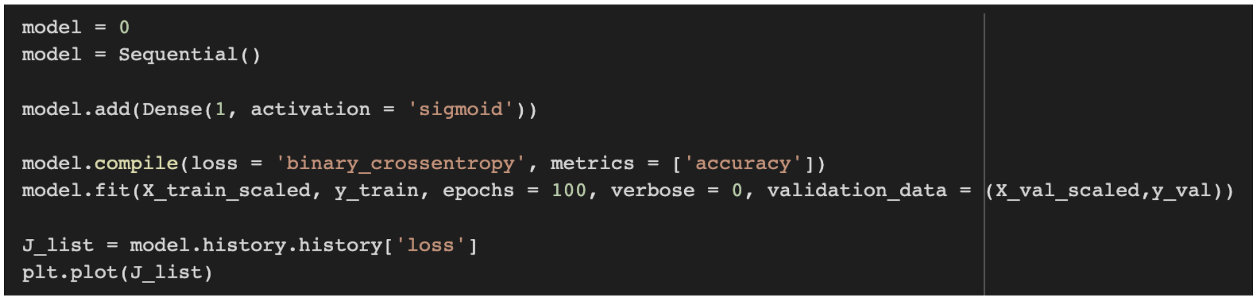
Figure 9. Code to split into training and validation



```
y = df['stroke'].values
X = (df.drop(columns = 'stroke')).values

X_train, X_val, y_train, y_val = train_test_split(X,y,test_size=0.25,random_state=4)
scaler = preprocessing.StandardScaler()
scaler.fit(X_train)

X_train_scaled = scaler.transform(X_train)
X_val_scaled = scaler.transform(X_val)
```

We use the imported library sklearn to split the data into the training set and the validation set for both the ‘X’ and ‘y’ values. The ‘X’ values are an array of all of the features, and the ‘y’ values are the actual stroke outcome. We then scale all of the features and labels by computing the z-score of each value (Fig 9). This standardizes all of the values so that they can be compared on the same scale. These values can be unscaled at the end to see the true numbers predicted.

Figure 10. Activating and training the model

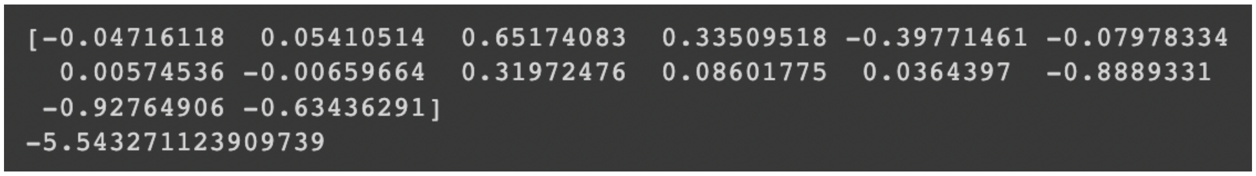


```
model = 0
model = Sequential()

model.add(Dense(1, activation = 'sigmoid'))

model.compile(loss = 'binary_crossentropy', metrics = ['accuracy'])
model.fit(X_train_scaled, y_train, epochs = 100, verbose = 0, validation_data = (X_val_scaled,y_val))

J_list = model.history.history['loss']
plt.plot(J_list)
```

Subsequently, we can train the model using the training and validation set. Using the imported library Keras from Tensorflow, we can use machine learning and logistic regression techniques to fit the model to the data. The model has one layer, and we choose a sigmoid function as the activation function for the model (Fig 10).

3. Results

Figure 11. Array of Parameters



```
[-0.04716118  0.05410514  0.65174083  0.33509518 -0.39771461 -0.07978334
  0.00574536 -0.00659664  0.31972476  0.08601775  0.0364397  -0.8889331
 -0.92764906 -0.63436291]
-5.543271123909739
```

Using all of the software described above paired with the aforementioned logistic regression techniques, the model that best fits the data was determined to have the parameters enumerated in Figure 11.

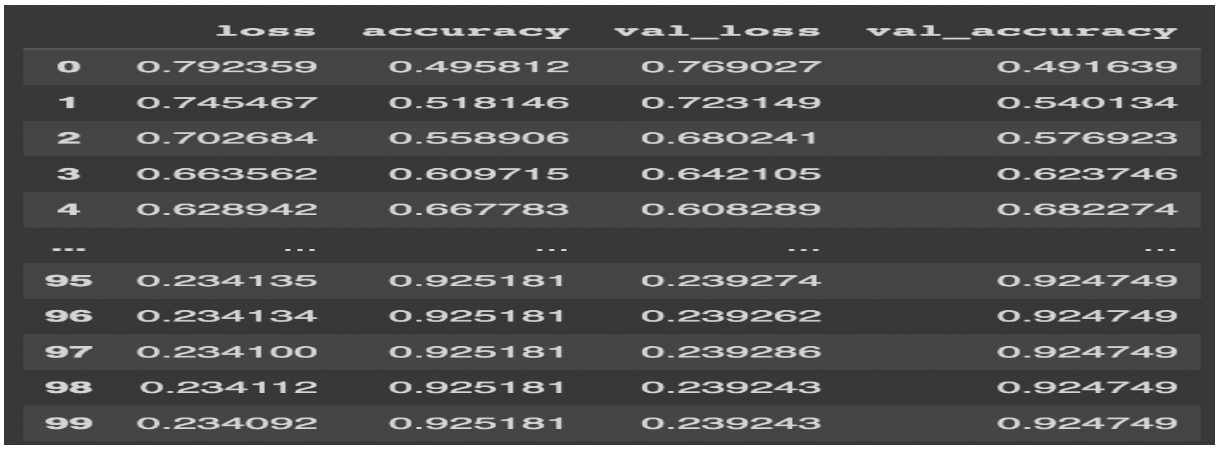
$\hat{y} = \sigma(c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 + c_6x_6 + c_7x_7 + c_8x_8 + c_9x_9 + c_{10}x_{10} + b)$

$= \sigma((-0.04716118)x_1 + (0.05410514)x_2 + (0.65174083)x_3 + (0.33509518)x_4 + (-0.39771461)x_5 + (-0.07978334)x_6 + (0.00574536)x_7 + (-0.00659664)x_8 + (0.31972476)x_9 + (0.08601775)x_{10} + (0.0364397)x_{11} + (-0.8889331)x_{12} + (-0.92764906)x_{13} + (-0.63436291)x_{14} - 5.543271123909739)$

where x_1 = gender, x_2 = age, x_3 = hypertension, x_4 = heart disease, x_5 = ever married, x_6 = residence type, x_7 = average glucose level, x_8 = bmi, x_9 = private job, x_{10} = self-employed, x_{11} = government job, x_{12} = formerly smoked, x_{13} = never smoked, and x_{14} = smokes.

This means that for any random patient, plugging in all fourteen of these values into the equation above would result in the model's predicted values between 0 and 1. If this value is greater than 0.5, the patient is predicted to have a stroke, and if it is below 0.5, the patient is predicted to be healthy.

Figure 12. Loss and accuracy of model

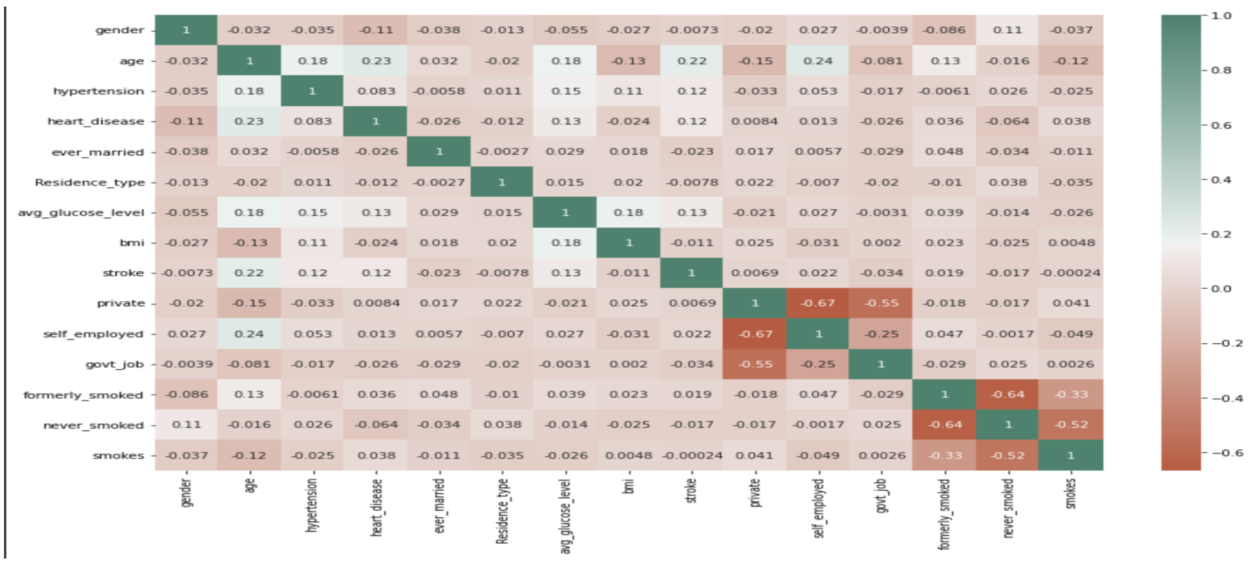


	loss	accuracy	val_loss	val_accuracy
0	0.792359	0.495812	0.769027	0.491639
1	0.745467	0.518146	0.723149	0.540134
2	0.702684	0.558906	0.680241	0.576923
3	0.663562	0.609715	0.642105	0.623746
4	0.628942	0.667783	0.608289	0.682274
...	...	...	...	...
95	0.234135	0.925181	0.239274	0.924749
96	0.234134	0.925181	0.239262	0.924749
97	0.234100	0.925181	0.239286	0.924749
98	0.234112	0.925181	0.239243	0.924749
99	0.234092	0.925181	0.239243	0.924749

The performance of a machine learning model is evaluated using the accuracy metric. Accuracy is calculated by dividing the total number of correct predictions by the total number of predictions. This metric is typically expressed as a number between 0 and 1 where a larger number closer to 1 indicates a model with better fit and a smaller number closer to 0 indicates a model with poor fit. Accuracy provides an especially useful and reliable insight into the model when the data is balanced, which we did in the preprocessing of our data.

Our model had an accuracy of 0.925181 on the training set and 0.924749 on the validation set after 100 epochs (Fig 11). In other words, the model's predictions were correct approximately 92.5% of the time. With each epoch the accuracy increased, but began to plateau at around 50 epochs. This means that 92.5% is a safe estimate, and running the model with additional epochs would simply require unnecessary calculations. The loss also reduced with each epoch, and ended at around 0.234092 after 100 epochs. Loss value implies how poorly or well a model behaves after each iteration (epoch) of optimization. In other words, the sum of all of the errors was 0.234092.

Figure 13. Correlation Matrix of the Data



A correlation matrix shows how all the features and labels are related to each other. A positive correlation between two variables means that as one increases or decreases, so does the other. Likewise, a negative correlation means that as one increases, the other decreases, and vice versa. Correlations are always between -1 and 1, and the closer the absolute value of the correlation is to 1, the higher correlation exists between the two variables. By observing which variables have a strong positive correlation, we see features have the biggest effect on the label as well as which features could be confounding with each other.

From the correlation matrix created using the seaborn package, the features that have the highest positive correlation with the label are age (0.22), hypertension (0.12), heart disease (0.12), and average glucose level (0.13).

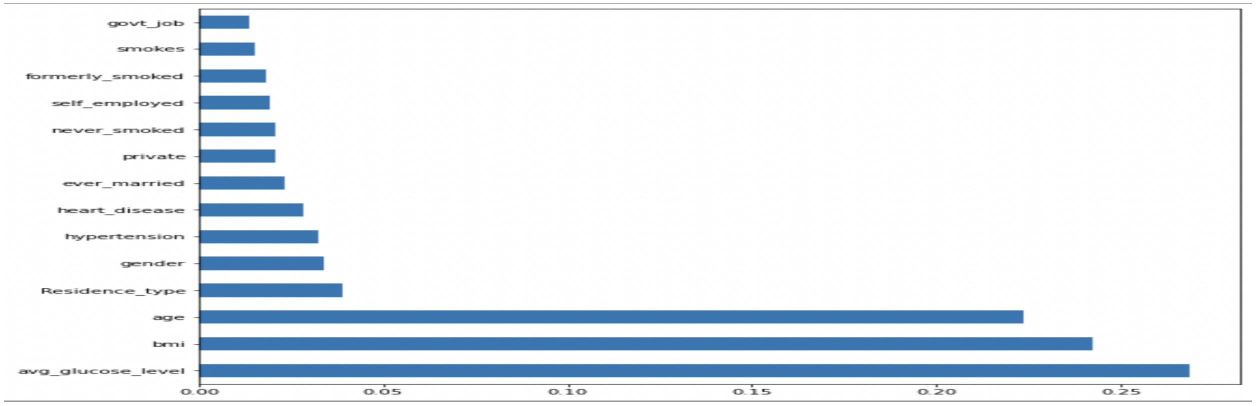
Age is also positively correlated with hypertension (0.18), heart-disease (0.23), average glucose level (0.18), self-employed (0.24), and formerly smoked (0.13).

Hypertension is also positively correlated with average glucose level (0.15), and BMI (0.11).

Heart disease is positively correlated with average glucose level as well (0.13).

Average glucose level is highly correlated with BMI (0.18) (Fig 12).

Figure 14. Feature Importances



Another way to analyze correlations is through feature importances. In order to implement this technique, we had to import Random Forest. Random Forest combines the output of multiple decision trees to reach a single result **[1]**. Adding another method for determining the relevance of each independent variable allows for better certainty.

According to the Random Forest feature importances graph, the only three features with significant correlation to having a stroke are age, bmi, and average glucose level at around 0.225, 0.245, and 0.275, respectively. This agrees with the correlation matrix in that average glucose level and age both factor into stroke risk, but the Random Forest algorithm classifies bmi as significant when the correlation matrix does not.

4. Discussion

Our model had a 92.5% accuracy in predicting stroke in patients. This is a very high value and shows that the algorithm models the data points very well. It is useful and precise in predicting the outcome for new patients. The loss or error was also relatively low, implying that not only does the model accurately predict the data points, but that it also understands the trends in the data. This means that there was little to no overfitting that would skew the results.

The features present in our study that have a visible effect on stroke risk are age, hypertension, heart disease, average glucose level, and bmi, according to the two different methods. Age is also correlated with hypertension, heart disease, and average glucose level, so it is plausible that all of them impact stroke risk. This result is reasonable because as humans age, they become more prone to different medical conditions, like heart disease and hypertension. Furthermore, bodily functions slow, and glucose levels and bmi can also rise. So, it is evident that age is the best metric for stroke prediction. As age rises, so do hypertension, heart disease, average glucose level, and bmi, which significantly increases stroke risk.

In the future, this model could be improved with more patients and more features to check if there are other factors that lead to strokes, with the ultimate goal of reducing the 140,000 deaths each year from this condition **[7]**.

References

[1] IBM Cloud Education. “What Is Random Forest?” *IBM*, https://www.ibm.com/cloud/learn/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems.

[2] IBM Cloud Education. “What Is Supervised Learning?” *IBM*, https://www.ibm.com/cloud/learn/supervised-learning.

[3] Kumawat, Dinesh. “7 Types of Activation Functions in Neural Network.” *Analytics Steps*, https://www.analyticssteps.com/blogs/7-types-activation-functions-neural-network.

[4] Kumawat, Dinesh. “Introduction to Logistic Regression - Sigmoid Function, Code Explanation.” *Analytics Steps*, https://www.analyticssteps.com/blogs/introduction-logistic-regression-sigmoid-function-code-explanation.

[5] Mitchell, Tom M. *Machine Learning*. MacGraw-Hill, 1997.

[6] Rossi, Richard, and Richard Rossi. *Mathematical Statistics: An Introduction to Likelihood Based Inference*. John Wiley & Sons, Inc., 2018.

[7] Stroke Awareness Foundation. “Stroke Facts & Statistics.” *Stroke Awareness Foundation*, 23 Jan. 2021, https://www.strokeinfo.org/stroke-facts-statistics/.

Acknowledgements

We would like to thank Dr. Guillermo Goldsztein for helping to teach us about machine learning and logistic regression, as well as helping us find a usable dataset. We would also like to acknowledge Kaggle for allowing us to use the dataset.