Stroke Prediction: Lessons About Biased Models Trained On An Unbalanced Dataset

Introduction

A stroke happens when a part of the brain receives a lack of blood flow, usually caused by a blocked artery or internal hemorrhage. If there is not a steady supply of blood to the cells, they will start to die from lack of oxygen, which can lead to death. Strokes are the number two cause of death worldwide.

This is why developing a model that can predict whether or not a person is going to have a stroke based on a number of factors is essential for the health industry. In this case, a sample stroke predictions dataset was obtained through kaggle.com, but the source of the dataset is confidential and thus the data source is unknown. The dataset contains various attributes of each data sample (or patient) and whether or not they have had a stroke in the past. These attributes are shown in the table below:

Feature	Description
ID	Unique identifier
Gender	"Male", "Female", or "Other"
Age	Age of the patient
Hypertension	0 if the patient doesn't have hypertension, 1 if the patient has hypertension
Heart disease	0 if the patient doesn't have any heart diseases, 1 if the patient does have a heart disease
Ever married	"No", or "Yes"
Work type	"Children", "Government job", "Never worked", "Private", "Self employed"
Residence type	"Rural" or "Urban"
Average Glucose Level	In the blood
BMI	Body mass index
Smoking status	"Fromerly smoked", "Never smoked", "Smokes", or "Unknown"*
Stroke	1 if the patient had a stroke, 0 if not

*The information is not available for this patient.

Data Cleaning and Preprocessing

Before developing a model for stroke prediction, it is important to consider that the imported dataset is in a raw format where the features have not been processed or cleaned. It is ideal to clean and process the data before model development so that we train the model using the data desired and make accurate predictions. I used Python as the main programming language and VS Code as the code editor. Multiple Python packages were used in the process, but will not be mentioned here for the sake of simplicity. The original data has shape (5110, 12), that is, it has 5110 rows and 12 columns. Each row is representative of one specific patient. 11 of the columns are the features describing each patient, and the last column, “Stroke”, will be the label output we would want to predict.

Dropping Columns and Checking for Duplicates

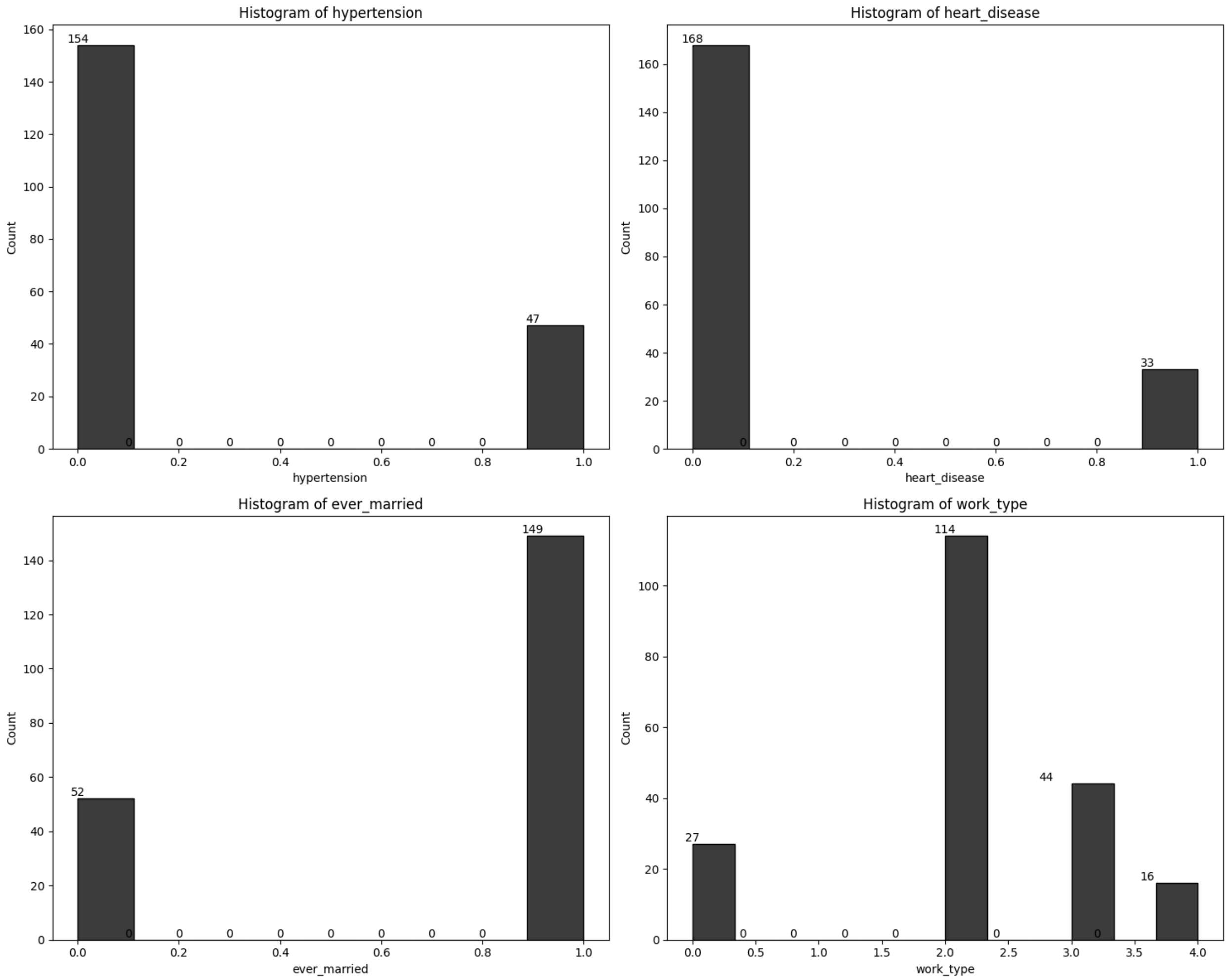
After importing the dataset, dropping unnecessary columns is needed.The “ID” column was dropped, as this is just a unique identifier for a specific patient and will not help in predicting stroke. Checking if there were any duplicates in the data is also important, but there weren’t any duplicates. There are no features that can be a combination of other features.

Dealing with N/A Values

It is seen in the dataset that the BMI column has 201 N/A values. These could be because there is not enough information about BMI for these patients. One way to approach this problem is by removing the data that contains N/A BMI values. However, considering that there are only 5110 data points, this is already a low enough dataset to remove 201 values which will make a more biased model, thus dropping this data shouldn’t be considered.

Another approach is finding the mean of the numerical BMI values and replacing the N/A values with the mean. This way, decreasing the number of values in the dataset and introducing more bias into the model is avoided. There are 249 total positive Stroke values in the original dataset, and 40 of those are in patients that contain N/A values in their BMI. This is 13% of the total amount of positive Stroke values, a significant amount of positive classes that shouldn’t be dropped, further suggesting the approach of replacing N/A values with their mean.

Let’s explore what relationships there are between the patients that contain N/A values for their BMI



For patients with N/A values for BMI, it appears that most were married at some point, haven't had hypertension, no heart disease detected, and the majority work for private companies. The correlation between these findings and why do these patients have N/A values for their BMI is difficult to pinpoint. However, it is possible that for people that work for private companies (the majority), their companies don't allow them to share too much personal information. Another reason could be that these patients don't feel comfortable sharing this information for the data collection, or simply don't know their BMI. These graphs show the labels as encoded features, which will be discussed in the next section.

Label Encoding

Label encoding is a technique that converts categorical data into numerical values. It is a common technique used in machine learning as models often require numerical input. Label encoding works by assigning each category to a unique integer. An example is assigning gender values as follows: Male = 0, Female = 1, Other = 2.

The following input features were label encoded in the given dataset: Gender, Ever Married, Work Type, Residence Type, Smoking Status.

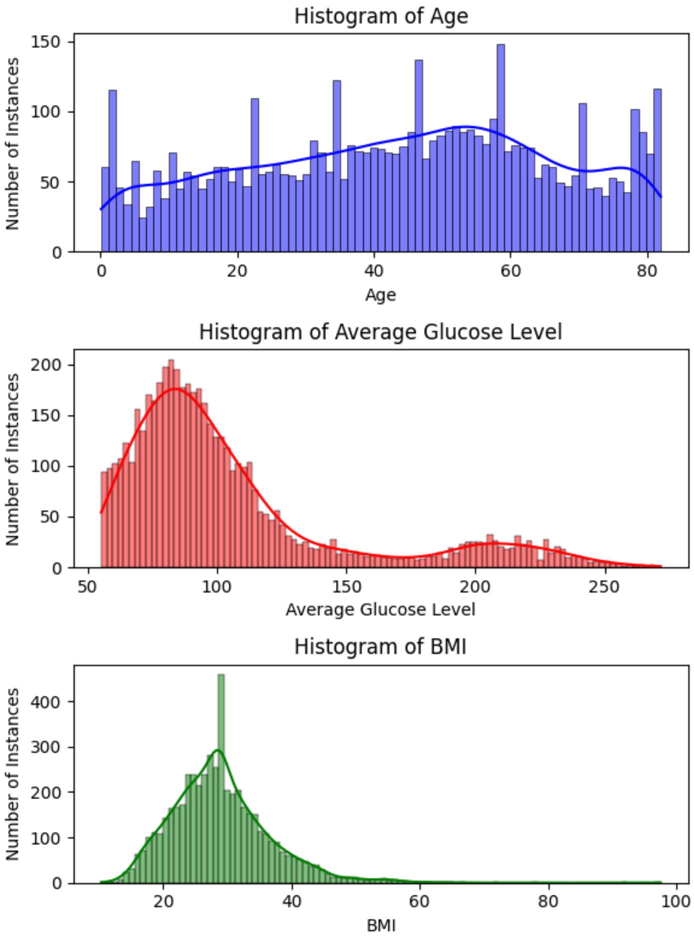
Here’s a legend for the encoded features:

Feature	Encoded value
Gender	“Male”=0, “Female”=1, “Other”=2
Ever_married	“No”=0, “Yes”=1
Work_type	“Gov job”=0, “Never worked”=1, “Private”=2, “Self employed”=3, “Children”=4
Residence_type	“Rural”=0, “Urban”=1
Smoking_status	“Unknown”=0, “Formally smoked”=1, “Never smoked”=2, “Smokes”=3

Scaling

Feature scaling is used when some features of the dataset have different units and magnitudes than others. If the dataset is not scaled, then the model will tend to become more biased to those values that have a bigger magnitude. Scaling data prevents this bias from happening, and makes the data easier and faster to process for the model.

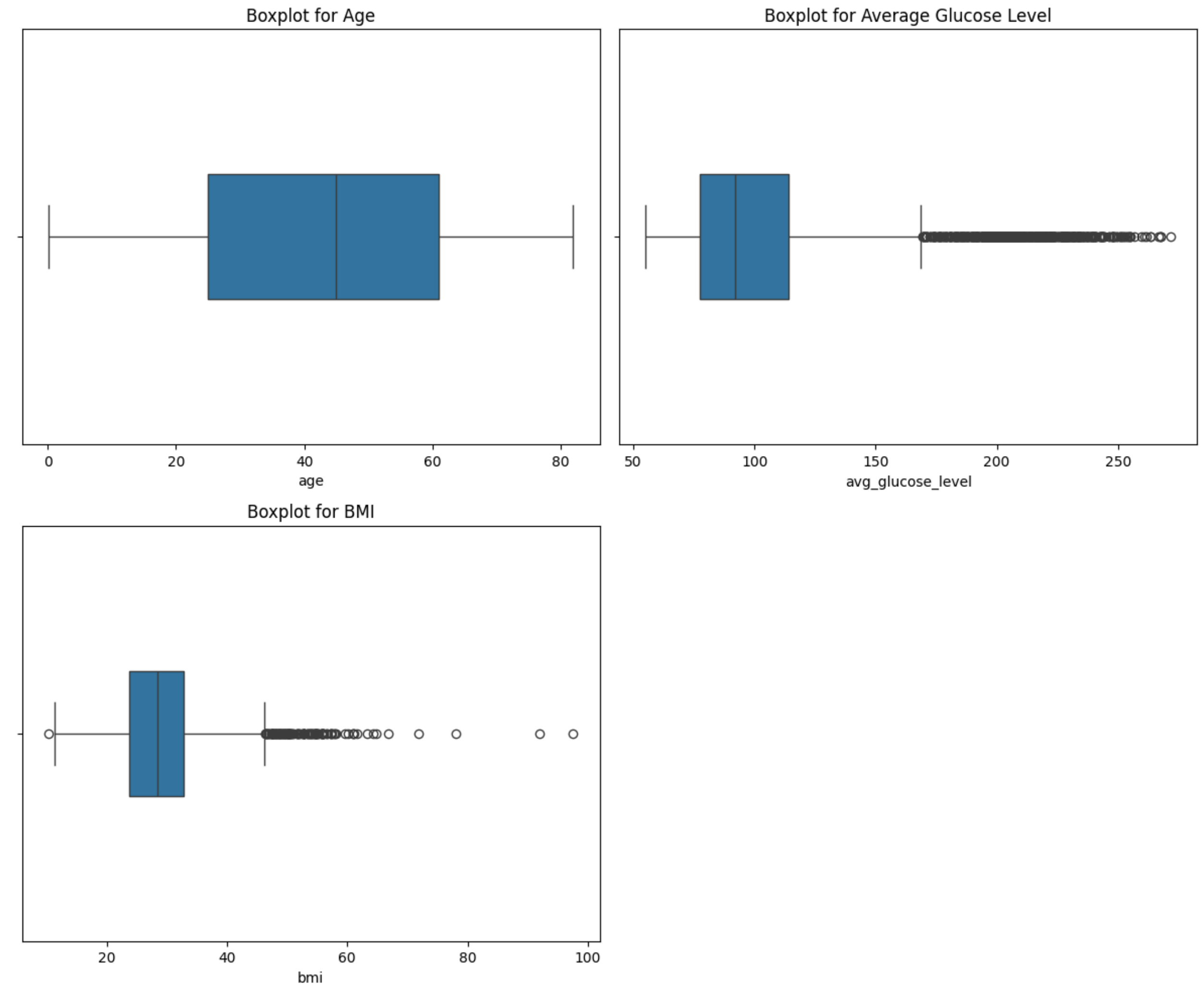
Data that contains discrete values, like in Gender where the values are “Male”, "Female", or “Other”, do not require scaling, as these values function as classifiers. Continuous values like BMI, Average Glucose Level, and Age do need scaling as they differ in ranges and magnitudes. First, let’s see if these features contain a normal, or Gaussian distribution:



A normal distribution is a type of distribution of data that is symmetric about its mean, meaning most data points cluster around the center, where the mean is located, with the probabilities decreasing equally on both sides the further you go from the mean. The standard deviation determines the spread of the distribution. A small STD means a steeper curve, while a larger STD means a wider curve.

There is a function used to scale normally distributed data, called StandardScaler from scikit learn, or the Z-score scale. The mathematical formula for the StandardScaler is: $X_{new} = (X_i - X_{mean})/STD$. This scaler is vulnerable to outliers because outliers affect the mean, so StandardScaler won't perform well if outliers are present. Additionally, a normal distribution has a distinct bell-shaped curve and is centered around the mean, but it appears that none of the features to be scaled look normally distributed. Therefore, we cannot use standardization.

Let's see if there are any outliers in the data to be scaled. We can visualize outliers by using boxplots as shown below:



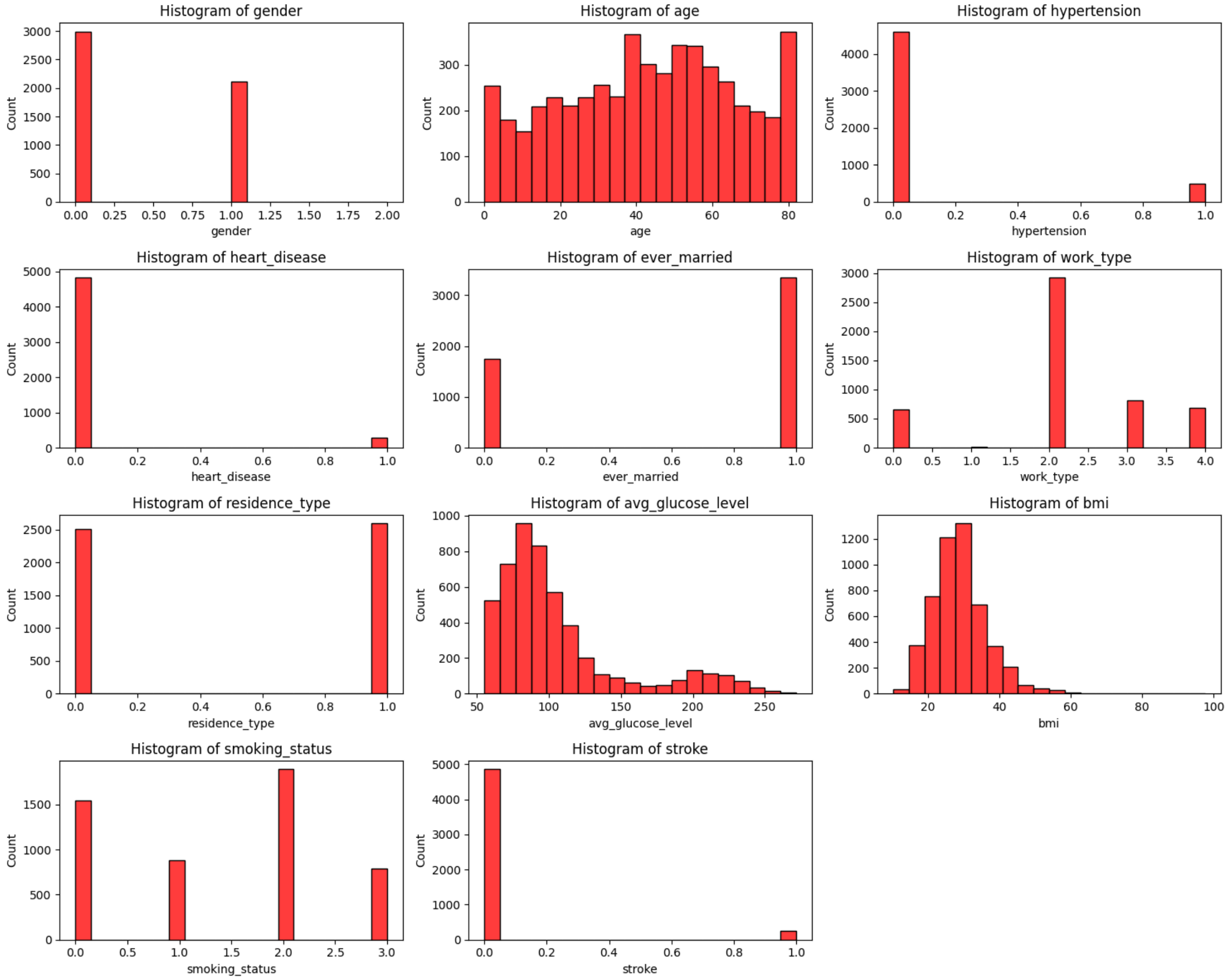
The whiskers of the box plot represent the minimum and the maximum values of the data. The line dividing the body of the box is the median, which is the value that divides the total data in half. If you look from the median line to the leftmost whisker, that is 50% of the data. Same goes for the right side of the box plot. The body of the box is composed of the 2nd and 3rd quartile, and that can also be considered 50% of the data. The 1st and 4th quartile are measured from the whiskers to the edges of the body. In simple terms, the box plot divides the total data into 4 parts, and each part contains 25% of the data. This is what is called the interquartile range, or IQR.

The dots that go beyond the whiskers are the outliers, and this is what we're looking for. It looks like "BMI" and "Average Glucose Level" have a significant number of outliers. A good scaler that handles outliers well is the Robust Scaler because it uses the median and the IQR instead of the mean and standard deviation, making it robust to outliers. Here's the formula to scale the data using Robust Scaler: $X_{new} = (X - X_{mean})/IQR$. As you can see, Robust Scaler is a median-based scaling method. Since it does not use the mean, it is not affected by outliers. Robust Scaler will be used to scale "BMI" and "Average Glucose Level".

The “Age” data doesn’t look like it has any outliers, therefore using MinMaxScaler is more appropriate. This scaler calculates the difference between the minimum and maximum values of the total data and divides each data point by this difference. MinMax Scaler will transform the data range from 0 to 1. Here’s the formula: $X_{new} = (X_i - min(X))/(max(X) - min(X))$.

There are other types of scalers, like the AbsMax Scaler, which divides the data by the absolute maximum of the data range, keeping the negative values; and the Log Scaler, which is used for data that increases (or decreases) exponentially, but these are outside the scope of this project and are not necessary.

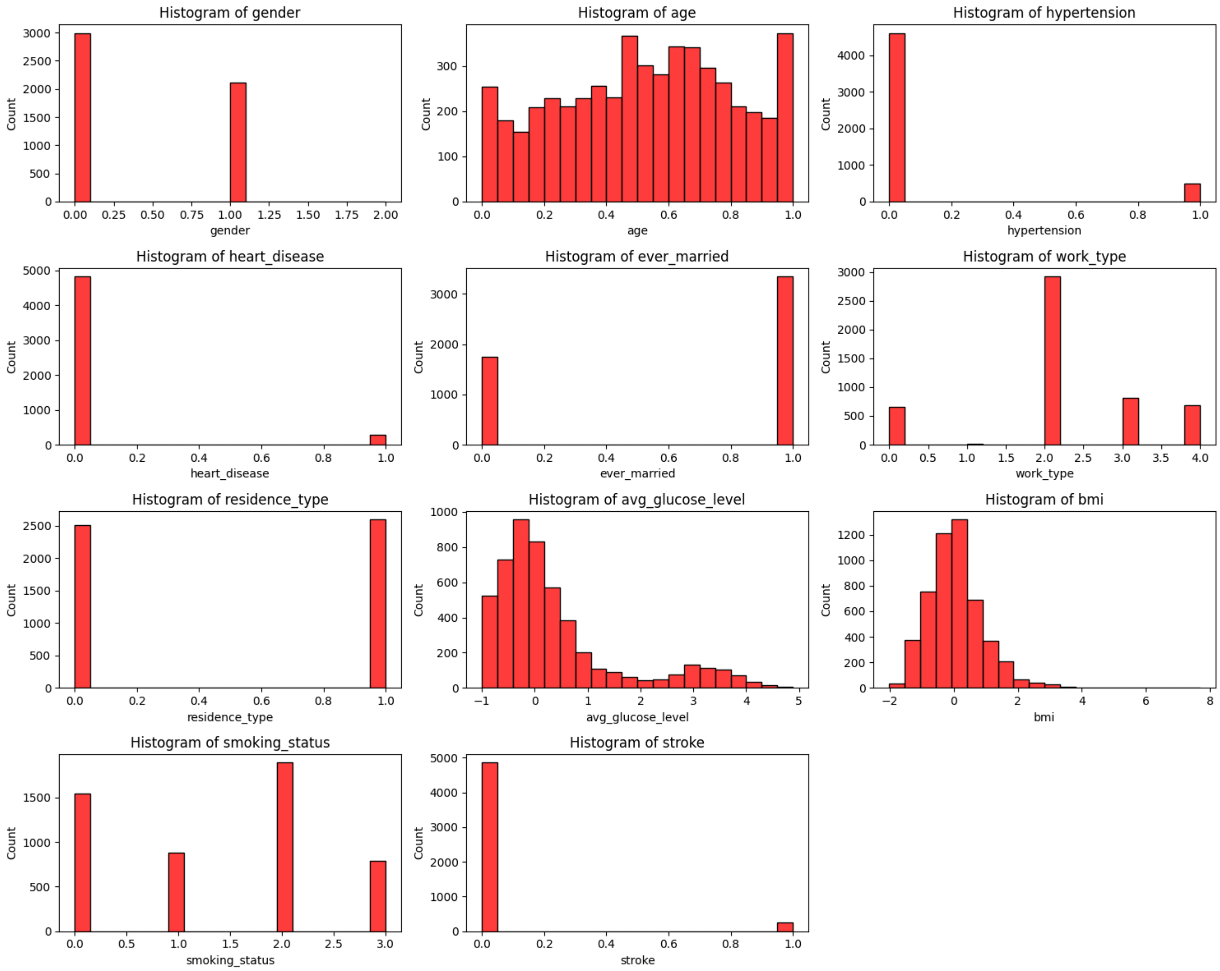
Here are the histograms of each feature **before** scaling continuous data:



Some statistics about the continuous data (before scaling):

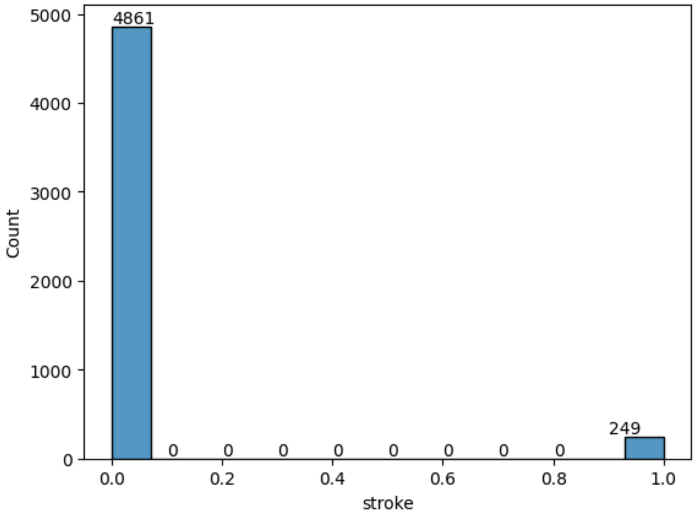
	Age	Avg glucose level	BMI
Count	5110	5110	5110
Mean	43.226	106.148	28.893
STD	22.613	45.284	7.698
Min	0.08	55.120	10.300
25%	25	77.245	23.800
50%	45	91.885	28.400
75%	61	114.090	32.800
Max	82	271.740	97.600

Here are the histograms of each feature **after** scaling continuous data:



Bias

It is important to consider the distribution between the output data, or stroke counts, amongst the entire data. There are a total of 5110 samples in this dataset, 4861 of which haven't had a stroke, and only 249 patients did have a stroke in the past. That is, 95% of the data is "No Stroke" while 5% is "Stroke".



This creates a huge imbalance in the output, causing our model to be biased towards the "No Stroke" label. Imbalances are not ideal when training a model, since you want the output data in a binomial model prediction like this one to be close to 50% one label, and 50% the other. I could have balanced out this data set by introducing more patient data from people who have had a stroke, either by conducting more studies or synthetically generating it, but considering that this is medical data, there are both moral and legal implications involved that I would prefer to stay away from.

I could have also cut down on the number of samples that contain the "No Stroke" label, but considering that this data set already has 5110 samples, this is a low enough number that if I remove just a small portion, it will make the model overfit the training data and be a terrible model for new data. I will keep the data as it is, keeping in mind my hypothesis that the model created will be biased. Let's explore that in the next section.

Creating the Model

The data used in this project has multiple inputs (also called features), and only one output with two classes: Stroke or No Stroke. The goal is to classify a patient if they have had a stroke in the past (1) or not (0), based on known stats about the person. This can be considered a binary classification problem. There are a number of models we can choose from for binary classification. One of the most common models used for this type of problem is logistic regression. As one can see, the focus is on Supervised Learning, since each input example is paired with a label or output that the model is expected to predict.

Splitting the Data

In order to train the model, it is important to split the data into training and test data. The training data is responsible for training the model, and the test data is responsible for introducing new and unseen data to the model in order to measure its accuracy. The data was split as follows: 80% training, 20% test.

Logistic Regression Model

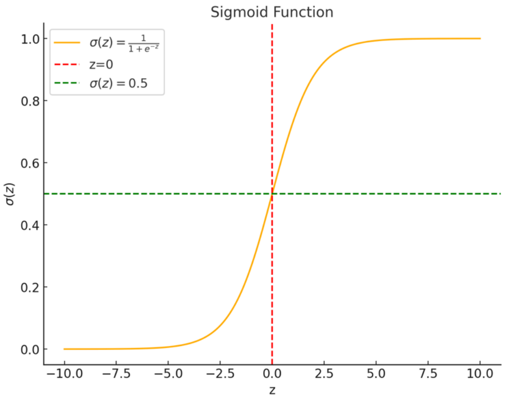
A logistic regression model is used when the outcome is categorical. It takes in the input features and combines them in a linear fashion using weights and biases as shown:

$$z = w_1x_1 + w_2x_2 + ... + w_nx_n + b$$

Where z is the weighted sum of features, w1, w2, …, wn are the weights, x1, x2, …, xn are the input features, and b is the bias term.

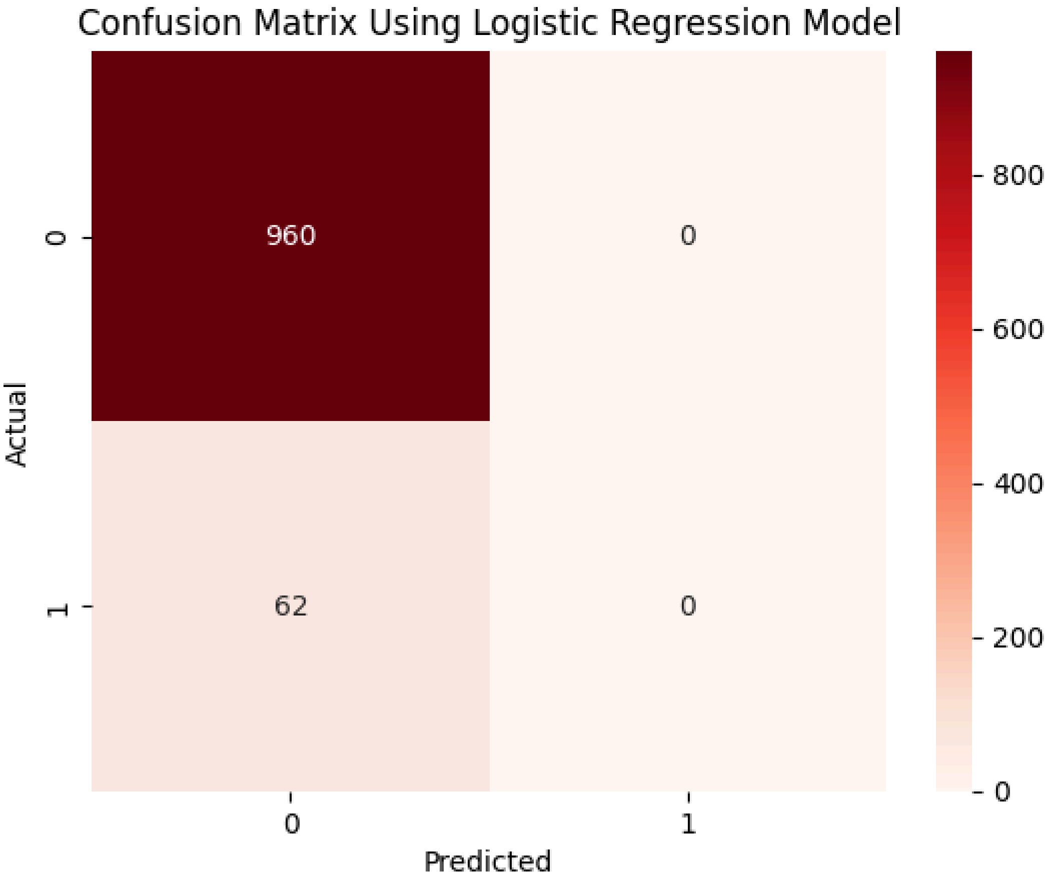
This model uses a function called the sigmoid, shaped like an S, and predicts the probability that a given input belongs to a specific class. The sigmoid function takes the output and turns it into a probability range that is between 0 and 1, and this is the probability of the outcome being one of the classes. The sigmoid function looks like this:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



A threshold is used, typically 0.5, in which the model relies on to classify a probability. If the probability of a given input is within 0.5 to 1 then the model will predict that the input belongs to class 1, whereas if the probability of a given input is within 0 to 0.5, then the model will predict that the input belongs to class 0. Logistic regression models work well for predicting binary categories, like True/False, Positive/Negative, or 0/1.

To evaluate the performance of the logistic regression model, I used accuracy and a confusion matrix. The model got an accuracy of 93.93%, which is pretty high. An accuracy this high means that the model has a high bias towards the majority class (0) and performs poorly on the minority class (1). The model is just predicting the majority class every time, so the accuracy will be higher than expected. A confusion matrix was used to evaluate the performance by comparing the predicted values to the actual values.



It is shown that the model predicted all 960 actual values of “0” correctly, but in reality, the other 62 values are “1” but the model predicted them to be “0”. This decision matrix shows the bias in the model, since there are 62 actual values that should have been labeled as “1”, but the model predicted them to be “0”. This problem arises from the imbalance in the dataset.

Potential future adjustments that can be made to the model include: adjusting the threshold value, adjusting the weights of the model, and checking AUC and ROC Curves. These are potential solutions to make the model more accurate; however, it might still be a stretch to have an accurate model even after adjustments due to the uneven nature of the labels.

Precision-Recall Tradeoff

An attempt to adjust the threshold value was made in order to balance the predictions. It is important to understand the concept of Precision-Recall Tradeoff to be able to choose the correct threshold value.

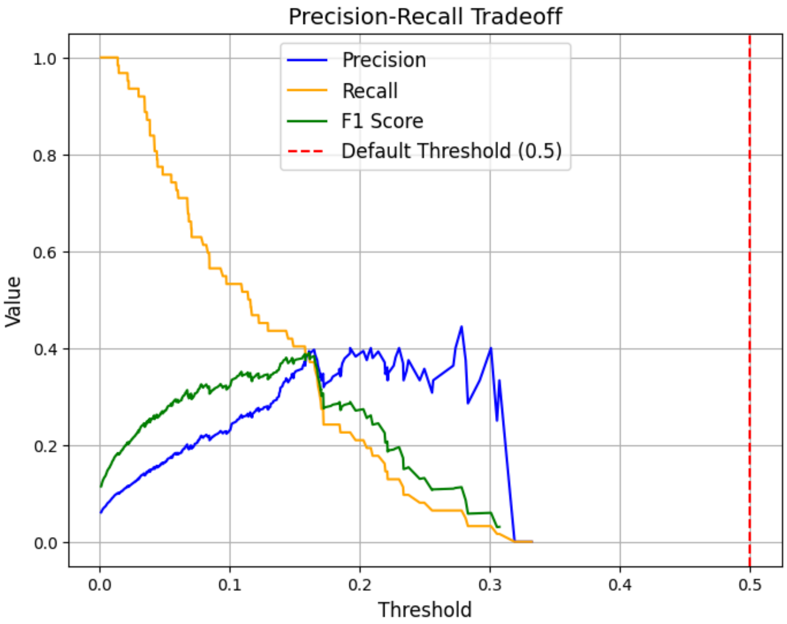
Precision is a measure of how many positive predictions were correct out of all positive predictions, while *recall* is a measure of how many positive predictions were correctly identified out of all actual positive cases. The *F1- Score* is the harmonic mean between precision and recall. A higher F1-Score indicates better overall performance.

$$Precision \ = \ \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)}$$

$$Recall \ = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Negative\ (FN)}$$

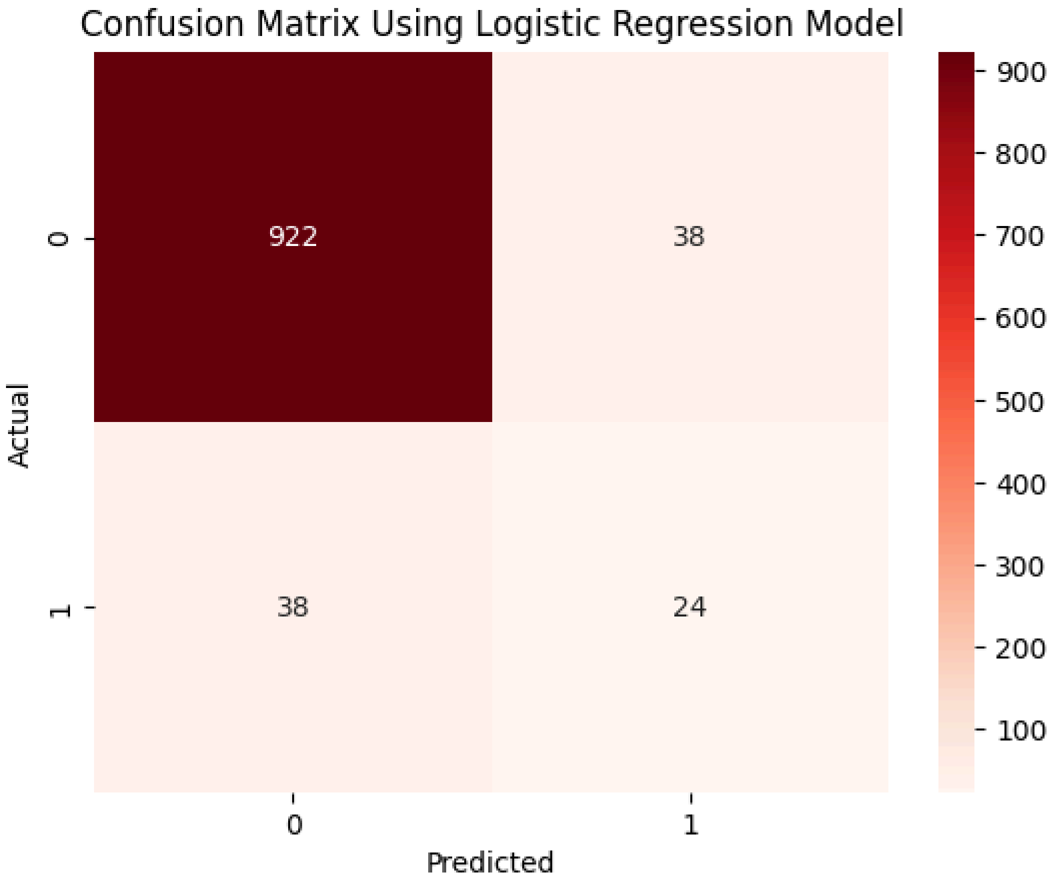
$$F1\ Score \ = \ 2 * \frac{Precision * Recall}{Precision + Recall}$$

In the case of stroke prediction, it is important to attain both high precision and high recall. This is because we would like to predict as many patients as possible who actually have a stroke (high recall), while also informing the doctor a higher number of predicted positive strokes compared to actual patients with strokes (high precision) can be useful to indicate other types of conditions. However, having both high precision and high recall is never the case. To determine the best ratio between precision and recall, the F1-Score is used. The relationship between precision, recall, and the F1-Score as the threshold increases is shown below:



One can see that precision increases as threshold increases and then sharply decreases, recall decreases continuously, and the highest F1-Score is attained around a threshold of 0.16. Since we are looking for a ratio of high precision and high recall, then choosing the highest F1-Score is necessary for better model performance. Therefore, a threshold of 0.16 was used for the logistic regression model.

Below is the new decision matrix using a threshold of 0.16 (or 16%) instead of the default 50%, along with the precision, recall, and F1-Score values:



	Precision	Recall	F1-Score
0	0.96	0.96	0.96
1	0.39	0.39	0.39

It is shown that the number of False Positives and True Positives increased, while the number of False Negatives decreased. This updated model has an accuracy of 92.6%. Even though the accuracy decreased slightly compared to the old model (93.9%), the new balance between precision and recall lets the model predict more positive values for stroke, making the model less biased even with an unbalanced dataset.

Additional Exploratory Data Analysis using PCA and K-Means Clustering

The model created using logistic regression gave us some insight on the distribution of the data. It is safe to say that the data used is highly unbalanced and therefore created a highly biased model, even if the classification threshold is lowered. We can explore further on the data to gain some more insight on the relationships between the features used to run the model by reducing the data's dimensionality and/or classifying the data into different clusters.

Principal Component Analysis (PCA)

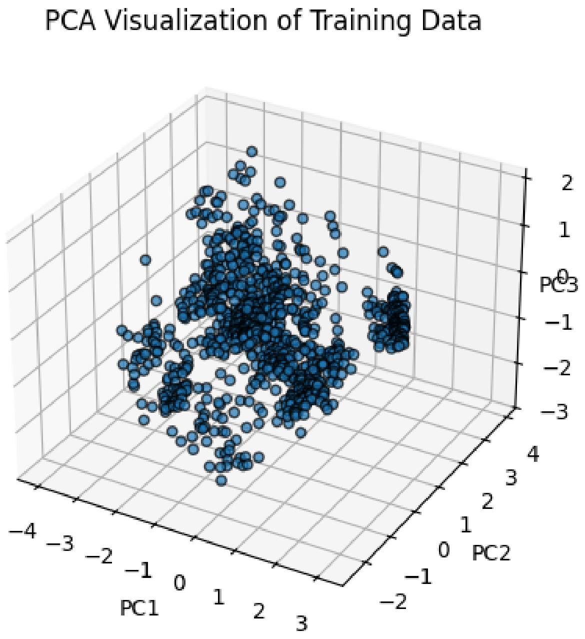
Principal component analysis, or PCA, is a dimensionality reduction technique used in datasets with a high number of dimensions while preserving as much information, or variance, as possible. It transforms the original features into uncorrelated variables called principal components, which are linear combinations of the original features. PCA is usually used in ML to reduce computational complexity, improve visualization, and to extract the most important features of a dataset.

The idea of PCA is to keep the highest information possible while simplifying the dataset by removing unnecessary features that do not contribute to the variance as much as the other features. The first feature selected using PCA will always have the highest variance, the second feature will be second in highest variance, and so on.

First, the output column for Strokes was dropped. This is because we would like to run K-Means Clustering later, an unsupervised learning method, and these models do not require an output label. Now there's a total of 11 features, making an 11 dimensional dataset. It is practically impossible to visualize an 11 dimensional space, highlighting the importance of PCA to reduce the data's dimensions from 11D down to 2D or 3D. This way we can visualize the data in a 2D or 3D graph.

A cumulative sum of the variances is calculated to determine the optimal reduction in dimensionality. The higher the dimensions, the higher the cumulative sum of variances. Using a 2 dimensional dataset was first attempted for easier visualization, but it turned out to have a 60% cumulative sum of variances. This means that the newly created principal components only retained 60% of the information in the dataset, which is low. An ideal cumulative sum of variances would be around 95%.

3 principal components were used instead. Using 3 principal components increased the cumulative sum of variances to 74%, which is still pretty low. However, if our goal is to visualize the data, going above 3 dimensions would be difficult to comprehend. Here is the 3D scatterplot of the 3 principal components:

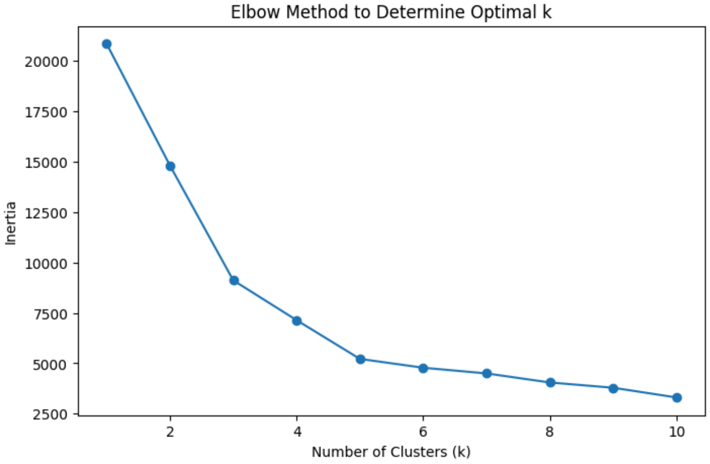


Here one can see that there are different clusters within this data. Performing K-Means Clustering will be helpful in identifying the clusters visually and how many there are.

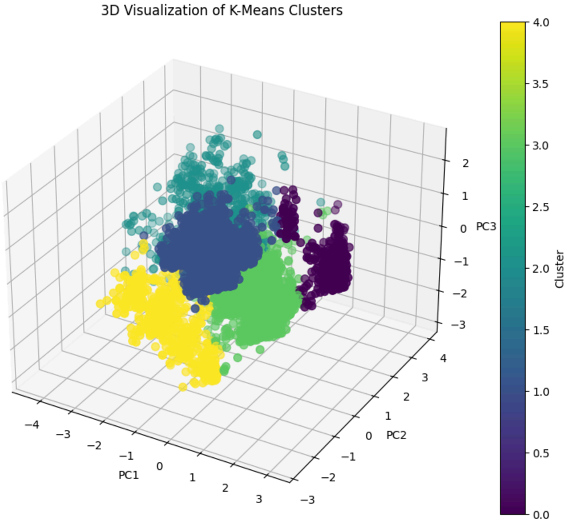
K-Means Clustering with PCA

K-Means clustering is a type of unsupervised learning that is used to cluster data into groups. The dataset is divided into k clusters, where each data point belongs to the cluster with the nearest mean (centroid). This model is used to separate data and group the data points that are similar to each other, while also finding patterns in the dataset.

To visualize K-Means Clustering, the PCA reduced dataset was used with 3 principal components. However, it is important to determine the number of clusters that will be used to group the data. This is called the k parameter, and can be found by using the elbow method. The elbow method is a graph that represents the inertia as the number of clusters (k) increases, as shown below:



The inertia decreases as the number of clusters increases. We choose the most optimal k value at the elbow point, where adding more clusters does not significantly reduce the inertia. In this case, the most optimal k is k = 5. Using this value, we can plot K-Means Clustering with 5 clusters:

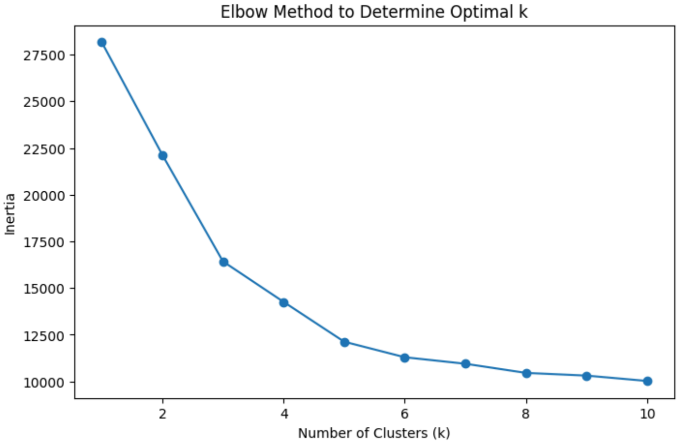


Here one can clearly see 5 clusters represented by 5 different colors. These clusters can potentially represent similarities in the features within the same cluster. The Silhouette Score is used to evaluate how well each data point fits into its assigned cluster compared to the other clusters. It has a range from 1 to -1. 1 means that the data is well structured and has a good separation. -1 means that the data is poorly clustered. 0 means that there are overlapping clusters with unclear separations.The Silhouette Score for the PCA reduced data is 0.433, meaning that our data is well clustered but there are some sections where the separations are unclear.

K-Means Clustering without PCA

Performing K-Means Clustering without PCA lets us use the total variance (information) of the original dataset, at the expense of not being able to visualize it. Using PCA with 3 principal components has a cumulative sum of variances of 74%, but to have more accurate results from K-Means, using the total variance would be ideal.

The elbow method was used again for the original dataset that has not undergone PCA, shown below:



Again, a total of 5 centroids is the most optimal to use. Performing K-Means Clustering on the non-PCA dataset gave us a Silhouette Score of 0.272. This is because we introduced more variance into the dataset, making the model more realistic (remember, a cumulative variance of 74% that was found during PCA is not really a good amount of variance).

The clustered data cannot be visualized at this dimensionality, but we can interpret the statistics of each centroid. Average glucose level, BMI, and age were the only features that were used for statistical analysis of centroid since these are the only continuous values. Let's look at the coordinates of each centroid:

k	gender	age	hypertension	Heart Disease	Ever Married	Work Type	Residence Type	Avg Glucose Level	bmi	Smoking Status
0	0.367	0.551	0.098	0.043	0.715	2.207	0.506	-0.044	0.147	2.302
1	0.479	0.727	0.255	0.156	0.883	1.982	0.512	3.137	0.500	1.603
2	0.372	0.591	0.088	0.043	0.789	0.024	0.528	-0.028	0.178	1.621
3	0.409	0.590	0.070	0.049	0.737	2.214	0.503	-0.044	0.156	0.453
4	0.526	0.087	0.001	0.001	0.001	3.978	0.503	0.031	-0.938	0.131

Now let's look at the statistics for the continuous features, both with the scaled and unscaled data for each feature:

Age (scaled)	
count	5110
mean	0.527
std	0.276
min	0
25%	0.304
50%	0.548
75%	0.744
max	1

Age (unscaled)	
count	5110
mean	43
std	23
min	0
25%	25
50%	45
75%	61
max	82

Avg Glucose Level (scaled)	
count	5110
mean	0.387
std	1.229
min	-0.998
25%	-0.397
50%	0
75%	0.603
max	4.881

Avg Glucose Level (unscaled)	
count	5110
mean	106.148
std	45.284
min	55.120
25%	77.245
50%	91.885
75%	114.090
max	271.740

bmi (scaled)	
count	5110
mean	0.055
std	0.855
min	-2.011
25%	-0.511
50%	0
75%	0.489
max	7.689

bmi (unscaled)	
count	5110
mean	28.893
std	7.698
min	10.3
25%	23.8
50%	28.4
75%	32.8
max	97.6

Interpretation:

- Cluster 1: People in this cluster tend to have an age close to the mean value of 43 years, have an average glucose level close to 50% of the distribution with a value of 91.885, and have a BMI that is close to the mean of 28.893.

- Cluster 2: People in this cluster tend to have an age at 75% of the distribution which is 61 years, have an average glucose level at the maximum of the distribution with about 271.74, and a BMI at 75% of the distribution with a value of 32.8.
- Cluster 3: People in this cluster tend to have an age at 50% of the distribution which is 45 years, have an average glucose level at 50% of the distribution which is 91.885, and a BMI at around the mean of 28.893.
- Cluster 4: People in this cluster tend to have an age at 50% of the distribution which is 45 years, have an average glucose level at 50% of the distribution with about 91.885, and a BMI at around the mean of 28.893.
- Cluster 5: People in this cluster tend to have an age at the minimum of the distribution in the range of younger people, an average glucose level at 50% of the distribution with about 91.885, and a BMI level close to 25% of the distribution at 23.8.

Conclusion

Predicting whether a person will have a stroke or not can be very helpful in the medical field. Doctors can benefit from a model like this to warn their patients in advance and prevent having a higher risk of stroke. Given an unbalanced dataset, with almost all of the output data being labeled as “No Stroke” and just a few being labeled as “Stroke”, an attempt to develop a model that can predict whether or not a person will have a stroke took place.

After preprocessing and visualizing the data, a logistic regression model was built. The model turned out to have a very high accuracy, which is a good sign for most models. However, for this specific data, having a high accuracy with very unbalanced data means that the model is biased towards the output that is most abundant, predicting nearly none of the data that is less abundant.

Different techniques to balance the data can be applied, including the introduction of more data from patients that had a stroke or synthetically generating it, but there are legal and moral implications involved, so these options were discarded. Lowering the classification threshold of the model is a good way to deal with this imbalance by introducing a higher recall, at the expense of precision. A threshold of 16% was used instead of the typical 50%. This gave a more reliable model, but still wouldn’t be useful for real medical implications.

Even though this model isn’t suitable for accurate predictions due to the unbalanced nature of the dataset, an exploratory approach of the data can be useful. PCA was performed to visualize the data, and K-Means Clustering was performed to classify the data into different clusters. After classifying 5 unique clusters in this data, some interesting relationships were observed between different continuous features within each cluster, giving more insight into the nature of this dataset.