Axel Chabbert and Sierra Rossman

Dr. Rhodes

DS 352: Machine Learning

14 December 2022

Stroke Risk Assessment: Final Project Submission

Project Description:

Stroke's are the second leading cause of death globally, according to the World Health Organization, so it is important to be able to predict the likelihood of an individual having a stroke in order to be proactive in prevention and treatment. It has been proven that over the past 30 years, the number of strokes experienced by adults 49 years old and younger have increased in the U.S. Nevertheless, it is true that people who experience strokes are more often older adult men and women. The average age of people who are affected by this health concern is 73 years old.

This is due to the progressive degeneration of the human body. With age, bodies have a longer recovery time and may not recover completely, leading to prolonged and fatal illnesses. However, it is obvious that people who do not consistently take care of their hearts (minimal physical activity, over-eating, smoking) increase their risk of having a stroke.

This data set looks at different attributes in order to predict if an individual is likely to have a stroke. Attributes that may predict this include if the individual smokes, their gender, their age, if they experience hypertension, and more. We expect to become more aware of what attributes contribute to a higher risk of having a stroke. We also expect to be able to view what ranges or categories specifically in each attribute leads to a higher risk of stroke. For example, if many patients who have hypertension experience strokes, that would suggest, without concluding causation, that hypertension may increase an individual's risk of stroke.

Our purpose in using the machine learning algorithms on this dataset is to identify which attributes have the biggest effect on an individual's risk of having a stroke. Our results may be used by those who are interested in their health status and by those who want to understand the potential causes behind strokes.

The project aims to apply to the medical world. It will thus be able to reach doctors and hospitals caring for patients with this kind of concern. However, the patient must have undergone tests to have a clear idea of this current state of health.

Using the data that the patient has either provided or that is present in his medical file (age, hypertension, heart disease, blood sugar, smoker, etc.), the doctor can then define if the patient's risk of stroke is high. In the world of health, this kind of dataset analysis is commonly used and enormously sought after, whether in clinics (private or public) or by the liberal professions. Although the technology is still improving, these detection algorithms are beginning to play a very important role in today's healthcare world.

Data Sources:

URL of the dataset found on Kaggle:

https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/discussion?resource=download

On Kaggle, the dataset was located by browsing through datasets with medical topics. Once the dataset was found, we looked through it to ensure it had an appropriate number of attributes (12) and observations (over 5000). The dataset was downloaded as a csv file.

Data Directory:

Name	Description	Data Type	Range/Limitation	Other Notes
id	Specific ID # for each patient	Numeric continuous	Positive integers	
gender	If the patient identifies as male, female, or other	Non-Numeric Categorical	Three categories: Male, Female, and Other	
age	The age of the patient	Numeric Discrete	Positive whole numbers	
hypertension	If the patient has hypertension, or if the patient does not have hypertension	Numeric Continuous	0 if the patient doesn't have hypertension 1 if the patient does have hypertension	
heart_disease	If the patient has a heart disease, or if the patient does not have a heart disease	Binary Boolean	0 if the patient does not have a heart disease 1 if the patient does have a heart disease	
ever_married	If the patient has ever been married	Binary Boolean	Two categories: Yes and No	

work_type	What type of work the patient does	Non-Numeric Arbitrary	Five categories: children, Govt_job, Never_worked, Private, and Self-employed	
Residence_type	What kind of area the patient lives in	Non-Numeric Categorical	Two categories: Urban and Rural	
avg_glucose_level	The average glucose level in the patient's blood	Numeric Statistical	Positive decimal number	
bmi	The patient's body mass index	Numeric Continuous	Positive decimal number	
smoking_status	If the patient has ever smoked	Non-Numeric Categorical	Four categories: formerly smokes, never smokes, smokes, unknown	
stroke	If the patient has had a stroke	Binary Boolean	1 if the patient has had a stroke, 0 if the patient has not had a stroke	

Snippet of dataset:

We used one unique .csv file for this project.

	A	B	C	D	E	F	G	H	I	J	K	L
1	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
2	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
3	51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A	never smoked	1
4	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
5	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
6	1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1
7	56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smoked	1
8	53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
9	10434	Female	69	0	0	No	Private	Urban	94.39	22.8	never smoked	1
10	27419	Female	59	0	0	Yes	Private	Rural	76.15	N/A	Unknown	1
11	60491	Female	78	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1
12	12109	Female	81	1	0	Yes	Private	Rural	80.43	29.7	never smoked	1
13	12095	Female	61	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
14	12175	Female	54	0	0	Yes	Private	Urban	104.51	27.3	smokes	1
15	8213	Male	78	0	1	Yes	Private	Urban	219.84	N/A	Unknown	1
16	5317	Female	79	0	1	Yes	Private	Urban	214.09	28.2	never smoked	1

Data merging:

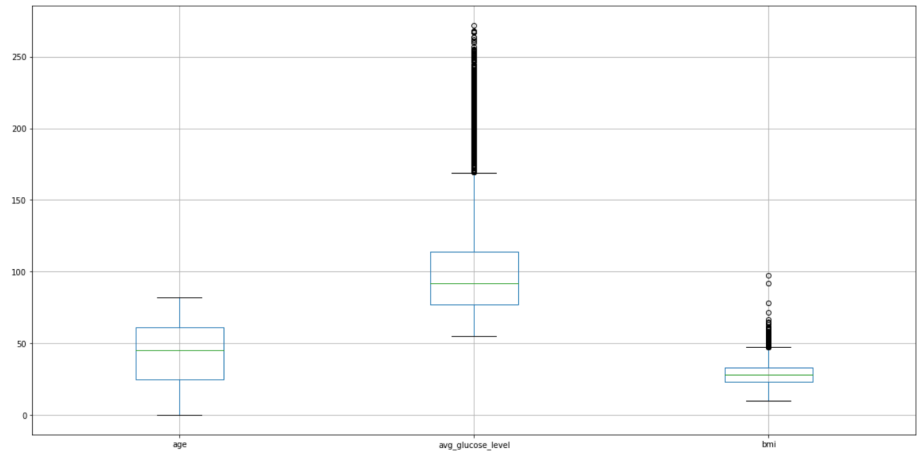
No merging of data collections is necessary for this dataset. The dataset has more than 5000 observations and twelve attributes to give us a good overall idea of what may lead to a higher risk of stroke for a patient.

Data Cleaning:

The first data cleaning algorithm we performed was to locate and discard any missing values in our dataset. The attributes that were determined to contain missing values are smoking_status and bmi. The missing values appear as “unknown” and “N/A” in their respective attributes.

In order to discard the observations with these values, we first had to replace each of the missing instances with NaN. Once they were replaced, they were discarded from the dataset. This left us with 3,426 observations in the dataset.

Outliers were then searched for next. We visualized outliers for the numeric attributes and decided to keep all of the outliers. This is to get a better overall picture of our dataset and keep the most data we can to perform our algorithms on. We also searched for duplicate values in this dataset, and there were no duplicate values present.



Originally, we wanted to take a random sample of the data, without replacement, to perform the algorithms on. After we took a sample of 200 observations, we decided to discard this sample and use our original observations for the algorithms. Sampling down to 200 samples was unnecessary since the number of observations in our dataset is not overwhelmingly large. The machine learning algorithms that we learned in this class would work fine with a dataset of 5000 observations.

The attribute that we eliminated was id. We eliminated the id attribute because it is a unique identifier and does not offer any useful data about the patient. This will also make later visualizations easier to interpret without clouding our results with the id attribute. For example, this will prevent us from thinking there is a correlation or association between an attribute and the id attribute.

In order to generate the final CSV file, we completed the data cleaning using Python in Jupyter. Through Python, we used functions from the pandas library. These functions include .drop to drop the appropriate attributes, .replace and .dropna to replace and drop missing values, and .duplicated to locate any duplicate values. .sample was used to create a sample of our dataset which was ultimately not used. The .to_csv function was used to export the new dataset into a csv file in order to promote easier access to the cleaned data.

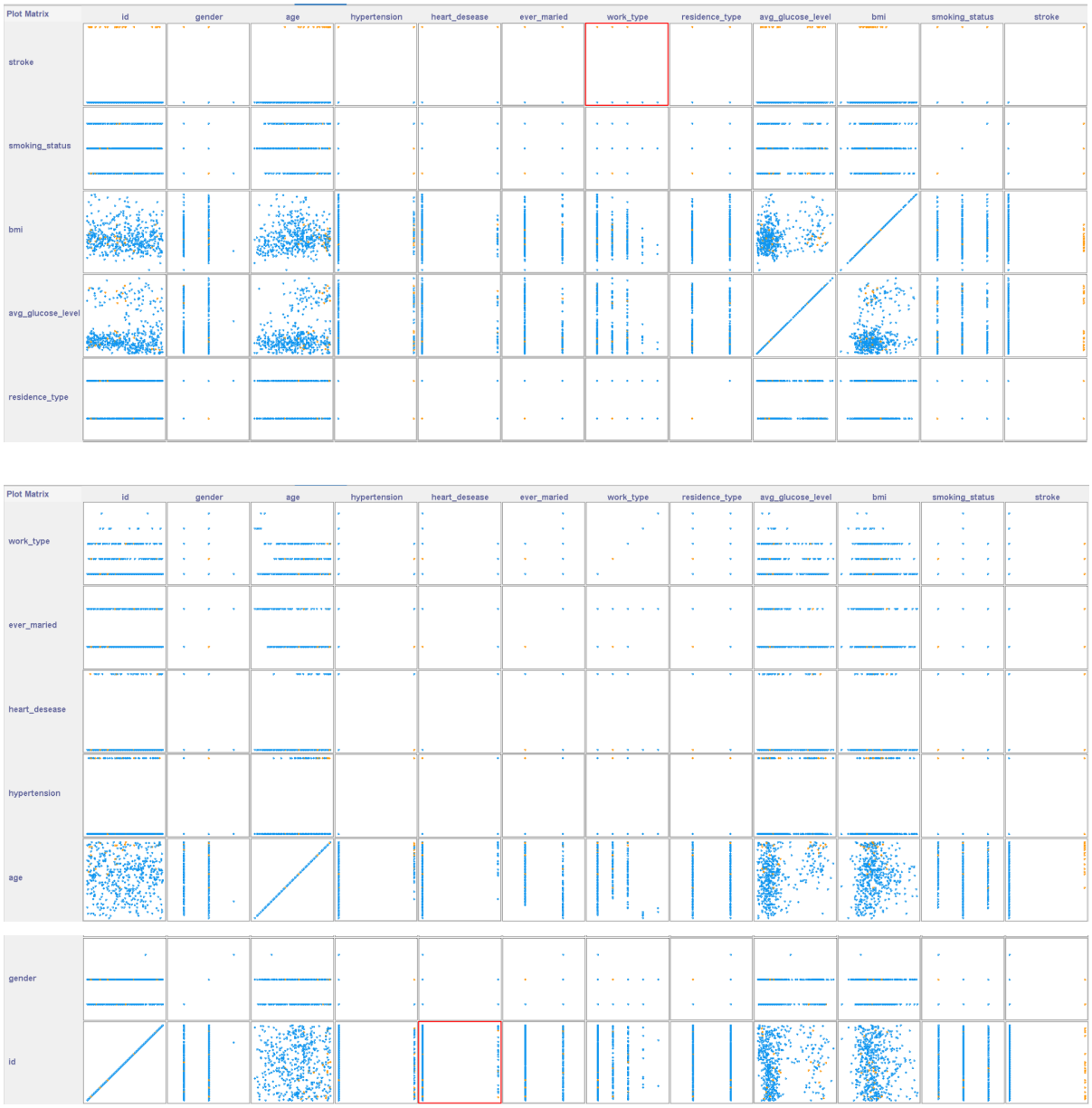
Transformations:

The only data transformation that was necessary was to create an alternate clean dataset where the binary attributes were changed from 0/1 values to No/Yes values. This was necessary when it came to creating decision trees. We didn't understand why the trees would not generate with our original data (using the 0/1 values), so we changed the binary attributes to be categorical to test and this transformation solved our problem.

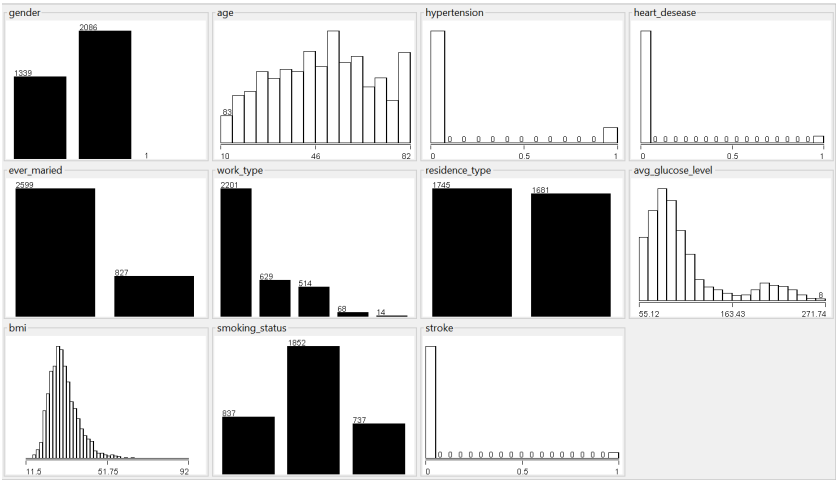
<u>Application of machine learning algorithms:</u>

Exploratory Visualizations:

In order to obtain a scatterplot matrix of all attribute pairs, the clean dataset was put into weka. From there, we could visualize all of the scatterplots. By being able to visualize all scatterplots, we were able to view potential correlations/similarity between attributes. We can also see that some attributes do not have any obvious correlation.



Univariate processing was also conducted on our dataset. Through this, we were able to visualize how the observations were distributed in each individual attribute. This gives us a better overall understanding of our dataset and allows us to make better interpretations of the attributes in the future. We created the visualization using Weka.

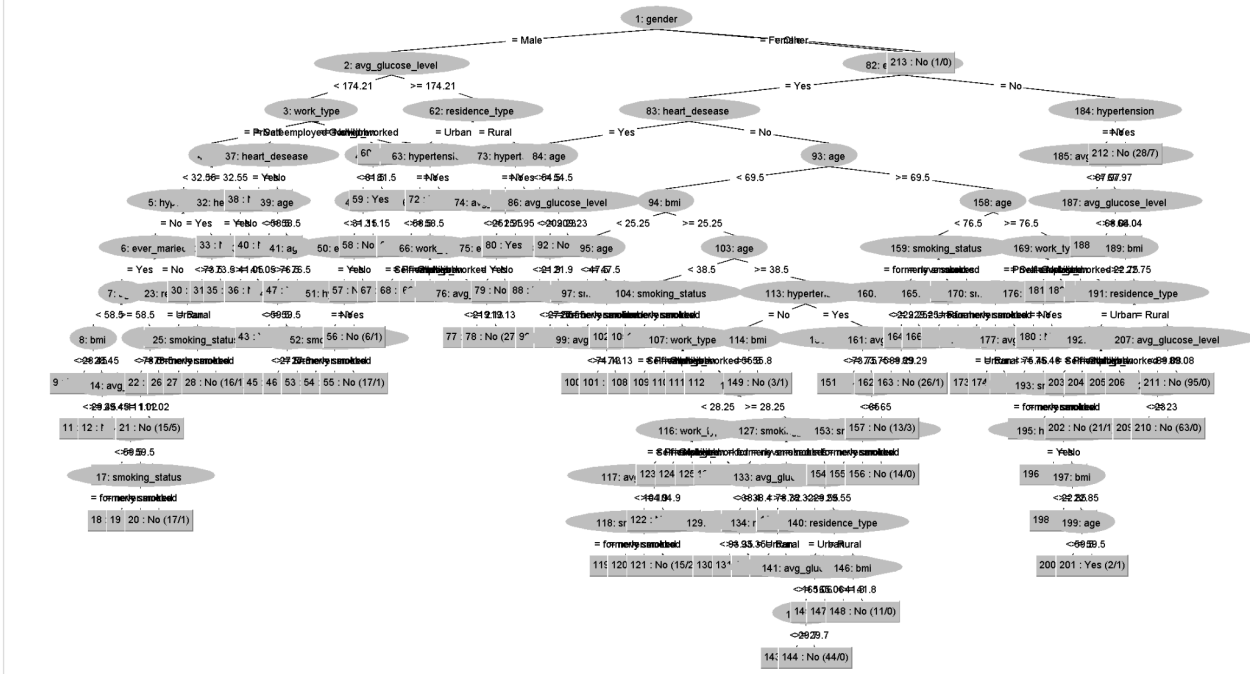


Decision Trees:

When it came to generating decision trees, we experienced a lot of difficulty. After playing around with the different algorithms and the dataset itself, we discovered that we needed to alter the binary attributes from 0/1 values to No/Yes values. This allowed us to be able to generate decision trees.

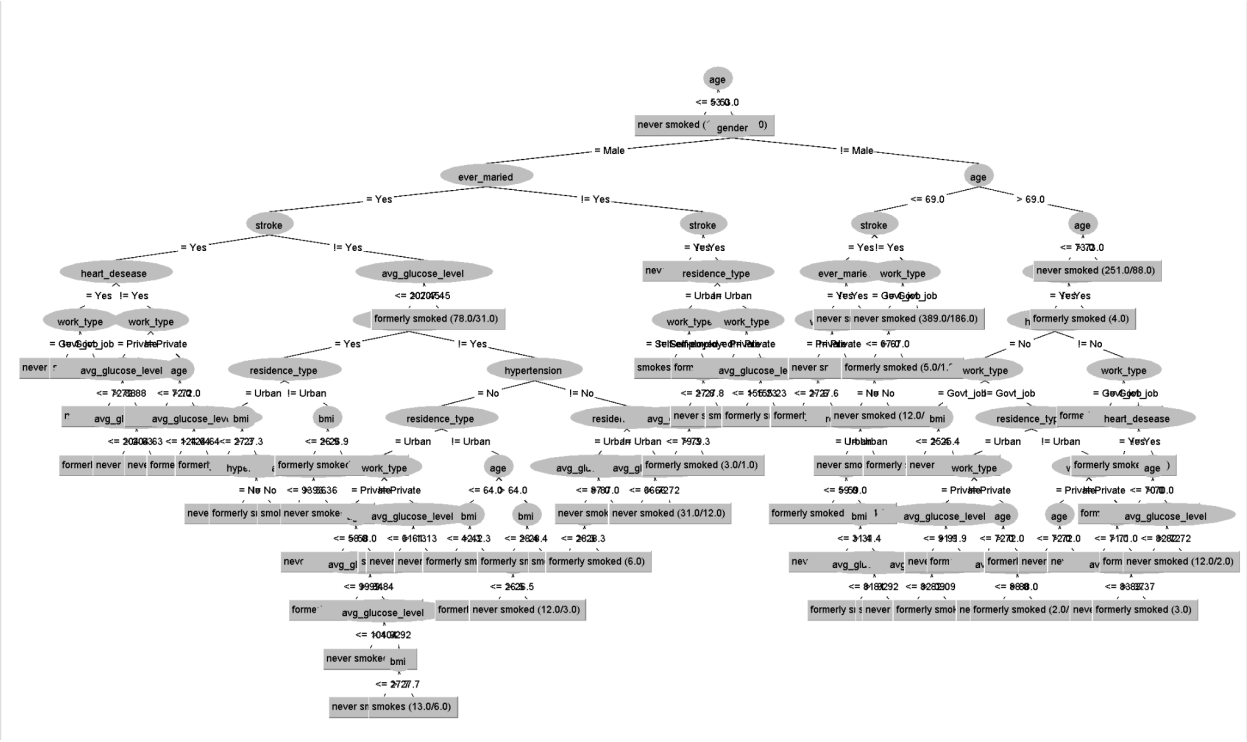
The attribute of interest when creating the trees was the stroke attribute. This would help us see how the other attributes are related to if a patient has or hasn't experienced a stroke. Using the J48 algorithm with the stroke attribute proved to be problematic due to it only creating one final branch: the no branch. We believe that this is due to the large number of observations with no in the stroke column compared to only about 5% of the data containing a yes in this column. Due to this, we attempted to use the RandomTree algorithm in Weka in order to get some kind of visualization. We were able to get a tree with a size of 213 and about 94% accuracy. Although the tree is accurate and does have more than one branch, it is difficult to visualize in the visualization window.

```
Correctly Classified Instances        3227               94.1915 %
Incorrectly Classified Instances       199                5.8085 %
Kappa statistic                          0.0503
Mean absolute error                      0.0906
Root mean squared error                  0.2275
Relative absolute error                 90.7726 %
Root relative squared error            101.9668 %
Total Number of Instances             3426

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.039    0.008    0.212      0.039    0.066      0.071    0.692     0.121     Yes
               0.992    0.961    0.949      0.992    0.970      0.071    0.692     0.969     No
Weighted Avg.  0.942    0.911    0.910      0.942    0.923      0.071    0.692     0.925

=== Confusion Matrix ===

    a    b   <-- classified as
    7  173 |    a = Yes
   26 3220 |    b = No
```



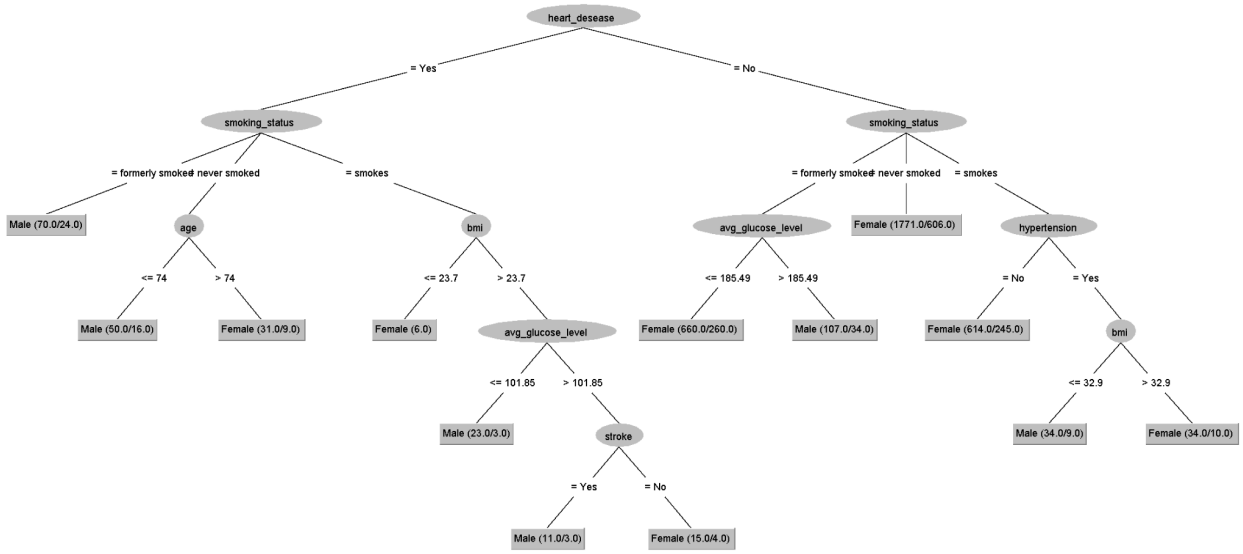
Since the attribute we originally wanted to create a decision tree on, stroke, produced a hard to visualize decision tree, we decided to also explore with a decision tree generated off of the smoking_status attribute. This is because it is interesting to see if having a stroke has any effect, not assuming causation, on the stroke variable and others through the branches. With this decision tree, using the original J48 algorithm, we get a more readable decision tree with a size of 145, but a lower accuracy of 52%.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1789               52.2183 %
Incorrectly Classified Instances      1637               47.7817 %
Kappa statistic                          0.0481
Mean absolute error                      0.3932
Root mean squared error                  0.4673
Relative absolute error                 97.9945 %
Root relative squared error            104.3384 %
Total Number of Instances             3426

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.148     0.090     0.347      0.148     0.208      0.082     0.571     0.293     formerly smoked
               0.883     0.833     0.555      0.883     0.682      0.072     0.527     0.545     never smoked
               0.041     0.035     0.244      0.041     0.070      0.014     0.515     0.221     smokes
Weighted Avg.  0.522     0.480     0.437      0.522     0.434      0.062     0.535     0.414

=== Confusion Matrix ===

   a    b    c   <-- classified as
 124  667   46 |    a = formerly smoked
 170 1635   47 |    b = never smoked
  63  644   30 |    c = smokes
```



Since all of the trees were too crowded, we created a new dataset eliminating the following attributes: ever_married, work_type, and residence_type. We tried to make a decision tree without these attributes so that we could get a smaller tree and because these attributes did not appear to have a significant influence on machine learning algorithms we performed later.

The best visualization of a decision tree was generated from the gender attribute. This decision tree is interesting because it lets us see how the different attributes/symptoms that may lead to a stroke affect men differently than women and vice versa.




```
Correctly Classified Instances        2161               63.0765 %
Incorrectly Classified Instances      1265               36.9235 %
Kappa statistic                          0.0986
Mean absolute error                      0.3087
Root mean squared error                  0.3957
Relative absolute error                 97.1322 %
Root relative squared error             99.2972 %
Total Number of Instances             3426

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.137     0.052     0.629       0.137   0.225      0.149    0.555     0.452     Male
               0.948     0.863     0.631       0.948   0.758      0.148    0.555     0.636     Female
               0.000     0.000     ?           0.000   ?          ?        0.415     0.000     Other
Weighted Avg.  0.631     0.546     ?           0.631   ?          ?        0.555     0.564

=== Confusion Matrix ===

   a    b    c   <-- classified as
 183 1156    0 |    a = Male
 108 1978    0 |    b = Female
   0    1    0 |    c = Other
```

Although this tree is significantly easier to visualize, it only has an accuracy of about 63.08%. The confidence level was dropped from 0.25 to 0.1 to get the accuracy from about 61% to 63.08%.

Through the extensive exploration we performed with decision trees, we have come to the conclusion that decision trees may not be the most appropriate classification algorithm for our dataset.

Rule Based Classification:

Classification rules were able to be generated with varying accuracy. Calculating classification rules on the stroke variable had a high accuracy, but only produced two rules which are not necessarily helpful. It is also easy to have accurate rules when one is a default rule, so these rules were not useful. Producing classification rules on the attributes of gender and smoking_status

produced more rules, with lower accuracy. We can see a trend in our explorations with the gender attribute. It seems to be the easiest attribute to work with in Weka. Although the rules are not very good, this is a very interesting trend to see.

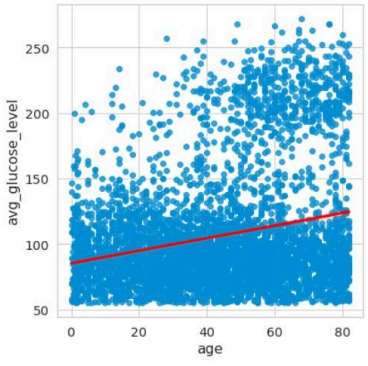
```
(age >= 68) and (avg_glucose_level >= 216.94) and (bmi >= 30.9) and (work_type = Private) => stroke=Yes (21.0/9.0)
 => stroke=No (3405.0/168.0)

Number of Rules : 2


Time taken to build model: 0.23 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3242               94.6293 %
Incorrectly Classified Instances       184                5.3707 %
Kappa statistic                          0.0262
Mean absolute error                      0.0985
Root mean squared error                  0.2234
Relative absolute error                 98.6487 %
Root relative squared error            100.1256 %
Total Number of Instances             3426

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,017    0,002    0,300      0,017    0,032      0,060    0,510     0,059     Yes
               0,998    0,983    0,948      0,998    0,972      0,060    0,510     0,948     No
Weighted Avg.  0,946    0,932    0,914      0,946    0,923      0,060    0,510     0,902

=== Confusion Matrix ===

    a    b   <-- classified as
    3  177 |    a = Yes
    7 3239 |    b = No
```

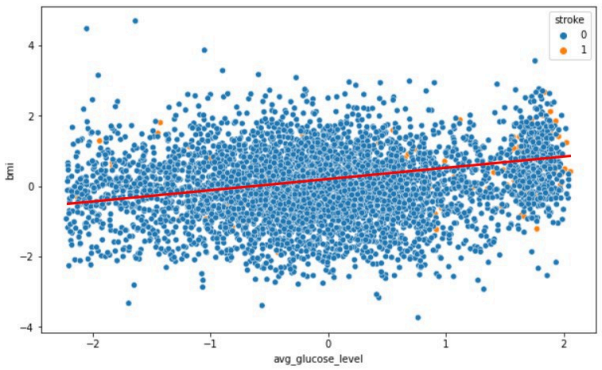
```
(bmi <= 32.4) and (heart_desease = Yes) and (age <= 75) and (bmi >= 27.1) => gender=Male (56.0/10.0)
(bmi >= 25.8) and (bmi <= 36.1) and (avg_glucose_level >= 105.92) and (smoking_status = formerly smoked) => gender=Male (164.0/70.0)
(bmi >= 25.8) and (bmi <= 32.4) and (avg_glucose_level <= 96.02) and (hypertension = Yes) => gender=Male (61.0/20.0)
 => gender=Female (3145.0/1159.0)

Number of Rules : 4


Time taken to build model: 0.16 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2054               59.9533 %
Incorrectly Classified Instances      1372               40.0467 %
Kappa statistic                          0.0708
Mean absolute error                      0.3128
Root mean squared error                  0.3987
Relative absolute error                 98.4265 %
Root relative squared error            100.0379 %
Total Number of Instances             3426

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,222    0,158    0,474      0,222    0,302      0,081    0,535     0,430     Male
               0,842    0,778    0,628      0,842    0,719      0,081    0,534     0,629     Female
               0,000    0,000    ?          0,000    ?          ?        0,128     0,000     Other
Weighted Avg.  0,600    0,536    ?          0,600    ?          ?        0,534     0,551

=== Confusion Matrix ===

    a    b    c   <-- classified as
  297 1042    0 |    a = Male
  329 1757    0 |    b = Female
    0    1    0 |    c = Other
```

```
(age >= 54) and (gender = Male) and (avg_glucose_level >= 195.03) => smoking_status=formerly smoked (139.0/67.0)
(age >= 54) and (gender = Male) and (bmi <= 24.6) => smoking_status=formerly smoked (55.0/24.0)
 => smoking_status=never smoked (3232.0/1437.0)

Number of Rules : 3


Time taken to build model: 0.21 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1853               54.0864 %
Incorrectly Classified Instances      1573               45.9136 %
Kappa statistic                          0.0167
Mean absolute error                      0.3991
Root mean squared error                  0.4477
Relative absolute error                 99.4603 %
Root relative squared error             99.9663 %
Total Number of Instances             3426

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,044    0,019    0,425      0,044    0,080      0,068    0,518     0,264     formerly smoked
               0,981    0,968    0,544      0,981    0,700      0,041    0,509     0,547     never smoked
               0,000    0,000    ?          0,000    ?          ?        0,509     0,219     smokes
Weighted Avg.  0,541    0,528    ?          0,541    ?          ?        0,511     0,407

=== Confusion Matrix ===

    a    b    c   <-- classified as
   37  800    0 |    a = formerly smoked
   36 1816    0 |    b = never smoked
   14  723    0 |    c = smokes
```

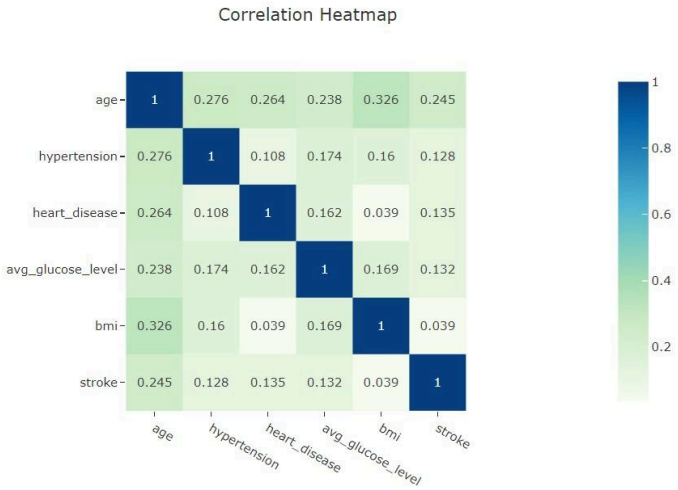
Linear Regression:

As a result of our scatterplot matrix, we wanted to look deeper into two attribute pairs that we believed to have a correlation. These pairs were avg_glucose_level and bmi, and age and avg_glucose_level. We created a regression line model for each of these suspected correlations. As a result of the regression lines, we actually found that the correlation between avg_glucose_level and bmi was not as distinct as we thought that it would be. However, the regression line showed an angle that we were expecting for age and avg_glucose_level.

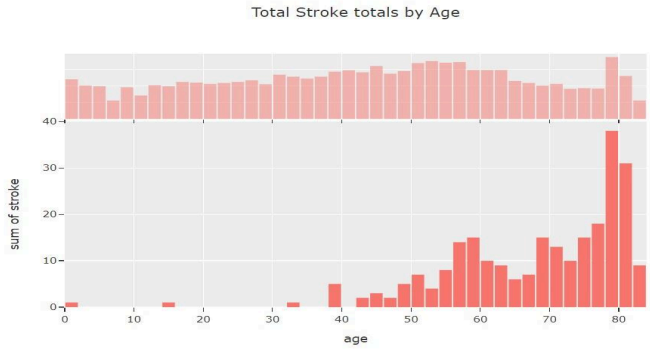


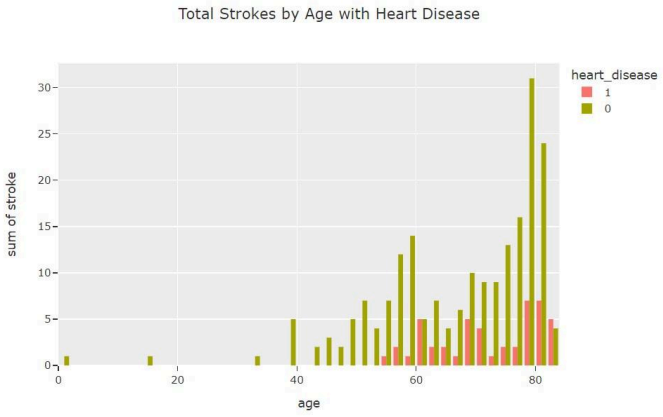


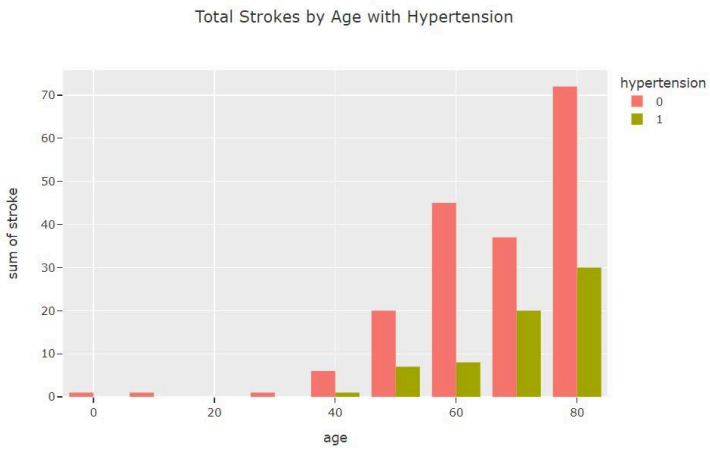
As a result of our findings, we thought it would be interesting to create a correlation heat map in order to get specific correlation strength for each attribute relationship. Through this, we found that the highest correlations were all related to the age attribute. This reinforces our original assumption that more health problems arise as you age. It also tells us that more things that could cause a stroke are present in older individuals which may explain why so many older individuals experience strokes. This heatmap can also be used as cluster validity for the clusters shown later in this report.



Furthermore, we explored deeper into the age attribute by using R and the ggplot library to visualize our data. Through each of these graphs, we can see more clearly than a scatterplot and regression line that there is a consistent increase in health problems and stroke frequency increases with age.





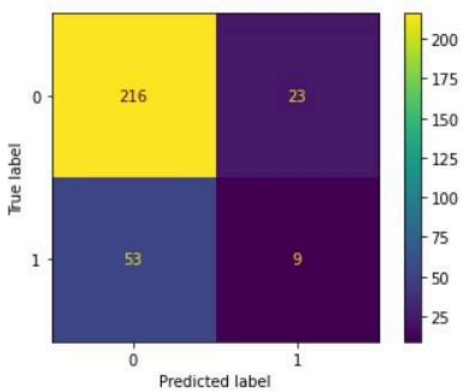


NN Classifiers:

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.
Here we also added some characteristics (precision,recall,accuracy,f1_score) to analyze their values to see how the k-nearest neighbors algorithm did on our data.
We also display a confusion matrix,a confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm. So we realized that a decent amount of our instances have been correctly classified.

```
KNN model evaluation:
Precision:  0.28125
Recall:  0.14516129032258066

Accuracy:  0.7475083056478405
f1_score:  0.1914893617021277
```

```
[62]:
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f01e7bc2f90>
```



Association Rules:

For the association rules we had to adjust our binary variable (0/1 to no/yes) because Weka needs only nominal variables to create the association rules because the apriori method doesn't work on numeric values.

We failed to make the correct rules. One reason could be that support and/or confidence values are too high. We tried low ones, e.g. a support and confidence level of 0.001%. Another reason could be that the data set just doesn't contain any association rules.We also tried another data set which certainly contains association rules from a set minimum support and confidence value to see if it was ours the problem and we think it is.

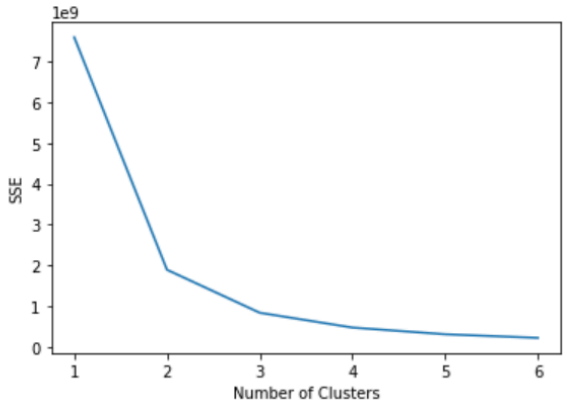
Sequential Patterns:

Sequential patterns are not appropriate to apply to our dataset because there is no timestamp attribute in our dataset. We also do not have specified order to the observations, so we would be unable to find subsequences.

K-Means Clusters:

After running our clean dataset through the cluster python code, we first evaluated that three clusters would be appropriate for our data since there is an elbow at the three clusters mark. Averages of each attribute were computed for each cluster and the dataset was put into clusters.

	Cluster ID
0	1
1	1
2	1
3	1
4	1
...	...
3421	0
3422	0
3423	0
3424	0
3425	0



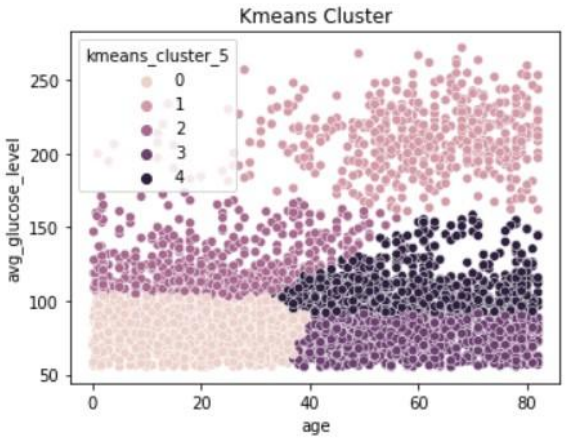
	Unnamed: 0	age	hypertension	heart_desease	avg_glucose_level	bmi	stroke
0	4263.763828	46.978929	0.103600	0.042142	105.914355	29.999210	6.106227e-16
1	836.730241	50.798110	0.137457	0.073024	111.552216	30.673196	1.546392e-01
2	2554.261799	48.105966	0.115761	0.065004	107.415467	30.187890	6.106227e-16

After further inspection, we wanted to explore how more clusters would distribute our data. The next number of clusters we explored was 4 clusters.

	Unnamed: 0	age	hypertension	heart_desease	avg_glucose_level	bmi	stroke
0	1891.419738	48.082045	0.109394	0.059453	108.472592	30.270511	5.273559e-16
1	4476.713287	46.599068	0.090909	0.039627	105.779615	30.120396	5.342948e-16
2	3176.604651	48.418605	0.124419	0.060465	106.163837	30.081395	5.342948e-16
3	617.339100	51.444060	0.151096	0.080738	112.832226	30.683852	2.076125e-01

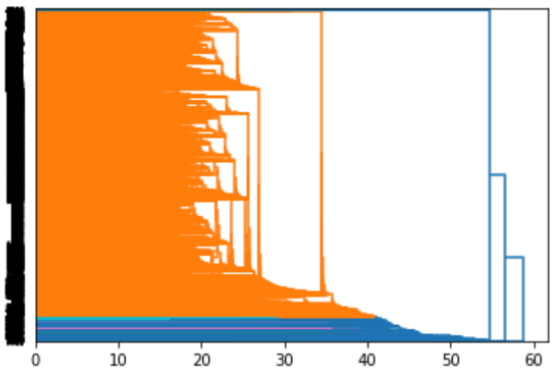
The final number of clusters we wanted to evaluate was 5 clusters. 5 clusters seemed to give our data the best distribution and the most clear graph. Due to this, 5 is the appropriate number of clusters for this dataset. In cluster 3, we can see higher numbers for every attribute which may be because the age is relatively higher compared to the other clusters.

	Unnamed: 0	age	hypertension	heart_desease	avg_glucose_level	bmi	stroke
0	2522.393275	48.292398	0.109649	0.070175	107.697047	30.504094	6.522560e-16
1	3550.016296	48.102222	0.130370	0.050370	106.634400	30.056296	6.522560e-16
2	1499.668175	47.992459	0.099548	0.048265	107.591629	30.162745	4.371503e-16
3	496.085837	52.191702	0.165951	0.092990	114.517010	30.772961	2.575107e-01
4	4590.347518	46.608511	0.089362	0.038298	105.088170	29.947092	5.273559e-16

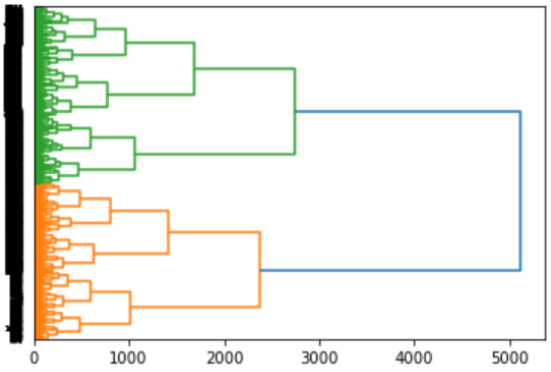


Hierarchical Clusters:

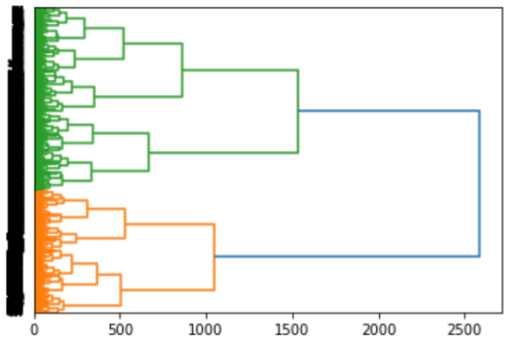
We first computed a single link cluster on our data:



Next, we performed a complete link cluster:

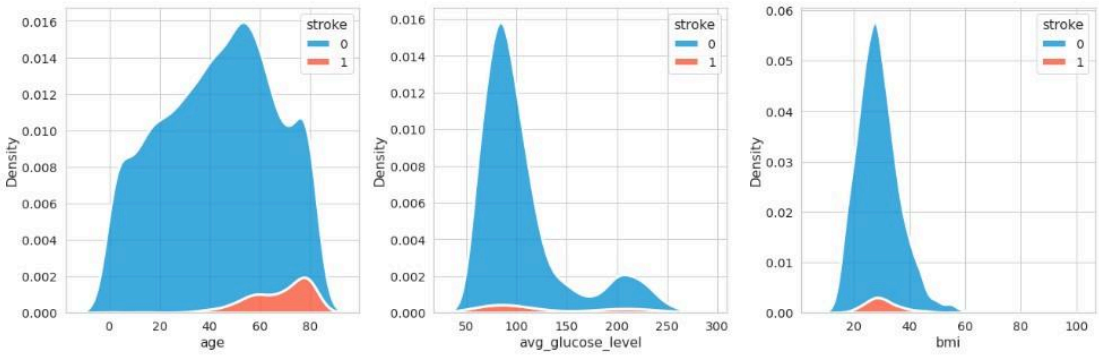


Finally, we performed a group average cluster:



After assessing all of these models, the complete and group average clusters have better visualizations. Due to this, they are better representations of how the observations work with one another.

DBSCAN Clusters:



During the project, we analyzed the density of our three important variables using the seaborn library on Python. The plots show us where most of our data is and what part of this data indicates a stroke.

We already analyzed this data before and saw that the euclidean distance between each of the values of these attributes are really close to each other. So even with a really small radius, we should probably display a graph that would take all the values in the graph. However, we didn't succeed in displaying a good DBSCAN graph.

It might come from our dataset, but as we saw previously it would be useless to use the DBSCAN approach we saw in class. We have high density values in our numerical attributes. Analyzing the clusters in this way would not help us to determine something.

Anomaly Detection:

An anomaly is a datapoint that differs significantly from the other datapoints in the same sample. An anomaly detection pattern produces two different results. The first is a categorical tag for whether the observation is abnormal or not; the second is a score or trust value. Score carries more information than the label. Because it also tells us how abnormal the observation is. The tag just tells you if it's abnormal. While labeling is more common in supervised methods, the score is more common in unsupervised and semi supervised methods.

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0
13	Male	78.0	0	1	Yes	Private	Urban	219.84	NaN
14	Female	79.0	0	1	Yes	Private	Urban	214.09	28.2
16	Male	64.0	0	1	Yes	Private	Urban	191.61	37.5
...	...	...	...	...	...	...	...	...	...
6	Male	74.0	1	1	Yes	Private	Rural	70.09	27.4
35	Female	82.0	1	1	No	Private	Rural	84.03	26.5
143	Female	69.0	1	1	No	Self-employed	Urban	72.17	36.8
171	Female	79.0	1	1	No	Self-employed	Rural	60.94	NaN
184	Male	61.0	1	1	Yes	Private	Urban	112.24	37.4

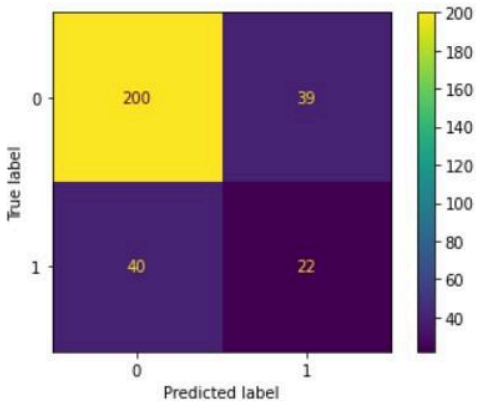
Neural Networks:
Multi-Layer Perceptron:

Multilayer perceptron is a type of artificial neural network organized in several layers. The information flows from the input layer to the output layer only: it is therefore a direct propagation network.

We first tried to apply this method on Weka, but after many tests we did not manage to have viable and correct results. Python with SKLearn and the MLP Classifier function was therefore a good option to try to apply this method and display a confusion matrix to see the effectiveness of the method (code in the last sheets).
We can still see that the model is not perfectly working as the accuracy value is really low.

```
MLP model evaluation:
Precision:  0.36065573770491804
Recall:  0.3548387096774194

Accuracy:  0.7375415282392026
f1_score:  0.3577235772357723
```

```
t[61]:
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f01f4aad090>
```



Bagging:

When computing bagging in Weka, we had to use the nominal version of our dataset in order to be able to bag on the stroke attribute. After this was completed, we received a 94.66% accuracy. We increased the number of iterations from 10 to 15 and the accuracy increased to 94.72%. After increasing the number of iterations further, the accuracy did not change.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3243               94.6585 %
Incorrectly Classified Instances       183                5.3415 %
Kappa statistic                          0.008
Mean absolute error                      0.0906
Root mean squared error                  0.2171
Relative absolute error                 90.7265 %
Root relative squared error             97.3007 %
Total Number of Instances             3426

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.006    0.001    0.200      0.006    0.011      0.025    0.795     0.165     Yes
               0.999    0.994    0.948      0.999    0.973      0.025    0.795     0.985     No
Weighted Avg.  0.947    0.942    0.908      0.947    0.922      0.025    0.795     0.942

=== Confusion Matrix ===

    a    b   <-- classified as
    1  179 |    a = Yes
    4 3242 |    b = No
```

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3245               94.7169 %
Incorrectly Classified Instances       181                5.2831 %
Kappa statistic                          0.0092
Mean absolute error                      0.0905
Root mean squared error                  0.2166
Relative absolute error                 90.6655 %
Root relative squared error             97.1    %
Total Number of Instances             3426

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.006    0.001    0.333      0.006    0.011      0.037    0.799     0.163     Yes
                 0.999    0.994    0.948      0.999    0.973      0.037    0.799     0.985     No
Weighted Avg.    0.947    0.942    0.915      0.947    0.922      0.037    0.799     0.942

=== Confusion Matrix ===

    a    b   <-- classified as
    1  179 |    a = Yes
    2 3244 |    b = No
```

Precision = 0.33
Recall = 0.0056
Specificity = 0.999
FPR = 0.0006

Boosting:

The accuracy for boosting was 94.75%. We used the AdaBoostM1 algorithm for this.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3246               94.7461 %
Incorrectly Classified Instances       180                5.2539 %
Kappa statistic                          0
Mean absolute error                      0.0927
Root mean squared error                  0.2161
Relative absolute error                 92.9085 %
Root relative squared error             96.8737 %
Total Number of Instances             3426

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.000    0.000    ?          0.000    ?          ?        0.771     0.125     Yes
                 1.000    1.000    0.947      1.000    0.973      ?        0.771     0.983     No
Weighted Avg.    0.947    0.947    ?          0.947    ?          ?        0.771     0.938

=== Confusion Matrix ===

    a    b   <-- classified as
    0  180 |    a = Yes
    0 3246 |    b = No
```

Precision = 0
Recall = 0
Specificity = 1
FPR = 0

Random Forest:

The accuracy for performing random forest was 94.57%.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        3240               94.5709 %
Incorrectly Classified Instances       186                5.4291 %
Kappa statistic                          0.0248
Mean absolute error                      0.0941
Root mean squared error                  0.2215
Relative absolute error                 94.2666 %
Root relative squared error             99.2909 %
Total Number of Instances             3426

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.017    0.003    0.250      0.017    0.031      0.052    0.756     0.145     Yes
               0.997    0.983    0.948      0.997    0.972      0.052    0.756     0.978     No
Weighted Avg.  0.946    0.932    0.911      0.946    0.923      0.052    0.756     0.935

=== Confusion Matrix ===

    a    b   <-- classified as
    3  177 |    a = Yes
    9 3237 |    b = No
```

Precision = 0.25
Recall = 0.0167
Specificity = 0.997
FPR = 0.0028

After performing each of the ensemble methods, we can see that boosting had the highest accuracy. Even though boosting had the best accuracy, it was only by 0.03%. All of the ensemble methods for our dataset were very similar in accuracy so it can be concluded that any one of them can be useful.

When looking at the imbalanced class calculations for each of the ensemble methods, we can see very interesting results. Although boosting has the highest accuracy, it has 0 precision, recall and FPR. This shows us that accuracy does not give the entire picture and boosting may not be the best ensemble method to use.

Comparing these results to our results from the decision tree made for the stroke attribute, boosting still has the highest accuracy. The decision tree's accuracy is around the same accuracy as the rest of the ensemble methods.

Most Viable Attribute:

After utilizing all of the machine learning algorithms that would apply to our dataset, we have determined the age attribute to be the most viable attribute. This can be concluded mainly from observing regression lines and the correlation heat map we generated. The other attributes, including the stroke attribute, seemed to be the most related to the age attribute. Although we can not conclude causation, age does seem to lead to a deterioration in overall health which makes a person more likely to have a stroke.

Code:

Preprocessing:

Machine Learning Project

First we start with data visualization: we first need to import our dataset

In [1]:
```
import pandas as pd

data = pd.read_csv('healthcare-dataset-stroke-data.csv',header=None)
data.columns = ['id','gender', 'age', 'hypertension', 'heart_desease', 'ever_maried','work_type','residence_type','avg_glucos
data = data.drop(labels=0, axis=0)

data.head()
```

Out[1]:

	id	gender	age	hypertension	heart_desease	ever_maried	work_type	residence_type	avg_glucose_level	bmi	smoking_status	stroke
1	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
2	51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
3	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
4	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
5	1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1

Cleaning the data:

Locating missing data points:

In [2]:
```
import numpy as np

data = data.replace('N/A',np.NaN)
data = data.replace('Unknown',np.NaN)

print('Number of instances = %d' % (data.shape[0]))
print('Number of attributes = %d' % (data.shape[1]))

print('Number of missing values:')
for col in data.columns:
    print('\t%s: %d' % (col,data[col].isna().sum()))
```

```
Number of instances = 5110
Number of attributes = 12
Number of missing values:
        id: 0
        gender: 0
        age: 0
        hypertension: 0
        heart_desease: 0
        ever_maried: 0
        work_type: 0
        residence_type: 0
        avg_glucose_level: 0
        bmi: 201
        smoking_status: 1544
        stroke: 0
```

We can see that there are missing values in smoking_status and in bmi. We will be discarding the missing data for both the bmi and smoking_status attributes.

In [4]:
```
print('Number of rows in original data = %d' % (data.shape[0]))
data2 = data.dropna()
print('Number of rows after discarding missing values = %d' % (data2.shape[0]))
```

```
Number of rows in original data = 5110
Number of rows after discarding missing values = 3426
```

Assessing if there are duplicate values:

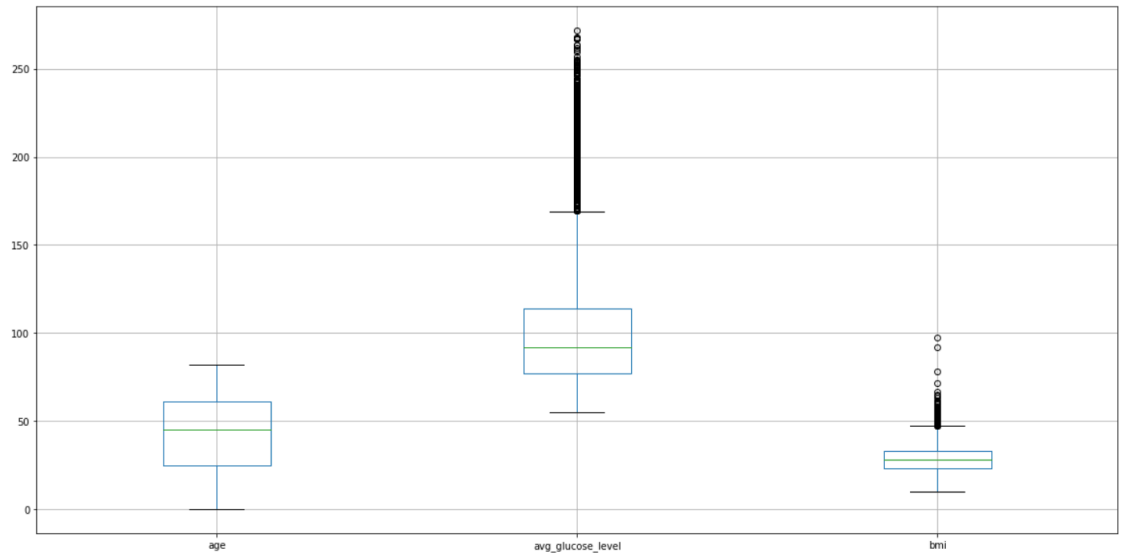
In [5]:
```
dups = data2.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
```

```
Number of duplicate rows = 0
```

We are trying to see if there is outliers. So to see that we create a variable named "data3" so we can remove the rows where there is stirng values so we can make the box plot.

In [6]:
```
data3 = data2
data3 = data.drop(['id','hypertension','heart_desease','stroke','gender','ever_maried','work_type','residence_type','smoking_
data3['avg_glucose_level'] = pd.to_numeric(data3['avg_glucose_level'])
data3['bmi'] = pd.to_numeric(data3['bmi'])
data3['age'] = pd.to_numeric(data3['age'])
data3.boxplot(figsize=(20,10))
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x1a04eac1438>



Now, our file is cleaned and we can use it to test or train the model we are going to build. We just need to export the new file as .csv file to use it.

In [9]:
```
data2.to_csv('CleanDataSet.csv')
```

In [10]:
```
cleanData = pd.read_csv('CleanDataSet.csv',header=None)
cleanData.columns = ['extra','id','gender', 'age', 'hypertension', 'heart_desease', 'ever_maried','work_type','residence_type
cleanData = cleanData.drop(labels=0, axis=0)
del cleanData['extra']

cleanData.head()
```

Out[10]:

	id	gender	age	hypertension	heart_desease	ever_maried	work_type	residence_type	avg_glucose_level	bmi	smoking_status	stroke
1	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
2	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1
5	56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smoked	1

In [11]:
```
sample = cleanData.sample(n=200)
sample
```

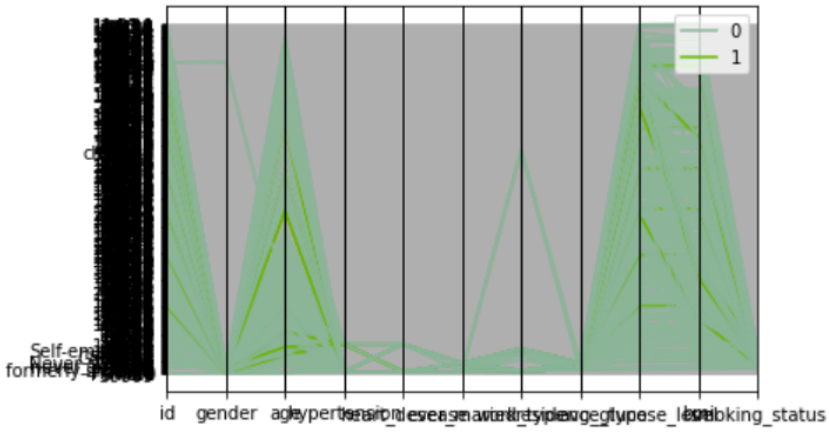
Out[11]:

	id	gender	age	hypertension	heart_desease	ever_maried	work_type	residence_type	avg_glucose_level	bmi	smoking_status	stroke
3003	50009	Female	17	0	0	No	Private	Urban	81.51	19.5	formerly smoked	0
1504	27017	Male	28	0	0	No	Private	Urban	75.92	22.8	never smoked	0
2424	4850	Male	51	0	0	Yes	Private	Rural	112.79	27.2	never smoked	0
2781	71966	Female	18	0	0	No	Never_worked	Urban	81.73	21.6	never smoked	0
1637	68141	Female	58	0	0	Yes	Private	Rural	65.66	24.6	formerly smoked	0
...	...	...	...	...	...	...	...	...	...	...	...	...
359	27757	Male	31	0	0	Yes	Private	Urban	88.78	35.8	smokes	0
1817	20165	Female	77	0	0	Yes	Private	Urban	250.8	32.9	never smoked	0
2736	60416	Female	57	0	0	Yes	Self-employed	Urban	106.84	29.6	never smoked	0
2181	47776	Female	57	0	0	Yes	Govt_job	Rural	176.78	50.4	never smoked	0
3167	25408	Female	24	0	0	Yes	Self-employed	Rural	114.54	30.1	smokes	0

Visualizing parallel coordinates based on the stroke attribute:

In [13]:
```
from pandas.plotting import parallel_coordinates
%matplotlib inline

parallel_coordinates(sample, 'stroke')
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1a050d6eef0>



Clusters: (The number of clusters was just adjusted in one code block)

In [1]:

```
import pandas as pd

dataset = pd.read_csv('CleanDataSet.csv')
dataset =  dataset.dropna();
dataset = dataset.drop(['id'],axis=1)
dataset = dataset.drop(['smoking_status'],axis=1)
dataset = dataset.drop(['residence_type'],axis=1)
dataset = dataset.drop(['work_type'],axis=1)
dataset = dataset.drop(['gender'],axis=1)
dataset = dataset.drop(['ever_maried'],axis=1)
print(dataset)
```

```
      Unnamed: 0  age  hypertension  heart_desease  avg_glucose_level   bmi  \
0              1   67             0              1             228.69  36.6
1              3   80             0              1             105.92  32.5
2              4   49             0              0             171.23  34.4
3              5   79             1              0             174.12  24.0
4              6   81             0              0             186.21  29.0
...          ...  ...           ...            ...                ...   ...
3421        5101   82             1              0              71.97  28.3
3422        5103   57             0              0              77.93  21.7
3423        5107   81             0              0             125.20  40.0
3424        5108   35             0              0              82.99  30.6
3425        5109   51             0              0             166.29  25.6

      stroke
0          1
1          1
2          1
3          1
4          1
...      ...
3421       0
3422       0
3423       0
3424       0
3425       0

[3426 rows x 7 columns]
```

In [2]:

```
from sklearn import cluster

data = dataset
k_means = cluster.KMeans(n_clusters=4, max_iter=50, random_state=1)
k_means.fit(data)
labels = k_means.labels_
pd.DataFrame(labels, columns=['Cluster ID'])
```

Out[2]:

	Cluster ID
0	3
1	3
2	3
3	3
4	3
...	...
3421	1
3422	1
3423	1
3424	1
3425	1

3426 rows × 1 columns

In [3]:

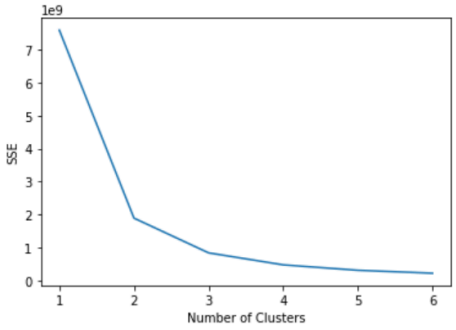
```
centroids = k_means.cluster_centers_
pd.DataFrame(centroids,columns=data.columns)
```

Out[3]:

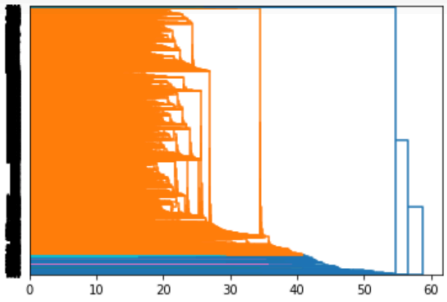
	Unnamed: 0	age	hypertension	heart_desease	avg_glucose_level	bmi	stroke
0	1891.419738	48.082045	0.109394	0.059453	108.472592	30.270511	5.273559e-16
1	4476.713287	46.599068	0.090909	0.039627	105.779615	30.120396	5.342948e-16
2	3176.604651	48.418605	0.124419	0.060465	106.163837	30.081395	5.342948e-16
3	617.339100	51.444060	0.151096	0.080738	112.832226	30.683852	2.076125e-01

In [4]:
```
import matplotlib.pyplot as plt
%matplotlib inline

numClusters = [1,2,3,4,5,6]
SSE = []
for k in numClusters:
    k_means = cluster.KMeans(n_clusters=k)
    k_means.fit(data)
    SSE.append(k_means.inertia_)

plt.plot(numClusters, SSE)
plt.xlabel('Number of Clusters')
plt.ylabel('SSE')
```

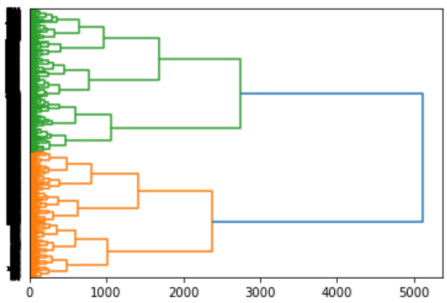
Out[4]: Text(0, 0.5, 'SSE')



In [5]:
```
from scipy.cluster import hierarchy
import matplotlib.pyplot as plt
%matplotlib inline

names = dataset['Unnamed: 0']
Y = dataset['stroke']
X=data

Z = hierarchy.linkage(X.values, 'single')
dn = hierarchy.dendrogram(Z,labels=names.tolist(),orientation='right')
```



In [8]:
```
Z = hierarchy.linkage(X.values, 'complete')
dn = hierarchy.dendrogram(Z,labels=names.tolist(),orientation='right')
```



In [9]:
```
Z = hierarchy.linkage(X.values, 'average')
dn = hierarchy.dendrogram(Z,labels=names.tolist(),orientation='right')
```

In [40]:
```
# Show the relationship by gender
plt.figure(figsize=(10, 8))
sns.scatterplot(x='bmi', y='avg_glucose_level', data=df, hue='gender')
```

Out[40]:
```
<AxesSubplot:xlabel='bmi', ylabel='avg_glucose_level'>
```

In [24]:
```
fig = plt.figure(figsize=(10,10))
sns.pairplot(df)
plt.show()
```

In [52]:
```
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
test_predictions_knn =knn_model.predict(X_test)
```

Performance metrics - KNN

In [53]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,f1_score
print('KNN model evaluation:')
print('Precision: ', precision_score(y_test,test_predictions_knn))
print('Recall: ', recall_score(y_test,test_predictions_knn))

print('\nAccuracy: ',accuracy_score(y_test,test_predictions_knn))
print('f1_score: ',f1_score(y_test,test_predictions_knn))
```

```
KNN model evaluation:
Precision:  0.4594594594594595
Recall:  0.2786885245901639

Accuracy:  0.7873754152823921
f1_score:  0.3469387755102041
```

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 5110 non-null   int64  
 1   gender             5110 non-null   object 
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64  
 4   heart_disease      5110 non-null   int64  
 5   ever_married       5110 non-null   object 
 6   work_type          5110 non-null   object 
 7   Residence_type     5110 non-null   object 
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object 
 11  stroke             5110 non-null   int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

Multi-Layer Perceptron model

In [61]:

```
mlp_model_all = MLPClassifier(activation='logistic',solver='lbfgs',hidden_layer_sizes=(1
5),max_iter=8000)
mlp_model_all.fit(X_training, y_training)

test_predictions_mlp_all = mlp_model_all.predict(X_testing)
train_predictions_mlp_all = mlp_model_all.predict(X_training)

print('MLP model evaluation:')
print('Precision: ', precision_score(y_testing, test_predictions_mlp_all))
print('Recall: ', recall_score(y_testing, test_predictions_mlp_all))
print('\nAccuracy: ',accuracy_score(y_testing, test_predictions_mlp_all))
print('f1_score: ',f1_score(y_testing, test_predictions_mlp_all))
print("\n")

plot_confusion_matrix(mlp_model_all,X_testing,y_testing)
```