INFOSYS 722 Data Mining and Big Data

Iteration 3

Semester 2, 2022

Qiong Zhou / 365217677 / qzho906

Table of Contents

1. Business / Situation Understanding

1.1 Business understanding Overview

Ischemic heart disease, stroke, and chronic obstructive pulmonary disease are the top three causes of death from disease, accounting for 16%, 11%, and 6 % of deaths worldwide respectively (WHO, 2020).

Stroke is a condition in which the blood supply to the brain is interrupted, resulting in a lack of oxygen, brain damage, and loss of function. Stroke can lead to permanent damage, including partial paralysis and impairments in speech, understanding, and memory, all of which affect the type and severity of disability depending on the part of the brain affected and the length of time the blood supply is stopped (World Stroke Organization, 2022). The prevalence of stroke has reached epidemic proportions beyond what is thought possible. Globally, one in four adults over the age of 25 will have a stroke in their lifetime (World Stroke Organization, 2022). This year 12.2 million people worldwide will have their first stroke and 6.5 million will die as a result. Worldwide, more than 110 million people have experienced a stroke (World Stroke Organization, 2022). The incidence of stroke increases significantly with age, but over 60% of strokes occur in people under the age of 70 and 16% in people under the age of 50 (World Stroke Organization, 2022).

Although stroke is an acute cerebrovascular disease with high morbidity, mortality, and disability rate. Many factors contribute to stroke, including age, race, gender, geography, and environment, which are not controllable. However, according to a review article published in the Journal of the American College of Cardiology, 90% of strokes are preventable and the key is to manage and treat controllable risk factors (Kleindorfer DO, Towfighi A, Chaturvedi S, et al., 2021). The controllable risk factors are blood pressure, blood lipids, blood sugar, and lifestyle (smoking, alcohol, etc.).

1.2 Determining Business Objectives

Although the disease itself can develop in a very short period of time and without any precursors, it is still possible to classify and predict patients based on their early personal history and to offer solutions and advice to patients while reducing the chances of a stroke. One of the main clinical risk factors for stroke is atherosclerosis-induced hypertension, also, there are many other risk factors including smoking, physical inactivity, unhealthy diet, harmful use of alcohol, atrial fibrillation, elevated blood lipid levels, obesity, genetic predisposition, stress, and depression (World Stroke Organization, 2022).

In this case, the study was commissioned with the following data business objectives:

- Be able to predict, prevent and reduce the occurrence of stroke disease based on the patient's current status.

The expected outcome of the study:

- Be able to provide current patients with detailed information on the causes of their current stroke in terms of current health indicators, age, and life status.
- Reduces the likelihood of a potential patient having a stroke at an early stage.

1.3 Accessing the Situation

In order to achieve the goal of being able to analyze in detail the causes of stroke in stroke patients and to detect early manifestations of signs of stroke. This study should review a large

number of existing patient cases and engage them as a dataset to find models, as well as relationships between signs, data mining specialists should be applied.
Personnel. It is clear and confident that it is important to consult doctors and specialists in the field of stroke research, as well as those working in the field of healthcare and rehabilitation for stroke patients, and to obtain clear and detailed information about the disease itself, including the primary, secondary and direct causes of stroke. About the data and information collected from these stroke specialists, a database specialist should be consulted. Since these specialists hope the results of the study will become part of a continuing studying and research process, data warehousing and data cleaning for analysis are required. Also, the information about the patients involves their privacy, the data management must also be considered.
Data. The data required for the study analysis will come primarily from the records of doctors and healthcare professionals, where detailed desensitized data on patients is recorded. For initial studies, the data can be easily found on the internet, furthermore, the study data can be expanded as the study is conducted.
Risks. As the project is primarily concerned with healthcare, the accuracy of the model depends to a large extent on the quality and quantity of the dataset used for the study. The model production process should also be concerned with the risk of privacy breaches to users. Once generated, the models should be further validated by a team of experts who specialize in the treatment and research of stroke patients.

1.4 Data Mining Objectives

Experts in data mining contributed by translating the project's business objectives into data mining objectives. All data mining objectives studied in the initial phase are listed below.
In this case, the study was commissioned with the following data mining objectives:

- Identify a clear relationship between the patient's current health Index (current illness and BMI) and the stroke.
- Find out the relationship between the patient's age and the chances of stroke.
- Find the relationship between the patient's current life status (smoking status, marital status, work type, living environment, etc.) and the stroke.
- Reduce the chances of stroke by providing predictions based on the patient's current illness and age.
- Use this data to train and fit the model to make it suitable for use in data prediction.
  The validation phase uses a selection of healthcare records as a validation process to see the prediction results of a given model, and if the model successfully passes a pre-determined threshold, validation by healthcare professionals is continued.
  The initial use of the model phase allows the healthcare professional to enter a new patient's medical history, view the patient's stroke risk rating, and have the doctor diagnose to see if the system passes the threshold.
  The enhancement and optimization phase should continue to collect the required data and retrain and improve the model to increase accuracy.

The expected outcome of the study:

- Provide detailed information on the causes of the stroke to current patients.
- Provide useful information and advice to people who are at risk of stroke.

- Reduce the likelihood of stroke in potential patients at an early stage.

If the above objectives are achieved, the system should provide the ability to diagnose the patient's risk level.

1.5 Project Plan

The entire project is planned to take ten weeks of work and is planned as follows, also the detailed project schedule is shown in Gantt Chart in Figure 1.

Phase	Time	Resources	Risks
Business Understanding - Business background research - Business case study	1 week	Analysts	- Project change - Find the unsuitable case to study
Data Understanding - Dataset selection - Data exploration	2 weeks	Analysts	- Data problems - Attribute selections in the dataset
Data preparation - Data selection - Data cleaning - Data construction - Data integration - Data formatting	2 weeks	Data mining experts	- Technical problems
Modeling - Train the model - Evaluation of the performance - improve the performance	2 weeks	Database analyst time	- Model fitting problems - Technical problems
Validation - Validate the model by test sets	1 week	Analysts and tests	- Incompatible model - Incompatible test sets - Technical problems
Evaluation	1 week	Analysts	- Technical problems
Deployment	1 week	- Data mining consultant - Database analyst time	- Inability to deploy the whole project

Table 1. Project Plan

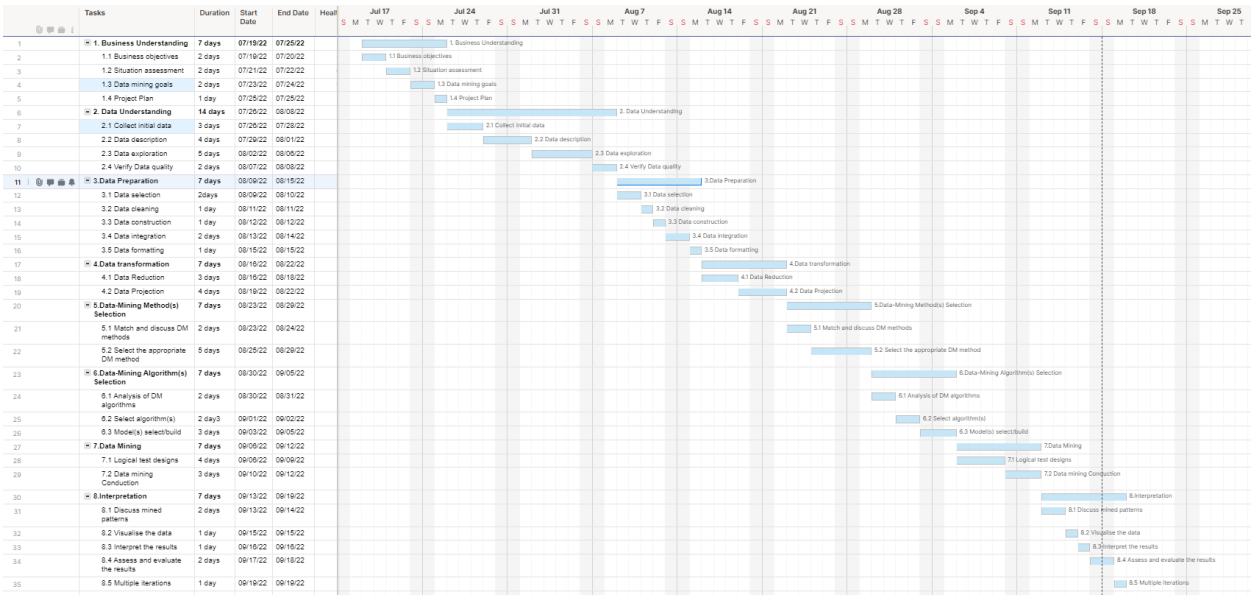


Figure 1. Gantt Chart of project schedule

2. Data Understanding

2.1 Data Understanding Overview

The healthcare information document is collected from the Internet, Kaggle. The data used will be accessed from a website link https://www.kaggle.com/datasets/lirilkumaramal/heart-stroke. The API command is kaggle datasets download -d lirilkumaramal/heart-stroke. It was last updated on 14 August 2021 (Amal, 2020).

2.2 Collecting Initial Data

The detailed information covered in the dataset is as below:

- Personal information. The raw data contain the basic information about individuals but does not relate to the disease itself, which includes age and gender.
- Health Index. The raw data contain a survey of diseases and basic physical indicators. Diseases investigated include the presence of hypertension and heart disease. The underlying physical indicators include mean glucose levels and BMI.
- Life status. The raw data contains living status and habits, whether married or not, type of work, type of residence, and smoking status.

In terms of data quality assumptions, firstly, the dataset is a desensitization dataset, as obviously the dataset did not include sensitive data, for example, name, address, postal code, and so on. Secondly, the dataset is rich enough for training a model for stroke prediction. Thirdly, the dataset is correctly collected from stroke patients or potential patients.

The main problem with this dataset is that it provides limited information about the origin of the dataset itself and only states that this dataset is widely used. This makes the source of this dataset, and in which region it was collected and generated, ambiguous in the study. Once the dataset has been fabricated, all of the findings and predictions in this article will be meaningless. After searching through the discussion section of this dataset, it was found that the data was actually collected by the clinic and the source of the data was deliberately

removed due to privacy concerns. After confirming the authenticity of the data sources, the reliability of this dataset was verified and approved for data mining.

2.3 Describing Data

Data Quantity

The format of the dataset has been packed as a .csv file, it can be easily accessed and converted to the desired format and used, for example, as a .xlsx file. The dataset contains 12 attributes and 43,400 records, in which, each record contains the information of individuals according to these 12 attributes, as shown below in Figure 2.

We use the Python Pandas framework to analyze data. Pandas is a Python library for working with datasets (W3Schools, 2022). It has functions for analyzing, cleaning, exploring, and manipulating data. Pandas enable us to analyze big data and make conclusions based on statistical theory. Pandas can clean up messy datasets and make them readable and relevant. And in data science, relevant data is very important.

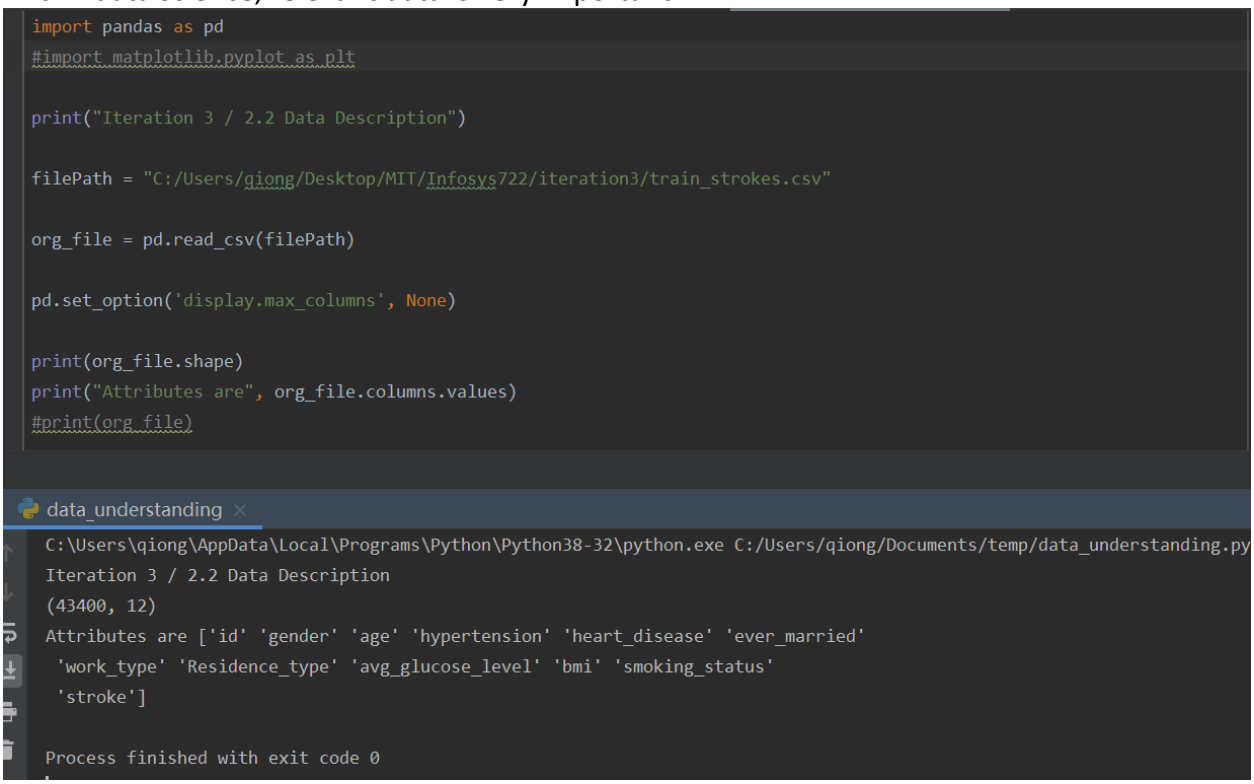
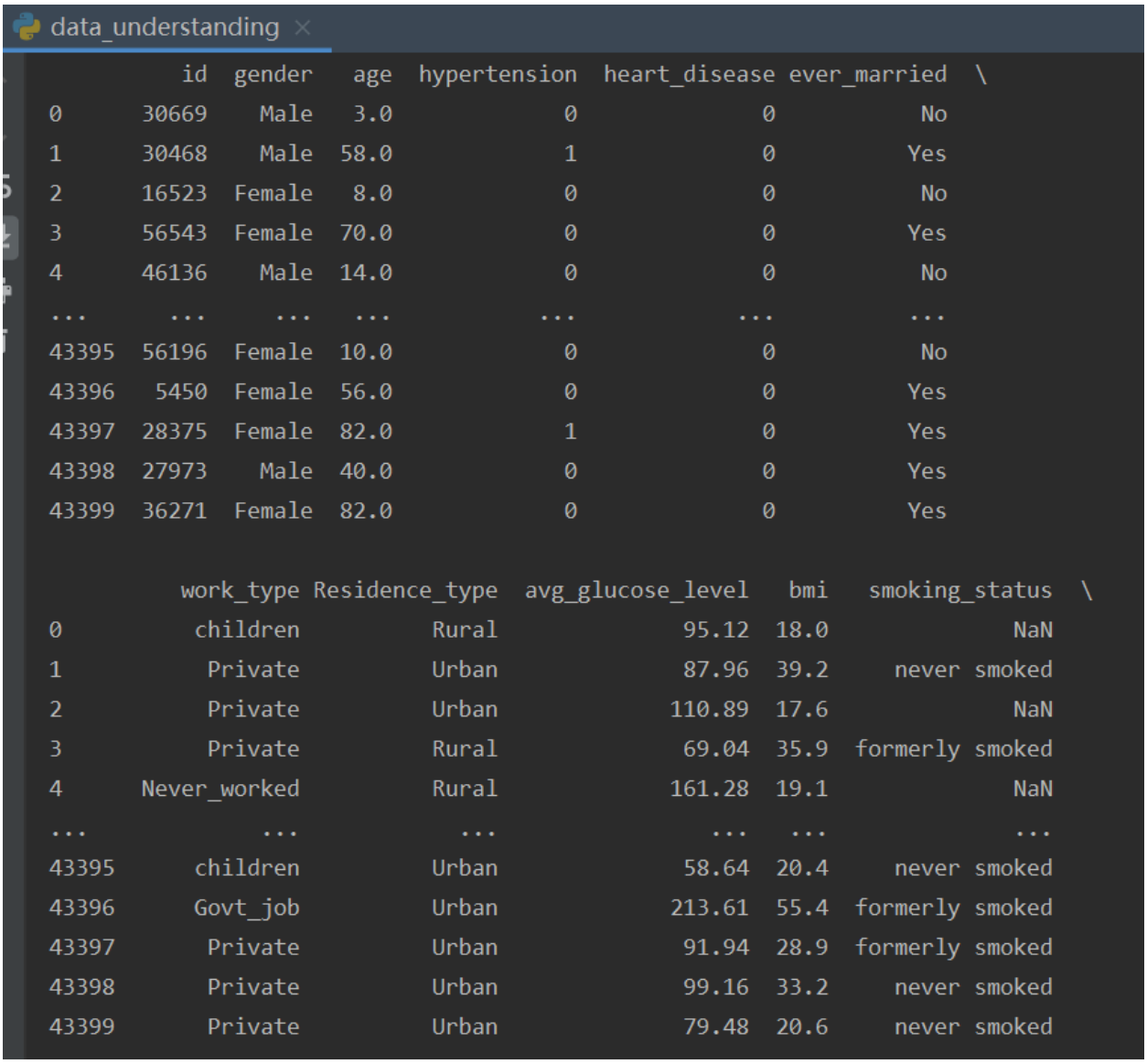


Figure 2. overview of the dataset using Python Pandas

The raw data is shown in Figure 3 & Figure 4.



	id	gender	age	hypertension	heart_disease	ever_married	\
0	30669	Male	3.0	0	0	No	
1	30468	Male	58.0	1	0	Yes	
2	16523	Female	8.0	0	0	No	
3	56543	Female	70.0	0	0	Yes	
4	46136	Male	14.0	0	0	No	
...	...	...	...	...	...	...	
43395	56196	Female	10.0	0	0	No	
43396	5450	Female	56.0	0	0	Yes	
43397	28375	Female	82.0	1	0	Yes	
43398	27973	Male	40.0	0	0	Yes	
43399	36271	Female	82.0	0	0	Yes	

	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	\
0	children	Rural	95.12	18.0	NaN	
1	Private	Urban	87.96	39.2	never smoked	
2	Private	Urban	110.89	17.6	NaN	
3	Private	Rural	69.04	35.9	formerly smoked	
4	Never_worked	Rural	161.28	19.1	NaN	
...	...	...	...	...	...	
43395	children	Urban	58.64	20.4	never smoked	
43396	Govt_job	Urban	213.61	55.4	formerly smoked	
43397	Private	Urban	91.94	28.9	formerly smoked	
43398	Private	Urban	99.16	33.2	never smoked	
43399	Private	Urban	79.48	20.6	never smoked	

Figure 3. Initial Data Overview 1

```
       stroke
0           0
1           0
2           0
3           0
4           0
...       ...
43395       0
43396       0
43397       0
43398       0
43399       0

[43400 rows x 12 columns]

Process finished with exit code 0
```

Figure 4. Initial Data Overview 2

Data Features

The objective of this project is to find the relationship between given information and the chance of stroke occurring. The information collected in the dataset, such as age, BMI, glucose, hypertension, history of heart disease, etc., were categorized early on as risk factors for stroke. The dataset is combining multiple formats of coding, the table 2 is showing the data type as well as the note of each attribute. Among these 12 attributes, 4 are processed as numeric data, 4 are processed as flags, and the rest 4 are processed as categories. These attributes provide sufficient information related to stroke. Therefore, based on medical research, it can be said that the information contained in the dataset has a strong relationship with stroke (Lim, 2021). The dataset provides a realistic picture of the individual, as each person is flagged stroke status and other related attributes as shown below.

Attribute	Value Type	Usage	Example
ID	Numeric	Unique Identifier	30468
Gender	Categorical (string)	Male, Female, Other	Male
Age	Numeric	Continues number	58
Hypertension	Boolean (Flag)	0 = No hypertension before; 1 = Yes	1 (Yes)
Heart_disease	Boolean (Flag)	0 = No heart disease before; 1 = Yes	0 (No)

Ever_married	Boolean (Flag)	0 = Never married; 1 = Married	Yes
Work_type	Categorical (string)	Type of work the individual has	Private
Residence_type	Categorical (string)	Rural/Urban	Urban
Avg_glucose_level	Numeric (Decimal)	The average glucose level of an individual	87.96
BMI	Numeric (Decimal)	The body mass index of an individual	39.2
Smoking_status	Categorical (string)	Formerly smoked/ smokes/never smoke	never smoked
Stroke	Boolean (Flag)	0 = No stoke happened before; 1 = Yes	0

Table 2. Attribute and Value types of Analysis

Inside this dataset, 41% of samples were recorded from male patients, and 59% of samples were from females. The average age of all patients was 42.2±22.5 with an average BMI of 28.6-±7.77, 64% of patients were ever married. 57% of patients worked for private companies, and 16% were self-employed. 50% of them lived in a rural place while others were urban. 90.64% of patients had a record of hypertension, and 95.25% of all patients were diagnosed with heart diseases and stroked happened on 98.20% of patients. The average glucose level of all patients was 104±43.1. In the records of smoking status for patients, 37% of them never smoked, the rest of them either did provide any records or were grouped into others.
At the current stage, the preference attributes associated with stroke are not known. Further exploration of the dataset is required.

2.4 Exploring Data

After observing the relationship between different attributes with target attribute stroke by pairplot using python seaborn package, as shown in Figure 5. With these attributes, we do not consider with the relationship between id and stroke, as it is obvious that ID is not an influential attribute toward stroke.

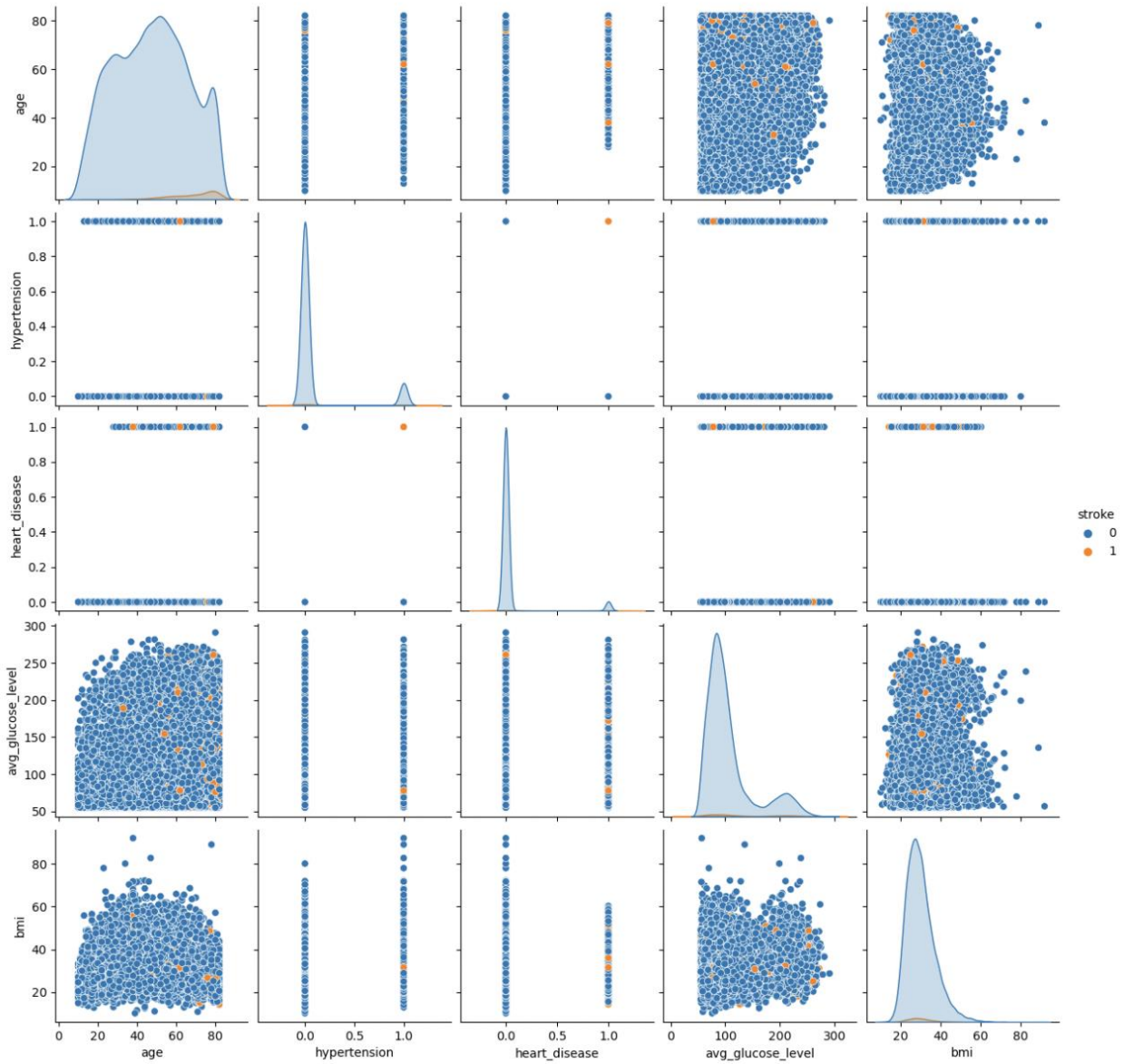


Figure 5. The pairplot between attributes

Then we take the first glance of the relationship between age, hypertension, heart disease, average glucose and BMI with target attribute stroke respectively.
From Figure 5, we can observe that

1. The intuitive hypothesis is that strokes are more likely to occur in older adults. This hypothesis was verified by the present dataset. The number of stroke cases among all different age patients were presented. A larger count circle indicates a higher records of stroke cases. Most of stroke cases were happened when age (>40) years old. And there are most stroke cases happened around 80 years old patients.
2. Patients without underlying conditions of hypertension are less likely to suffer a stroke.
3. Patients without underlying heart disease are less likely to have a stroke, and patients are more likely to have a stroke underlying heart disease.

4. The number of stroke cases among all different glucose level (56 - 260) patients were presented, but glucose level from 55 to 116 has most patients who did not get stroke before. And the patients who has glucose level higher than 271 did not get stroke in this dataset.
5. The number of stroke cases were analyzed in patients with different BMI. From the statistical result, patients with from 14.3 to 56.6 showed more cases of a stroke happening, while patients with a lower BMI (<15) or higher BMI (>55) showed less cases of stroke.

Figure 5 shows the predictors that are important for the wind in the target attributes and also shows the relationship between each attribute. The figure shows that age, hypertension, heart disease, blood glucose, and BMI all have a direct and observable effect on stroke. Of these, age is the single most important risk factor for stroke, and as stated in the figure, older people have a greater chance of having a stroke. Hypertension and heart disease and other such underlying conditions are also important predictors for stroke, while a clear relationship between blood glucose and BMI for stroke is difficult to observe at this stage and further exploration is needed.

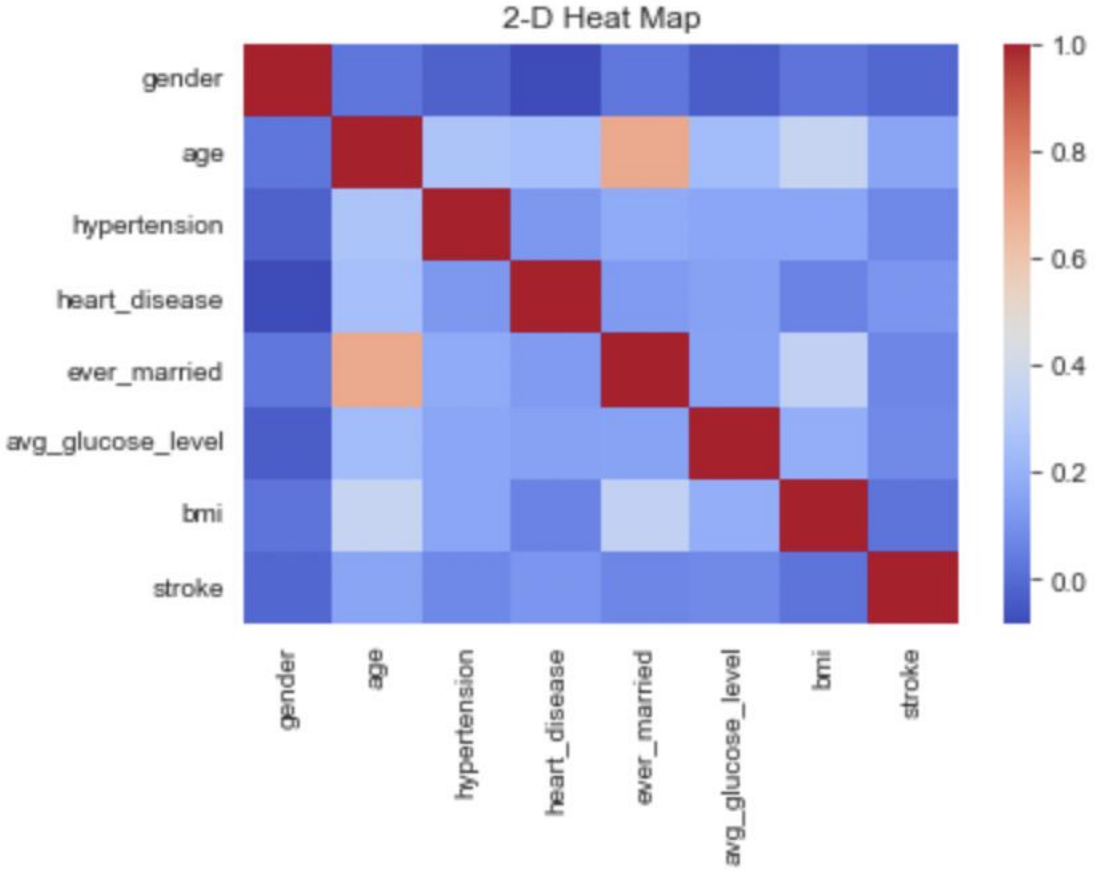


Figure 6: The correlation heatmap between attributes

Figure 6 shows the heat map of correlations between the attributes of the dataset, the most influential attribute is the age of the individual, followed by history of heart disease, average blood glucose level and hypertension status, the least relevant attribute is the type of residence, which can be considered as the least important attribute.

In the next data cleaning process, we will filter and remove some less relevant attributes, such as the type of work mentioned earlier. And keeping the goal of data mining will remain the same, which is to discover the relationship between the attributes and the final result. In addition, sensitive data will be processed, such as extreme values. another aspect, it is necessary to use a subset of instances. Too much difference between the two labels can affect the final accuracy. Therefore, certain instances and attributes need to be selected to eliminate potential waste of time and processing costs. And for this amount of data, we need to perform further filtering.

Furthermore, we can see from Figure 7, the stroke cases are extremely imbalance, only 783 (1.8%) patients got stroke, other 42,617 (98.2%) patients did not get stroke. Though the relationship between the different attributes is not yet clear, the attributes gender, age, health index (current illness: hypertension, heart disease, average glucose level; BMI) and work type needs more investigation as they may not be removed due to the imbalance in stroke cases.

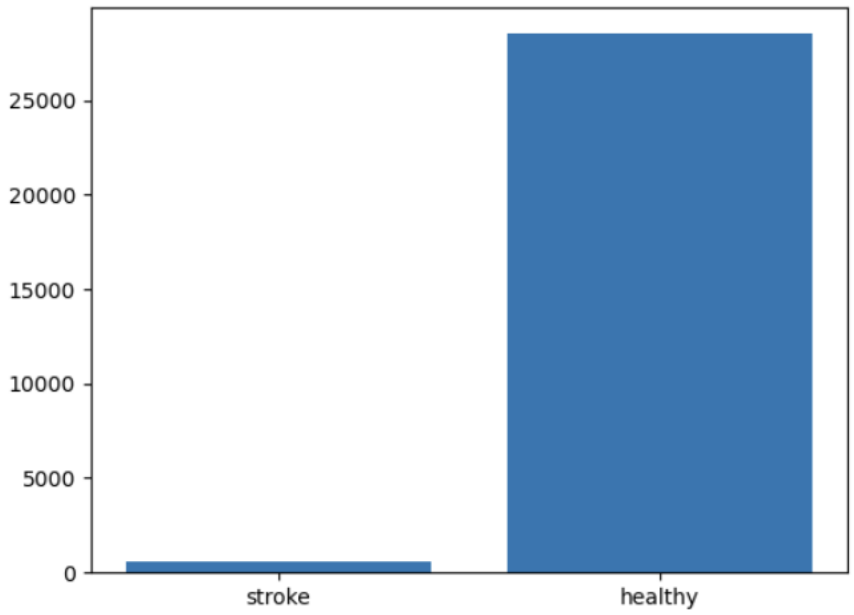


Figure 7. Distribution of stroke cases

2.5 Verifying Data Quality

Identifying the quality of the entire dataset, the percentage of attribute completion is 83.33%, and the percentage of records completion is 66.99%. The main reason for lowering the records completion rate is the incompletion of attribute smoking status of individuals, since only 69% of the patients provided their status of smoking. Also, 96.631% completion rate of BMI for individuals also lowers the total completion rate compared to the 100% completion rate for other attributes. As shown in Figure 8, we can see there are missing values in attributes BMI and smoking status. Because this data set is a median size dataset, for later identifying the

relationship between the attribute of smoking status and the possibility of stroke, we could only use these 66.99 % datasets (with smoking status and BMI completed) to train the related prediction model.
Furthermore, there are 1,462 and 13,292 records are missing/null values in BMI and smoking status respectively. Totally there are 14,326 records contain missing values in the raw dataset, as shown in Figure 9.

```
#-------------------------------------------------------------------------------
print("Iteration 3 / 2.4 Data Quality")
# Completion (Missing data for each attributes)

# a变量用了isnull函数，返回值如果是false说明不是空值，如果是true就有空值
# missing_status with the isnull function, the return value if it is false means not null,
# if it is true then there is a null value
missing_status = pd.isnull(raw_data)
# any函数，就是对每个列去找有没有TRUE，true也就是存在空值，结果是全FALSE
# any, that is, for each column to find whether there is TRUE, true that is,
# the existence of null values
print(missing_status.any())
```

```
Iteration 3 / 2.4 Data Quality
id                  False
gender              False
age                 False
hypertension        False
heart_disease       False
ever_married        False
work_type           False
Residence_type      False
avg_glucose_level   False
bmi                  True
smoking_status       True
stroke              False
dtype: bool
```

Figure 8. Show whether have missing values for each attribute

```
remove_missing_data = raw_data.copy()
remove_missing_data.dropna(axis=0, how='any', inplace=True)
missing_bmi = missing_status[missing_status['bmi']==True]
missing_smoke = missing_status[missing_status['smoking_status']==True]
```

```
print('There are' ,missing_bmi.shape[0],'missing data in bmi.')
print('There are' ,missing_smoke.shape[0],'missing data in smoking.')
print('There are' ,raw_data.shape[0]-remove_missing_data.shape[0],'missing data in the raw data.')
```

```
There are 1462 missing data in bmi.
There are 13292 missing data in smoking.
There are 14328 missing data in the raw data.
```

Figure 9. Show number of missing values

The outliers/extremes are calculated and identified in python as shown in Figure 10. We can see there are 0 outliers in age, 412 outlier data in BMI, 645 outliers in average glucose level. We can only identify outliers for attribute age, BMI and average glucose level, because there is no mean to identify outliers for id, it is a unique identifier, we will not analyze the relationship between id and stroke; for attribute hypertension, heart disease, ever married, and stroke, they are Boolean values, there are no outlier values; for attribute gender, work type, residence type, smoking status, their values are recorded as string, which the outlier value cannot be identified using computer program. Because the way computer calculated outliers for attribute is that computer finds the mean value of each attribute and finds the range where most of the values lie according to the normal distribution function and calculates the deviation for each value in the attribute, and values with a large deviation (those that differ too much from the group characteristics are considered outliers). The presence of outliers can have a significant impact on data analysis.

```
# calculate outliers
threshold = 3
zscore_data = raw_data.copy()
zscore_data = zscore_data.drop(columns=['id'])
for col in ['age', 'avg_glucose_level', 'bmi']:
    series = zscore_data[col]
    zscore = (series - series.mean()) / series.std()
    zscore_data[col] = zscore.abs() > threshold
```

```
outlier_age = zscore_data[zscore_data['age']==True]
outlier_bmi = zscore_data[zscore_data['bmi']==True]
outlier_ave_glucose = zscore_data[zscore_data['avg_glucose_level']==True]
print('There are' ,outlier_age.shape[0],'outlier data in age.')
print('There are' ,outlier_bmi.shape[0],'outlier data in bmi.')
print('There are' ,outlier_ave_glucose.shape[0],'outlier data in avg_glucose_level.')
```

```
There are 0 outlier data in age.
There are 412 outlier data in bmi.
There are 645 outlier data in avg_glucose_level.
```

Figure 10. Show outliers

Based on the observation of data quality inside python, we found missing data and extreme data (most likely data errors) in this database. It is not difficult to understand that in the cycle of information collection it is unlikely that perfect data will be available, due to privacy or sensitivity issues etc. And problems with the data will often be discovered when the data is analyzed as the project progresses. Here are some of the potential issues that can arise during the execution of a project.

- **Missing data.** As can be seen from this database, the missing data is obvious and expected. In the questionnaires used to collect data, privacy concerns and other concerns caused patients to fill in questionnaires without filling in detailed personal information such as BMI, marital status and place of work, among other things.
- **Data errors.** There are two possibilities for causing data errors and incorrect data entry at the time of recording. In the medical dataset used in this study, each record was manually recorded into the system by the healthcare professionals and a small amount of incorrect data entry was inevitable. In addition, machine misinterpretation, resulting in incorrect data being read in, as different machines use different ways of reading data. And most of these small amounts of incorrect data can be eliminated in most cases by filtering/processing outliers.
- **Measurement errors.** When data is measured without strict adherence to the measurement rules, small errors do occur from time to time or there is also a risk that units are reported incorrectly.
- **Imbalance numbers of labels.**
  As mentioned earlier, the distribution of labels was inconsistent. Only 783 instances were labeled as strokes, while the remaining 42,617 samples were labeled as non-strokes. The large discrepancy is likely to make the model fit less good and insensitive to unseen values.

Attribute	Value Type	Completeness	Extreme / Outlier
ID	Numeric	N/A	N/A
Gender	Categorical (string)	100%	N/A
Age	Numeric	100%	0
Hypertension	Boolean (Flag)	100%	N/A
Heart_disease	Boolean (Flag)	100%	N/A
Ever_married	Boolean (Flag)	100%	N/A
Work_type	Categorical (string)	100%	N/A
Residence_type	Categorical (string)	100%	N/A
Avg_glucose_level	Numeric (Decimal)	100%	645
BMI	Numeric (Decimal)	96.631% （1462/43400）	412
Smoking_status	Categorical (string)	69.373%（13292/43400）	N/A
Stroke	Boolean (Flag)	100%	N/A

Table 3. Analysis raw dataset quality

3. Data Preparation

3.1 Data Selection

Data selection, as the initial step in the data cleaning process, aims to remove some unnecessary attributes and eliminate the number of instances. It can be divided into two main parts, selecting items (row selection) and selecting attributes (column selection).

Items (row) selection: All instances of the initial dataset will be used to have a taste of how biased the dataset is as below in Figure 11 using Tableau. The results are categorized as 0 and 1, with 0 indicating no stroke and 1 indicating the occurrence of a stroke. However, there is a significant imbalance between two categories, as only 783 instances were marked as stoke and the remaining 42,617 records were cited as no stoke as below in Figure 12.

For the row selection, we tried to keep all the original data as much as possible to reduce the impact of noise and deletion of real data on the later model building. Because more real data, build better model, which will predict more accurate for target attribute for long-term and future use purpose. We try to keep the data of each entry in this session, and then reduce the data rows as needed when doing data cleaning, handling empty data, data errors and noise in the following sessions.

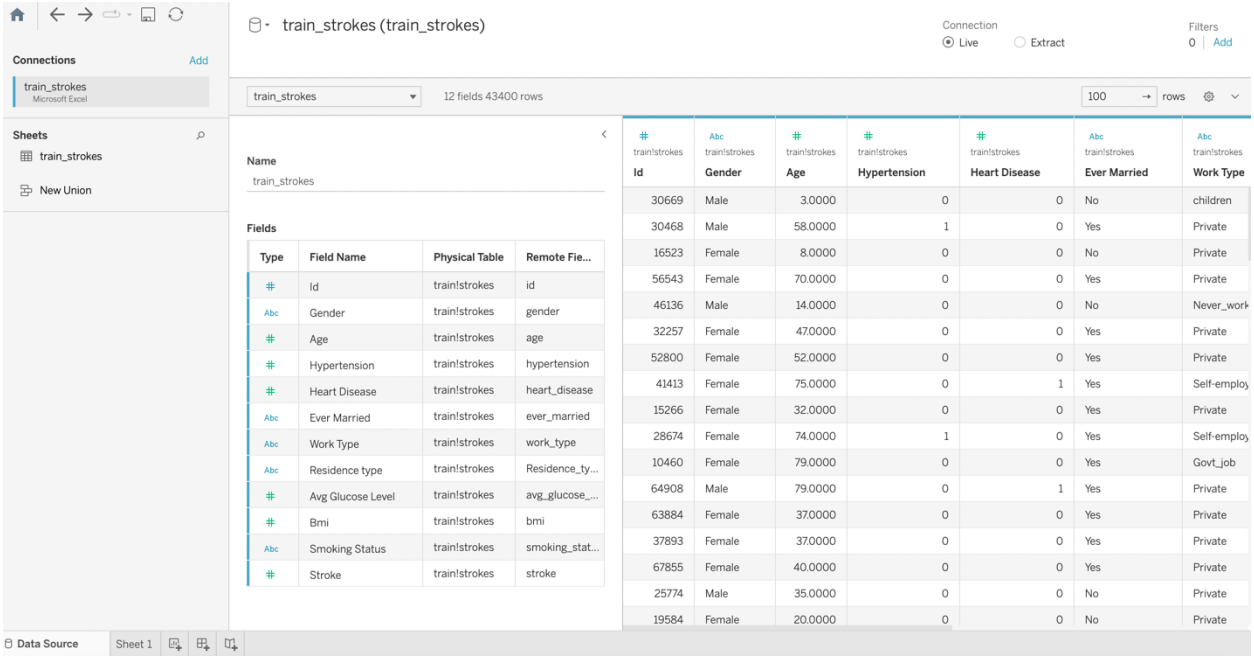


Figure 11. Items (row) selection of the dataset

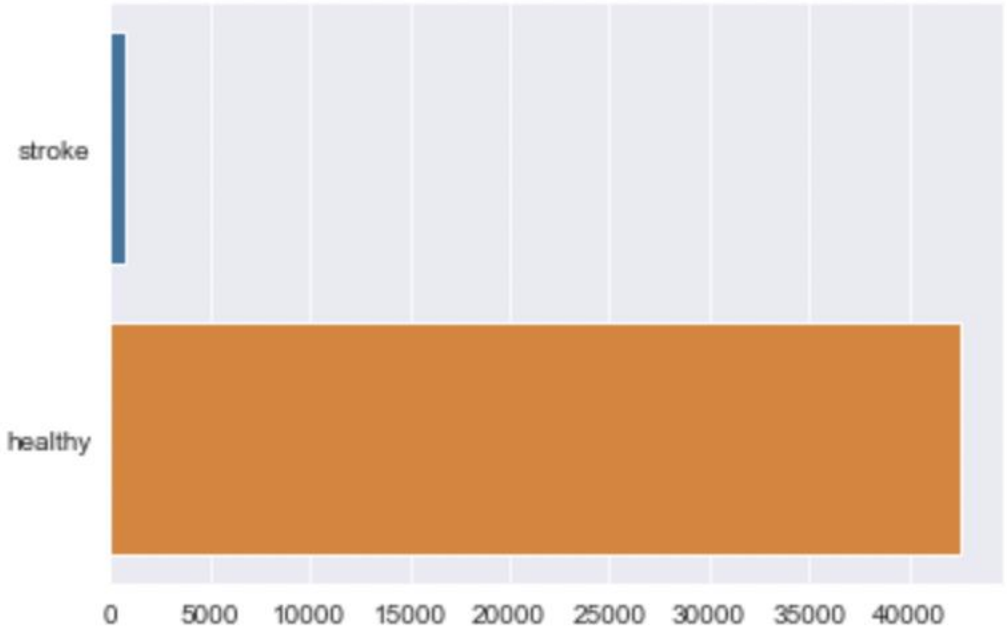


Figure 12. Analyze the target attribute based on Items (row) selection

Attribute (column) selection: According to (Barnett & H, 2005; Chong, 2020; Wajngarten & Silva, 2019), risk factors for stroke include heart disease, hypertension, smoking habits, diabetes, and obesity, but of these risk factors, the type of dwelling is not one of them. Furthermore, according to section 2.4, some attributes contribute less to stroke, which means that they are less critical attributes. The presence of these values has the potential to influence the judgements and patterns of the algorithm. Once these tributes have been removed, the algorithm will start to move along the right track towards the correct and robust pattern.
As we did data exploration in section 2.4, residence type is not a risk, which has no effect on the outcome that ultimately leads to stroke. Therefore, it can be removed from further studies.
Also, the attribute id is not needed for further studies, as id is an unique identifier, we do not need it to build the model, and it has no relationship with target attribute stroke.
Figure 13 below illustrates the attribute selection process. We have deleted the attribute of Residence type, and can see the data frame after column selection, there are 10 attributes left.
Figure 14 below illustrates the dataset after column selection.

```
In 15  1  #-------------------------------------------------------------------------------
       2  print("Iteration 3 / 3.1 Data Selection")
       3  # step 1, row selection, keep all rows
       4  # step 2, column selection: drop Residence_type
       5  column_selection = drop_id.copy()
       6  column_selection = column_selection.drop(columns=['Residence_type'])

          Iteration 3 / 3.1 Data Selection

yter:  Server: Infosys 722   Variables: understanding.ipynb ×
> column_selection = {DataFrame: (43400, 10)} gender  age  hypertension  heart_disease  ever_married  work_type
```

Figure 13. Attribute selection process

```
        gender   age  hypertension  heart_disease  ever_married     work_type  \
0            0   3.0             0              0             0      children
1            0  58.0             1              0             1       Private
2            1   8.0             0              0             0       Private
3            1  70.0             0              0             1       Private
4            0  14.0             0              0             0  Never_worked
...        ...   ...           ...            ...           ...           ...
43395        1  10.0             0              0             0      children
43396        1  56.0             0              0             1      Govt_job
43397        1  82.0             1              0             1       Private
43398        0  40.0             0              0             1       Private
43399        1  82.0             0              0             1       Private

       avg_glucose_level   bmi   smoking_status  stroke
0                  95.12  18.0              NaN       0
1                  87.96  39.2     never smoked       0
2                 110.89  17.6              NaN       0
3                  69.04  35.9  formerly smoked       0
4                 161.28  19.1              NaN       0
...                  ...   ...              ...     ...
43395              58.64  20.4     never smoked       0
43396             213.61  55.4  formerly smoked       0
43397              91.94  28.9  formerly smoked       0
43398              99.16  33.2     never smoked       0
43399              79.48  20.6     never smoked       0

[43400 rows x 10 columns]
```

Figure 14. Attribute (column) selection of the dataset

3.2 Cleaning Data

The original files of the dataset had gaps and unfinished areas and these errors were often missed or deliberately ignored by those who did not record them in the dataset due to privacy concerns or other reasons of consideration. Two types of errors were found in the processed data, missing values and data errors. These values will be removed or replaced in preparation for further use of the data.

Missing values:

A total of two attributes were found to contain missing values, with integrity values of 96.631% for BMI and 69.373% for smoking status. The inclusion of 14 missing values can reduce the efficiency of the module's execution. Therefore, they need to be removed before being applied to any model. This step eliminates the problem when applying data to certain algorithms, making the algorithm less skewed in the wrong direction.

A null or empty value is not very compatible with some algorithms. It is usually represented in many different forms, such as empty cells as well as N/A markers. In this dataset, both smoking status and BMI contain these missing values. There are three ways to remove them: directly from the record or populated according to some pattern (usually copied from the last populated instance) or populated with a fixed number derived from the averaging algorithm. For the missing BMI and smoking status values in this project, the best approach is to delete these instances directly, because there are three smoking statuses and the missing values occupy 31% of the existing dataset, which is too large a proportion, and filling the values randomly or copying from later instances would likely cause the dataset to lose its original characteristics and lead to much less accurate prediction results.

The BMI value is also difficult to fill with a fixed number and it is also difficult to fill with the following pattern. Since most of the instances in this empty subset were marked as non-stroke, deleting these instances would not have a significant impact on the shortage of stroke cases. Therefore, it was decided to remove these instances where the missing values for BMI and smoking status were located. Figure 15 show the process of removing null/missing values in the dataset, Figure 16 shows the results of removing the null values from the BMI and smoking status attribute.

```
1  #----------------------------------------
2  print("Iteration 3 / 3.2 Data Cleaning")
3  # Remove empty values
4  row_selection = column_selection.dropna()
5  print(row_selection)
```

Figure 15. Cleaning missing values process

Selected dataset after removing missing values as Figure 16 below:

```
Iteration 3 / 3.2 Data Cleaning
       gender   age  hypertension  heart_disease  ever_married      work_type  \
1           0  58.0             1              0             1        Private
3           1  70.0             0              0             1        Private
6           1  52.0             0              0             1        Private
7           1  75.0             0              1             1  Self-employed
8           1  32.0             0              0             1        Private
...       ...   ...           ...            ...           ...            ...
43395       1  10.0             0              0             0       children
43396       1  56.0             0              0             1       Govt_job
43397       1  82.0             1              0             1        Private
43398       0  40.0             0              0             1        Private
43399       1  82.0             0              0             1        Private

       avg_glucose_level   bmi   smoking_status  stroke
1                  87.96  39.2     never smoked       0
3                  69.04  35.9  formerly smoked       0
6                  77.59  17.7  formerly smoked       0
7                 243.53  27.0     never smoked       0
8                  77.67  32.3           smokes       0
...                  ...   ...              ...     ...
43395              58.64  20.4     never smoked       0
43396             213.61  55.4  formerly smoked       0
43397              91.94  28.9  formerly smoked       0
43398              99.16  33.2     never smoked       0
43399              79.48  20.6     never smoked       0

[29072 rows x 10 columns]
```

Figure 16. Dataset after cleaning missing values

The method used to remove the missing values is to use dropna() function in python, which selects all valid values and filters out those uncompleted values in BMI and smoking status values(rows). After this step, there are 29,072 records left, which means 14,328 records are reduced. When this algorithm is applied, the completeness of all attributes is increased to 100%.

Data Errors:

The only data errors contained in the file are extreme/outlier values, which can affect the module's predictions as the module may be biased towards outliers as it tries to cover many different possible values.

From Figure 10, we know that there are 0 outliers in age, 412 outlier data in BMI, 645 outliers in average glucose level. These extreme values are removed using forced deletion. This method will attempt to drag the values from the extreme range to a reasonable range within the range.

Figure 17 show the process of removing outlier values in the dataset, Figure 18 shows the results of removing the outliers from the age, BMI and average glucose level attribute.

```
# Remove outlier
# calculate outliers
threshold = 3
zscore_data = row_selection.copy()
for col in ['age', 'avg_glucose_level', 'bmi']:
    series = zscore_data[col]
    zscore = (series - raw_data[col].mean()) / raw_data[col].std()
    zscore_data[col+'_flag'] = zscore.abs() > threshold
zscore_data.drop(zscore_data[(zscore_data.age_flag == True) | (zscore_data.bmi_flag == True) |
                             (zscore_data.avg_glucose_level_flag == True)].index, axis=0, inplace=True)
row_selection = zscore_data.drop(columns=['age_flag', 'bmi_flag', 'avg_glucose_level_flag'])
```

Figure 17. Cleaning outliers process

```
Iteration 3 / 3.2 Data Cleaning
       gender   age  hypertension  heart_disease  ever_married work_type  \
1           0  58.0             1              0             1   Private
3           1  70.0             0              0             1   Private
6           1  52.0             0              0             1   Private
8           1  32.0             0              0             1   Private
11          0  79.0             0              1             1   Private
...       ...   ...           ...            ...           ...       ...
43394       0  47.0             0              0             0  Govt_job
43395       1  10.0             0              0             0  children
43397       1  82.0             1              0             1   Private
43398       0  40.0             0              0             1   Private
43399       1  82.0             0              0             1   Private

       avg_glucose_level   bmi   smoking_status  stroke
1                  87.96  39.2     never smoked       0
3                  69.04  35.9  formerly smoked       0
6                  77.59  17.7  formerly smoked       0
8                  77.67  32.3           smokes       0
11                 57.08  22.0  formerly smoked       0
...                  ...   ...              ...     ...
43394              68.52  25.2  formerly smoked       0
43395              58.64  20.4     never smoked       0
43397              91.94  28.9  formerly smoked       0
43398              99.16  33.2     never smoked       0
43399              79.48  20.6     never smoked       0

[28259 rows x 10 columns]
```

Figure 18. Dataset after cleaning missing values

The method used to remove the outliers is to use zscore method to calculate the deviation of the value with normal distribution, which selects all values in the rage of normal distribution and filters out those outlier values in age, BMI and average glucose(rows). After this step, there are 28,259 records left, which means 813 records are reduced.

Figures 19 and 20 are the distribution boxplot (before and after) outlier removal status for the attribute BMI. It totally removed five extreme value and modified 11 instances.
Figures 21 and 22 are the distribution boxplot (before and after) outlier removal status of the attribute of Average glucose level. The boxplot is still containing some outliers that are being removed.

We can see that after remove outliers from BMI and average glucose attribute, the distribution of values in these 2 attributes are more normally distributed, we reduced the potential noise for future model building and training.

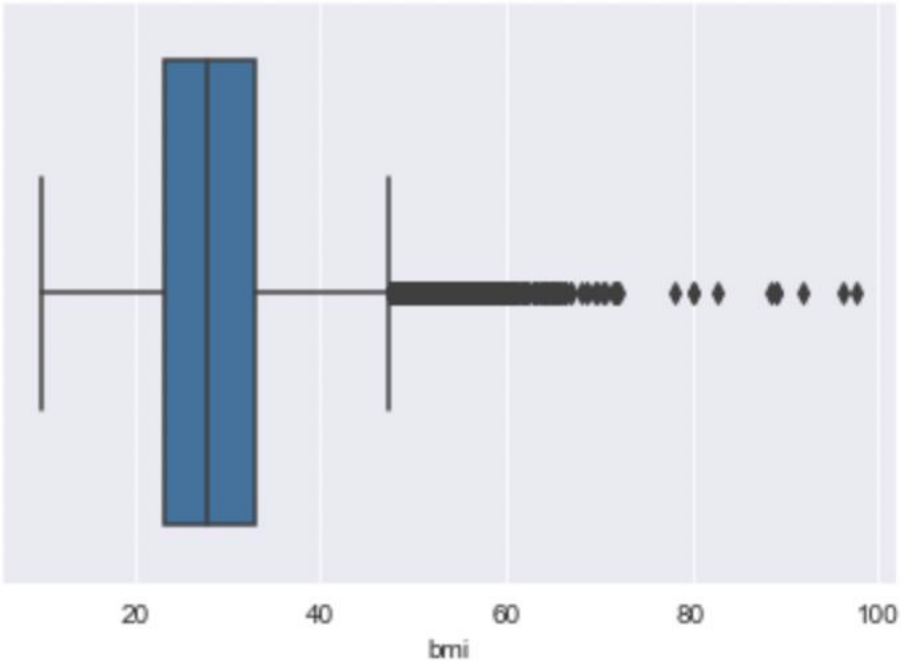


Figure 19: The boxplot of BMI (before clean outliers)

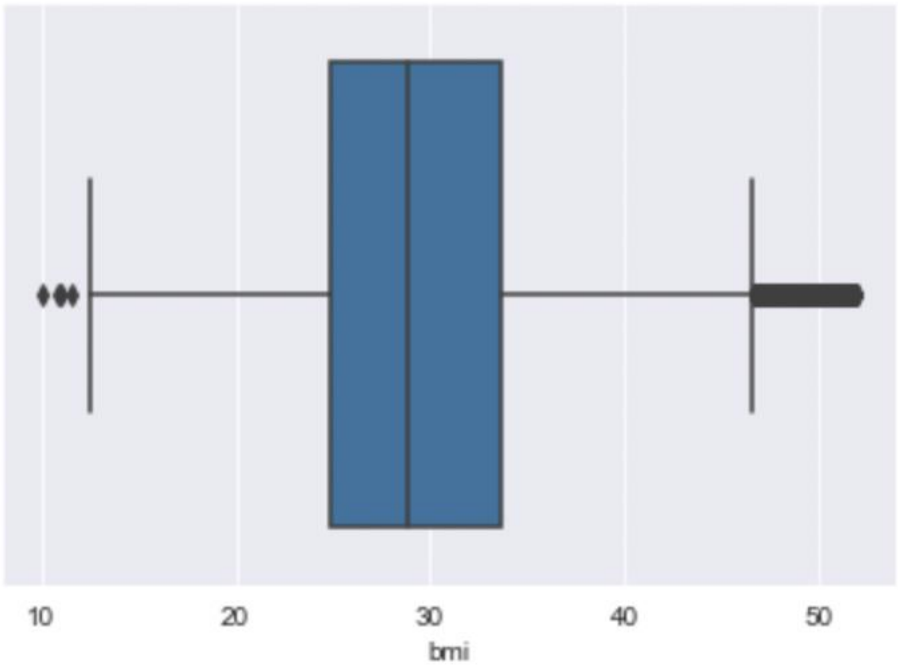


Figure 20: The boxplot of BMI (after clean outliers)

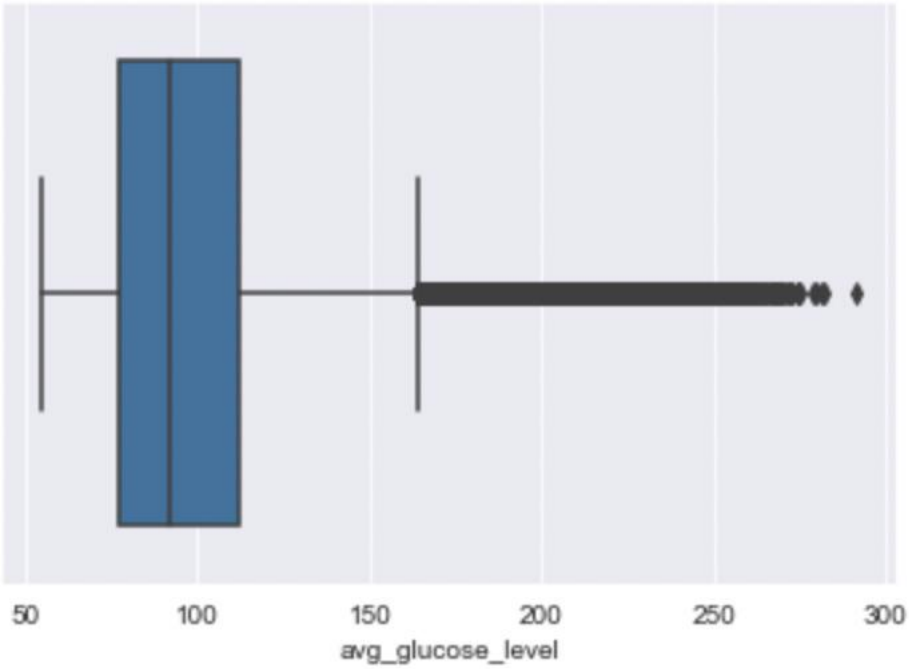


Figure 21: The boxplot of Average glucose level (before clean outliers)

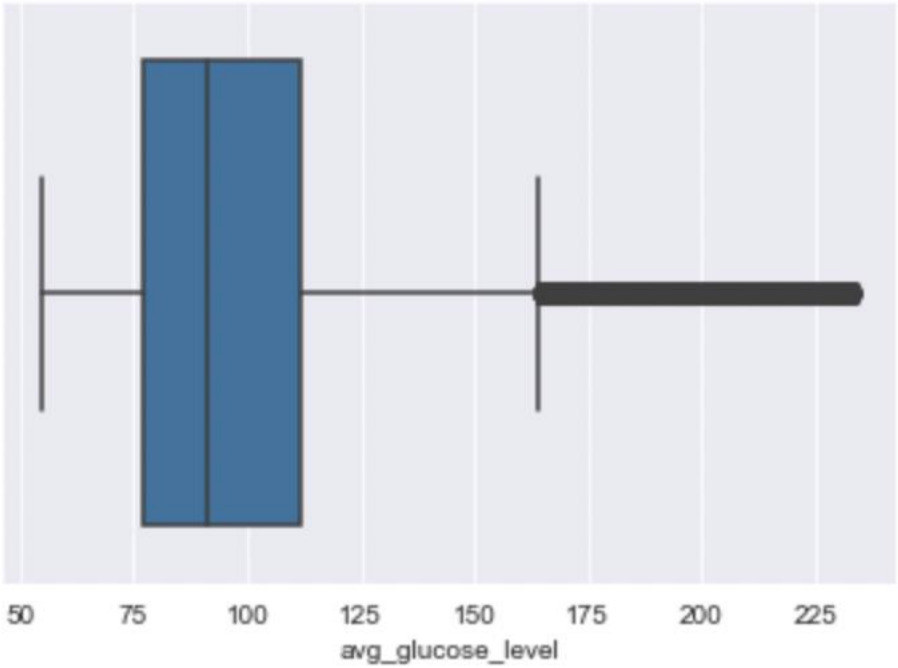


Figure 22: The boxplot of Average glucose level (after clean outliers)

From Figure 23 and Figure 24, we can see the difference of before cleaning dataset and after cleaning dataset.
Figure 23 is showing the dataset before data cleaning.

Figure 24 is showing the dataset after data cleaning.

```
       gender   age  hypertension  heart_disease  ever_married     work_type  \
0           0   3.0             0              0             0      children
1           0  58.0             1              0             1       Private
2           1   8.0             0              0             0       Private
3           1  70.0             0              0             1       Private
4           0  14.0             0              0             0  Never_worked
...       ...   ...           ...            ...           ...           ...
43395       1  10.0             0              0             0      children
43396       1  56.0             0              0             1      Govt_job
43397       1  82.0             1              0             1       Private
43398       0  40.0             0              0             1       Private
43399       1  82.0             0              0             1       Private

       avg_glucose_level   bmi   smoking_status  stroke
0                  95.12  18.0              NaN       0
1                  87.96  39.2     never smoked       0
2                 110.89  17.6              NaN       0
3                  69.04  35.9  formerly smoked       0
4                 161.28  19.1              NaN       0
...                  ...   ...              ...     ...
43395              58.64  20.4     never smoked       0
43396             213.61  55.4  formerly smoked       0
43397              91.94  28.9  formerly smoked       0
43398              99.16  33.2     never smoked       0
43399              79.48  20.6     never smoked       0

[43400 rows x 10 columns]
```

Figure 23. Dataset before data cleaning

```
Iteration 3 / 3.2 Data Cleaning
       gender   age  hypertension  heart_disease  ever_married work_type  \
1           0  58.0             1              0             1   Private
3           1  70.0             0              0             1   Private
6           1  52.0             0              0             1   Private
8           1  32.0             0              0             1   Private
11          0  79.0             0              1             1   Private
...       ...   ...           ...            ...           ...       ...
43394       0  47.0             0              0             0  Govt_job
43395       1  10.0             0              0             0  children
43397       1  82.0             1              0             1   Private
43398       0  40.0             0              0             1   Private
43399       1  82.0             0              0             1   Private

       avg_glucose_level   bmi   smoking_status  stroke
1                  87.96  39.2     never smoked       0
3                  69.04  35.9  formerly smoked       0
6                  77.59  17.7  formerly smoked       0
8                  77.67  32.3           smokes       0
11                 57.08  22.0  formerly smoked       0
...                  ...   ...              ...     ...
43394              68.52  25.2  formerly smoked       0
43395              58.64  20.4     never smoked       0
43397              91.94  28.9  formerly smoked       0
43398              99.16  33.2     never smoked       0
43399              79.48  20.6     never smoked       0

[28259 rows x 10 columns]
```

Figure 24. Dataset after data cleaning

3.3 Constructing New Data

To construct new data by creating new features, here three new features are created, new_smoking_status, new_bmi, new_work_type. New smoking status is set as a flag type from the nominal type of smoking status, the New BMI is re-grouped by a certain range of values and the new work type is re-classified by general knowledge base respectively.

1. **The new smoking status**
   The initial smoking status includes formerly smoked, smoking, and never smoking. Studies show whether smoke or not is a key predictor of our target attribute stroke. Therefore, smoking status is flagged as smoking and never smoking, as Figure 25 shows.

```
#-------------------------------------------------------------------------------
print("Iteration 3 / 3.3 Feature Creation")
smoking_list = [i for i in list(row_selection['smoking_status'])]
smoke = []

for s in smoking_list:
    if s.strip() == 'formerly smoked':
        smoke.append(1)
    elif s.strip() == 'smokes':
        smoke.append(1)
    else:
        smoke.append(0)

featured_data = row_selection.drop(['smoking_status'], axis=1)
featured_data = featured_data.assign(new_smoking_status=np.array(smoke))
```

Figure 25. New Smoking status after set a flag

2. **BMI groups:**

BMI stands for body mass index and is a simple calculation using a person's height and weight. The formula is BMI = kg/m2, where kg is a person's weight in kilograms and m2 is the square of their height in meters. A BMI of 25.0 or higher is considered overweight, while a healthy range is 18.5 to 24.9. And one of the risk factors for stroke, the target attribute of this study, is BMI. for these reasons, I created the characteristics of the weight categories based on the BMI results as Figure 26 shows.

```
for b in bmi_list:
    if b < 18.5:
        bmi.append('Underweight')
    elif b <= 24.9:
        bmi.append('Normal')
    elif b <= 29.9:
        bmi.append('Overweight')
    else:
        bmi.append('Obese')

featured_data = featured_data.drop(['bmi'], axis=1)
featured_data = featured_data.assign(new_bmi=np.array(bmi))
```

Figure 26. New BMI groups

3. **New Work type:**
   The work type initially includes five categories: children, never worked, self-employed, government jobs, and private. Some of these types should be grouped together because these job categories have a high degree of similarity to each other. Also, by merging these similar job types, the complexity of the module is reduced at the same time. It was decided to combine the categories Child and Never Worked into Never Worked and Self-Employed and Private Company Work into self-employed, as shown in Figure 27.

```
job_list = [i for i in list(featured_data['work_type'])]
job = []

for j in job_list:
    if j == 'Private':
        job.append('Self-employed')
    elif j == 'Self-employed':
        job.append('Self-employed')
    elif j == 'children':
        job.append('Never_worked')
    elif j == 'Never_worked':
        job.append('Never_worked')
    else:
        job.append('Govt_job')

featured_data = featured_data.drop(['work_type'], axis=1)
featured_data = featured_data.assign(new_work_type=np.array(job))
print(featured_data)
```

Figure 27. New Work categories

3.4 Integrating Data

Following the data cleaning process and constructing new features, the data cleaning process succeeded in removing the existing extreme/outlier and missing values. However, the imbalanced label problem is not solved, and this problem needs to be handled before building a model and do data mining.

After the data cleaning process and the construction of new features, the data cleaning process successfully eliminated the existing extreme/outliers and missing values. However, the imbalanced labeling problem was not solved, and this issue needs to be addressed before

building the model and performing data mining. Unbalanced labels, with too large a difference in the number of stroke patients and non-stroke patients, can lead to the prior probability of the data itself, which can have a bias towards negative samples of stroke. And according to Bayesian theory, the prior probability is the probability that can be obtained before the model experiment or sampling based on previous experience and analysis. To put it in layman's terms, there are now 513 samples labeled as stroke and 27,746 samples labeled as not stroke in the dataset as shown in Figure 28. Using this dataset to train the model, the model will most likely predict the patients who do not have a stroke accurately, while the patients with high risk of stroke are predicted to have a high probability of not having a stroke. This model, however, has a high risk of not being able to accurately predict the likelihood of stroke for patients at an early stage in practical use, which is contrary to our goal of doing this project and modeling each other. To address the potential dangers of such label imbalances, we need to rebalance the data set between stroke and healthy (non-stroke) patients.

```
# Draw featured_data stroke distribution
featured_data_stroke = featured_data[featured_data['stroke']==1]
featured_data_healthy = featured_data[featured_data['stroke']==0]
num_stroke = len(featured_data_stroke)
num_healthy = len(featured_data_healthy)

print("There are",num_stroke,"records are marked stroke.")
print("There are",num_healthy,"records are marked as healthy (non-stroke).")
```

```
There are 513 records are marked stroke.
There are 27746 records are marked as healthy (non-stroke).
```

Figure 28. Current stroke distribution

Before balancing the stroke and non-stroke samples, we need to perform data partitioning first, because we only want to balance the data inside the training set so that the model can have enough balanced data for training and can predict the results of the real data more accurately. The data in the test set maintains the characteristics of the original real data set as much as possible, which can also better test the results of the model training. Therefore, we split the dataset to train and test datasets based on 7:3 ratio.

The main idea of partitioning the dataset into validation sets is to prevent our model from overfitting, i.e., the model becomes very good at classifying samples in the training set but cannot generalize and accurately classify data that it has not seen before.

```
#--------------------------------------------------------------------------------
print("Iteration 3 / 3.4 Integrating Data")
# dataset split to train and test
from sklearn import model_selection
y = featured_data['stroke']
X = featured_data.drop(columns=['stroke'])
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, y, test_size=0.3)
X_train = X_train.reset_index()
X_test = X_test.reset_index()
Y_train = Y_train.reset_index()
Y_test = Y_test.reset_index()
del X_train['index']
del X_test['index']
del Y_train['index']
del Y_test['index']
```

Figure 29. Train, test split to 7:3

After splitting the dataset into train dataset and test dataset. We use the algorithm of ADASYN to solve the imbalance label problem, and generate more data for stroke, which makes the stroke patients and non-stroke patients' ratio around 1: 1. ADASYN (Adaptive Synthesis) is an algorithm that generates synthetic data with the great advantage of not replicating the same minority data and generating more data for "harder to learn" examples.
The ADASYN algorithm works by calculating the ratio d = ms /ml of the minority (patients who had a stroke) to the majority (patients who did not have a stroke), and if d is below a certain threshold, the algorithm will calculate the total number of synthetic minority data to be generated. According to the formula ri = #majority / k, the k-Nearest Neighbours of each minority example is found and the ri value is calculated. after completing this step, each minority example should be associated with a different community. the ri value indicates the dominance of the majority class in each particular neighborhood. Higher ri communities contain more majority class examples and are more difficult to learn. Instead, we set up to find 6 neighbors in the neighborhood and then normalize the ri values so that the sum of all ri values equals 1 (the number of strokes and non-strokes is a total of 100%, and each is roughly 50%). Since the ri is higher in majority-class dominated neighborhoods, more synthetic minority group examples will be generated for these neighborhoods. Thus, this gives the ADASYN algorithm adaptability; more data is generated for the "harder to learn" communities.

```
from collections import Counter
import imblearn
from sklearn.neighbors import KNeighborsClassifier
from sklearn import neighbors

ct = Counter(Y_train['stroke'])
ada = imblearn.over_sampling.ADASYN(random_state=0, n_neighbors=6)
X_train_reimb, Y_train_reimb = ada.fit_resample(X_train, Y_train)
```

Figure 30. Apply ADASYN algorithm rebalances stroke attribute

Through this way, we have generated new data of stroke patients to balance the uneven of stroke and non-stroke distribution and integrate new data records into the dataset. We can see the new stroke distribution in the dataset is balanced as shown in Figure 31, and Figure 32. There are 19,324 records are marked as stroke, and there are 19,406 records are marked as non-stroke.

```
# Draw stroke distribution
raw_data_stroke = Y_train_reimb[Y_train_reimb['stroke'] == 1]
raw_data_healthy = Y_train_reimb[Y_train_reimb['stroke'] == 0]
num_stroke = len(raw_data_stroke)
num_healthy = len(raw_data_healthy)

print("There are", num_stroke, "records are marked stroke.")
print("There are", num_healthy, "records are marked as healthy (non-stroke).")
# plt.bar(['stroke','healthy'],[num_stroke,num_healthy])
# plt.show()
# plt.bar(['stroke','healthy'],[num_stroke,num_healthy])
# plt.show()
seaborn.barplot(y=['stroke', 'healthy'], x=[num_stroke, num_healthy], orient='horizon')
```

```
There are 19324 records are marked stroke.
There are 19406 records are marked as healthy (non-stroke).
```

Figure 31. After rebalances stroke attribute

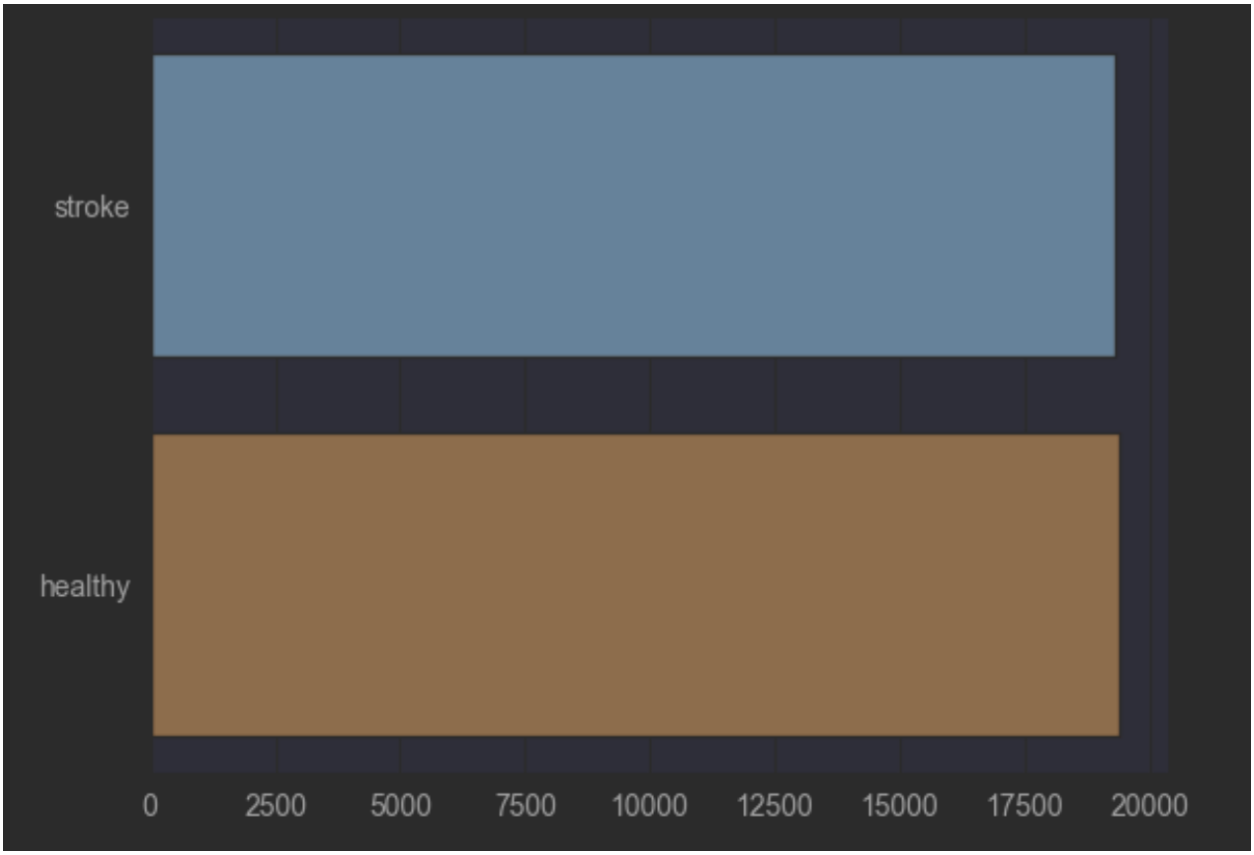


Figure 32. Stroke distribution after rebalances

3.5 Formatting Data

Data sets have numeric and categorical features. Categorical features are string data types, which can be easily understood by humans. However, machines cannot interpret categorical data directly. Here, unlike the professional process application of SPSS modeler, categorical data needs to be converted to numerical data because many models/algorithms require that categories are represented numerically and can only be calculated and executed on a numerical basis. Based on this requirement, data formatting is applied to this dataset for further data mining. It is used the method of Label Encoding in python to convert string type to number.
In this dataset, there are three category attributes that need to be converted from strings to numbers, i.e., gender (male, female, other), job type (private job, never worked, government job), and BMI (underweight, normal, overweight, obese) need to be converted.
The issue to note here is that when transforming categorical features to numerical features, numbers are size differentiated, while gender and job type should not be size differentiated or unequal. So we use the unique method to tag each value, where the number assigned to each value has no size or inequality, but just a tag for each value in the attribute.
Table 4shows the reference table for these categories. Figure 33 shows the results of the implementation of the labeling process.

Gender			
Male	Female	Other	

1	0	2	
Work Type			
Self-employed	Never Worked	Government Job	
1	2	0	
BMI			
Underweight	Normal	Overweight	Obese
3	2	0	1

Table 4. The data formatting referencing table

```
#-------------------------------------------------------------------------
print("Iteration 3 / 3.5 Formatting Data")
columns = ['gender','new_bmi','new_work_type']
# number_feature = featured_data.copy()
for column in columns:
    classes = X_train[column].unique()
    print(classes)
    for i in range(len(classes)):
        print(classes[i],i)
        X_train[column] = X_train[column].replace(classes[i],i)
        X_test[column] = X_test[column].replace(classes[i],i)

```

```
Iteration 3 / 3.5 Formatting Data
['Female' 'Male' 'Other']
Female 0
Male 1
Other 2
['Overweight' 'Obese' 'Normal' 'Underweight']
Overweight 0
Obese 1
Normal 2
Underweight 3
['Govt_job' 'Self-employed' 'Never_worked']
Govt_job 0
Self-employed 1
Never_worked 2
```

Figure 33. Python implementation of labeling process

4. Data Transformation

4.1 Data Reduction

There are 14 properties in this dataset, including the three new ones created in 3.3. In this session, we will remove some attributes that are not used, such as BMI before update, smoking status before update, and job type before update; and id and target attribute stroke, which are not needed when training the model, because the association between id and stroke is 0, while the target attribute is the result.

After delete these 5 attributes, we get the dataset as below Figure 34. There are 9 attributes left, and 19,781 records.

```
       gender   age  hypertension  heart_disease  ever_married  \
0           0  69.0             0              0             1
1           1  52.0             0              0             1
2           0  61.0             0              0             1
3           0  42.0             0              0             1
4           1  37.0             0              0             1
...       ...   ...           ...            ...           ...
19776       0  77.0             0              0             1
19777       0  78.0             0              0             1
19778       0  32.0             0              0             1
19779       0  80.0             0              0             0
19780       0  74.0             0              0             1

       avg_glucose_level  new_smoking_status  new_bmi  new_work_type
0                  74.34                   0        0              0
1                  93.60                   1        0              0
2                  69.88                   0        0              1
3                 123.28                   0        0              0
4                  66.17                   0        0              1
...                  ...                 ...      ...            ...
19776              93.04                   0        0              1
19777              91.87                   0        2              1
19778             107.51                   0        1              1
19779             230.74                   1        1              1
19780              86.25                   0        0              1

[19781 rows x 9 columns]
```

Figure 34. dataset after deleting unneeded attributes

Next, among these nine attributes, there are also attributes that need to be evaluated for some of the smaller contributions to the final results. Most of the time, these attributes can mislead the algorithm and ultimately reduce the overall accuracy of the model. Therefore, some amount of data reduction/dimensionality reduction is also required to eliminate the number of these attributes, using a method called PCA. principal component analysis (PCA) is a popular technique for analysing large datasets containing a large number of dimensions/features per observation, improving the interpretability of the data while retaining the maximum amount of information, and visualizing multidimensional data. Formally, PCA is a statistical technique that reduces the dimensionality of a data set. This is achieved by linearly transforming the data into a new coordinate system in which (most) changes in the data can be described in fewer dimensions than the initial data.

PCA is used for exploratory data analysis and building predictive models. It is typically used for dimensionality reduction, projecting each data point onto only the first few principal components to obtain low-dimensional data while preserving as much variation in the data as possible. The first principal component can be equivalently defined as the direction that maximizes the variance of the projected data. The i-th principal component the i-th principal component can be used as the direction orthogonal to the i-1st principal component that maximizes the variance of the projected data.

For either objective, it can be shown that the principal components are the eigenvectors of the data covariance matrix. Therefore, principal components are usually calculated by eigendecomposition of the data covariance matrix or singular value decomposition of the data matrix. PCA is the simplest of the true eigenvector-based multivariate analyses and is closely related to factor analysis. Factor analysis typically incorporates more domain-specific assumptions about the underlying structure and resolves the eigenvectors of a slightly different matrix.

We do the data reduction using PCA as shown in Figure 35 and Figure 36. We project the 9 latitudes to a 2-dimensional data graph.

```python
#-------------------------------------------------------------------------------
print("Iteration 3 / 4.1 Data Reduction")
# dimension reduction, 新的特征space之间相互正交，保持最大方差的特征
# dimension reduction, new features space are orthogonal to each other,
# keeping the maximum variance of the features
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X_train)
X_train_pca = pca.transform(X_train)
```

```
Iteration 3 / 4.1 Data Reduction
```

```python
point_number = 500
Y_train_pos = Y_train[Y_train.stroke==1].index
Y_train_neg = Y_train[Y_train.stroke==0].index
plt.scatter(X_train_pca[Y_train_pos][:point_number//5, 0],
 X_train_pca[Y_train_pos][:point_number//5, 1], alpha=0.5, s=5, c='red')
plt.scatter(X_train_pca[Y_train_neg][:point_number, 0],
 X_train_pca[Y_train_neg][:point_number, 1], alpha=0.5, s=5, c='green', marker='x')
plt.show()
```

Figure 35. Use PCA to do data reduction by reducing latitudes

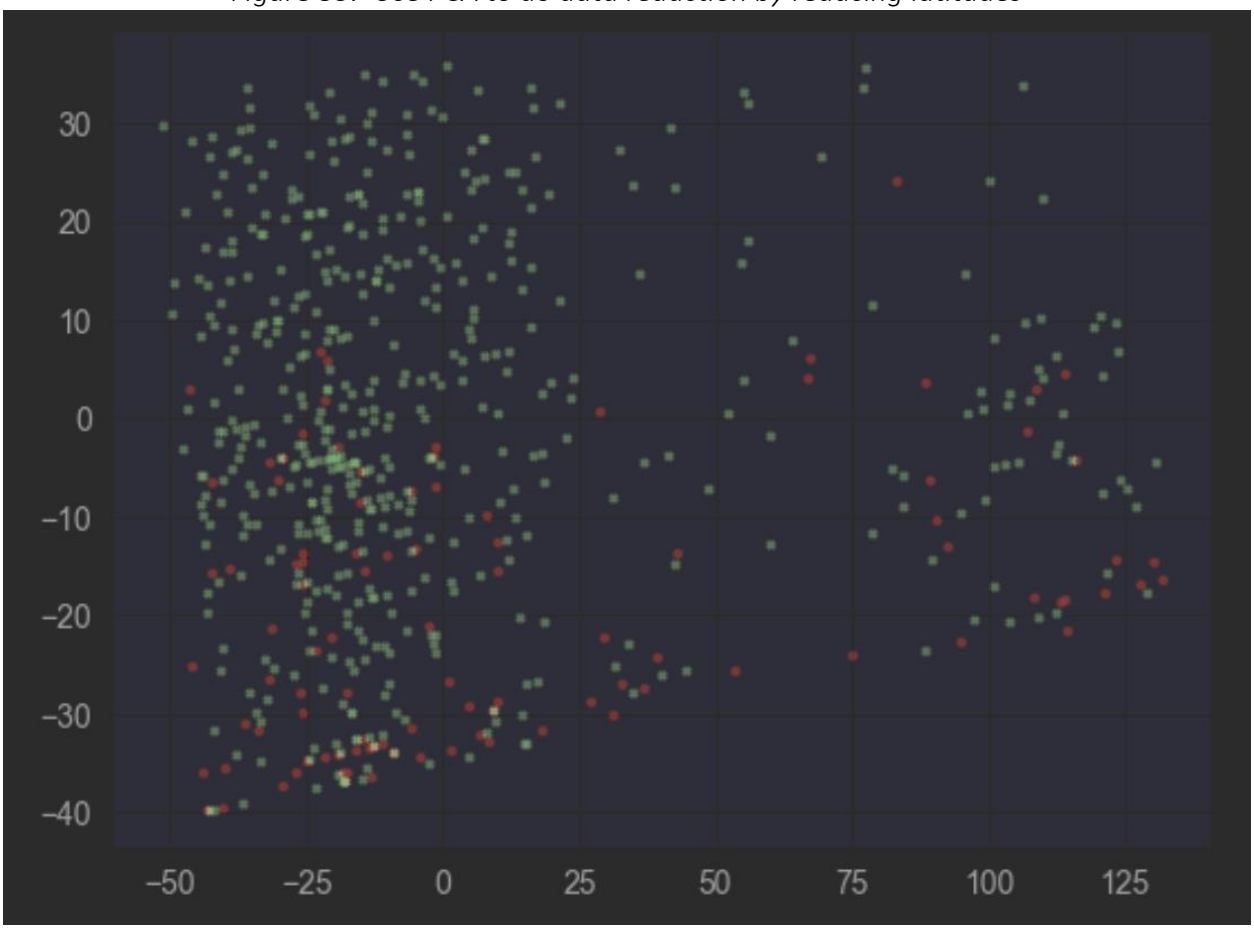


Figure 36. Result of using PCA in 2D-demision

4.2 Data Projection

As mentioned above, the data project after using PCA to do data reduction is as below in Figure 37. PCA evaluated the importance of each attribute for the target attribute and reduced the dataset from 9 latitudes to 3 latitudes.

```
#-------------------------------------------------------------------------------
print("Iteration 3 / 4.2 Data Projection")
pca = PCA(n_components=3)
pca.fit(X_train_reimb)
print(pca.explained_variance_ratio_)

X_train_reimb_pca = pca.transform(X_train_reimb)
X_test_pca = pca.transform(X_test)

```

```
Iteration 3 / 4.2 Data Projection
[8.75971132e-01 1.23429855e-01 2.15592856e-04]
```

```
print(X_train_reimb_pca)
```

```
[[-2.43148238e+01  3.27088888e+01 -1.33466973e+00]
 [-1.53069312e+01  3.25981034e+01 -2.74440113e-01]
 [-1.50370797e+01  4.06659785e+01 -1.39645595e+00]
 ...
 [-2.49077864e+01 -1.08452244e+01  7.67292642e-02]
 [-2.63936907e+01 -1.08054948e+01 -2.22815399e-02]
 [-2.57162750e+01 -1.06354319e+01  1.03165963e+00]]
```

Figure 37. Data visualisation of using PCA do data reduction

As described in previous chapters, the selected train dataset contains an unbalanced number of labels between the stroked and non-stroked instances. Figure 38 is a bar chart showing the difference between the two. Assume that these datasets are directly applied to the algorithm. In this case, the algorithm would obtain a low recall, i.e., the algorithm is good at detecting those non-stroke values, but very insensitive to those stroke data, since the stroke sample size is much too small to allow the algorithm to study these patterns.

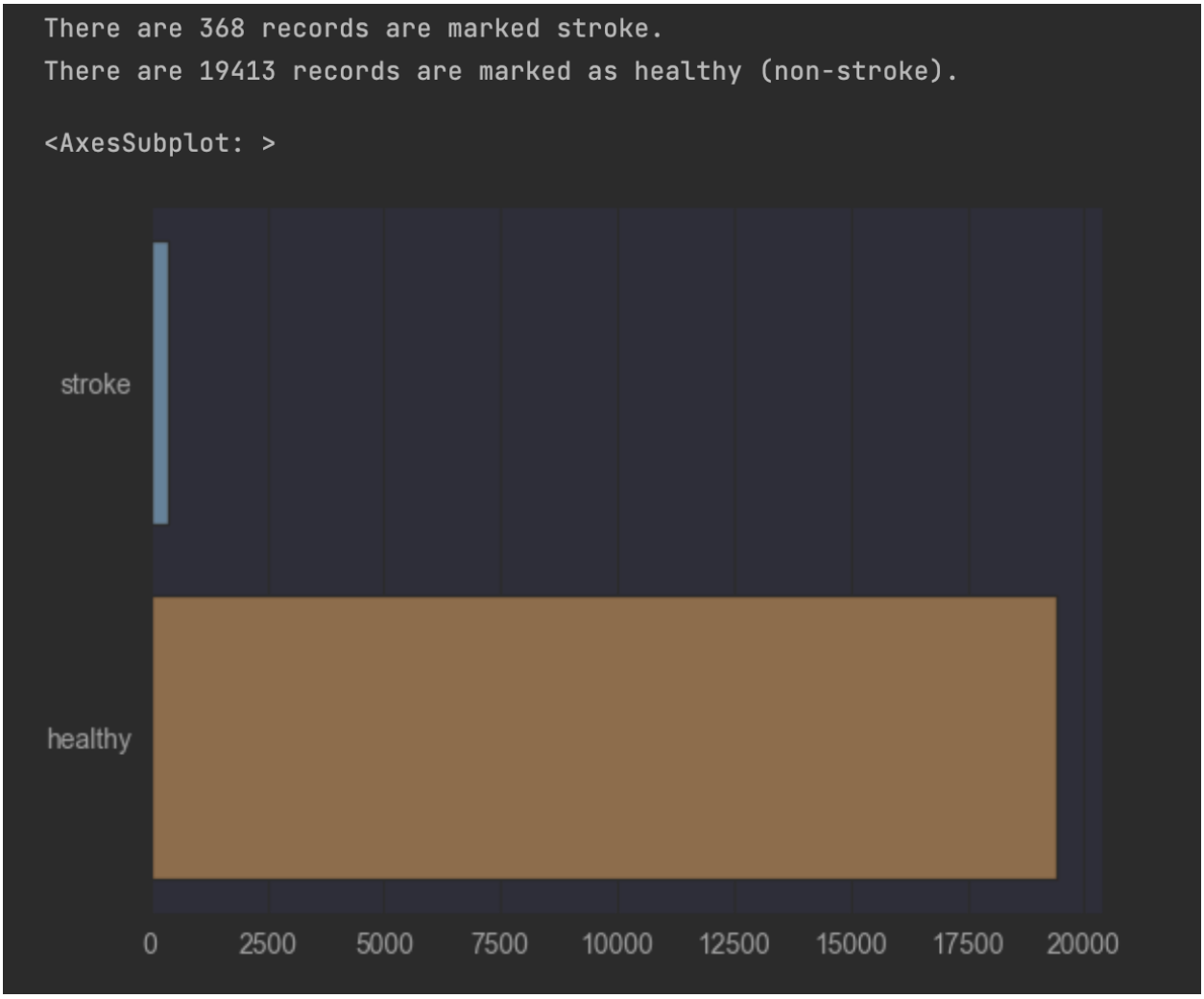


Figure 38. Stroke distribution in training dataset before re-balance

Based on this concern and considering trying to maintain the characteristics of the original data in the test set, we performed the rebalancing operation only on the training data. The dataset was further processed to fill in the gaps between the stroke and non-stroke data. In total, 348 cases were classified as stroke instances and 19,433 cases were classified as non-stroke in the training set. On this basis, the oversampling method is more appropriate for this case compared to under sampling. It will increase the number of stroke instances without decreasing the number of non-stroke instances. Using this method, the algorithm itself will have a large enough record to find and refine the pattern.

As mentioned before, after rebalancing the stroke attributes inside the training set using the ADASYN algorithm, Figure 39 shows the histogram, which shows the number of stroke instances, which are balanced at this time. There are 19,307 stroke cases in training dataset, and there are 19,433 non-stroke cases in training dataset.

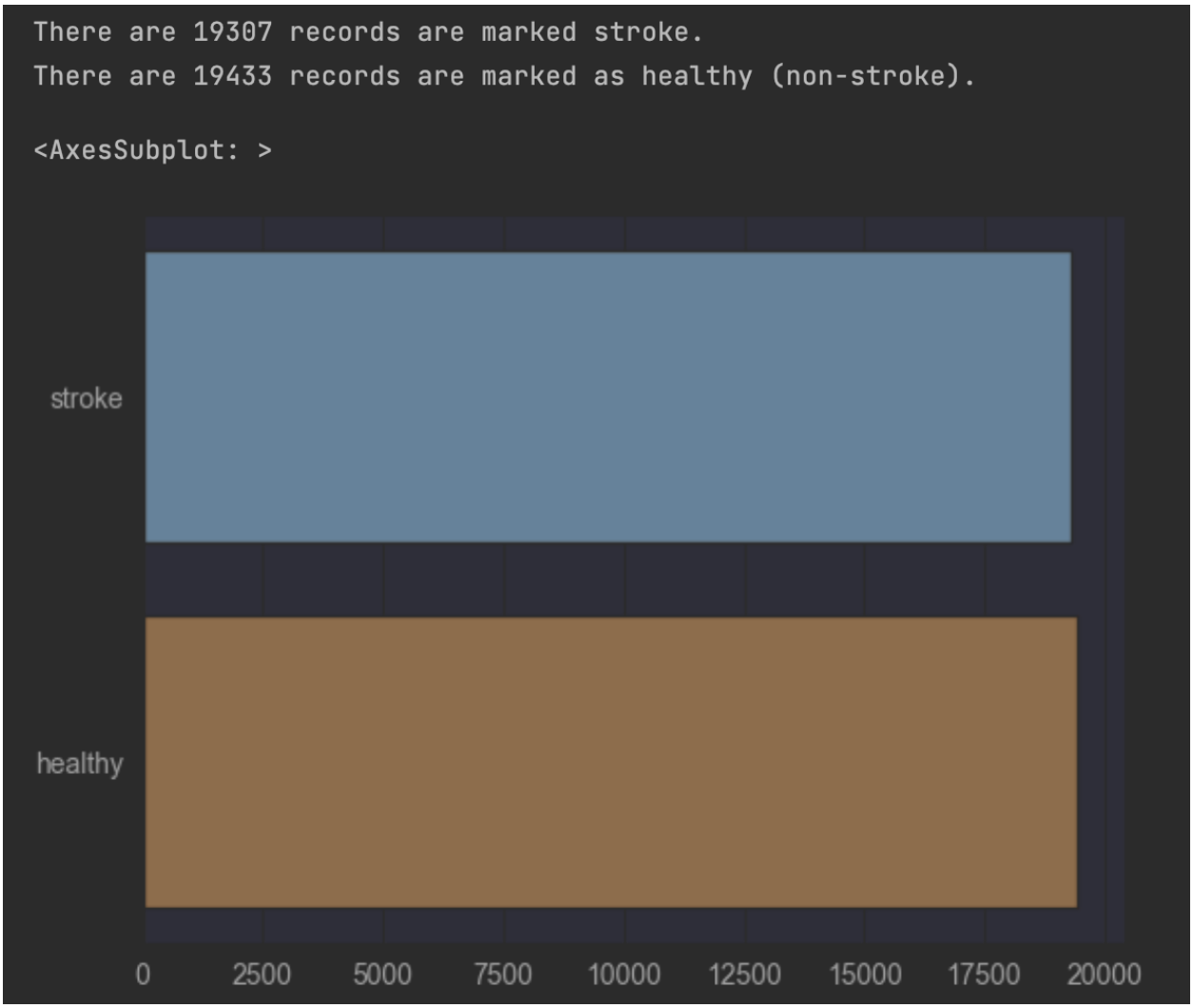


Figure 39. Stroke distribution in training dataset after re-balance

5. Data-Mining Method(s) Selection

5.1. Discussion of Data Mining Methods in Context of Data Mining Objectives

As mentioned earlier in the business objectives, we wanted to predict, prevent and reduce the occurrence of stroke disease based on the current status of the patient. In conjunction with the initial data mining goals, we want to clarify the relationship between a patient's current health index (current disease and BMI), age, and current life status (smoking status, marital status, type of work, living environment, etc.) and stroke. And by analyzing these relationships, the data is used to train and fit models suitable for data prediction, by providing these predictions to analyze the causes of current patient strokes and to reduce the chances of potential patient strokes at an early stage.

Before building the module, it is important to review our original intention of doing this study and choose to narrow or keep this goal for the module construction. The prediction system is based on a model that follows a number of patterns that are between the target and all relevant information relationships.

The assumptions of the modeling

In the preparation phase, nine attributes were selected: gender, age, average blood glucose level, hypertension, heart disease, whether married or not, whether smoking or not, type of work and weight type (BMI group), excluding the record identification number and the target attribute. We assume that these attributes are highly correlated with the target attributes, and the attributes that were removed and reconstructed for various reasons, the original smoking status of the dataset, job type and BMI, gender, etc., which will not affect the accuracy of the model's prediction of the target attributes. We keep as many attributes as possible and use pca's method to descend the latitude because having more information and keeping more attributes can provide more information for the model to train and understand the case more fully and build the model to predict more accurately.

Making predictions

The main function of this prediction system is to find the cause of stroke or to predict the level of stroke risk for an individual. In other words, it needs to collect information from individuals and use existing training models to assess the level of risk based on these collected data for consumption. The nine attributes to be predicted are to be collected as candidate factors or potential influences for stroke. The total number of instances prepared for the data mining process was 47218 as shown in Figure 40.

And before we build a model for the prediction system, we need to create a test design for the validation of the prediction model. When creating the test design, the entire data will be used to divide the training set and the test set in the 7:3 ratio mentioned in section 3.4 and use the shuffle parameter inside python to make each test and training set division random as a way to better train the model.

```
print("The total number of instances prepared for the data mining process was",
      len(X_train_reimb)+len(X_test),
      ". \n There are",len(X_train_reimb),"training instances in training dataset, \n and "
      "There are", len(X_test), "test instances in test dataset.")
```

```
The total number of instances prepared for the data mining process was 47218 .
 There are 38740 training instances in training dataset,
 and There are 8478 test instances in test dataset.
```

Figure 40. Number of instances

Choosing the Right Modeling Techniques:

1. Association

   It is used to find correlations between two or more items by identifying hidden patterns in a dataset, hence the term relationship analysis. This method is used in shopping cart analysis to predict customer behavior. For example, the classic case, purchase (x, "beer") -> purchase (x, "diaper") [support = 1%, confidence = 50%], where x represents a customer who buys beer and diapers together. Confidence indicates that if a customer buys beer, there is a 50% chance that he/she will also buy diapers as well. Support implies that 1% of all transactions analyzed indicate that beer and diapers are purchased together.

There are two types of association rules.
Unidimensional association rules: These rules contain repeated individual attributes.
Multidimensional association rules: These rules contain multiple attributes that are repeated.

2. Clustering
   A cluster is a collection of data objects; these objects are similar in the same cluster. This means that objects in the same group are similar to each other, and they are quite different, or they are different or unrelated to objects in other groups or other clusters. Cluster analysis is the process of discovering groups and clusters in the data, making the degree of association between two objects the highest if they belong to the same group and the lowest between them otherwise. The results of this analysis can be used to create customer analysis.

3. Classification
   This data mining method is used to distinguish items in a dataset into classes or groups. It helps to accurately predict the behavior of entities within the group. It is a two-step process.
   Learning step (training phase): Here, the classification algorithm constructs the classifier by analyzing the training set.
   Classification step: Test data is used to estimate the accuracy or precision of the classification rules.

Conclusions are drawn based on the relationship between each field value and the final target. For example, a medical researcher analyzes cancer data to predict which drug to prescribe to a patient, groups patients according to their lives, and finds relationships or patterns between attributes.

5.2. Selecting the appropriate Data-Mining method(s)

According to the objectives of data mining mentioned in section 1.4, the three data mining methods mentioned in section 5.1, clustering, association and classification methods. Considering the suitability of this project, the classification method was selected as the data mining method.

Both classification methods are more suitable for this project than the other two methods, clustering and association. In terms of prediction of the target attributes, the association method analyses the relationship between two or more attributes and draws conclusions about association rules to predict the probability of one attribute occurring when another occurs. When there are multiple attributes, association methods are less sensitive to the data than classification methods. Classification methods, on the other hand, combine all the important attributes to make predictions about the future target data.

In terms of data grouping, association methods only absorb continuous values and make predictions. Classification methods also provide the ability to use non-continuous values as predictors. And a larger range of predictors can produce more accurate answers to this.

However, cluster analysis, the process of discovering clusters and clustering in the data, is not compatible with our project objectives and is therefore not chosen.
In conclusion, the method of classification is the most suitable one for this case.

6. Data-Mining Algorithm(s) Selection

6.1 Algorithms analysis

According to the algorithm of Classification discussed in section 5.2, includes supervised learning and unsupervised learning algorithms. The difference between these two types is the target label. The information from supervised learning will be given a result label and it is mainly used in terms of pattern search. Unsupervised learning will not be given any outcome labels and it focuses on the relationships in the dataset. It is mainly used in clustering methods (Mohamed, Yap, & Berry, 2019). In this data mining process, algorithms categorized as supervised learning will be used depending on the data mining objectives. Based on the limitations mentioned above, three data mining algorithms are being selected and discussed based on the data mining objectives.
Based on the above limitations, three data mining algorithms are being selected and discussed in accordance with the data mining objectives. The algorithms/models Random Tree, Random Forest, KNN, SVM will be discussed.

1. **Random Tree**
   The random tree is a supervised classifier; it is an integrated learning algorithm that generates a large number of individual learners. It uses the idea of packing to construct a random set of data to build a decision tree. In a standard tree, each node uses the best split of all variables. In a random forest, each node is split using the best node in a randomly selected subset of predicates for that node.
   Leo Breiman and Adele Cutler (2001) introduced the random tree. This algorithm can handle classification and regression problems. The classification mechanism is as follows: the random tree classifier takes the input feature vector, classifies it with each tree in the forest, and outputs the class labels that receive the majority of "votes". In the case of regression, the classifier response is the average of the responses of all trees in the forest. Random trees are essentially a combination of two existing algorithms in machine learning: a single model tree merged with the idea of a random forest. Model trees are decision trees in which each leaf has a linear model that is optimized for the local subspace explained by that leaf.

2. **Random Forest**
   A random tree is a set (set) of tree predictors called a random forest. Random forests or stochastic decision forests are an integrated learning method for classification, regression and other tasks that operates by constructing multiple decision trees at training time. It also limits its drawbacks. Most importantly, different combinations of data are used to eliminate the uncertainty of a single decision tree (the order in which the data arrive makes the decision trees different). For classification tasks, the output of a random forest is the class of most tree choices. For regression tasks, the mean or

average prediction of individual trees is returned. Random decision forests correct the habit of overfitting decision trees to their training set. Random forests usually outperform decision trees. However, data characteristics may affect its performance. Random forests have been shown to greatly improve the performance of individual decision trees: tree diversity is created by two types of randomizations. First, the training data is sampled, and each tree is replaced. Second, instead of always computing the best split for each node when planting a tree, a random subset of all attributes is considered at each node and the best split for that subset is computed. These trees combine for the first-time model trees and random forests for classifying stochastic model trees. Random trees use this product for split selection to induce reasonably balanced trees where one global setting of ridge values applies to all leaves, thus simplifying the optimization process.

3. **KNN**

   The k-nearest neighbor algorithm, also known as KNN or k-NN, is a nonparametric supervised learning classifier that uses proximity to classify or predict groupings of individual data points. It is a model that follows the basic rules of clustering, where instances (cases) with similar attributes approach each other, or are grouped together, and using this feature, unseen variables are projected into this space to produce predictions based on the state of the nearest k-number (the k-number is set manually, usually odd) of neighbors. Information about patients in the dataset is actually shared, and the KNN solution is able to detect these groups and generate labels based on this information.

   For classification problems, the labels for a category are assigned on a majority vote basis, meaning that the most frequently occurring labels around a given data point are used. While this is technically considered "majority voting", the term "majority voting" is more often used in the literature. While it can be used for both regression and classification problems, it is often used as a classification algorithm whose working assumption is that points that are close to each other can be found.

4. **SVM**

   SVC, or support vector classifier, is a supervised machine learning algorithm typically used for classification tasks. SVC works by mapping data points into a high-dimensional space and then finding the best hyperplane to classify the data into two classes. SVM, or support vector machine, is a powerful model that looks to separate data with invisible lines. It ensures that the distance between two types of data is uniform and away from each other. It is also able to do nonlinear separation by projecting the data into three dimensions and separating them. This model can be used because it meets the goals of data mining and is able to find patterns.

   SVM is particularly suitable for generalized datasets, that is, datasets with a large number of prediction domains. A classification algorithm is usually used to deal with two sets of classification problems. After giving SVM models a labeled training data set for each category, they are able to classify new text. They have two main advantages over

newer algorithms such as neural networks: higher speed and better performance with a limited number of samples (in the thousands). This makes the algorithms very suitable for text classification problems, where datasets with up to a few thousand labeled samples are usually available.

All of those mentioned algorithms will be experienced in python using SK-learn, and the performance and accuracy of each algorithm will be evaluated in the following chapters.

6.2 Algorithms analysis

1. Random Tree

   According to DM objectives, the decision tree divides the importance level of attributes based on the attributes of this dataset, with the most important attribute as the first branch node condition, and then branches down in layers to partition the best nodes in the subset. Each leaf is a case. It is possible to clearly show what the most influential predictors of stroke are for the target attribute, stroke.

   I set different preconditions to find the parameters that make the random tree present the best performance.

   Figure 41 shows the overall accuracy, confusion matrix, recall score, and time consumption used for random tree after rebalancing the target attribute stroke.

```
from sklearn import tree
from sklearn.metrics import confusion_matrix
clf = tree.DecisionTreeClassifier()
clf.fit(X_train_reimb, Y_train_reimb)
# entropy = - p log p

Y_pred = clf.predict(X_test)

# data after re-balance
print("Data after rebalance")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

```
Random Tree
Data after rebalance
Accuracy score 0.9386647794291106
[[7940  383]
 [ 137   18]] Confusion Matrix
Recall score 0.11612903225806452
ROC 0.5350559855511157
time cost 0.09638714790344238 s
```

Figure 41. Random tree process and result after rebalancing

Figure 42 shows the overall accuracy, confusion matrix, recall score, and time consumption used for random tree after rebalancing the target attribute stroke and running PCA method.

```
from sklearn import tree
from sklearn.metrics import confusion_matrix
clf = tree.DecisionTreeClassifier()
clf.fit(X_train_reimb_pca, Y_train_reimb)
# entropy = - p log p

Y_pred = clf.predict(X_test_pca)

# data after re-balance and pca into dimension=3
print("Data after rebalance and pca into dimension=3")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

```
Random Tree
Data after rebalance and pca into dimension=3
Accuracy score 0.8941967445152158
[[7543  780]
 [ 117   38]] Confusion Matrix
Recall score 0.24516129032258063
ROC 0.5757225411122695
time cost 0.12729597091674805 s
```

Figure 42. Random tree process and result after rebalancing and PCA

We can see that the accuracy is very high around 90%, but the recall score is not that good, which means the accuracy (positive predictive value) of classifying the data instances was not very good, meaning that the model was not sensitive enough to predict the sample of stroke, the probability of predicting the sample of stroke as the correct stroke was only 11.6% after data balancing, while performing PCA pairs with recall had an increase to 24.5%.

2. Random Forest

A decision forest is a collection of multiple decision trees and tends to perform better than a decision tree.
Figure 43 shows the overall accuracy, confusion matrix, recall score, and time consumption used for random forest after rebalancing the target attribute stroke and running PCA.

```
time_start=time.time()
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
clf = RandomForestClassifier()
clf.fit(X_train_reimb_pca, Y_train_reimb)
Y_pred = clf.predict(X_test_pca)

# data after re-balance and pca into dimension=3
print("Data after rebalance and pca into dimension=3")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

```
 Random Forest

/var/folders/xl/y162rqwx3hl6tfsd_fc2z51c0000gn/T/ipykernel_46345/409188

 Data after rebalance and pca into dimension=3
 Accuracy score 0.9033970276008493
 [[7621  702]
  [ 117   38]] Confusion Matrix
 Recall score 0.24516129032258063
 ROC 0.5804083515171716
 time cost 2.746000051498413 s
```

Figure 43. Random forest process and result after rebalancing and PCA

We can see that the accuracy is 90.34%, the recall score is 24.5%. Although the random forest improves the performance compare with random tree, but the improvement is small, the model needs to be improved more. The recall score is not that good, which means the accuracy

(positive predictive value) of classifying the data instances was not very good, meaning that the model was not sensitive enough to predict the sample of stroke, the probability of predicting the sample of stroke as the correct stroke was 24.5%.

3. KNN

   According to DM objectives, it is a model that follows the basic rules of clustering, where instances (cases) with similar attributes are close to each other, or are grouped together. Using this feature, unseen variables are projected into this space, generating predictions based on the state of the nearest k-number (the k-number is set manually and is usually odd) neighbours. Information about patients in the dataset is actually shared, and the KNN solution is able to detect groups of these stroke patients and generate labels based on this information.

   Figure 44 shows the overall accuracy, confusion matrix, recall score, and time consumption used for KNN.

```python
# KNN
print("KNN")
import time
time_start=time.time()
from sklearn import neighbors
from sklearn.metrics import confusion_matrix
clf = neighbors.KNeighborsClassifier(n_neighbors=1, weights='distance')
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)

print("data after re-balance and pca into dimension=3")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

```
KNN
data after re-balance and pca into dimension=3
Accuracy score 0.9660297239915074
[[8187  136]
 [ 152    3]] Confusion Matrix
Recall score 0.01935483870967742
ROC 0.5015072883924453
time cost 0.044014692306518555 s
```

Figure 44. KNN process and result

We can see that the accuracy is 96.6%, the recall score is 1.9%. From the performance of KNN model, we can see that it is not a good model for this case, it seems like the model did not learn from dataset, the prediction is not precise. The recall score is not that good, which means the accuracy (positive predictive value) of classifying the data instances was not good, meaning that the model was not sensitive enough to predict the sample of stroke, the probability of predicting the sample of stroke as the correct stroke was only 1.9%.

The reason why KNN is a very bad model is the data itself, where all the samples with and without stroke are too close together, as shown in Figure 36, in PCA result.

The reason why decision trees, decision forests and KNN models do not perform well is that these models are based on spatial partitioning, with decision trees and decision forests dividing

the population based on area in space, and KNN dividing the population based on distance in space.
More models need to be discovered.

4. SVM
   According to DM objectives, SVM is a powerful model that wants to separate data with invisible lines. It ensures that the distance It ensures that the distance between the two types of data is uniform and away from each other. It is also able to do nonlinear separation by projecting the data into 3D space and separating them. It is a good choice for the database of this project, which is difficult to classify from a spatial perspective.
   In this model, I did not use the rebalanced training dataset, which is done by myself, instead, I used the parameter to do class weight balance in the SVM algorithm.
   Figure 40 shows the overall accuracy, confusion matrix, recall score, and time consumption used for SVM with balanced weight.

```
import time
time_start=time.time()
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
clf = SVC(class_weight='balanced')
clf.fit(X_train, Y_train)

Y_pred = clf.predict(X_test)

# original data with balance weight
print("original data with balance weight")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

```
SVM - SVC

/Users/anbyzhou/Desktop/Infosys 722/venv/lib/python3.9/site-packages/

original data with balance weight
Accuracy score 0.7204529370134466
Recall score 0.7806451612903226
ROC 0.7499885664675812
time cost 6.72917628288269 s
```

Figure 45. SVM with balance weight process and result

We can see that the accuracy is 72%, the recall score is 78%. From the performance of SVM model, we can see that although the accuracy is not very high, but the recall score is much improved and better than other models we discussed before, which means the accuracy (positive predictive value) of classifying the data instances was better, meaning that the model was more sensitive to predict the sample of stroke, the probability of predicting the sample of stroke as the correct stroke was 78%.

Compare with other models, we discussed before, SVM model is learning from the dataset, and the learning result/output is much better. We can improve the prediction accuracy score by adjusting the parameters of this algorithm in the following section.

6.3 Parameter Tuning

Most machine learning and deep learning algorithms have some parameters that can be tuned, called hyperparameters. Before training the model, we need to set the hyperparameters. Hyperparameters are crucial for building robust and accurate models. By helping us find a balance between bias and variance, they prevent the model from being overfitted or underfitted. In order to be able to tune hyperparameters, we need to understand what they mean and how they change the model.

We will choose SVM Algorithm based on our data mining objective, as it has the most satisfiable performance (accuracy score: 72%, recall score: 78%) compare with random tree (accuracy score: 89.4%, recall score: 24.5%), random forest (accuracy score: 90.3%, recall score: 24.5%) and KNN (accuracy score: 96.6%, recall score: 1.9%).
SVM was chosen as the algorithm for the following data mining process. Although the current accuracy is not high, there is still room for further improvement and parameter tuning. The algorithm has several parameters to control the performance of the algorithm.

In this section, the C, kernel, degree and gamma will be evaluated and tested to understand the impact of each parameter. The reasons for choosing these four parameters are discussed below.

C: It is the regularization parameter, C, of the error term.
kernel: It specifies the kernel type to be used in the algorithm. It can be 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed', or a callable. The default value is 'rbf'. The main function of the kernel is to transform the low-dimensional input space into a higher-dimensional space. It is mainly useful in nonlinear separation problems.
degree: It is the degree of the polynomial kernel function ('poly') and is ignored by all other kernels. The default value is 3.
gamma: It is the kernel coefficient for 'rbf', 'poly', and 'sigmoid'. If gamma is 'auto', then 1/n_features will be used instead. Gamma defines how far influences the calculation of plausible line of separation. Large gamma means only near points are considered; low gamma means far away points are also considered.
When the gamma is high, nearby points will have a higher impact; low gamma means that distant points will also be considered to obtain decision boundaries. The values of gamma would be normally considered are 1, 0.1, 0.01, 0.001. And the default value is 'auto' 1/n features.

C parameter optimization:
From Figure 41, we could see the performance and result using SVM model to run the original data without rebalanced process and PCA. The accuracy is very high, 98.17%, however, the

recall score is 0%. It seems like the model did not learn from dataset; the prediction is not precise at all. The recall score is bad, which means the accuracy (positive predictive value) of classifying the data instances is bad, meaning that the model was not sensitive enough to predict the sample of stroke, the probability of predicting the sample of stroke as the correct stroke was only 0%.
To optimize this situation, there are 2 ways, either use the rebalance method that we created, or use the class weight balance method parameter in SVM algorithm.

```
time_start=time.time()
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
clf = SVC()
clf.fit(X_train, Y_train)

Y_pred = clf.predict(X_test)

# Original data
print("Original data")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

```
SVM - SVC

/Users/anbyzhou/Desktop/Infosys 722/venv/lib/python3.9/site-pa...

Original data
Accuracy score 0.9817173861759849
Recall score 0.0
ROC 0.5
time cost 0.49842000007629395 s
```

Figure 46. SVM train original data

Use the class weight balance parameter in SVM:
In SVM, C is the penalty for misclassification. The general idea is to increase the penalty for misclassification to prevent them from being 'swamped' by the majority of classes.
In scikit-learn, when using SVC, we can automatically set the value of C by setting class_weight='balanced' and the balanced parameter will automatically weigh the classes.

```
time_start=time.time()
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
clf = SVC(class_weight='balanced')
clf.fit(X_train, Y_train)

Y_pred = clf.predict(X_test)

# original data with balance weight
print("original data with balance weight")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

```
SVM - SVC

/Users/anbyzhou/Desktop/Infosys 722/venv/lib/python3.9/site-pa...

original data with balance weight
Accuracy score 0.6988676574663836
[[5812 2517]
 [  36  113]] Confusion Matrix
Recall score 0.7583892617449665
ROC 0.7280960596154294
time cost 6.46691370010376 s
```

Figure 47. SVM with balance weight process (C)

We can see that the accuracy is 69.9%, the recall score is 75.8%. From the performance of SVM model, we can see that although the accuracy is not very high, but the recall score is much improved and better than other models we discussed before, which means the accuracy (positive predictive value) of classifying the data instances was better, meaning that the model was more sensitive to predict the sample of stroke, the probability of predicting the sample of stroke as the correct stroke was 75.8%.

Use the rebalance method that we created and default C= 1:

```
time_start=time.time()                                          ⚠ 30  ✓ 15
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
clf = SVC(class_weight='balanced')
clf.fit(X_train_reimb, Y_train_reimb)

Y_pred = clf.predict(X_test)

# data after re-balance
print("data after re-balance")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

```
 SVM - SVC

/Users/anbyzhou/Desktop/Infosys 722/venv/lib/python3.9/site-pa...

 data after re-balance
 Accuracy score 0.6983958480773768
 [[5807 2522]
  [  35  114]] Confusion Matrix
 Recall score 0.7651006711409396
 ROC 0.7311516082322539
 time cost 22.876780033111572 s
```

Figure 48. SVM with rebalance method & default C= 1 process

We can see that the performance of using the rebalance method that we created, and default C value is better than just using the class weight balance parameter in SVM, we will justify C parameter in this way by using the rebalance method we created and default C value 1.

Then, we add PCA function that we did before.
Use the rebalance method that we created and default C= 1 and PCA:

```
time_start=time.time()
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
clf = SVC()
clf.fit(X_train_reimb_pca, Y_train_reimb)

Y_pred = clf.predict(X_test_pca)

# data after re-balance and pca into dimension=3
print("data after re-balance and pca into dimension=3")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

```
SVM - SVC

/Users/anbyzhou/Desktop/Infosys 722/venv/lib/python3.9/site-pa...

data after re-balance and pca into dimension=3
Accuracy score 0.7029959896201935
[[5843 2486]
 [  32  117]] Confusion Matrix
Recall score 0.785234899328859
ROC 0.7433798461105815
time cost 18.137819051742554 s
```

Figure 49. SVM with rebalance method & default C= 1 process & PCA

We can see that the performance is improved and more importantly, the time cost reduced, which saves the cost when we use the model.

Experiment and optimization of these parameters:

C	Kernel	Degree	Gamma	Accuracy	Recall	Time cost
1 (Default)	rbf (Default)	N/A	1/n features (Default)	98.17%	0	0.49 s
Balance weight	rbf (Default)	N/A	1/n features (Default)	69.89%	75.84%	6.47 s
Customized rebalance method + Balance weight	rbf (Default)	N/A	1/n features (Default)	69.84%	76.51%	22.88s
Customized rebalance method + 1 (Default) + PCA	rbf (Default)	N/A	1/n features (Default)	70.30%	78.52%	18.14 s
Customized rebalance method + 1 (Default) + PCA	linear	N/A	1/n features (Default)	71.26%	76.51%	175.05 s
Customized rebalance method + 1 (Default) + PCA	poly	3 (Default)	1/n features (Default)	81.89%	61.74%	14.24 s
Customized rebalance method + 1 (Default) + PCA	poly	9	1/n features (Default)	29.09%	95.3%	389.64 s
Customized rebalance method + 1 (Default) + PCA	poly	3 (Default)	0.001	81.91%	61.07%	23.91 s
Customized rebalance method + 1 (Default)	rbf	3 (Default)	1/n features (Default)	71.22%	82.35%	21.15 s

As we can see from above optimizations for every parameter, there are 2 algorithms/models' performance is better than others.
Which is

- SVM model with C uses Customized rebalance method + 1 (Default), kernel uses rbf, gamma uses default value 1/n features. The time cost is 21.15 s. Also, this model has the highest ROC score in these all models experimented above, which is 0.7668.

```
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
clf = SVC(kernel = 'rbf')
clf.fit(X_train_reimb, Y_train_reimb)

Y_pred = clf.predict(X_test)

# data after re-balance into dimension=3 & kernel = 'rbf'
print("data after re-balance & kernel = 'rbf'")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

```
 SVM - SVC

/Users/anbyzhou/Desktop/Infosys 722/venv/lib/python3.9/site-packages/s

 data after re-balance & kernel = 'rbf'
 Accuracy score 0.7121962727058269
 [[5912 2413]
  [  27  126]] Confusion Matrix
 Recall score 0.8235294117647058
 ROC 0.766839780957428
 time cost 21.474217176437378 s
```

Figure 50. SVM with rebalance method & default C= 1 process & rbf

7. Data Mining

7.1 Logical test designs

Before the data mining was conducted, we first spited the dataset into a training and testing dataset as shown in section 3.4. A method for assessing a machine learning algorithm's performance is the train-test split. It may be applied to issues involving classification or regression as well as any supervised learning technique. The process entails splitting the dataset into two subgroups. The training dataset is the initial subset, which is used to fit the model. The model is not trained using the second subset; rather, it is given the input element of the dataset, and its predictions are then produced and contrasted with the expected values. The test dataset is the second dataset in question. For fitting the machine learning model, use the train dataset. Test dataset used to assess how well a machine learning model fits the data. The dataset is split into two parts, the training and test datasets. This operation simulates the actual environment faced by the model. Furthermore, by using the training/testing set, it is possible to assess the sensitivity of the model in the face of unseen data. The content inside the training dataset is the knowledge possessed before being labelled, while the content inside the test set is all the inputs, then the model needs to predict the answer based on the given information. Finally, the performance of this model can be evaluated based on various methods.

The split data ratio was set at 7:3, with the training data accounting for 70% of the total records and 30% reserved for the test data set. the 7:3 ratio allows the model to be adequately trained as it contains multiple cases where patterns can be found, and the 30% reset is large enough to cover sufficient instances (cases) for potential input and evaluation. Importantly, the data structure of test datasets should remain the same with raw dataset, especially for target attribute, so we did rebalance method on training dataset only. The target attribute distribution of training and testing dataset as shown in Figure 51.

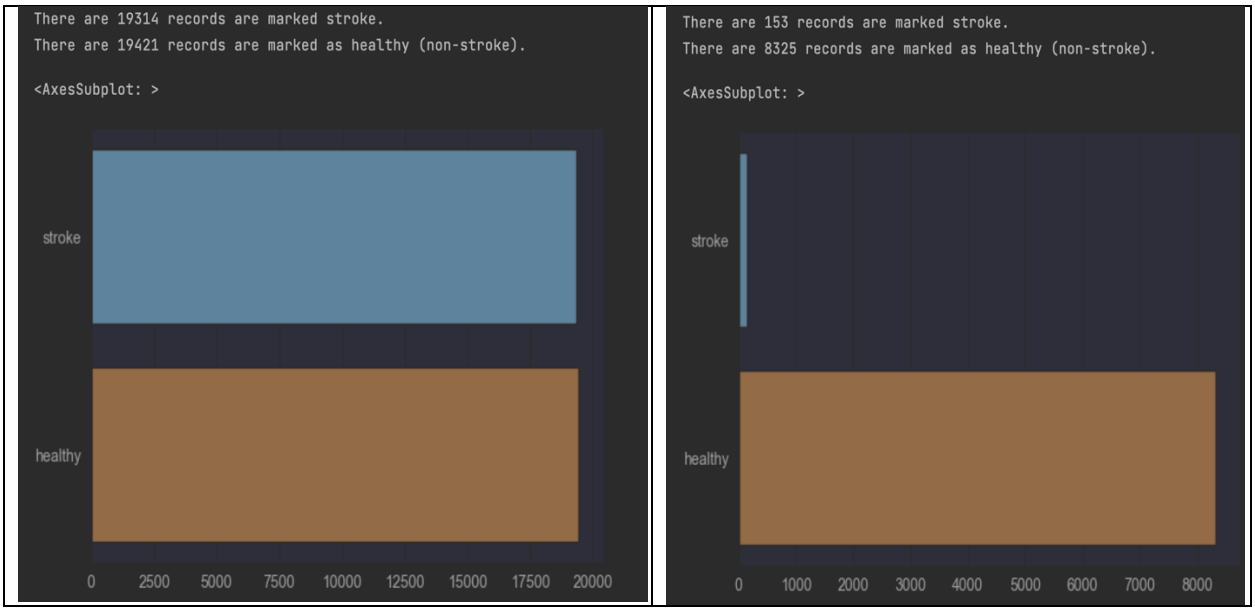


Figure 51. The data structure of stroke for the train & test set

As we can see, there are 19,314 records are marked as stroke, and 19,421 records marked as non-stroke in the training dataset.
There are 153 records are marked as stroke, and 8,325 records marked as non-stroke in the testing dataset.

7.2 Conduct Data mining

We run the partition that we have split in section 3.4, which split the train set and test set with a 7:3 ratio randomly. Then, we do data mining with selected SVM algorithm.

The data mining model at this stage is being prepared to produce results. The input parameters will be gender, age, hypertension, heart disease, ever married, average blood glucose level, new BMI, new smoking status, new job type, and the target is set to be stroke as shown in Figure 52. The total amount of data used was 47,213 instances, which were split into training/testing datasets. The split ratio was set to 7:3. Training dataset after using the rebalance method has 38,735 records as shown in Figure 53, and testing dataset has 8,478 records as shown in Figure 54.

```
print(X_train_reimb.columns, Y_train_reimb.columns)
```

```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'avg_glucose_level', 'new_smoking_status', 'new_bmi', 'new_work_type'],
      dtype='object') Index(['stroke'], dtype='object')
```

Figure 52. input parameters of the model

```
print(X_train_reimb.all, Y_train_reimb.all)

 38733          70.306056                  0        1             0
 38734          72.950738                  0        2             0

 [38735 rows x 9 columns]> <bound method NDFrame._add_numeric_operations.<locals>.all of
     stroke
 0           0
 1           0
 2           0
 3           0
 4           0
 ...       ...
 38730       1
 38731       1
 38732       1
 38733       1
 38734       1

 [38735 rows x 1 columns]>
```

Figure 53. training dataset after rebalance

```
print(X_test,Y_test.all)

 8476             86.38                  0        2             0
 8477            104.12                  0        1             0

 [8478 rows x 9 columns] <bound method NDFrame._add_numeric_operations.<locals>.all of
  stroke
 0           0
 1           0
 2           0
 3           0
 4           0
 ...       ...
 8473        0
 8474        0
 8475        0
 8476        0
 8477        0

 [8478 rows x 1 columns]>
```

Figure 54. testing dataset

Figure 55 shows the entire data flow as well as the structure of the model. Also, the model with the predefined parameters and settings.

Figure 56 shows after the model running, the results were successfully generated, which shows the results of the execution of the model. The figure includes the accuracy of the model, recall score, confusion matrix, and time measurements.

Figure 57 is showing the overall ROC (receiver operating characteristic curve) of the model. The ROC is 0.7668, the closer the value of ROC is to 1, the better the model is chosen.

Figure 58 shows the importance of each attribute in the model. Average glucose level is the most important predictor, and age is the second important predictor and new BMI group is the third important predictor.

```
print("########################################")
print("Selected model for the project")
print("########################################")
# SVM
# kernel support vector machine
print("SVM - SVC")
import time
time_start=time.time()
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
clf = SVC(kernel = 'rbf')
clf.fit(X_train_reimb, Y_train_reimb)

Y_pred = clf.predict(X_test)

# data after re-balance into dimension=3 & kernel = 'rbf'

print("data after re-balance & kernel = 'rbf'")
print("Accuracy score", sl.metrics.accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred), "Confusion Matrix")
print("Recall score", sl.metrics.recall_score(Y_test, Y_pred))
print("ROC",sl.metrics.roc_auc_score(Y_test, Y_pred))

time_end=time.time()
print('time cost',time_end-time_start,'s')
```

Figure 55. Selected SVM model with predefined parameters

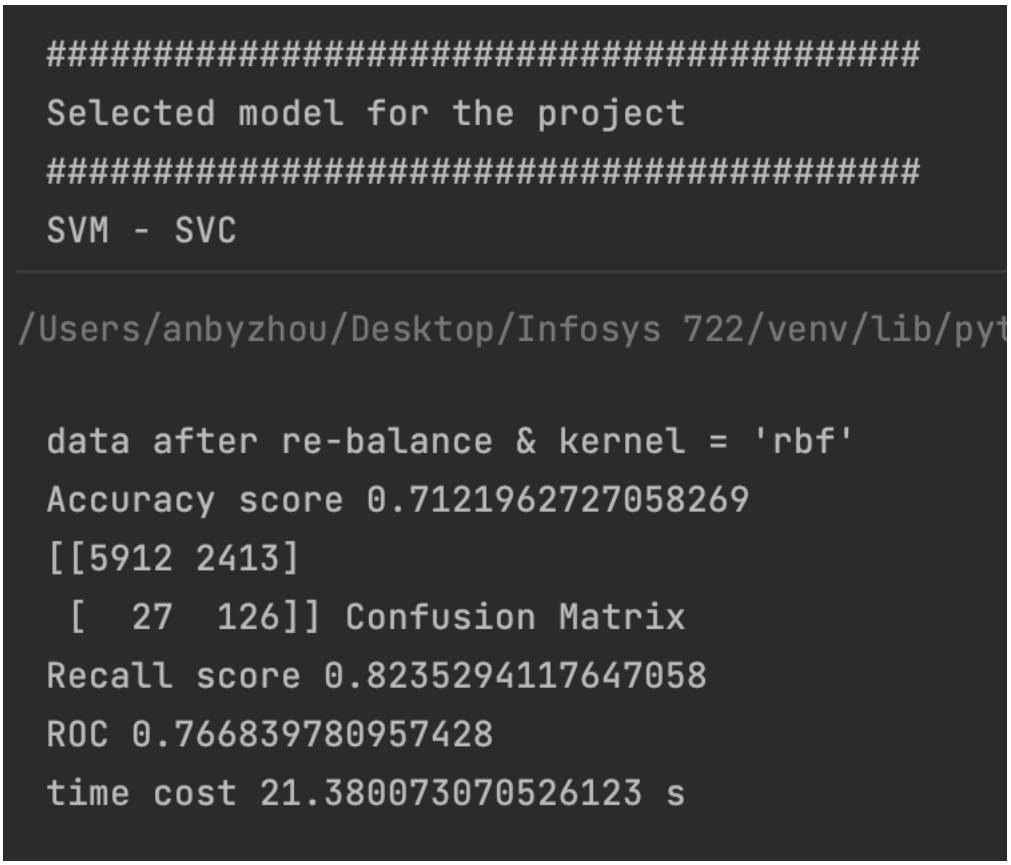


Figure 56. Selected SVM model running result

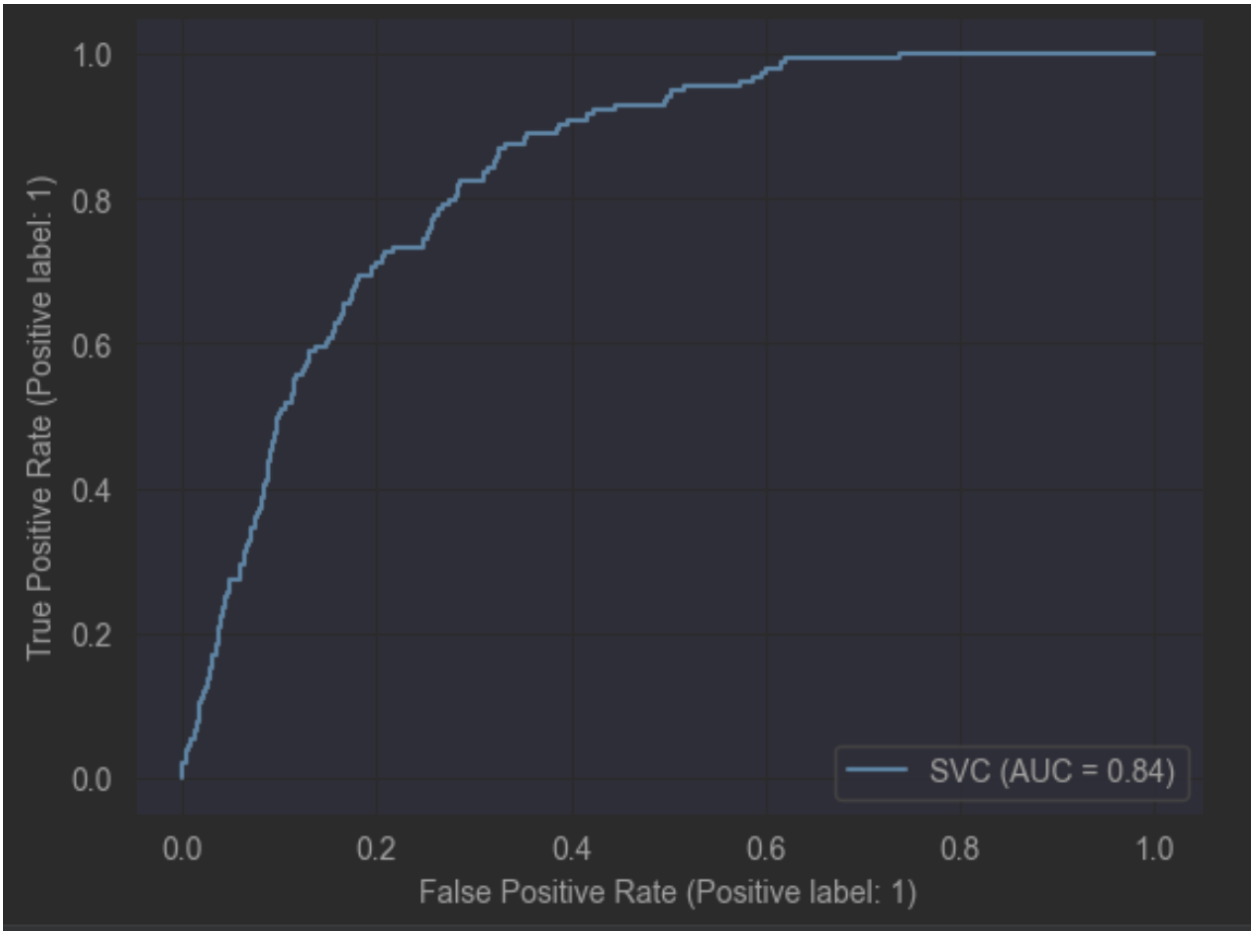


Figure 57. ROC curve

```
importance = dict(zip(keys, feature_importance))
sorted_importance = sorted(importance.items(), key=lambda x: x[1],reverse=True)
print(sorted_importance)
```

```
[('avg_glucose_level', 0.6894617280208079), ('age', 0.13732114470307533), ('new_bmi',
0.03535753461424832), ('gender', 0.03370248007289918), ('new_smoking_status', 0
.03221516371638127), ('ever_married', 0.02368496277624564), ('new_work_type', 0
.021385640091579457), ('hypertension', 0.013972425479202244), ('heart_disease', 0
.012898920525560659)]
```

Figure 58. Feature Importance

7.3 The output of models (Search for patterns)

From the result, the algorithm accuracy was 71.22%, recall score is 82.35, and ROC is 0.77, which indicated a good performance of the used SVM algorithm for stroke prediction in the project.

Recall that the business goal of this data mining process is to discover patterns among attributes and their impact on stroke. This process is also considered to assess the importance of the attributes and draw a conclusion about the risk factors for each feature.
The main findings are listed below:

1. The average glucose is the most important predictor as the medical condition of stroke. Patients with higher average glucose level is more likely to get stroke. Most patients get stroke with the average glucose level in the range 113 – 180 mg/dL as shown in Figure 59. The normal blood glucose range is 70 to 99* mg/dL (Cleveland Clinic medical professional, 2018).

```
#plot relationship between attribute and stroke
seaborn.boxplot(x=featured_data['stroke'],y=featured_data['avg_glucose_level'])
```

```
<AxesSubplot: xlabel='stroke', ylabel='avg_glucose_level'>
```

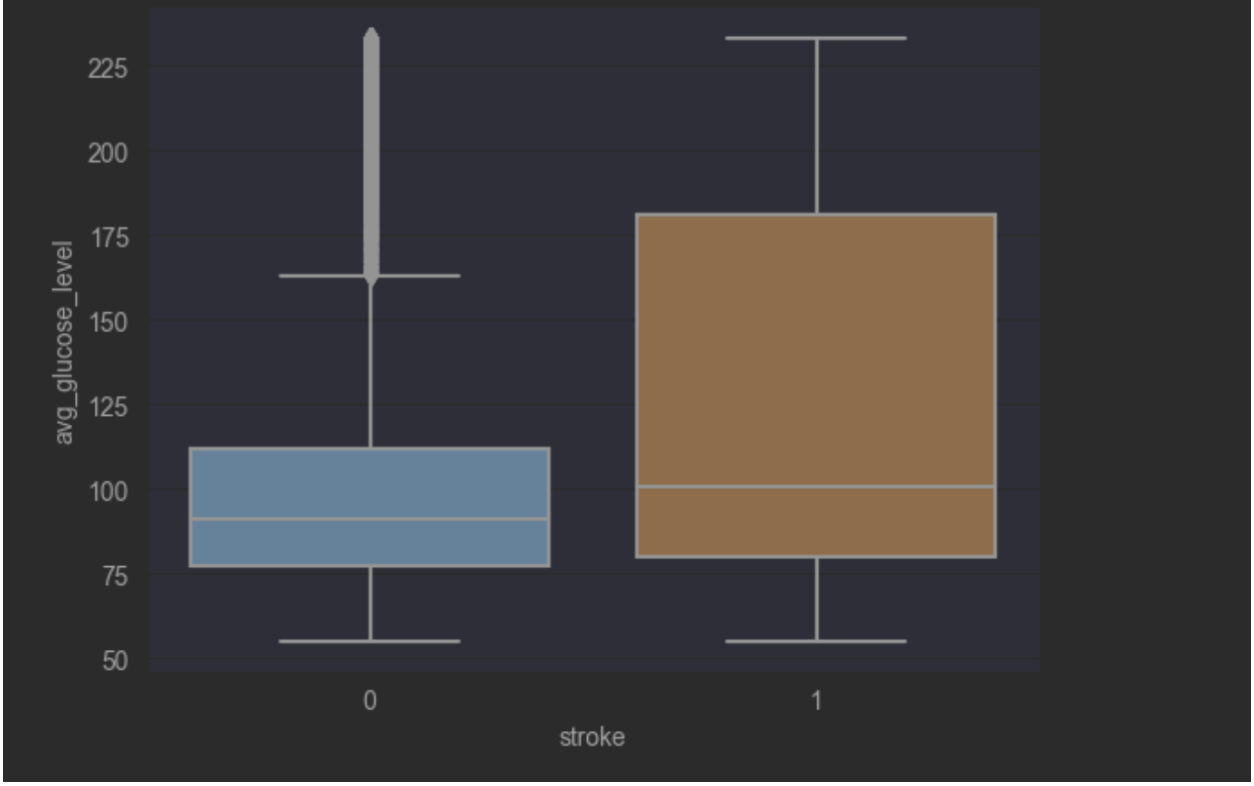


Figure 59. relationship of average glucose level and stroke

2. The age factor is much more influential than other medical condition (hypertension, heart disease). As seen in Figure 60, age is much more important than other medical-related information. Most patients get stroke with in an age range of 60 – 80 years old.

Figure 60. relationship of ageand stroke

3. From the result in PCA, we did data reduction/dimensionality reduction is also required to avoid the mislead of algorithm/model. We can see that the distance between stroke patients and non-stroke patients is too close, which means most stroke patients has very similar features as non-stroke patients, as shown in Figure 61. So we can justify SVM model's parameter kernel to make decision surface, which can cluster stroke patients better. So that we can make better predictions.

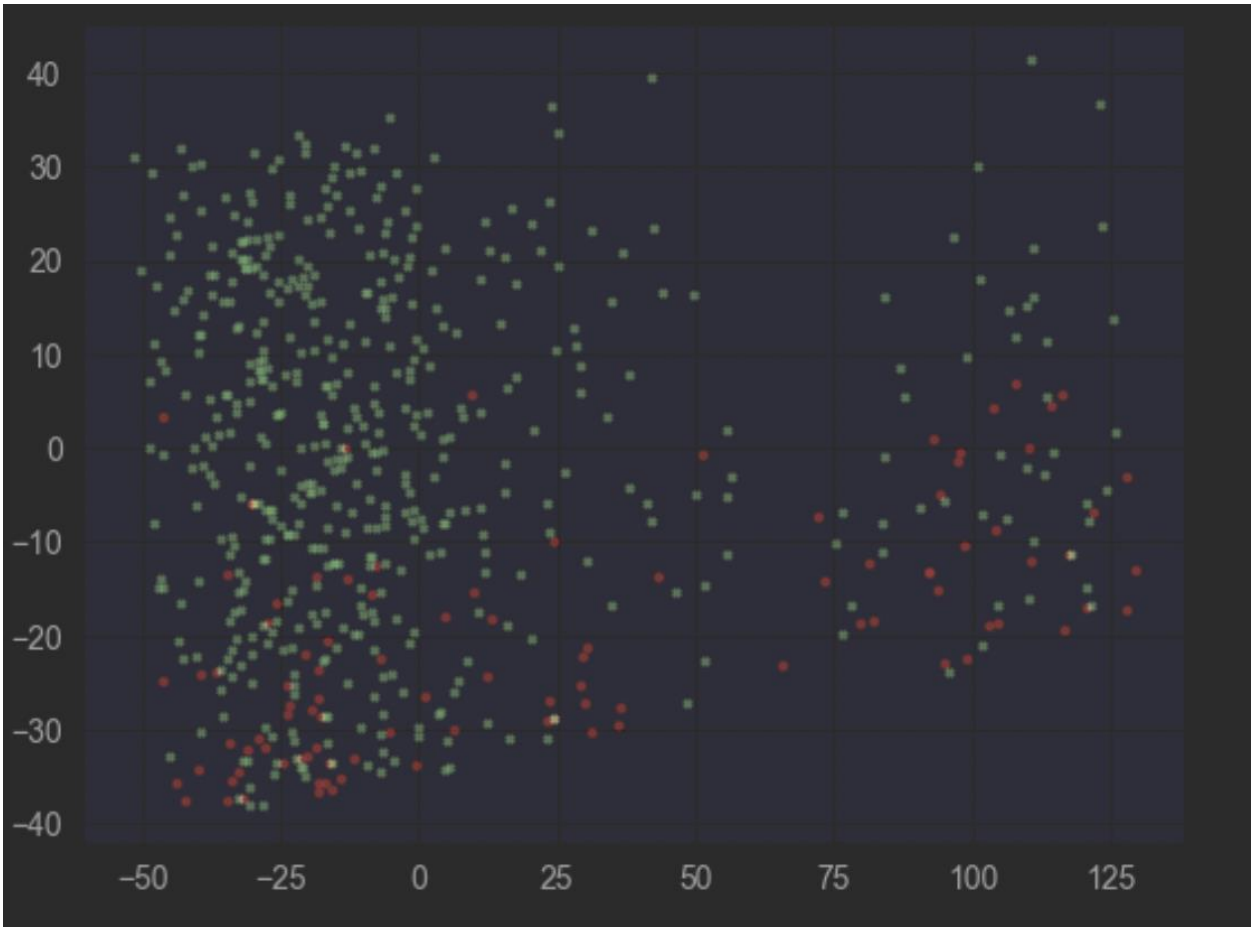


Figure 61. PCA data reduction

4. According to pattern 3, we get to know that stroke patients and non-stroke patients have similar features. So, in addition, a combination of several factors will have an impact on the final stroke conditions. As the patients who has higher glucose level than normal range and elder age would have more chance to get stroke.

8. Interpretation

8.1 Data Mining Pattern

The quantity and quality of the dataset is good because it provides a sufficient number of cases for training and the test sample covers most of the cases. The use of the oversampling technique ADASYN algorithm balanced the difference between the two labels without causing overfitting problems.

The model used for this dataset is svm. It addresses the shortcomings of random trees, random forests, knn, and spatial-based classification because the characteristics of patients with and without stroke are similar. svm projects the data to a higher latitude for classification and is able to predict the results more accurately. The recall and f1 score of 77% proved that the model is usable and that the model matches well with the dataset. Therefore, it is a good model.

Pattern 1：

Average blood glucose is the most important predictor of the medical condition of stroke. Patients with higher average blood glucose levels were more likely to have a stroke. As shown in Figure 59, stroke occurs in most patients with an average blood glucose level in the range of 113-180 mg/dL. The normal blood glucose range is 70 to 99* mg/dL (Cleveland Clinic Medical Specialists, 2018). The svm model is also run to separate the hyperplane based on this most important parameter, project to higher latitudes, and then combine it with other parameters for classification.

Pattern 2：

The age factor is much more influential than other medical conditions (hypertension, heart disease). As shown in Figure 60, age is much more important than other medically relevant information. Most patients had a stroke between the ages of 60 and 80 years. This is indisputable, and in combination with blood glucose, the probability of occurrence of high blood glucose is higher in the elderly and therefore causes a greater chance of stroke.

Pattern 3：

The analysis of the importance of different attributes concluded that the average blood glucose level and BMI had a greater degree of influence than other medically relevant factors. the BMI value as well as the average blood glucose level showed the lifestyle of a person. According to the research paper, a person who is obese has a higher chance of getting a stroke than others, and a person with a lower BMI and average blood sugar maintains a good lifestyle. Therefore, they are less likely to have a stroke.

Pattern 4：

From the results of PCA, we also need to perform data reduction/deviation to avoid misleading algorithms/models. We can see that the distance between stroke patients and non-stroke patients is too close, which means that most stroke patients and non-stroke patients have very similar characteristics, as shown in Figure 61. So we can argue the parameter kernel of the SVM model to make a decision surface, which can better cluster the stroke patients. In this way, we can make better predictions.

Pattern 5：

According to model 4, we can know that stroke patients and non-stroke patients have similar characteristics. So, in addition to that, the combination of several factors will have an impact on the final stroke situation. This is because patients with higher-than-normal blood glucose levels and older age will have a higher chance of having a stroke.

Pattern 6：

The type of work and smoking habits has a smaller impact on stroke. Figure 58 shows the importance of the attributes. It is clear from the figure that job category and smoking habits influence stroke, but the effect is limited. As a factor, these two attributes have no direct effect

on stroke. However, these attributes as part of lifestyle, job type represents the stress and environment a person is exposed to, while smoking status influences the presence of hypertension and heart disease. Thus, the effects of job type and smoking habits on stroke are indirect.

8.2 Data Visualization

There are several visualisations being applied to the current model.
Figure 62 is showing the model execution result.
Figure 63 is showing the ROC curve of the model execution. A ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. The curve plots two parameters: True Positive Rate and False positive rate.
Figure 64 is the bar plot showing the importance of each attribute.
According to the importance of attributes, the most 2 important predictor will show the relationship with stroke.
Figure 65 is showing relationship between average glucose level and stroke.
Figure 66 is showing relationship between age and stroke.

```
########################################
Selected model for the project
########################################
SVM - SVC
/Users/anbyzhou/Desktop/Infosys 722/venv/lib/pyt

data after re-balance & kernel = 'rbf'
Accuracy score 0.7121962727058269
[[5912 2413]
 [  27  126]] Confusion Matrix
Recall score 0.8235294117647058
ROC 0.766839780957428
time cost 21.380073070526123 s
```

Figure 62. Selected SVM model running result

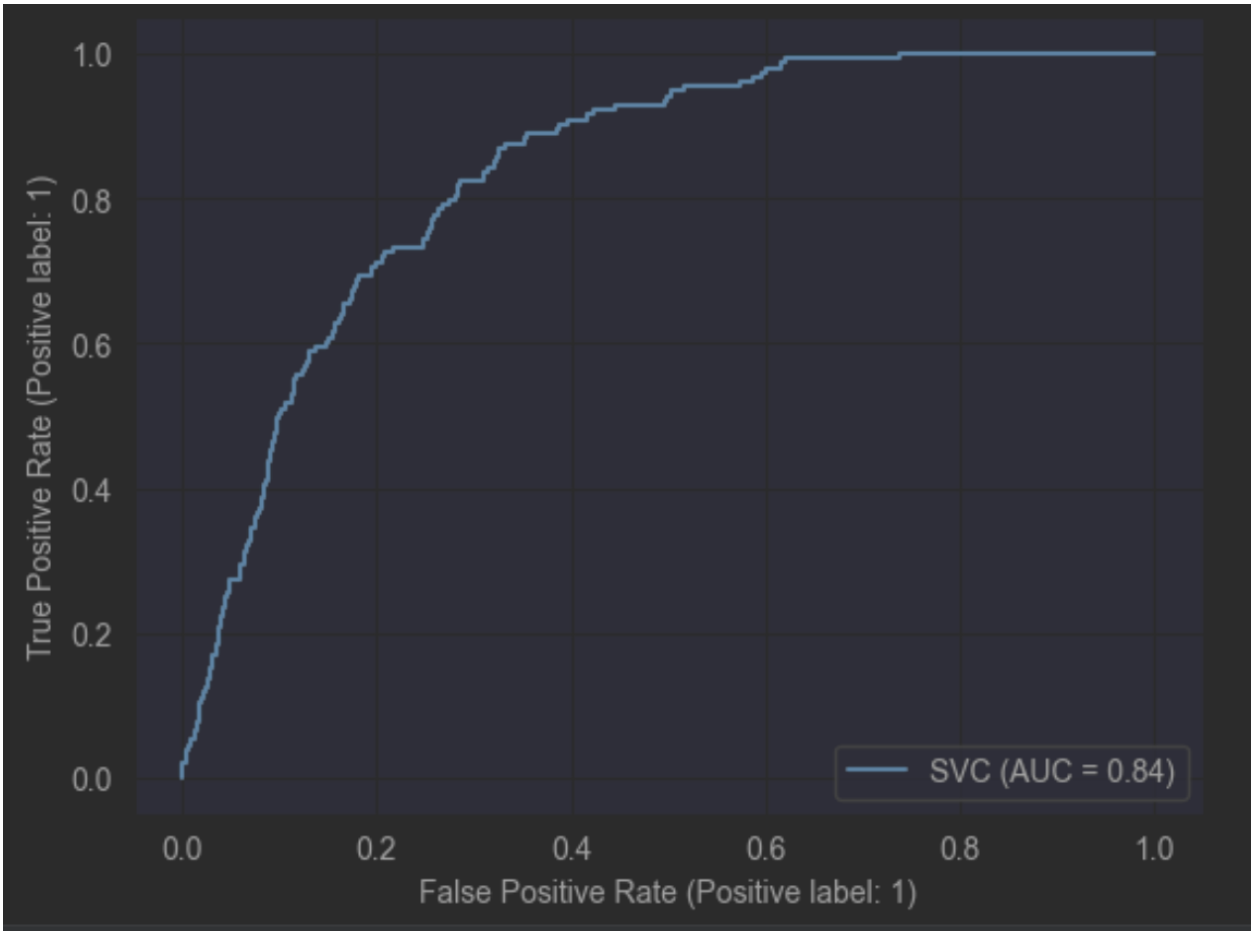

1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
SVC (AUC = 0.84)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

Figure 63. ROC curve

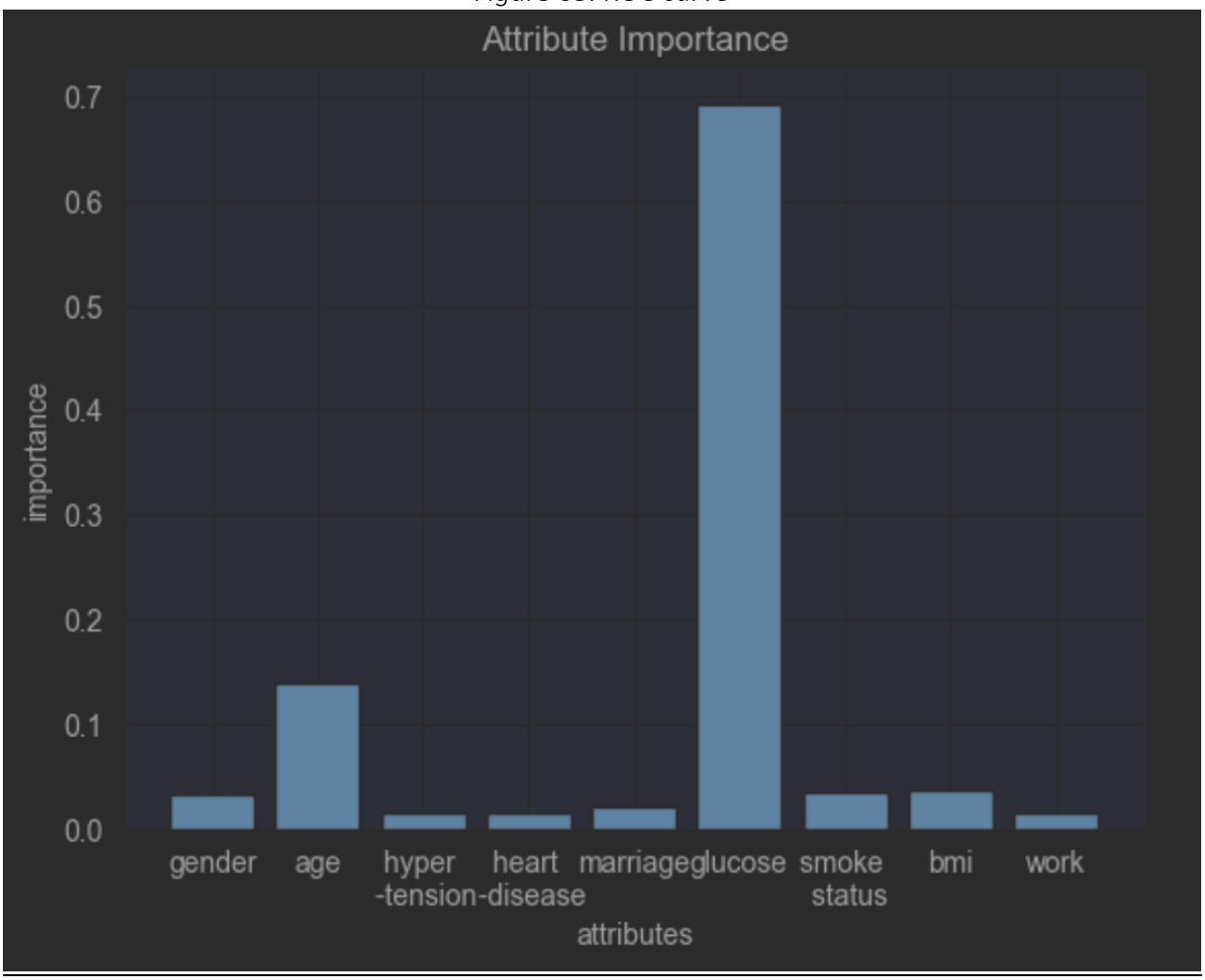


Figure 64. Attribute importance

```
#plot relationship between attribute and stroke
seaborn.boxplot(x=featured_data['stroke'],y=featured_data['avg_glucose_level'])
```

```
<AxesSubplot: xlabel='stroke', ylabel='avg_glucose_level'>
```

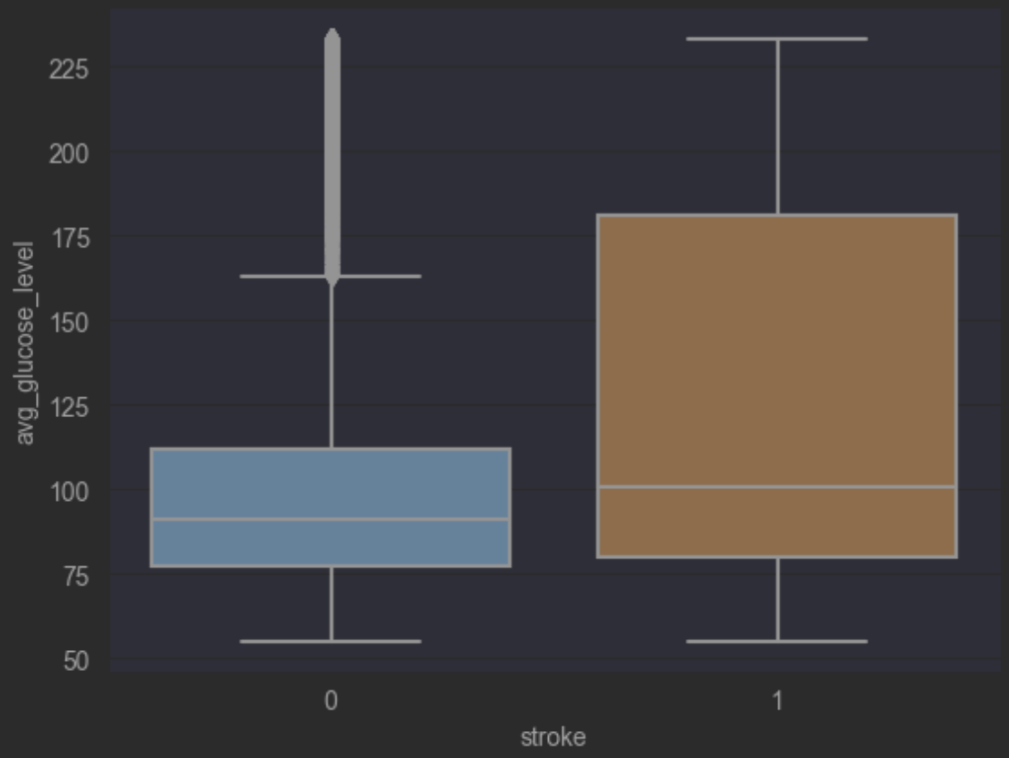


Figure 65. relationship of average glucose level and stroke

Figure 66. relationship of age and stroke

8.3 Interpretation of result, models and patterns

Patterns:

The patterns discussed above suggest that medical-related information is not the most risk factor for stroke. The original hypothesis focused on the influence of medical information, but the factors found to be more influential are now blood glucose and age. Based on the research, it is recommended that potential patients with blood glucose and age above a certain threshold undergo regular physical examinations to reduce the risk of stroke.

Model and result:

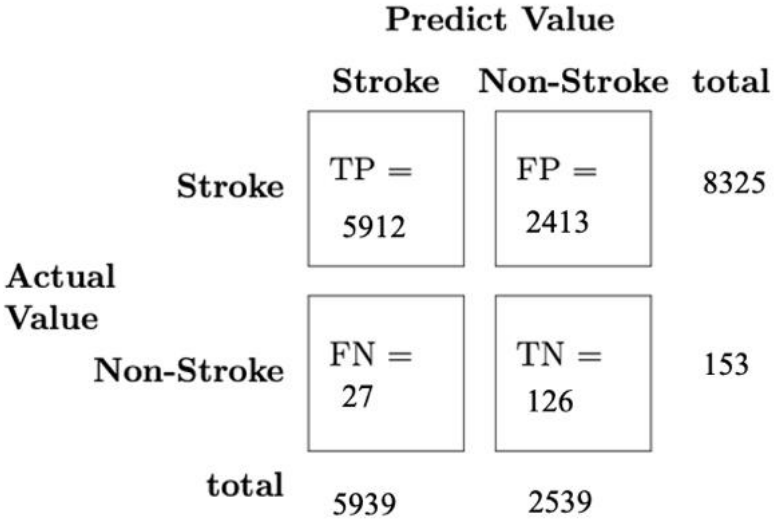


Figure 67. Model running result confusion matrix

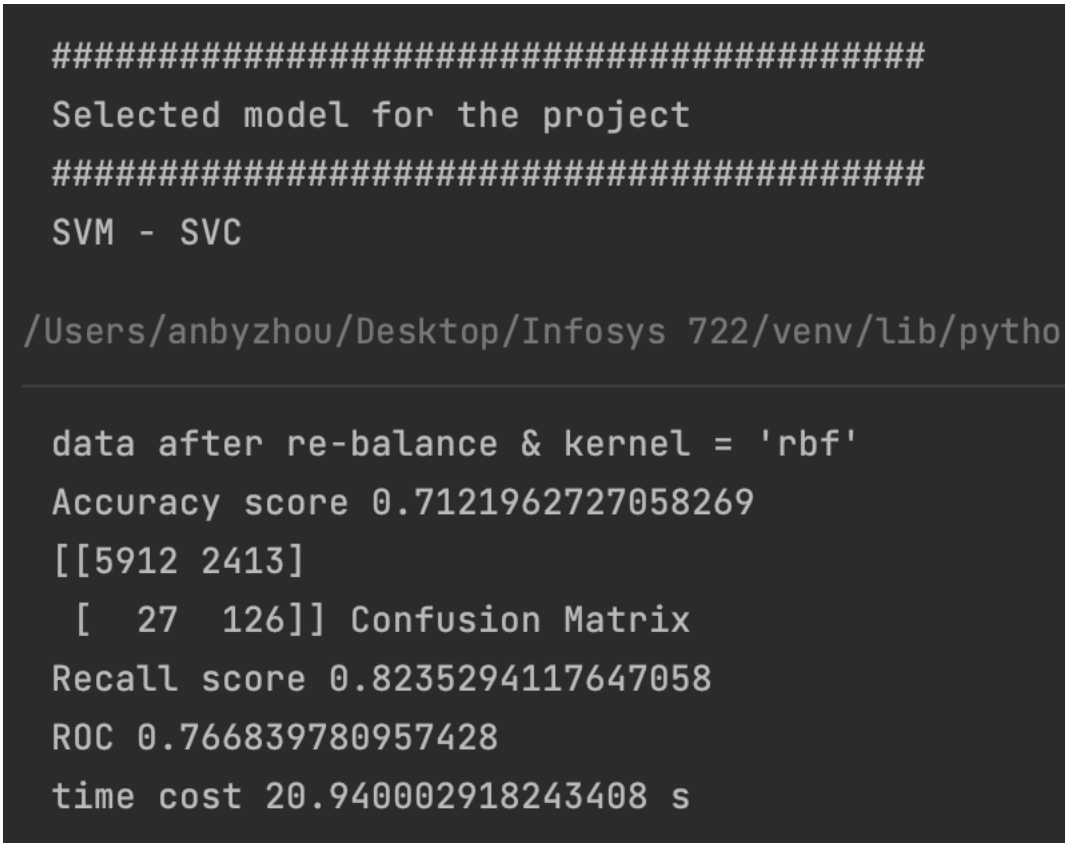


Figure 68. Model and result

The overall performance of the model was quite satisfactory, in dealing with an unbalanced database, we had to perform a rebalancing, meaning that noise was added to the dataset. However, the model still achieves an average accuracy of 75%. At the same time, the recall rate (calculated by the confusion matrix above Figure 67) reached 82%, which is quite satisfactory. Figure 68 shows the ROC curve, indicating that the model performed well. The model is able to perform data prediction for both stroke and non-stroke, and the model itself is usable and valid.

However, overall, the accuracy and recall of this model is still not high enough to accurately achieve most predictions, but it is far from being ready for use for medical purposes, and more data needs to be collected for input and training, and more data is needed for testing. However, this model needs further improvement and involves more predictors and adjusting more parameters to make the model more accurate and usable for medical purposes in the future. From a data perspective, the model still sees relatively little information about the candidates for stroke. Even with the test dataset for detection, overfitting problems may still occur, resulting in an insensitive model.

8.4 Assess and Evaluations of result, model and patterns

The assessment and evaluation process is similar to an after-action review, focusing on errors that occurred during the evaluation process and potential ways to improve them.

Evaluation of patterns:

The model finds patterns that provide relationships between attributes and targets, but the relationships themselves are not as clear. The model analysed the importance of other attributes on the target attribute and the relationship between other attributes and the target attribute. However, the effect of combining attributes with each other on the target attribute stroke is not clear; for example, the combination of heart disease and hypertension has not been assessed as a strong predictor of stroke, but there is now much literature strongly supporting these factors. Therefore, this will affect the accuracy of the results and conclusions. Potential solutions are to add additional measurements to the current algorithm or to use multiple algorithms to perform the task.
The patterns generated by this data model are less supported by medically relevant studies. These models found that hyperglycemia and age were the largest influencing factors for stroke. However, the importance of these attributes is generated by the data mining process, and limited medical studies are being reported to support these ideas. Further improvements to this would be to add more attributes and large amounts of data, using mixed models of models to predict outcomes and gather opinions supported by medical studies.

Performance evaluation of the model and results:
Current data models are good at providing good predictions from training datasets, but are not sufficient for use in medical purposes. A broader sample of data should be taken and the training success of the model should be examined further. Step-by-step adjustment of parameters to enhance the model can also introduce a mixture of multiple models. In addition, the data model should involve a large sample of many actual stroke data rather than predicted data. Predicted data are more likely to overfit the model and ultimately make the model insensitive to data that has not been seen before.

And for the results, all in all, the overall performance of the model is satisfactory. To improve the accuracy of the results, recall, roc, the accuracy of the algorithm can be further improved by adding more cases or using a larger database to do data mining, there or by doing a larger training data set. More, this result only predicts whether the patient will have a stroke or not, and for patients who have already had a stroke, this prediction is not very meaningful, so we need further improvement of the final result should be for the individual risk level of stroke. Because for patients who have already had a stroke, we are giving a prediction for what has already happened, rather than predicting the individual risk of stroke and giving warnings and recommendations.

8.5 Iterations and Improving Models

We could use the method of re-split training and test data set to improve the model, for example, re-split the training model and test model by 7.5:2.5 or 8: 2 ratios. Also, data pre-processing is quite important for SVM model training. A suitable normalisation method could be used. In this dataset, a great number of smoking records are missing, a better way of dealing with the missing value should be explored, for example, we could set the smoke status of patients with an age smaller than 20 as non-smokers.

Iteration of the model will continue to improve the performance of the model and bring clarity to the data mining process. In order to improve the model, a review of the current steps is necessary.

Step 1: The first step in data mining is to understand the context of the dataset and to develop a business understanding of this dataset to have a grasp of the current situation. By understanding the dataset, the overall structure of this dataset will be clear, the business objectives will be clarified, and the expected results will be listed. In this step, the precise positioning of the business objectives is crucial.

Step 2: The second step in data mining. The initial data collection involves finding and identifying suitable datasets and importing them into Tableau and Python. The overall characteristics and quality of the data will then be assessed and analyzed. In this step, an understanding of the structure and quality of the data set is an important step before data mining.

Step 3: The third step in data mining. This step involves data cleaning and improving the quality of the dataset. In this step all missing values will be removed and extreme this will also be processed to ensure the quality of the dataset. In addition, feature combination or integration will also take place in this step. The data cleaning and processing process will influence the final model selection and prediction discovery.

Step 4: The fourth step in data mining. In this step, imbalances in the data will be eliminated, especially in the target attributes. All unnecessary attributes with low relevance to the final result will be filtered out and removed and the unbalanced data will be rebalanced.

Step 5: The fifth step in data mining. In this step, we need to make a judgement based on the dataset and the expected results and determine the data mining method. The choice of data mining method will influence the choice of data mining algorithm. And the train and test set will be splinted before we do data mining.

Step 6: The sixth step in data mining. In this step, based on the training dataset and test dataset segmentation of the dataset, we perform model/algorithm selection and building. Data mining algorithms will be evaluated and selected based on business objectives and data mining goals.

Step 7: The seventh step in data mining. In this step, the data mining model/algorithm will be run, and the patterns are being identified.

Step 8: The eighth step of data mining. The results of the data mining, models and patterns are evaluated, and the results are analysed for validity and correctness. Attempts are made to improve the accuracy of the model, and the most convenient and easy updates can occur in the training/test data split. As the number of training sets increases, the efficiency of the algorithm increases. The new data splitting ratio was set to 8:2 as in Figure 69.

```python
# dataset split to train and test
from sklearn import model_selection
y = featured_data['stroke']
X = featured_data.drop(columns=['stroke'])
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, y, test_size=0.2)
X_train = X_train.reset_index()
X_test = X_test.reset_index()
Y_train = Y_train.reset_index()
Y_test = Y_test.reset_index()
del X_train['index']
del X_test['index']
del Y_train['index']
del Y_test['index']
```

Figure 69. New train/test dataset split

```
########################################
Selected model for the project
########################################
SVM - SVC

/Users/anbyzhou/Desktop/Infosys 722/venv/lib/python3
.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y
 was passed when a 1d array was expected. Please change the shape of y to (n_samples, ),
for example using ravel().
  y = column_or_1d(y, warn=True)

data after re-balance & kernel = 'rbf'
Accuracy score 0.719037508846426
[[5992 2347]
 [  35  104]] Confusion Matrix
Recall score 0.7482014388489209
ROC 0.7333764119535406
time cost 20.605851888656616 s
```

Figure 70. The accuracy report after new data splitting

The application of the new data proportions resulted in an increase in overall training accuracy and in the accuracy of the test set. It can be seen that more training data will give the model higher accuracy and also validate the stability of the SVM model.

9. Reference

2021 Guideline for the Prevention of Stroke in Patients With Stroke and Transient Ischemic Attack: A Guideline From the American Heart Association/American Stroke Association. *Stroke* 2021; May 24: [Epub ahead of print].

Amal, L. (2020, October 26). *Heart stroke*. Kaggle. Retrieved September 18, 2022, from https://www.kaggle.com/datasets/lirilkumaramal/heart-stroke/discussion/225077

Barnett, H. J. (2005). Stroke by Cause. *Stroke, 36* (12), 2523-2525. doi: 10.1161/01.STR.0000194560.65809.47.

Berry, M. W., Mohamed, A., & Yap, B. W. (Eds.). (2019). *Supervised and unsupervised learning for data science*. Springer Nature.

Breiman Leo (2001). "Random Forests". Machine Learning 45 (1): 5–32.

Chong, J. Y. (2022, August 4). *Overview of stroke - brain, spinal cord, and nerve disorders.* MSD Manual Consumer Version. Retrieved August 15, 2022, from https://www.msdmanuals.com/home/brain,-spinal-cord,-and-nerve-disorders/stroke-cva/overview-of-stroke

Cleveland Clinic medical professional. (2018, February 21). *Blood glucose test: Levels & What They mean*. Cleveland Clinic. Retrieved September 19, 2022, from https://my.clevelandclinic.org/health/diagnostics/12363-blood-glucose-test#:~:text=A%20blood%20glucose%20test%20is,indicate%20pre%2Ddiabetes%20or%20diabetes.

Lim, M. (2021, Jun 19). Effective ways to prevent a stroke: Take pre-emptive measures by understanding the common causes of stroke. *The Business Times* http://ezproxy.auckland.ac.nz/login?url=https://www.proquest.com/newspapers/effective-ways-prevent-stroke/docview/2542628647/se-2

W3Schools. (2022). *Pandas Tutorial*. Pandas tutorial. Retrieved September 18, 2022, from https://www.w3schools.com/python/pandas/default.asp

Wajngarten, M., & Silva, G. S. (2019). Hypertension and stroke: Update on treatment. *European Cardiology Review, 14(*2), 111–115. https://doi.org/10.15420/ecr.2019.11.1

WHO. (2020, December 9). *The top 10 causes of death*. World Health Organization. Retrieved July 29, 2022, from https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death.

10. Disclaimer

"I acknowledge that the submitted work is my own original work in accordance with the University of Auckland guidelines and policies on academic integrity and copyright. (See: https://www.auckland.ac.nz/en/students/forms-policies-and-guidelines/student-policies-and-guidelines/academic-integrity-copyright.html, Links to an external site.).

I also acknowledge that I have appropriate permission to use the data that I have utilized in this project. (For example, if the data belongs to an organization and the data has not been published in the public domain then the data must be approved by the rights holder.) This includes permission to upload the data file to Canvas. The University of Auckland bears no responsibility for the student's misuse of data."