Stroke Prediction Using Machine Learning Methods

GR5291 Project Report

Lichun He (lh3094@columbia.edu)

Yuqi Zhang (yz4091@columbia.edu)

Shiang XuanYuan (sx2292@columbia.edu)

Long Lin(ll3340@columbia.edu)

Weijia Wang (ww2589@columbia.edu)

Ran Zhang (rz2568@columbia.edu)

Table of Content

1 Introduction

As technology continues to evolve, medical care is no longer just about a doctor identifying diseases through his experience. With the advent of annotated Electronic Health Record (EHR) records of patients, we can now use data mining techniques to identify trends in datasets that can help medical physicians make accurate predictions. It improves medical conditions and lowers treatment costs, as well as assists doctors in identifying the onset of the disease at an early stage. We are particularly interested in stroke and plan to implement Machine Learning knowledge to propose a model that could assist clinicians in determining if a patient is at risk for stroke.

The main contributions of this report are as follows – (a) we give a complete overview of the various risk variables for stroke prediction. We investigate the various features presented in the dataset of Electronic Health Record (EHR) records of patients, which is an open-access Stroke Prediction dataset generated from Kaggle; and under the fact of the unbalanced dataset, we provide two datasets to use for stroke prediction - undersampling and oversampling; (b) we also use dimensionality reduction technique to firstly identify patterns in the low-dimensional subspace of the feature space, and secondly adopt three classes of feature selection to keep feature subsets for later model construction use; and (c) we compare popular machine learning models for stroke prediction in a publicly available dataset.

The structure of the report is as follows. The following Section 2 describes the dataset used in our project and covers the exploratory data analysis. Dimensionality reduction and feature selection analysis are explained in Section 3. Section 4 describes the predictive modeling techniques utilized and their performance on the dataset. Finally, Section 5 wraps up the findings and looks ahead to future research.

2 Exploratory Data Analysis

Before we start to do data explorations, we want to give a brief introduction to the dataset we use. The dataset is generated from Kaggle with 5110 records of participants and 10 explanatory variables along with 1 target variable - "stoke". Table 2.1 shows the description of 11 variables and some useful insights generated from python.

Variables	Description	Non-Null Count	Dtype
gender	"Male", "Female" or "Other"	5110 non-null	object
age	age of the patient	5110 non-null	float64
hypertension	0 if the patient doesn't have hypertension, 1 if the patient has hypertension	5110 non-null	int64
heart_disease	0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease	5110 non-null	int64
ever_married	"No" or "Yes"	5110 non-null	object
work_type	"children", "Govt_jov", "Never_worked", "Private" or "Self-employed"	5110 non-null	object
Residence_type	"Rural" or "Urban"	5110 non-null	object
avg_glucose_level	average glucose level in blood	5110 non-null	float64
bmi	body mass index	4909 non-null	float64
smoking_status	"formerly smoked", "never smoked", "smokes" or "Unknown"	5110 non-null	object
stroke	1 if the patient had a stroke or 0 if not	5110 non-null	int64

Table 2.1: Summarize of the dataset

From Table 2.1, we find that in the 'non-null count' column, there exist some missing values for the feature "bmi". Next step, we believe it is considerable to check whether this dataset contains null before going any further.

2.1 Check Missing Value

We formally check the missing values for each feature and below is the resulting Figure 2.1. We find that only BMI contains the missing value and it has 201 missing values in this column.

Then, we fill these rows by the median value of BMI. We check again and conclude that there are no missing values in the dataset now (Fig. 2.2).

gender	0
age	0
hypertension	0
heart_disease	0
ever_married	0
work_type	0
Residence_type	0
avg_glucose_level	0
bmi	201
smoking_status	0
stroke	0

Fig. 2.1: Missing values in total for each category

gender	0
age	0
hypertension	0
heart_disease	0
ever_married	0
work_type	0
Residence_type	0
avg_glucose_level	0
bmi	0
smoking_status	0
stroke	0

Fig. 2.2: Missing values in total in each category after filling in replaced values

2.2 Check Numerical and Categorical Variables

By the overall information mentioned above, the data type in this dataset is separated into two types: integer, and object. Thus, we first want to check the categorical columns, as well as the numerical columns, excluding "stroke".

- Categorical columns: ['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status']

- Numerical columns: ['age', 'hypertension', 'heart_disease', 'avg_glucose_level', 'bmi']

However, 'hypertension' and 'heart_disease' are classified as numerical variables but, in fact, they are categorical variables with binary outcomes (0,1). Hence, we move these two features from numeric class to category class. And as a result, we get the numerical variables and categorical variables as follows,

- Categorical columns after adding binary features: ['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status', 'hypertension', 'heart_disease']
- Numerical columns after deleting binary features: ['age', 'avg_glucose_level', 'bmi']

Since we have the numerical columns, we could derive the descriptive statistics of those columns (Table 2.2).

	count	mean	std	min	25%	50%	75%	max
age	5110.0	43.226614	22.612647	0.08	25.000	45.000	61.00	82.00
avg_glucose_level	5110.0	106.147677	45.283560	55.12	77.245	91.885	114.09	271.74
bmi	5110.0	28.862035	7.699562	10.30	23.800	28.100	32.80	97.60

Table 2.2: The statistics of numerical data

2.3 Check Uniqueness

We plan to check the distribution for the target variable “stroke”. Table 2.3 shows that our target variable “stroke” is a binary variable that contains only "0" and "1" as outcomes. To be specific, "1" means the patient had a stroke and “0” means the opposite. Furthermore, we can also conclude that our dataset is unbalanced because the records of patients without stroke are approximately 19 times more than those with stroke. Thus, in the following Section 2.9 Dealing with Imbalanced Dataset, we are going to present a balanced sample based on the outcomes here.

	Stroke: Yes	Stroke: No
Counts	249	4861

Table 2.3: Distribution of target variable `stroke`

We continue to check the uniqueness of each categorical column (Fig. 2.3). After comparing the results to the dataset description, we find something worthy to note that there are three unique outcomes in “gender”. With further checking, we find there exists one row with “other” outcome in the “gender” column and we delete this special case.

```
gender             3
ever_married       2
work_type          5
Residence_type     2
smoking_status     4
hypertension       2
heart_disease      2
```

Fig. 2.3: Unique values in total of each categorical column

2.4 Check Correlation

Next, we will check the correlation between the numerical variables and generate a heatmap to help us visualize and demonstrate the result (Fig. 2.4).

We don't observe any negative value in the heatmap. Hence, all the features are positively correlated with the target variable "stroke". Moreover, we summarize some interesting findings as follows:

- 'age' is correlated to stroke by a factor of 0.25
- 'avg_glucose_level' is correlated to stroke by a factor of 0.13
- 'bmi' is correlated to stroke by a factor of 0.036

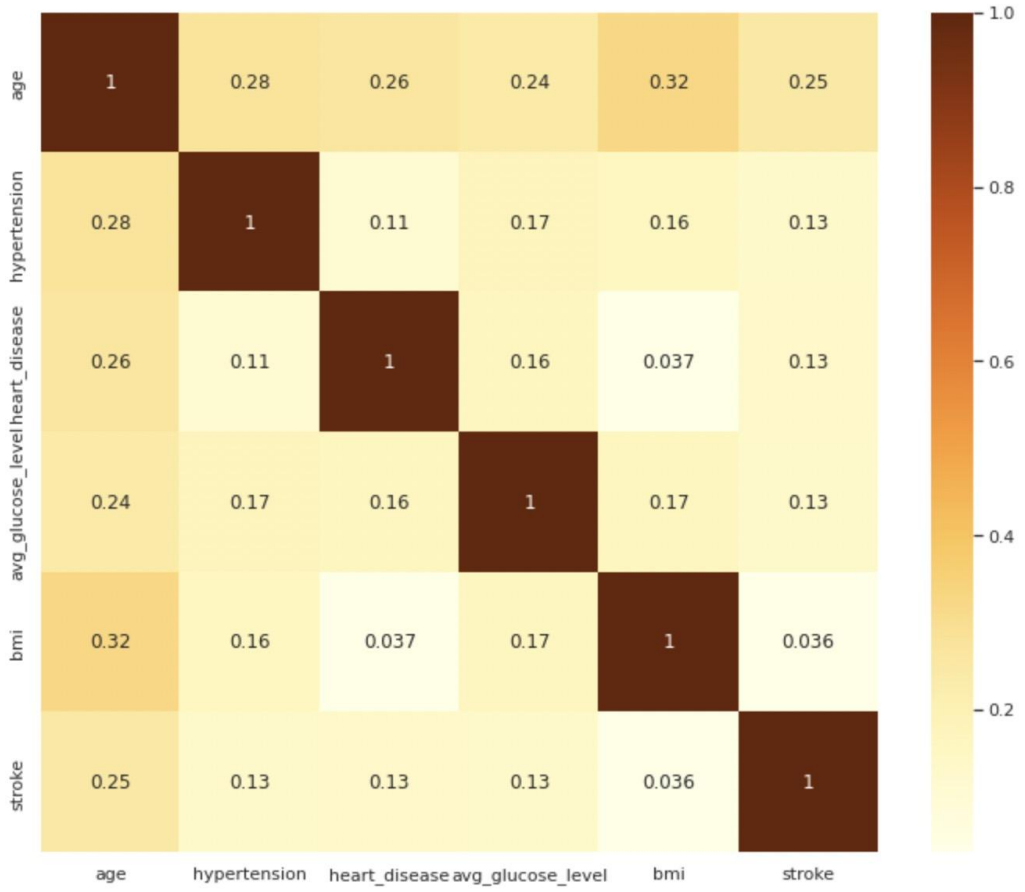


Fig. 2.4: Heatmap for correlation matrix of each numerical variable

2.5 Distribution of Numerical Variables

After observing the correlation between numerical variables and `stroke`, we want to see the distribution of those numerical variables, **['age', 'avg_glucose_level', 'bmi']**. We use histograms with a smooth silhouette line as Fig. 2.5 to visualize these distributions.

If we ignore extreme values in age, such as 0-year-old and 80-year-old, the distribution of age in this dataset could be regarded as an approximately normal distribution. The distribution of avg_glucose_level has two modes and is right-skewed. The distribution of BMI is similar to the distribution of avg_glucose_level, right-skewed, but only has one peak.

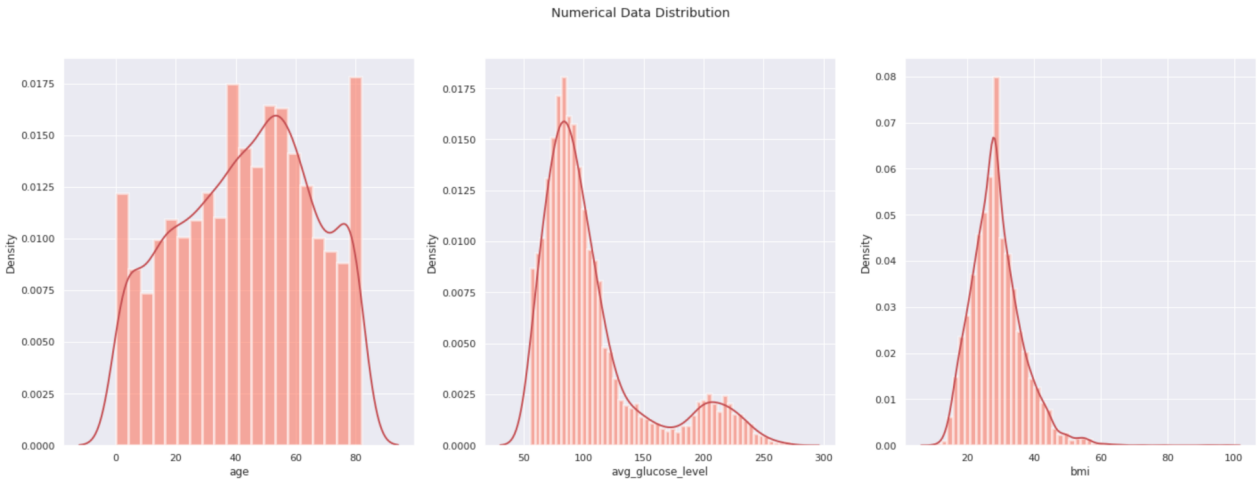
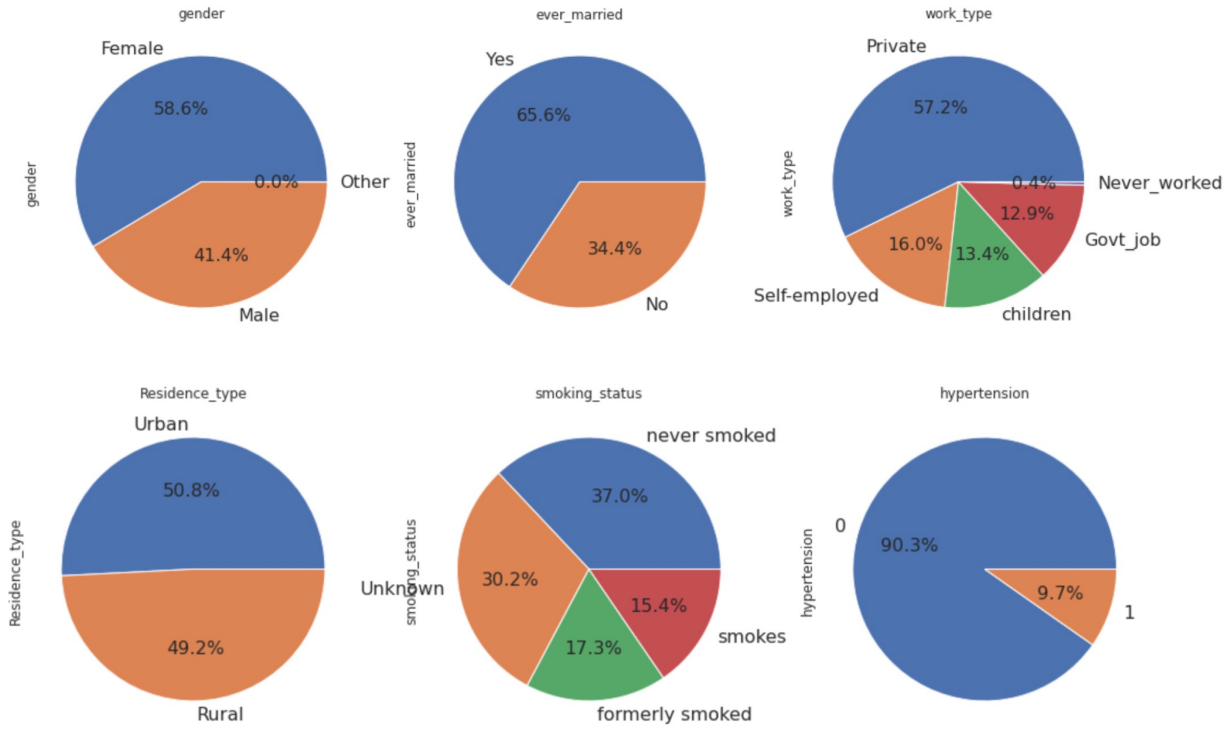


Fig. 2.5: Density histograms for each numerical variable

2.6 Distribution of Categorical Variables

We believe pie charts would be helpful for visualizing the distribution of categorical variables, **['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status', 'hypertension', 'heart_disease', 'stroke']**. And the results are shown in Fig. 2.6.



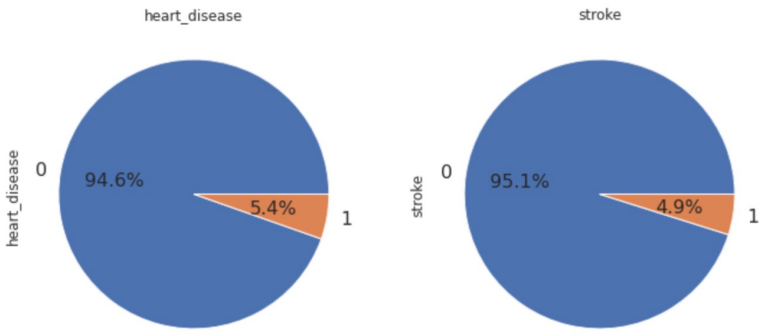


Fig.2.6: Pie charts for categorical variables

From the above 8 pie charts, Fig. 2.6, we can draw the following conclusions:

1. In this dataset, females account for 58.6% with only one record of "other" in the dataset. To avoid under-fitting, we will drop this record later.
2. 65.6% of observations are in the marital or have married before.
3. For "work_type", the outcome "private" accounts for more than half of the entire data (57.2%). What's more, we find observations who are children or adolescents might have work_type outcomes as "never_worked".
4. The two types of "residence_type", Urban or Rural, are separated approximately evenly.
5. For the "smoking_status", 37% of the observations have never smoked, 30.2% are unknown, 17.3% formerly smoked, and 15.4% smoke.
6. 0 indicates No, 1 indicates Yes, thus, 90.3% of observations do not have hypertension.
7. 94.6% of observations in this dataset do not have heart disease.
8. 95.1% of observations do not have strokes. This pie chart shows that, if we want to avoid the effects of this unbalanced dataset, we need to deal with this imbalance. The related part of this project would be discussed later.

2.7 Exploratory Analysis

Here histograms and boxplots are used to visually view and analyze the relationship between each numerical variable and our target variable "stoke".

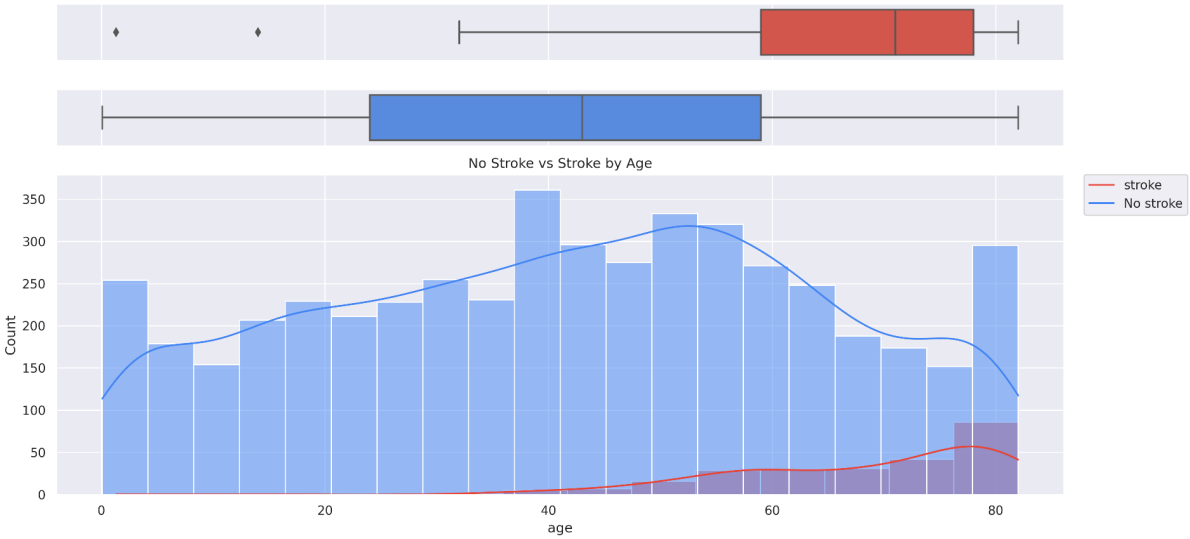


Fig. 2.7: No Stroke vs. Stroke by Age

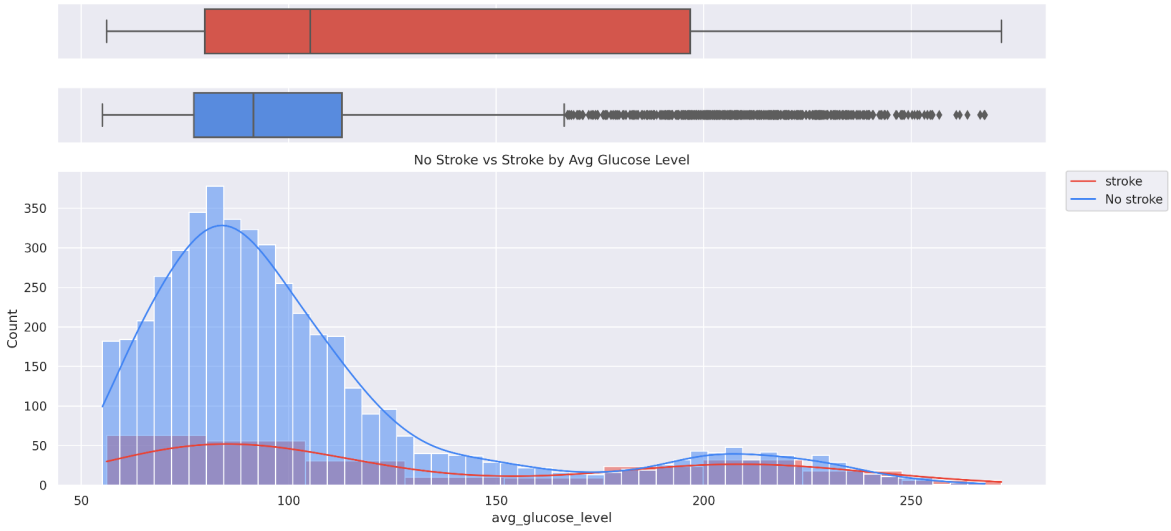


Fig. 2.8: No Stroke vs. Stroke by Average Glucose Level in Blood

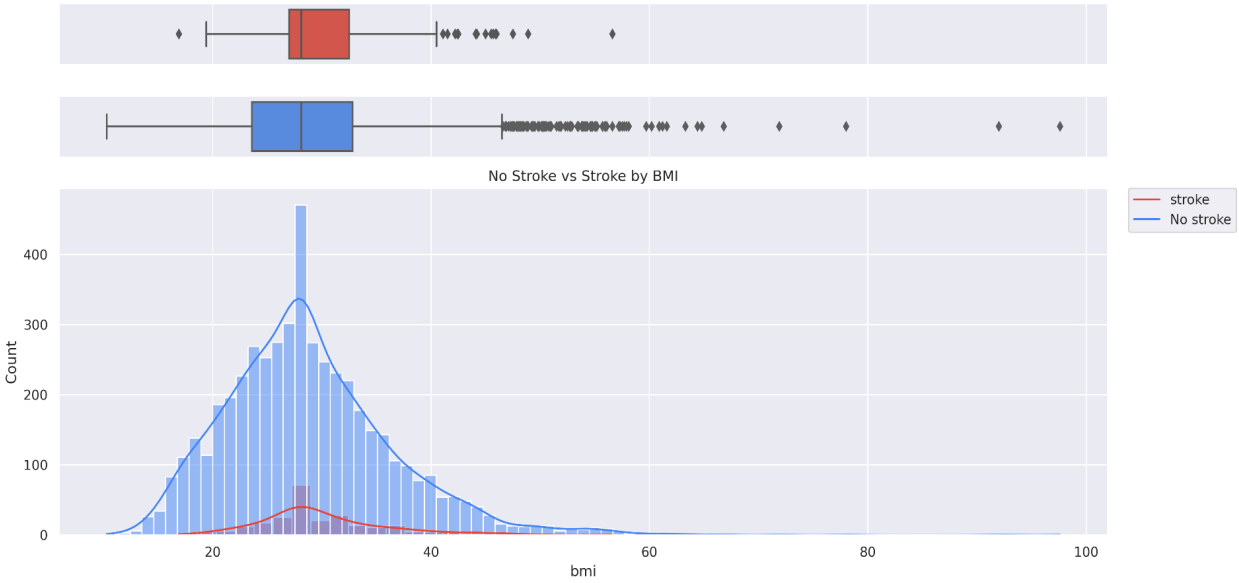


Fig. 2.9: No Stroke vs. Stroke by Body Mass Index

From Fig. 2.7, we observe that aged patients are more likely to have a stroke, and there exist some outliers in the "stroke" case.

From Fig. 2.8, we can derive that most people having a stroke usually have a higher average glucose level than those who didn't have a stroke. We could observe some outliers in the boxplot of the "no stroke" case.

Fig. 2.9 demonstrates that people who had a stroke would have a slightly higher BMI than those who didn't have a stroke. For this BMI feature, outliers exist in both "stroke" and "no stroke" cases.

Then we use the bar chart to help visualize the relationship between each categorical variable and our target variable "stoke".

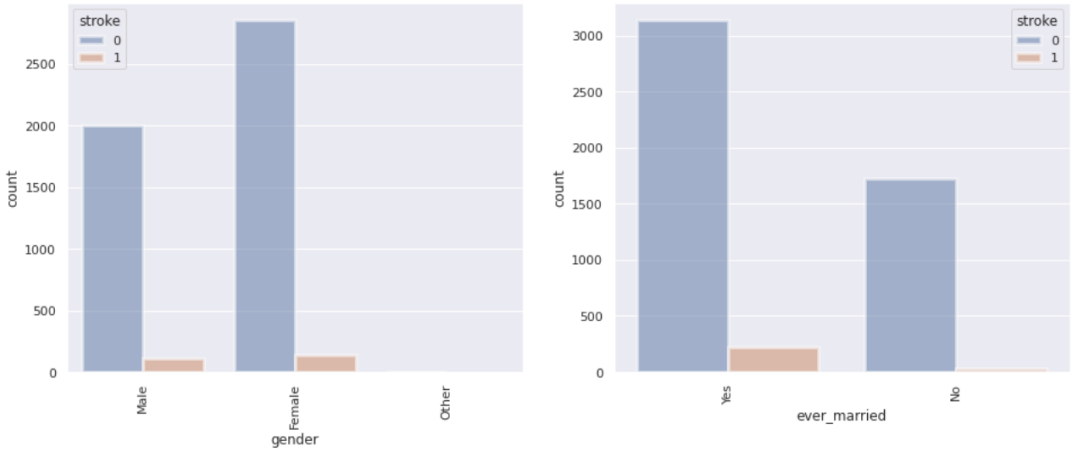


Fig. 2.10: The bar chart of gender, ever_married, grouped by `stroke`

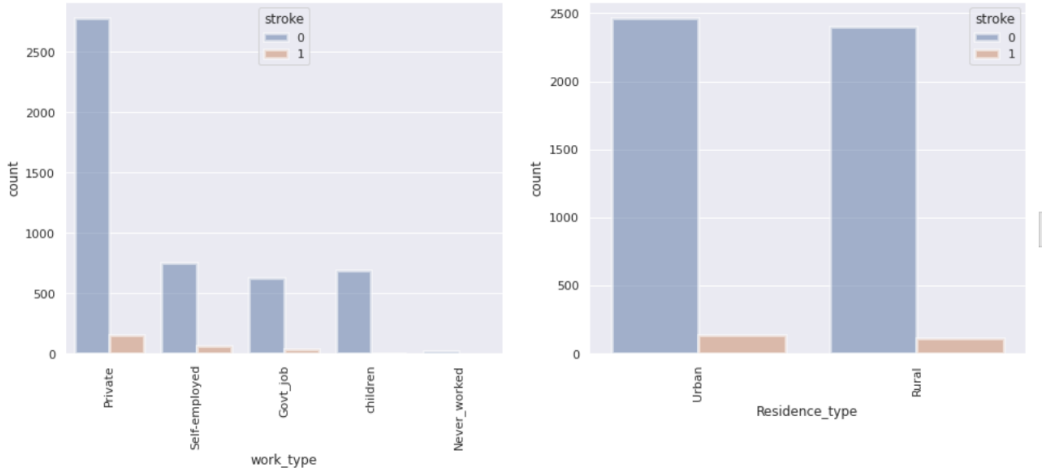
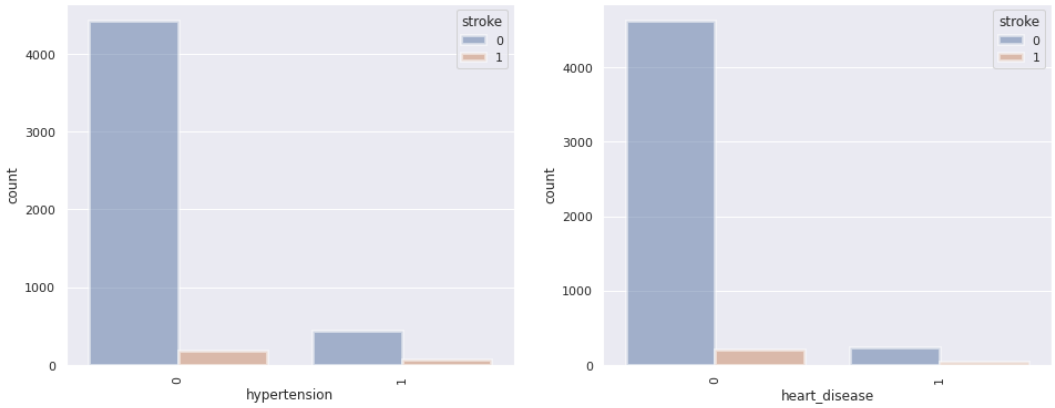


Fig. 2.11: The bar chart of work_type, Residence_type, grouped by the `stroke`



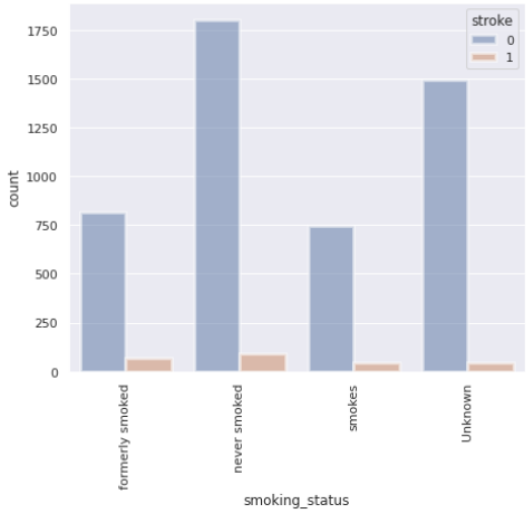


Fig. 2.12: The bar chart of smoking_status, hypertension, heart_disease, grouped by the `stroke`

From Fig. 2.10, Fig. 2.11, and Fig. 2.12, we could have the following observations:

- 141 females have a stroke, and 108 males also suffer from a stroke. It cannot be concluded that females would have higher probability to have a stroke than males because the number of females tested in this dataset is much higher than the number of males tested.
- In this dataset, most people who have a stroke are married or have married before. However, for the same reason as gender, this conclusion could not be concluded by just observing this bar chart.
- It seems that working at a private company would have a high probability of having a stroke.
- There is no relationship between residence type and stroke since the proportion of these two likelihoods is almost the same.
- From the smoking status, it looks like being a smoker doesn't seem to factor too much.

- The number of people with hypertension is considerably lower than the number of people without it. The number of persons who have had a stroke in the hypertension category is substantially higher than the number of people who have had a stroke in the non-hypertension category. However, because the number of people who had hypertension was significantly fewer than the number of people who didn't, it's difficult to say definitively that people with hypertension are more likely to have a stroke than people who don't.
- The number of persons who have heart disease is significantly less than the number who do not. The proportion of patients who suffered a stroke in the heart disease category is much higher than in the No heart disease category. Since heart_disease would have the same problem as hypertension, the imbalanced distribution, it's impossible to say with certainty that people with heart disease are more likely to have a stroke than people who don't.

We can observe the rough distribution of strokes from the above. Meanwhile, at this stage we find that the distributions of the patient having a stroke are not clear enough in each plot due to that the original dataset now is unbalanced and the results of "0" occupied about 19.5 times the results of "1". We will make it balanced in the following Dealing with Imbalanced Dataset section.

2.8 Encoding & Data cleaning

As we observed that there exist three types of values in the "gender" column in Section 2.3 Check Uniqueness, here we drop the "other" in values, since this is the only special case in observations in gender.

From the original dataset, we have many categorical variables which have certain important associations with the results of whether a patient is having a stroke. Therefore, we need to encode these categorical columns into numerical values as shown below, in Table 2.4, after encoding all the categorical features into integers.

id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
9046	1	67.0	0	1	1	2	1	228.69	36.6	1	1
51676	0	61.0	0	0	1	3	0	202.21	28.1	2	1
31112	1	80.0	0	1	1	2	0	105.92	32.5	2	1
60182	0	49.0	0	0	1	2	1	171.23	34.4	3	1
1665	0	79.0	1	0	1	3	0	174.12	24.0	2	1

Table 2.4: The dataset after encoding, changing each categorical feature to integers

- Gender: Female=0 Male=1
- Ever_married: No=0 Yes=1
- work_type: Govt_job=0 Never_worked=1 Private=2 Self-employed=3 children=4
- Residence_type: Rural=0 Urban=1
- smoking_status: Unknown=0 formerly smoked=1 never smoked=2 smokes=3

2.9 Dealing with Imbalanced Dataset

Before we deal with the imbalance, we want to check the size of having a stroke and not having a stroke. There are 249 observations that have a stroke, and 4860 observations that do not have a stroke. This original dataset is imbalanced since the samples possessing stroke labels of "0" are about 19.5 times larger than the stroke labels of "1". Due to our research interests in the stroke samples, we first make sure to keep all the results of stroke existing and undersampling the non-stroke amount to create a balanced sample. The methods we use to make it balanced will always keep the new dataset with an equal number of "1" and "0" levels of stroke samples.

We begin with the pure undersampling method that randomly chose 249 data from a subset (stroke = 0) to construct a balanced dataset (498 data in total) shown in Table 2.5.

After undersampling: Counter({0: 249, 1: 249})

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	12.0	0	0	0	4	1	67.06	16.1	0	0
1	0	52.0	0	0	1	0	1	80.88	23.8	3	0
2	0	81.0	1	0	1	0	1	216.07	43.4	2	0
3	1	42.0	0	0	1	2	1	68.24	33.1	1	0
4	0	61.0	0	0	1	2	0	71.40	29.2	1	0
...	...	...	...	...	...	...	...	...	...	...	...
493	1	57.0	0	0	1	2	0	84.96	36.7	0	1
494	0	14.0	0	0	0	4	0	57.93	30.9	0	1
495	0	75.0	0	0	1	3	0	78.80	29.3	1	1
496	1	71.0	1	0	1	3	0	87.80	28.1	0	1
497	0	78.0	0	0	1	2	0	78.81	19.6	0	1

498 rows × 11 columns

Table 2.5: The small balanced sample by using undersampling, with size 498*11

By using the pure oversampling method to expand, the stroke=1 amount is correlated to the non-stroke one to create a balanced sample with more data values (9720 data in total) shown in Table 2.6.

After oversampling: Counter({1: 4860, 0: 4860})

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.000000	0	1	1	2	1	228.690000	36.600000	1	1
1	0	61.000000	0	0	1	3	0	202.210000	28.100000	2	1
2	1	80.000000	0	1	1	2	0	105.920000	32.500000	2	1
3	0	49.000000	0	0	1	2	1	171.230000	34.400000	3	1
4	0	79.000000	1	0	1	3	0	174.120000	24.000000	2	1
...	...	...	...	...	...	...	...	...	...	...	...
9715	1	69.484878	0	0	1	3	1	195.200907	28.845488	2	1
9716	0	78.000000	0	0	1	2	0	109.144693	31.062152	1	1
9717	0	71.807701	0	0	1	2	0	193.460552	35.069241	1	1
9718	0	61.737350	0	0	1	0	0	206.147048	41.852771	1	1
9719	0	79.000000	0	1	1	2	0	129.029493	24.402077	1	1

9720 rows × 11 columns

Table 2.6: The large balanced sample by using oversampling, with size 9720*11

The main purpose of the pure undersampling method is to reduce the majority of data, while the pure oversampling method is to create the minority samples. Both under-sampling and oversampling change the distribution of original data to a certain extent, which may lead to the overfitting of the model. Therefore, we test several times to compare the accuracy of these two methods and derive that the undersampling method is much better and with higher accuracy.

Fig. 2.13 and Fig. 2.14 give the AUC score with respect to undersampling and oversampling.

```
Before undersampling:  Counter({0: 2913, 1: 152})
After undersampling:  Counter({0: 152, 1: 152})
ROC AUC score for original data:  0.5
ROC AUC score for undersampled data:  0.7792003558210093
```

Fig. 2.13. The AUC scores before undersampling and after undersampling

```
Before oversampling:  Counter({0: 2909, 1: 156})
After oversampling:  Counter({0: 2909, 1: 2909})
ROC AUC score for original data:  0.5
ROC AUC score for oversampling data:  0.7405107940234674
```

Fig. 2.14. The AUC scores before oversampling and after oversampling

There are many other methods to create a balanced new sample as well, but most of them are just combining the undersampling and oversampling with different rates.

```
SMOTE oversampling rate:0.3, Random undersampling rate:0.7 , Mean ROC AUC: 0.840
SMOTE oversampling rate:0.3, Random undersampling rate:0.6 , Mean ROC AUC: 0.840
SMOTE oversampling rate:0.3, Random undersampling rate:0.5 , Mean ROC AUC: 0.840
SMOTE oversampling rate:0.4, Random undersampling rate:0.7 , Mean ROC AUC: 0.839
SMOTE oversampling rate:0.4, Random undersampling rate:0.6 , Mean ROC AUC: 0.840
SMOTE oversampling rate:0.4, Random undersampling rate:0.5 , Mean ROC AUC: 0.840
SMOTE oversampling rate:0.5, Random undersampling rate:0.7 , Mean ROC AUC: 0.839
SMOTE oversampling rate:0.5, Random undersampling rate:0.6 , Mean ROC AUC: 0.839
SMOTE oversampling rate:0.5, Random undersampling rate:0.5 , Mean ROC AUC: 0.838
```

Fig. 2.15. The results of combining undersampling and oversampling

We set a search test for the rates which shows that the proportion of undersampling is always higher than or equal to the one of oversampling (Fig. 2.15). Thus, we decide to just use the undersampling for comparatively higher accuracy and more efficiency with smaller data, and the comparison in cases with oversampling.

3 Feature Engineering

In this section, we apply several dimensionality reduction techniques we learned in Machine Learning courses, with the goal of simplifying and speeding up data transformations while also enhancing model accuracy.

Dimensionality reduction is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data. There are several dimensionality reduction methods that can be used with different types of data for different requirements. Generally, based on the resulting features' format, dimensionality reduction methods can be roughly divided into two categories: (1) creates a combination of new features, or (2) only keeps the most important features in the data. The latter one we would call the "feature selection method" in this report.

In this report, we use Principal Component Analysis (PCA) as a method of finding a new combination of features; meanwhile, we adopt different techniques based on three classes of feature selection methods. According to different combinations of algorithm and model construction, feature selection methods can be basically divided into three classes: filter method, wrapper method, and embedded method. The following Table 3.1 shows a simple comparison between these three methods of feature selection.

	Filter Method	Wrapper Method	Embedded Method
Explanation	Uses proxy measure	Uses predictive model	Selects features during model building
Example	Correlation, Chi-square test, (Tree) Information gain	Stepwise selection, Forward selection, Backward elimination	Regularization methods, such as LASSO, Ridge regression
Speed	Computationally faster	Slower	Medium
Overfitting	Avoids overfitting	Prone to overfitting	Less prone to overfitting

Table 3.1 Comparison of three methods of feature selection

3.1 Principal component analysis (PCA)

Principal component analysis is a linear dimensionality reduction algorithm using Singular Value Decomposition (SVD) of the data to project it to a lower-dimensional space. It transforms a set of correlated variables (p) into a smaller k (k<p) number of uncorrelated variables called principal components while retaining as much of the variation in the original dataset as possible. PCA is known as an unsupervised machine learning algorithm that is used for dimensionality reduction.

We perform the PCA on the normalized matrix of X_i, which is normalized using its corresponding means μ_i and standard deviation σ_i. The following Fig 3.1 and Fig 3.2 are the scree plots we generated from PCA both on undersampling and oversampling datasets. A scree plot is used to help select the components which explain the most variability in the dataset. Here we find that the first five principal components can explain over 85% of the total variance: 87.2% for undersampling and 90.7% for oversampling.

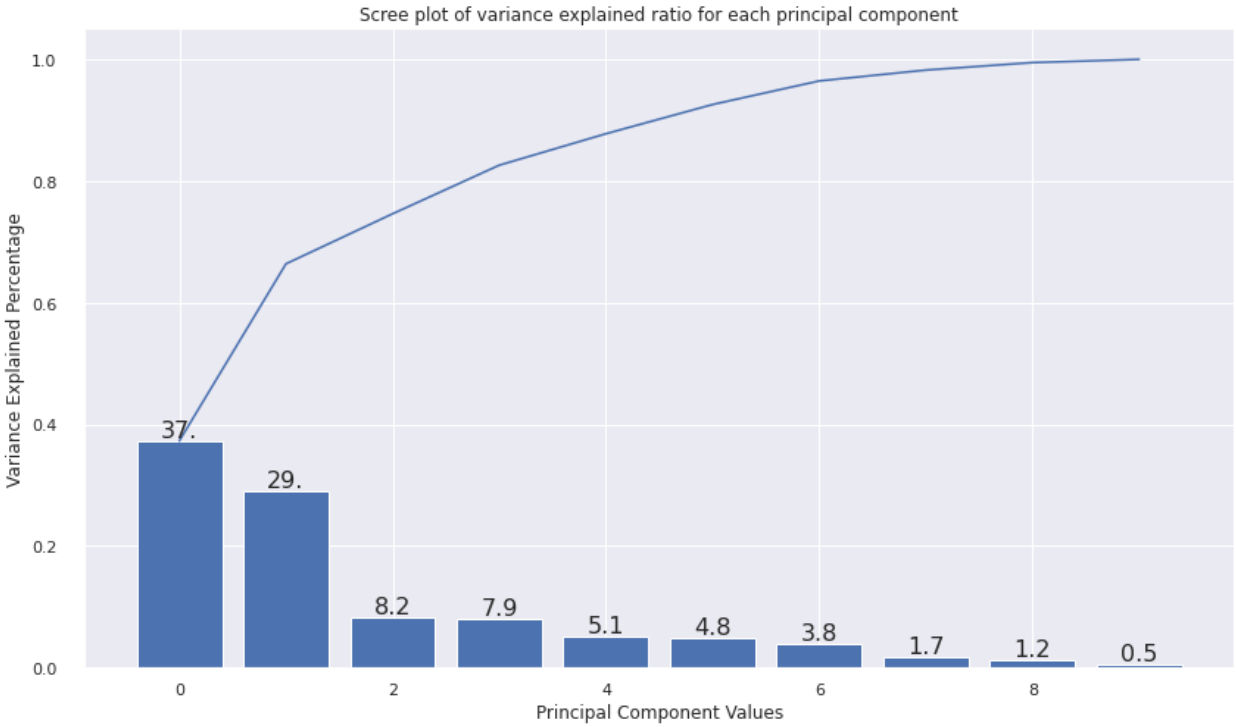


Fig 3.1 Scree plot of variance explained ratio for each principal component based on undersampling data

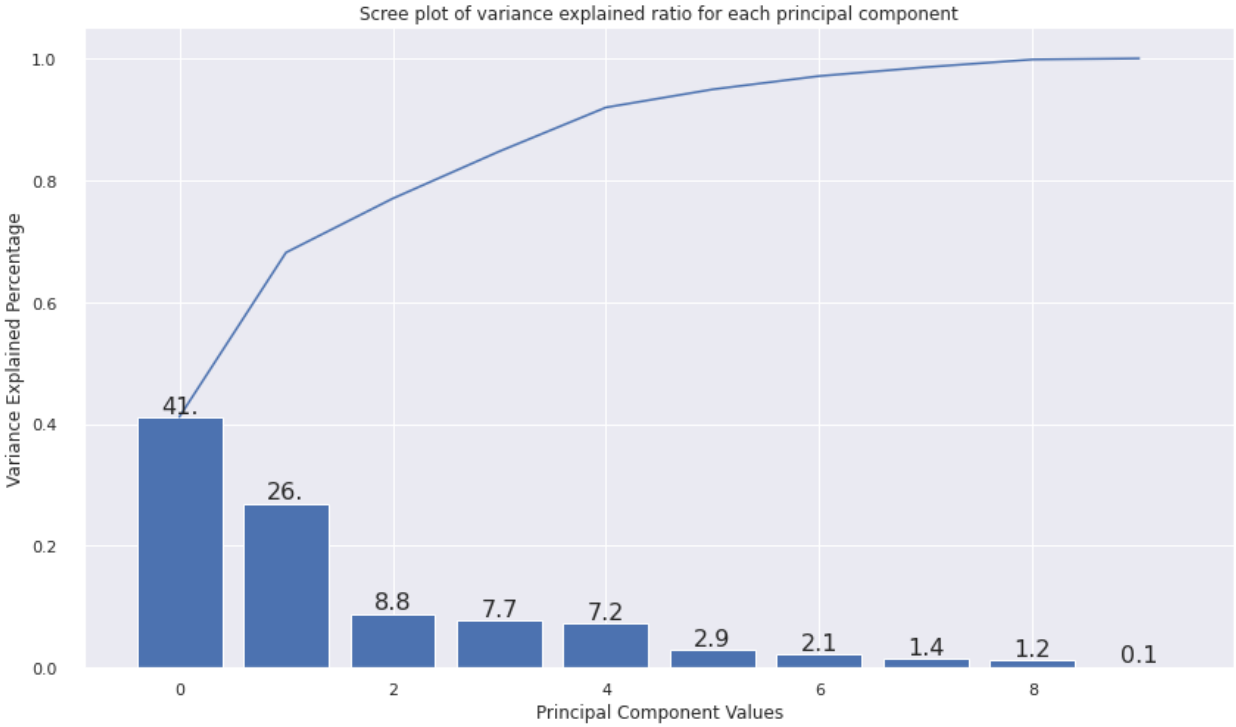


Fig 3.2 Scree plot of variance explained ratio for each principal component based on oversampling data

3.2 Feature Selection

3.2.1 Filter Method

Filter feature selection methods apply statistical tests that can be used to score features' correlation with the output variable, and then features are ranked and selected on the basis of their scores. For this part, we choose to use the univariate selection.

Univariate Selection

After data cleaning, the dataset does not contain constant features and it contains 10 features and 1 target variable. We perform the feature selection on both the undersampling and oversampling datasets. Since our target variable is categorical, we plan to use these two score methods to test the dataset: chi-squared stats and ANOVA f-value. The first method we used performed a chi-square test, which estimates the degree of dependence between features and target variable, on the training set. The second method we used calculated the ANOVA f-value, which represents the ratio between the explained and the unexplained variance, as a measure of how informative each feature is for the dataset.

ss_Features	chi2 Score	ss_Features	f-Score	ls_Features	chi2 Score	ls_Features	f-Score
age	1716.24	age	287.02	age	29958.50	age	5170.53
avg_glucose_level	586.18	ever_married	31.30	avg_glucose_level	15092.04	avg_glucose_level	629.52
heart_disease	20.76	heart_disease	24.62	gender	250.31	work_type	474.37
hypertension	18.47	hypertension	23.58	work_type	220.50	gender	383.64
ever_married	6.41	avg_glucose_level	23.54	Residence_type	216.14	Residence_type	381.14

Table 3.2 Comparison of Univariate selection

Table 3.2 shows the top five features and scores under different sample subsets and different scores methods. We aim to select features that are highly dependent on the response, so we

manually set the threshold of the scores and retrieve the top five features with the highest scores. We find that the results of both methods are consistent for different sample subsets.

- Undersampling with chi-square score and f-score: ['age', 'avg_glucose_level', 'heart_disease', 'hypertension', 'ever_married']
- Oversampling with chi-square score and f-score: ['age', 'avg_glucose_level', 'gender', 'work_type', 'Residence_type']

3.2.2 Wrapper Method

Wrapper feature selection method based on a specific machine learning algorithm that we are trying to fit on the dataset. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion. For our dataset, we use accuracy as the evaluation criterion since we are dealing with a classification problem. For this part, we choose to apply the Recursive feature elimination algorithm with logistic regression to our dataset.

Recursive Feature Elimination with Cross-Validation (RFECV)

We use the Recursive Feature Elimination with Cross-Validation feature selection technique to select the best subset of features based on the accuracy of the logistic regression model. This technique eliminates the features from a model by fitting the model multiple times and at each step, removing the weakest features. From Fig 3.3, for the undersampling data, selecting 6 features is reasonable enough compared to the similar accuracy of the full features. As well for the oversampling data, Fig 3.4 shows that selecting 8 features is also reasonable enough to obtain approximately similar accuracy to the full features but with a reduced set of features.

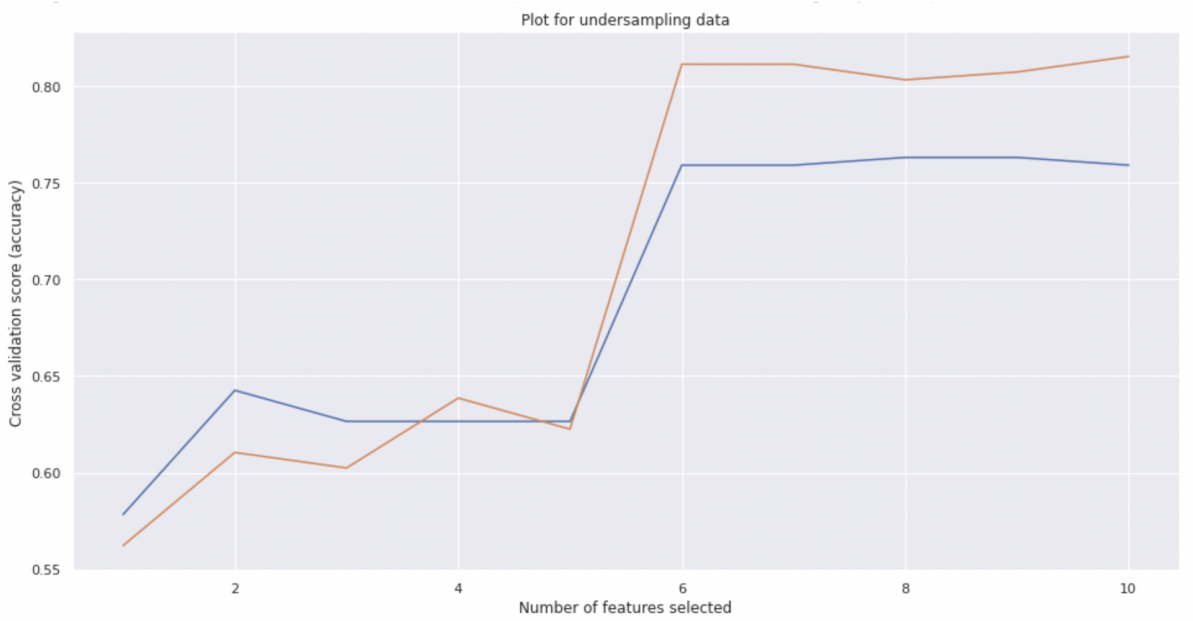


Fig 3.3 Cross-validation score based on undersampling data

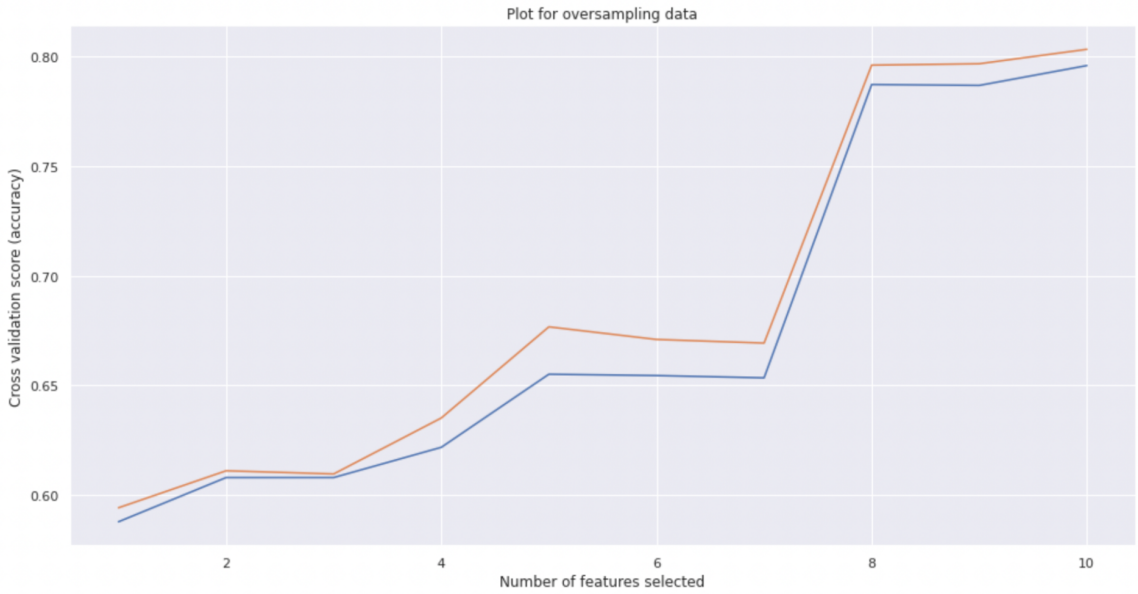


Fig 3.4 Cross-validation score based on oversampling data

- Undersampling with RFECV: ['gender', 'age', 'hypertension', 'heart_disease', 'ever_married', 'work_type']
- Oversampling with RFECV: ['gender', 'age', 'hypertension', 'heart_disease', 'ever_married', 'work_type', 'Residence_type', 'avg_glucose_level']

3.2.3 Embedded Method

Embedded feature selection methods are implemented by algorithms that have their own built-in feature selection methods. In this part, we choose to use two popular methods: Lasso and Random forest feature importance.

Lasso

Least Absolute Shrinkage and Selection Operator (Lasso) is one of the powerful methods that help perform regularization with the L1 penalty, so it can shrink features' coefficients to zero. It helps us to reduce the overfitting problem. We scale our data in advance and optimize the hyperparameter of Lasso regression using Grid Search. Finally, we get the values of the coefficients of Lasso regression and choose features with non-zero coefficients.

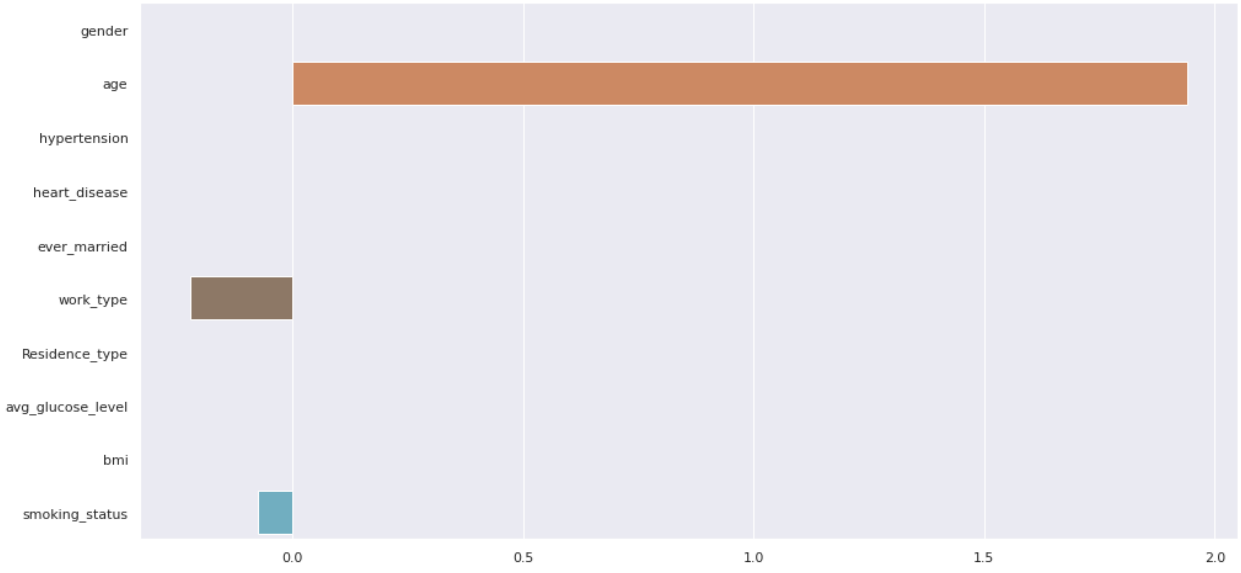


Fig 3.5 Lasso coefficients in barplot based on undersampling data

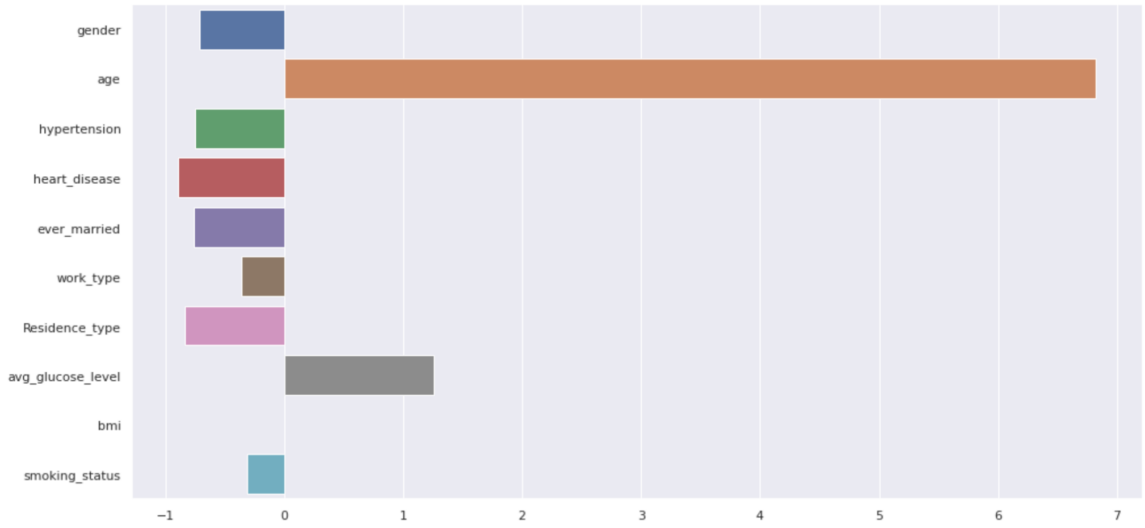


Fig 3.6 Lasso coefficients in barplot based on oversampling data

From Fig 3.5 and 3.6, we can choose the feature subsets:

- Undersampling with Lasso: ['age', 'work_type', 'smoking_status']
- Oversampling with Lasso: ['gender', 'age', 'hypertension', 'heart_disease', 'ever_married', 'work_type', 'Residence_type', 'avg_glucose_level', 'smoking_status']

Random forest importance

Using a random forest algorithm, the average impurity decrease computed from all decision trees in the forest measured the feature importance. The advantage of it is that no matter whether the data are linearly or nonlinearly (linearly inseparable), the method still applies. From Fig 3.7 and 3.8, we can see each feature's degree of importance. We manually keep the top 5 features with the highest feature importance as our favored feature subsets. Moreover, we can see from Fig 3.7 and 3.8, that the feature subsets contain the same features for both undersampling and oversampling data, and they are ['age', 'bmi', 'avg_glucose_level', 'smoking_status', 'work_type'].

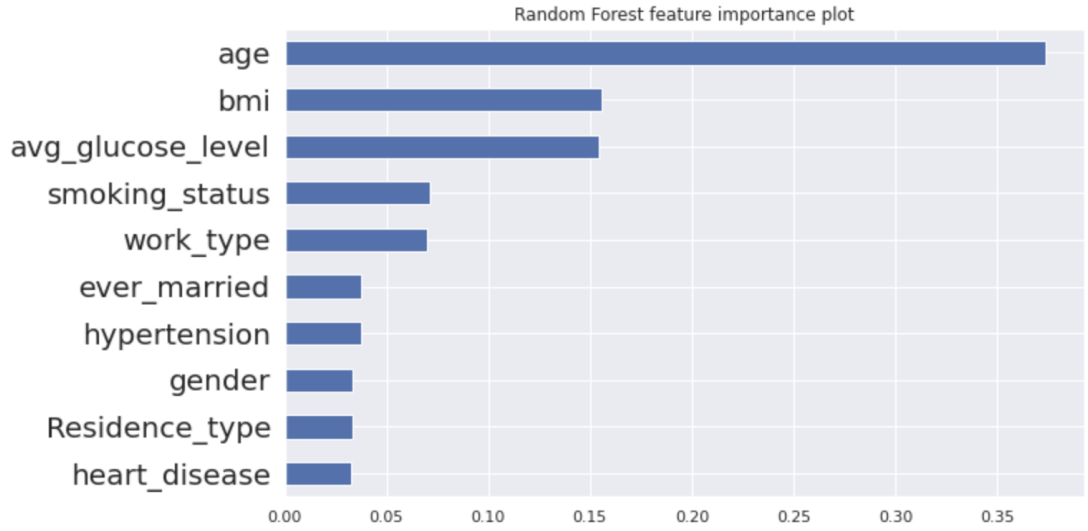


Fig 3.7 Random forest feature importance barplot based on undersampling data

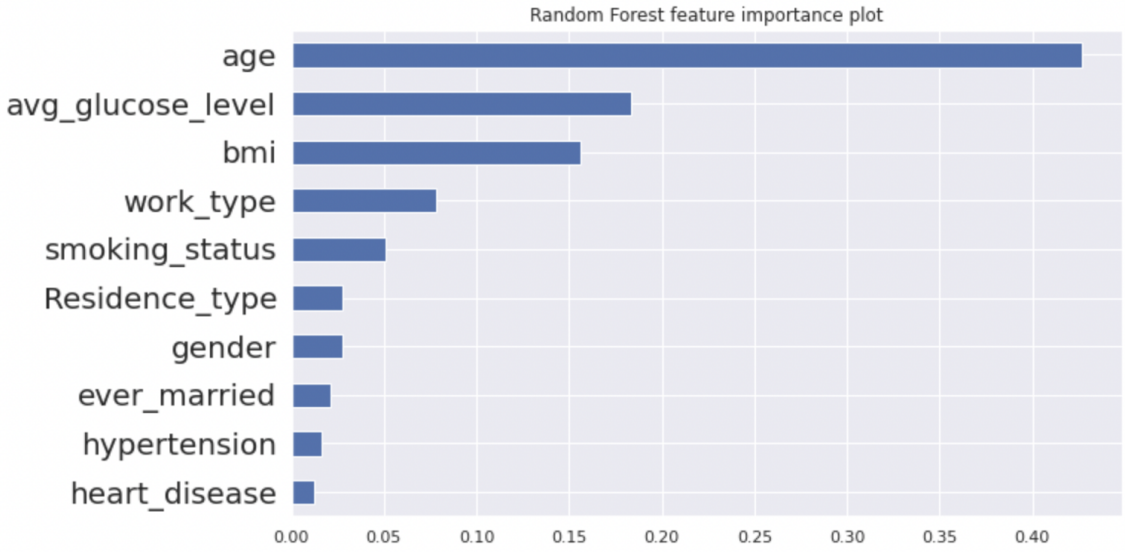


Fig 3.8 Random forest feature importance barplot based on oversampling data

3.3 Feature Selection Summary

This section summarizes the results of the previous analysis of dimensionality reduction methods.

1. **PCA**: The first five components can explain over 85% of the total variance for both undersampling and oversampling datasets. Hence, we use the first five principal components for predictive modeling of stroke occurrence.
2. **Univariate Selection**
    a. Undersampling with chi-square score and f-score: ['age', 'avg_glucose_level', 'heart_disease', 'hypertension', 'ever_married']
    b. Oversampling with chi-square score and f-score: ['age', 'avg_glucose_level', 'gender', 'work_type', 'Residence_type']
3. **Recursive Feature Elimination with Cross_Validation (RFECV)**
    a. Undersampling with RFECV: ['gender', 'age', 'hypertension', 'heart_disease', 'ever_married', 'work_type']
    b. Oversampling with RFECV: ['gender', 'age', 'hypertension', 'heart_disease', 'ever_married', 'work_type', 'Residence_type', 'avg_glucose_level']
4. **Lasso**
    a. Undersampling with Lasso: ['age', 'work_type', 'smoking_status']
    b. Oversampling with Lasso: ['gender', 'age', 'hypertension', 'heart_disease', 'ever_married', 'work_type', 'Residence_type', 'avg_glucose_level', 'smoking_status']
5. **Random Forest Importance:** ['age', 'bmi', 'avg_glucose_level', 'smoking_status', 'work_type']

Next step, we are going to try a range of different models fit on different subsets of features chosen via different statistical measures and discover what works best for our dataset.

4 Model Selection

In this section, we have two main tasks. Firstly, we hope to decide which machine learning method could yield the best result when trying to predict stroke. Secondly, based on the results from Section 3, we hope to find out the best feature subset which could produce the most accurate prediction.

To do the comparison, we firstly develop 6 machine learning methods on all 11 variables to see the base accuracy of each method. The 6 methods are Logistic Regression, Decision Tree, K Nearest Neighborhood (KNN), Support Vector Machine (SVM), AdaBoost, and Naive Bayes. Secondly, we adopt these methods on data that have been dimensionally reduced by the 5 feature engineering methods in Section 3. Therefore, we would have 36 models in total, for each of the 6 variable sets have used 6 methods to do the training.

As for the dataset, we chose to use the undersampling data from Section 2 which contains 249 observations, as it is a balanced dataset with better performance than oversampling data. For accuracy calculations, we adopt the “accuracy_score” function from the “metrics” package in Python to calculate the difference between the test y value and the predicted y value.

In predictive analytics, a table of confusion (sometimes also called a confusion matrix) is a table with two rows and two columns that reports the number of true positives, false negatives, false positives, and true negatives. This allows for a more detailed analysis than simply observing the proportion of correct classifications (accuracy). So, we produce confusion matrices for each one of the models to help us gain more information about the precision of models and the choice of feature subsets.

4.1 Logistic Regression

LR Accuracy: 0.824

	precision	recall	f1-score	support
0	0.86	0.77	0.81	62
1	0.80	0.87	0.83	63
accuracy			0.82	125
macro avg	0.83	0.82	0.82	125
weighted avg	0.83	0.82	0.82	125

PCA Accuracy: 0.584

	precision	recall	f1-score	support
0	0.58	0.58	0.58	62
1	0.59	0.59	0.59	63
accuracy			0.58	125
macro avg	0.58	0.58	0.58	125
weighted avg	0.58	0.58	0.58	125

Lasso Accuracy: 0.784

	precision	recall	f1-score	support
0	0.83	0.71	0.77	62
1	0.75	0.86	0.80	63
accuracy			0.78	125
macro avg	0.79	0.78	0.78	125
weighted avg	0.79	0.78	0.78	125

Recursive Accuracy: 0.672

	precision	recall	f1-score	support
0	0.70	0.60	0.64	62
1	0.65	0.75	0.70	63
accuracy			0.67	125
macro avg	0.68	0.67	0.67	125
weighted avg	0.68	0.67	0.67	125

Univariate Accuracy: 0.816

	precision	recall	f1-score	support
0	0.85	0.76	0.80	62
1	0.79	0.87	0.83	63
accuracy			0.82	125
macro avg	0.82	0.82	0.82	125
weighted avg	0.82	0.82	0.82	125

Tree Accuracy: 0.672

	precision	recall	f1-score	support
0	0.70	0.60	0.64	62
1	0.65	0.75	0.70	63
accuracy			0.67	125
macro avg	0.68	0.67	0.67	125
weighted avg	0.68	0.67	0.67	125

Table 4.1 Results for logistic regression based on different feature subsets

The above Table 4.1 shows the accuracy of different feature subsets when adopting logistic regression. We can see that logistic regression is giving us the best result and PCA is giving us the worst result.

4.2 Decision Tree

DecisionTree Accuracy: 0.736

	precision	recall	f1-score	support
0	0.75	0.71	0.73	62
1	0.73	0.76	0.74	63
accuracy			0.74	125
macro avg	0.74	0.74	0.74	125
weighted avg	0.74	0.74	0.74	125

PCA Accuracy: 0.56

	precision	recall	f1-score	support
0	0.55	0.66	0.60	62
1	0.58	0.46	0.51	63
accuracy			0.56	125
macro avg	0.56	0.56	0.56	125
weighted avg	0.56	0.56	0.56	125

Lasso Accuracy: 0.728

	precision	recall	f1-score	support
0	0.72	0.74	0.73	62
1	0.74	0.71	0.73	63
accuracy			0.73	125
macro avg	0.73	0.73	0.73	125
weighted avg	0.73	0.73	0.73	125

Recursive Accuracy: 0.656

	precision	recall	f1-score	support
0	0.64	0.69	0.67	62
1	0.67	0.62	0.64	63
accuracy			0.66	125
macro avg	0.66	0.66	0.66	125
weighted avg	0.66	0.66	0.66	125

Univariate Accuracy: 0.688

	precision	recall	f1-score	support
0	0.69	0.66	0.68	62
1	0.68	0.71	0.70	63
accuracy			0.69	125
macro avg	0.69	0.69	0.69	125
weighted avg	0.69	0.69	0.69	125

Tree Accuracy: 0.656

	precision	recall	f1-score	support
0	0.64	0.69	0.67	62
1	0.67	0.62	0.64	63
accuracy			0.66	125
macro avg	0.66	0.66	0.66	125
weighted avg	0.66	0.66	0.66	125

Table 4.2 Results for Decision Tree based on different feature subsets

The above Table 4.2 shows the accuracy of different feature sets when adopting Decision Tree. We can see that the full model is giving us the best result and PCA is still giving us the worst result.

4.3 K Nearest Neighborhood

KNN Accuracy: 0.64

	precision	recall	f1-score	support
0	0.67	0.53	0.59	62
1	0.62	0.75	0.68	63
accuracy			0.64	125
macro avg	0.65	0.64	0.64	125
weighted avg	0.65	0.64	0.64	125

PCA Accuracy: 0.576

	precision	recall	f1-score	support
0	0.60	0.42	0.50	62
1	0.56	0.73	0.63	63
accuracy			0.58	125
macro avg	0.58	0.57	0.56	125
weighted avg	0.58	0.58	0.57	125

Lasso Accuracy: 0.744

	precision	recall	f1-score	support
0	0.80	0.65	0.71	62
1	0.71	0.84	0.77	63
accuracy			0.74	125
macro avg	0.75	0.74	0.74	125
weighted avg	0.75	0.74	0.74	125

Recursive Accuracy: 0.624

	precision	recall	f1-score	support
0	0.63	0.60	0.61	62
1	0.62	0.65	0.64	63
accuracy			0.62	125
macro avg	0.62	0.62	0.62	125
weighted avg	0.62	0.62	0.62	125

Univariate Accuracy: 0.776

	precision	recall	f1-score	support
0	0.84	0.68	0.75	62
1	0.73	0.87	0.80	63
accuracy			0.78	125
macro avg	0.79	0.78	0.77	125
weighted avg	0.79	0.78	0.77	125

Tree Accuracy: 0.624

	precision	recall	f1-score	support
0	0.63	0.60	0.61	62
1	0.62	0.65	0.64	63
accuracy			0.62	125
macro avg	0.62	0.62	0.62	125
weighted avg	0.62	0.62	0.62	125

Table 4.3 Results for KNN based on different feature subsets

The above Table 4.3 shows the accuracy of different feature sets when adopting the KNN method. We can see that Univariate selection is giving us the best result and PCA is giving us the worst result.

4.4 Support Vector Machine

SVM Accuracy: 0.808

	precision	recall	f1-score	support
0	0.90	0.69	0.78	62
1	0.75	0.92	0.83	63
accuracy			0.81	125
macro avg	0.82	0.81	0.81	125
weighted avg	0.82	0.81	0.81	125

PCA Accuracy: 0.504

	precision	recall	f1-score	support
0	0.50	0.39	0.44	62
1	0.51	0.62	0.56	63
accuracy			0.50	125
macro avg	0.50	0.50	0.50	125
weighted avg	0.50	0.50	0.50	125

Lasso Accuracy: 0.768

	precision	recall	f1-score	support
0	0.84	0.66	0.74	62
1	0.72	0.87	0.79	63
accuracy			0.77	125
macro avg	0.78	0.77	0.77	125
weighted avg	0.78	0.77	0.77	125

Recursive Accuracy: 0.696

	precision	recall	f1-score	support
0	0.69	0.69	0.69	62
1	0.70	0.70	0.70	63
accuracy			0.70	125
macro avg	0.70	0.70	0.70	125
weighted avg	0.70	0.70	0.70	125

Univariate Accuracy: 0.824

	precision	recall	f1-score	support
0	0.88	0.74	0.81	62
1	0.78	0.90	0.84	63
accuracy			0.82	125
macro avg	0.83	0.82	0.82	125
weighted avg	0.83	0.82	0.82	125

Tree Accuracy: 0.696

	precision	recall	f1-score	support
0	0.69	0.69	0.69	62
1	0.70	0.70	0.70	63
accuracy			0.70	125
macro avg	0.70	0.70	0.70	125
weighted avg	0.70	0.70	0.70	125

Table 4.4 Results for SVM based on different feature subsets

The above Table 4.4 shows the accuracy of different feature sets when adopting SVM. We can see that Univariate selection is giving us the best result and PCA is still giving us the worst result.

4.5 AdaBoost

ADA Accuracy: 0.752

	precision	recall	f1-score	support
0	0.78	0.69	0.74	62
1	0.73	0.81	0.77	63
accuracy			0.75	125
macro avg	0.76	0.75	0.75	125
weighted avg	0.75	0.75	0.75	125

PCA Accuracy: 0.536

	precision	recall	f1-score	support
0	0.53	0.53	0.53	62
1	0.54	0.54	0.54	63
accuracy			0.54	125
macro avg	0.54	0.54	0.54	125
weighted avg	0.54	0.54	0.54	125

Lasso Accuracy: 0.776

	precision	recall	f1-score	support
0	0.81	0.71	0.76	62
1	0.75	0.84	0.79	63
accuracy			0.78	125
macro avg	0.78	0.78	0.77	125
weighted avg	0.78	0.78	0.77	125

Recursive Accuracy: 0.672

	precision	recall	f1-score	support
0	0.70	0.60	0.64	62
1	0.65	0.75	0.70	63
accuracy			0.67	125
macro avg	0.68	0.67	0.67	125
weighted avg	0.68	0.67	0.67	125

Univariate Accuracy: 0.776

	precision	recall	f1-score	support
0	0.84	0.68	0.75	62
1	0.73	0.87	0.80	63
accuracy			0.78	125
macro avg	0.79	0.78	0.77	125
weighted avg	0.79	0.78	0.77	125

Tree Accuracy: 0.672

	precision	recall	f1-score	support
0	0.70	0.60	0.64	62
1	0.65	0.75	0.70	63
accuracy			0.67	125
macro avg	0.68	0.67	0.67	125
weighted avg	0.68	0.67	0.67	125

Table 4.5 Results for AdaBoost based on different feature subsets

The above Table 4.5 shows the accuracy of different feature sets when adopting AdaBoost. We can see that Lasso and Univariate selection are both giving us the best result and PCA is still giving us the worst result.

4.6 Naive Bayes

GaussianNB Accuracy: 0.768

	precision	recall	f1-score	support
0	0.78	0.69	0.74	62
1	0.73	0.81	0.77	63
accuracy			0.75	125
macro avg	0.76	0.75	0.75	125
weighted avg	0.75	0.75	0.75	125

PCA Accuracy: 0.616

	precision	recall	f1-score	support
0	0.61	0.63	0.62	62
1	0.62	0.60	0.61	63
accuracy			0.62	125
macro avg	0.62	0.62	0.62	125
weighted avg	0.62	0.62	0.62	125

Lasso Accuracy: 0.776

	precision	recall	f1-score	support
0	0.89	0.63	0.74	62
1	0.72	0.92	0.81	63
accuracy			0.78	125
macro avg	0.80	0.77	0.77	125
weighted avg	0.80	0.78	0.77	125

Recursive Accuracy: 0.648

	precision	recall	f1-score	support
0	0.60	0.87	0.71	62
1	0.77	0.43	0.55	63
accuracy			0.65	125
macro avg	0.69	0.65	0.63	125
weighted avg	0.69	0.65	0.63	125

Univariate Accuracy: 0.824

	precision	recall	f1-score	support
0	0.82	0.82	0.82	62
1	0.83	0.83	0.83	63
accuracy			0.82	125
macro avg	0.82	0.82	0.82	125
weighted avg	0.82	0.82	0.82	125

Tree Accuracy: 0.648

	precision	recall	f1-score	support
0	0.60	0.87	0.71	62
1	0.77	0.43	0.55	63
accuracy			0.65	125
macro avg	0.69	0.65	0.63	125
weighted avg	0.69	0.65	0.63	125

Table 4.6 Results for Naive Bayes based on different feature subsets

The above Table 4.6 shows the accuracy of different feature sets when adopting Naive Bayes. We can see that Univariate selection is giving us the best result and PCA is always giving us the worst result.

4.7 Model comparison

To gain a more intuitive understanding of the performance of different models with different datasets, we made a comparison in Table 4.7 to show all the accuracy scores of the models.

Model/Feature Selection Tool	Full Model	PCA	LASSO	Recursive	Univariate	Tree Based Selection
Logistic Regression	0.824	0.584	0.784	0.672	0.816	0.672
Decision Tree	0.712	0.568	0.744	0.656	0.672	0.656
KNN	0.688	0.576	0.744	0.624	0.776	0.624
SVM	0.808	0.504	0.768	0.696	0.824	0.696
AdaBoost	0.752	0.536	0.776	0.672	0.776	0.672
Naive Bayes	0.768	0.616	0.776	0.648	0.824	0.648

Table 4.7 Comparison of the accuracy of 36 models

From the accuracy table above, we can compare the performance of different classifiers when features are selected by different methods.

From the perspective of feature subset choice, we can see that univariate selection is giving the highest accuracy among all 5 feature selection methods. Except for logistic regression and Decision Tree which yield the best results when all 11 features are applied, univariate selected features can give better results when other models are applied. Therefore, the features we choose eventually should be "age", "avg_glucose_level", "heart_disease", "hypertension", and "ever_married".

From the perspective of model selection, we can see that logistic regression with all the features, SVM with univariate features, and Naive Bayes with univariate selected features are given the same accuracy score which is 0.824. Thus, logistic regression, Support Vector Machine, and Naive Bayes should be our selected models.

5 Results

From the analysis we develop in this report, we conclude several major results as follows.

Firstly, from the exploratory analysis, we find out that older people, people with a higher average glucose level, people with higher BMI, and people who work in private companies are more likely to get a stroke. Therefore, it is important for older people to keep doing exercises to maintain a lower glucose level and keep slim. They should also avoid working in high-intense private companies.

When predicting whether a person is going to have a stroke or not, we try 5 feature engineering methods and extract different features with each method. After applying 6 different machine learning methods to do the different datasets, we find out that univariate selection can provide us with the best result, and the features it chooses are age, average glucose level, heart disease history, hypertension, and whether they ever got married or not. This partially coincides with our findings in exploratory data analysis. In terms of model selection, we found out that logistic regression, Support Vector Machine, and Naive Bayes are giving us the same testing accuracy which is 82.4%. If we want to use the original data for analysis, then logistic regression should be used; if we adopt the selected dataset, then Support Vector Machine or Naive Bayes should be adopted.