Table of Contents

Project Introduction 2

Data Dictionary 2

Data Cleaning 3

Dealing with missing values 5

Correlation Analysis 7

Data Visualization 8

Pre-modelling preparations 11

Data Modelling 14

Randomnized Search 16

Feature Importance 17

Conclusion 17

Reference 19

Project Introduction and aims

This project is done as part of my assessment of this subject (Machine Learning for Developers) for my student life in Temasek. In this MLDP Project, I need to first raise a topic of my interest and look for a suitable dataset online. Next, I will aim to clean, transform, and visualize the dataset chosen before selecting relevant and/or best features to build and improve my machine learning models. Lastly, I aim to be able to predict an outcome with my trained model and deploying it as a web application using Flask.

The dataset I have chosen is ‘Stroke Prediction Dataset’ that is taken from Kaggle. Each row in the dataset provides some relevant information about the patient. The aim of this dataset is to train a machine learning model which can be used to predict whether a patient is likely to get stroke based on the given parameters.

Data Dictionary

1. Id: Unique identifier
2. Gender: Patient gender, including ‘Male’, ‘Female’ and ‘Other’
3. Age: Age of patient
4. hypertension: 0 if the patient does not have hypertension, 1 if the patient has hypertension
5. heart\_disease: 0 if the patient does not have any heart diseases, 1 if the patient has a heart disease
6. ever\_married: Has the patient ever been married? "No" or "Yes"
7. work\_type: Work type of patient. "children", "Govt\_jov", "Never\_worked", "Private" or "Self-employed"
8. Residence\_type: Residence type of patient. "Rural" or "Urban"
9. avg\_glucose\_level: Average glucose level in blood
10. bmi: Body mass index
11. smoking\_status: Smoking status of patient. "Formerly smoked", "never smoked", "smokes" or "Unknown"
12. stroke: 1 if the patient had a stroke or 0 if not

\*Note: "Unknown" in smoking\_status means that the information is unavailable for this patient. In this case I will be treating it as missing value

Data cleaning

Firstly, importing packages and reading the dataset. Text

Description automatically generated

Here are the libraries that I have imported at the start of the project. Pandas and NumPy will be used for data related operations, such as reading csv file and creating new dataframe. Matplotlib.pyplot and seaborn are imported for plotting visualizations such as heatmap. Scipy.stats was imported for a specific operator that I needed to use for calculating Cramer’s V correlation later in the project. The KNN Imputer was imported for imputing missing datas, the Standard Scaler and LabelEncoder are used to encode the dataset before modelling. Train\_test\_split is used for splitting dataset for training models. Lastly, warnings were imported to ignore unfatal warnings.

Graphical user interface, application

Description automatically generated

Moving on, I first checked whether there are duplicated columns in the dataset. There was no duplicated rows.

Text

Description automatically generated with medium confidence

The columns ‘id’ was not needed and was dropped.

Then, I checked through every column to see whether there are irrelevant values. While doing so, I did slight edit to data in columns ‘work\_type’ and ‘smoking\_status’ for better readability. At the same time, I replaced the unknown values in ‘smoking\_status’ with np.nan.

Table

Description automatically generated with low confidence

After checking through all columns, I checked the sum of nullity in the dataset. There were missing values in bmi and smoking\_status

A picture containing text

Description automatically generated

Then, I removed outliers in column ‘bmi’. To clarify, a person is considered obese with a Body-Mass Index(BMI) above 30. In usual cases, a normal person's BMI is capped below 50, but the maximum value in the column 'bmi' is 97.6. Hence, these numbers should be seen as outliers and should be capped at 50 based on my knowledge.

Dealing with missing values

I created a copy of dataset for encoding. Text

Description automatically generated

For encoding the dataset, I have defined a class. Create an empty dictionary. If encoding is called, the dataset will first go through a for loop. For each column, if the data type is not float, do label encoding, Else do Standard Scaling for the float columns, at the same time store column name and function used for decoding later. When decoding is called, for each column in the dataset to decode, it will check whether the column was previously encoded, if yes, then perform inverse\_transform() the decode the dataset.

Here is dataset after encoding

Calendar

Description automatically generated

After encoding the dataset, use KNNImputer to impute missing values. A picture containing text

Description automatically generated

A dictionary and a for loop is used to ensure that the dataset maintains its data type for each column as Label Encoder requires integer in order to perform inverse\_transform(), but running the KNNImputer will cause the dataset to turn to float for all columns.

After imputing missing values, decode the dataset. Double check if there are still missing values Graphical user interface, table

Description automatically generated

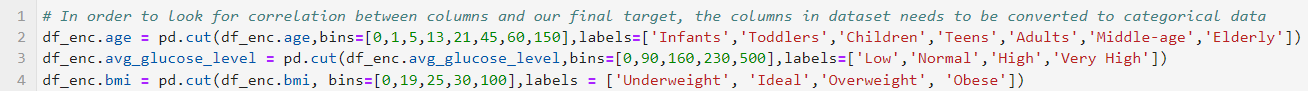
Perfect, there are no more missing values in the dataset

Graphical user interface, text, application, Word

Description automatically generated

Store the treated dataset as a CSV file. Now we are done with cleaning the data.

Correlation Analysis

Since our final target ‘stroke’ is a binary data (categorical data), I will convert the remaining columns to categorical data in order to do correlation analysis. I chose to bin the only 3 numerical data columns. 

In order to do analysis, the data have to be encoded. I made use of the class I defined in previous section and encoded the dataset with Label Encoder.

First correlation method I will be using is Cramer’s V. I have refered to a notebook on kaggle[1] to implement the function.

Cramer’s V is a correlation analysis method that aims to analyse the correlation between categorical data, which is exactly what I need. Text

Description automatically generated with medium confidence

I also used Spearman’s Correlation for analysis as Spearman’s Correlation is also a method for analysing correalation between categorical columns. Graphical user interface, application

Description automatically generated with medium confidence

Then, I plotted a heatmap for both methods.Chart, histogram

Description automatically generated

After looking at the heatmap, I realise that the features highly related to ‘stroke’ are age, average glucose level an hypertension

Data Visualization

Text

Description automatically generatedImported various functions from plotly

First: As shown in the pie chart below, the dataset we have is very biased. The percentage of stroke patients is only roughly 5%. I will be using SMOTE(Synthetic Minority Over-sampling Technique) to balance the dataset in later stages

Chart, pie chart

Description automatically generated

As shown in the graph, there are slightly more female than male patients in the datasetet. The age distribution is pretty even for both male and female, but there is an obvious outlier which is females aged 78-80.

Chart, histogram

Description automatically generated.

A visible observation that can me made is the fact the in both scatterplots the individuals who had a stroke are mostly located in the BMI value region under 60 and in high glucose levels as well as old age.

Chart, scatter chart

Description automatically generated Chart, scatter chart

Description automatically generated

Chart

Description automatically generated

From the graph, we can conclude:

* The percentage of female who had gotten stroke is slightly higher than the percentage of male who had gotten stroke.
* From the histograms, female stroke samples start to increase from age 50 and above, but male stroke samples only start to increase from age 55 onwards.
* Males seems to be more prone to strokes from their early 50s but the median stroke age for female is around 75-79.
* There are two noticeable drops in stroke occurrence. One of them is at females aged ranging 60-64, and the other is at males aged ranging 65-69.
* The two histograms further proofs that age is one of the major factors that contributes to stroke.
* A picture containing diagram

  Description automatically generated
* Looking at the pie charts, there seems to show no correlation between them and the stroked patients. The residence type is close to even distribution and most patients are married. The different work types is also not a controllable factor.
* These graphs are evidence that can support the conclusion I have drawn after correlation analysis, which is age, average glucose level and bmi are the highest contributor to potential stroke

Lastly, looking at smoking status of patients

Chart

Description automatically generated

Seems like a good portion (roughly 39%) of patients who suffered from stroke were formal smokers. There is 17% of patients who are still smokers and roughly 44% have never smoked.  
Since the percentage of smokers and non-smokers does not have a big difference, hence we are unable to conclude that smoking status is a factor that contributes to stroke.

Pre-Modelling preparations

Declaration: Since the number of columns present in the dataset is very little, feature selection and dropping of columns might not be a suitable act for this dataset. Hence, I decided not to do feature selection to after modelling.

Firstly, the functions imported from sklearn library: Text, letter

Description automatically generated

Since the outcome that we are predicting is a categorical variable, hence classification models will be used for this project and functions used for evaluating classification models are also imported.

The models to be evaluated are:

How are the models evaluated?

1. Classification Report
2. Confusion Matrix
3. Accuracy Score
4. F1 Score
5. ROC AUC Score
6. Logistic Regression (LR)
7. Decision Tree Classifier (DT)
8. Random Forest Classifier (RF)
9. Linear Discriminant Analysis (LDA)
10. XGBoost Classifier (XGB)
11. K-Neighbors Classifier (KN)
12. Support Vector Machines (SVM)

Models are separated and trained using pipeline

Diagram

Description automatically generated

First, I took reference to this algorithm selection cheat-sheet.

The following steps are taken:

Thus, the following are chosen:

1. Decision Tree Classifier
2. KNeighbors Classifier
3. SVM

* >50 samples: Yes
* Predicting a category: Yes
* Labelled Data: Yes
* Therefore, this is a classification problem
* <100k samples: Yes

What about the others?

1. Logistic Regression

LR is an easily implemented, interpreted and an efficient model to train. It performs well when the dataset is linearly separable (which it is in this project)

1. Random Forest Classifier

RF is a great model to use when the dataset is highly complicated as it has a parameter that allows it to split the dataset and work on subsets of the data. This also results in it being quick and more accurate as it is a combined effort of various randomized decision tree classifiers.

1. Linear Discriminant Analysis

LDA is primarily used in machine learning to reduce the number of features to a more manageable number before classification.

1. XGBoost Classifiers

Gradient Boosting Classifiers is a group of machine learning algorithms that combines many weak learning models together to create a strong model. It is effective and simple to be used.

How are these models evaluated?

1. Classification Report

Classification report is used to measure the quality of predictions from a classification algorithm.

1. Confusion Matrix

Confusion matrix provides an insight to the predictions of the model. It displays the correct and incorrect predictions on each class.

1. Accuracy Score

Accuracy score is simply just finding the percentage of correct prediction of a model.

1. F1 Score

F1 Score is an average between Precision (measuring how good model is when prediction is positive) and Recall (measuring how good model is at correctly predicting positive classes) scores

1. ROC AUC Score

ROC curve summarizes the performance of model at different threshold values. AUC is the area under ROC curve between (0,0) and (1,1).

After importing the functions needed, encode the dataframe again with label encoder for categorical values and standard scaler for numerical values Graphical user interface, text, application

Description automatically generated

Then, do train test split with test size 25% and a fixed random state 6. Next, import SMOTE. For both training and testing dataset, do resampling to reduce biasness of data.

Data Modelling

Text

Description automatically generated with medium confidence

Firstly, define different pipelines for different models. We will first evaluate the model with training set via different evaluation methods. Then, store these results into a dataframe for easier visualization. The function sum\_of\_scores() is to add up the scores of each model for training and testing dataset. This can give us a more straightforward comparison of how well the models performed.

Table

Description automatically generated

Random Forest Classifier seems to be the model that performed the best on training dataset.

Next, we move on to training these models and testing them with test dataset. Graphical user interface, text

Description automatically generated

After getting the scores on testing dataset, store them into the same dataframe created just now.

Graphical user interface, text, application, chat or text message

Description automatically generated

Random Forest is still the best for testing dataset. Hence, we will take the second-best model for randomized search too, which is Logistic Regression.

Trying to improve the models with randomized search

Firstly, Random Forest Classifier.

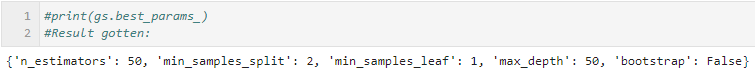
Then, Logistic Regression:

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated



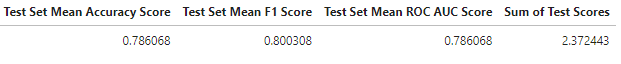
RF after tuning:

LR after tuning:

Text

Description automatically generated

Results before tuning:



Logistic Regression has a great improvement in the various scores!

Text

Description automatically generated

Results before tuning:

Graphical user interface, text, application

Description automatically generated

There seems to be a slight decrease in all the scores...

Feature Importance

Before Tuning:

After Tuning:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated

The results gotten from feature importance supports the conclusion I have drawn after correlation analysis. Age, average glucose level and BMI are the three highest factors that are related to stroke.

Conclusion

Table

Description automatically generated

Timeline

Description automatically generated

As shown from the visualizations, all the models have a relatively high accuracy, with Logistic Regression and Random Forest Classifier being the best ones.

With reference to various results of the models, I will be choosing Random Forest Classifier as my final prediction model. Chart

Description automatically generated with medium confidence

Prediction

I tried with values from a row that exist in the dataset. It returns the correct prediction.

Text

Description automatically generated

Now, the model is ready to be deployed as web application using flask.

End of Report

References:

How to Best Evaluate a Classification Model.

Link: <https://towardsdatascience.com/how-to-best-evaluate-a-classification-model-2edb12bcc587>

Stroke Prediction Dataset

Link: <https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset>

Cramer’s V reference[1]

Link: <https://www.kaggle.com/code/chrisbss1/cramer-s-v-correlation-matrix/notebook>