

1 Organise Data

The dataset comprises 43,400 records and 12 columns, including Patient's ID, Gender, Age, Hypertension status, Heart Disease status, Marital status, Work Type, Residence Type, Average Glucose Level, BMI, Smoking status, and Brain Stroke occurrence. Each row represents an individual's demographic and health-related information, sourced from openly available health data.

Sr No	Name	Description	Data Type
1	ID	Patient's ID number	Nominal
2	Gender	Patient's gender	Nominal
3	Age	Age of patient	Numerical (Int)
4	Hypertension	Patient's high blood pressure status	Nominal
5	Heart Disease	Indication of patient's heart disease status	Nominal
6	Married	Indicates whether the patient is married	Nominal
7	Occupation	Patient's occupation	Nominal
8	Residence	Type of residence	Nominal
9	Average Glucose Level	Average blood glucose level of the patient	Numerical (Float)
10	BMI	Body Mass Index (BMI) of the patient	Numerical (Float)
11	Smoking	Indicates whether the patient is a smoker	Nominal
12	Brain Stroke	Indicates whether the patient has experienced a brain stroke	Nominal

Table 1: Data Dictionary

Preprocessing steps, such as renaming variables and addressing data quality issues (including handling missing values and detecting outliers), were undertaken to ensure the dataset's readiness for predictive modelling and analysis. These steps are described in the following section .

1.1 Data Preprocessing

The dataset is preprocessed using Python. We have used Google Colab for the implementation .

1.1.1 Data Exploration

- The dataset is imported using Pandas library as shown in Figure 1.

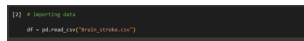


Figure 1: Loading the dataset

- Firstly, we will check the size of the dataframe as shown in Figure 2.



Figure 2: Size of the dataframe

The Dataset contains 43,400 row and 12 columns.

```

[1]: # checking the data type
df.info()

Out[1]: Int64Index: 43400 entries, 0 to 43399
Data columns (total 12 columns):
#   column      non-null count  dtype
#   --
0   id           43400 non-null      int64
1   gender       43400 non-null      object
2   age          43400 non-null      float64
3   hypertension 43400 non-null      object
4   heart_disease 43400 non-null      object
5   stroke       43400 non-null      object
6   avg_glucose_level 43400 non-null  float64
7   bmi          43400 non-null      float64
8   smoking_status 43400 non-null  object
9   no_of_children 43400 non-null  int64
10  average_education_level 43400 non-null  float64
11  average_income_level 43400 non-null  float64

```

Figure 3: Data Type of each columns

- The type of variables can be seen using info() command. See Figure 3.
Since, Stroke, heart disease or hypertension describes the plausibility of occurrence. These can be Nominal variables. So, the dataframe contains 4 Numerical fields and rest as Nominal fields.
- Then, we will find the summary of all columns.
There is a possibility of outliers in the BMI and Average Glucose Level columns as shown in Figure 4(a).
Figure 4(b) doesn't show any spelling mistakes. .

```

[5]: # checking the statistical data for numerical column
df[['id', 'age', 'avg_glucose_level', 'bmi']].describe()

```

	id	age	avg_glucose_level	bmi
count	43400.000000	43400.000000	43400.000000	43308.000000
mean	36326.142900	42.217894	104.482750	28.500038
std	21072.154879	22.519649	43.191751	7.770020
min	1.000000	0.000000	55.000000	10.100000
25%	18038.500000	24.000000	77.500000	23.200000
50%	36326.500000	44.000000	91.500000	27.700000
75%	54014.250000	60.000000	112.070000	32.900000
max	72943.000000	62.000000	291.000000	97.600000

(a)

```

[6]: # checking the value count for categorical column
selected_df = df[['gender', 'stroke', 'heart_disease', 'hypertension']]
for column in selected_df.columns:
    print(selected_df[column].value_counts())

```

```

gender
male      43308
female    9092
Name: count, dtype: int64

stroke
0      43308
1       9092
Name: count, dtype: int64

heart_disease
0      43308
1       9092
Name: count, dtype: int64

hypertension
0      43308
1       9092
Name: count, dtype: int64

```

(b)

Figure 4: Summary of all columns

1.1.2 Data Cleaning

- Excluding the *ID* column, as it is not necessary for our analysis. See Figure 5.

```
[9] # removing id column
df_cleaned = df.drop('id', axis = 1)
```

Figure 5: Dropping the *ID* Column

- Renaming column headers to enhance understanding of all dataset columns. Printing and inspecting the first few columns of the dataset as shown in Figure 6.

```
[10] # renaming the headers
df_cleaned = df_cleaned.rename({'gender': 'Gender',
                                'heart_disease': 'Heart Disease',
                                'ever_married': 'Ever Married',
                                'work_type': 'Occupation',
                                'Residence_type': 'Residence Type',
                                'avg_glucose_level': 'Avg Glucose Level',
                                'bmi': 'BMI',
                                'smoking_status': 'Smoking Status',
                                'stroke': 'Stroke Type', gender: 'Gender'})
```

(a)

```
# Printing first 5 rows to inspect data
df_cleaned.head()
```

	Gender	Age	Hypertension	Heart Disease	Stroke	Ever Married	Occupation	Residence Type	Avg Glucose Level	BMI	Smoking Status	Stroke Type
0	Male	55	1	1	1	No	Officer	Sub	99.92	32	Never	None
1	Male	66	1	1	1	No	Officer	Urban	101.02	32	Never	None
2	Female	51	1	1	1	No	Officer	Urban	103.02	32	Never	None
3	Female	59	1	1	1	No	Officer	Urban	99.92	32	Never	None
4	Male	60	1	1	1	No	Officer	Urban	101.02	32	Never	None

(b)

Figure 6: Summary of all columns

- Finding NAs in the dataset. See Figure 7.

```
[8] # finding NA's
df.isna().sum()
```

id	0
gender	0
age	0
hypertension	0
heart_disease	0
ever_married	0
work_type	0
Residence_type	0
avg_glucose_level	0
bmi	1462
smoking_status	13292
stroke	0
dtype: int64	

Figure 7: Finding NAs in the Dataset

We have missing values (NA's) in the *BMI* and *Smoking Status* columns. We have decided to impute these missing values in the next step.

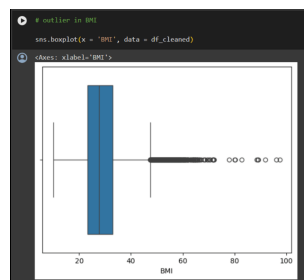
- Addressing missing values in the *Smoking* column by imputing NA values with an unknown category as shown in Figure 8.

```
[28] # solving NA's in smoking column
df_cleaned['Smoking'] = df_cleaned['Smoking'].fillna('unknown')
```

Figure 8: Solving in NAs in *Smoking*

- Checking Outliers in the *BMI* column. See Figure 9(a).

We have observed outliers in the *BMI* column and have chosen to address them by imputing the BMI values with the mean, grouped by age group as shown in Figure 9(b). We are removing ages below 5, assuming that children under 5 years are not prone to brain stroke.



(a)

```
# Solving BMI values with the mean grouped by age group
def categorize_age(df):
    if age < 5:
        return 'Child'
    elif 5 <= age < 18:
        return 'Teen'
    elif 18 <= age < 25:
        return 'Young Adult'
    elif 25 <= age < 35:
        return 'Adult'
    elif 35 <= age < 45:
        return 'Middle-aged'
    elif 45 <= age < 55:
        return 'Older Adult'
    elif 55 <= age < 65:
        return 'Senior'
    elif 65 <= age < 75:
        return 'Elderly'
    elif 75 <= age < 85:
        return 'Very Elderly'
    elif 85 <= age < 95:
        return 'Oldest'
    else:
        return 'Unknown'

df_cleaned = df_cleaned.assign(
    df_cleaned.groupby('Age')['BMI'].transform('mean')
)
df_cleaned['BMI'] = df_cleaned['BMI'].fillna(df_cleaned.groupby('Age')['BMI'].transform('mean'))
df_cleaned['Age'] = df_cleaned['Age'].fillna('unknown')
```

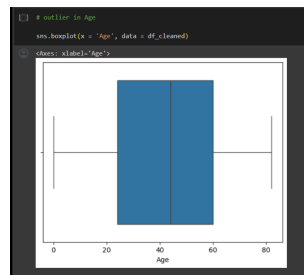
(b)

Figure 9: Solving NAs in *BMI*

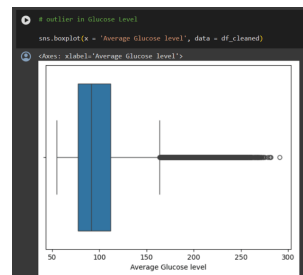
- Checking Outliers in the *Age* column

There are no outliers present in the *Age* Columns seen in Figure 10(a).

- Checking Outliers in the *Glucose Level* column



(a)



(b)

Figure 10: NA's in Age and Glucose Level

We have identified outliers in the glucose level column as seen in the Figure 10(b), suggesting values beyond the typical range for most candidates in the dataset. To ensure data integrity, we have decided to remove all glucose level values exceeding 260.