A

PROJECT REPORT

"Brain Stroke Prediction Using Machine Learning"

BY

Mr. Jibraan Attar

Mr. Tushar Jadhav

Submitted in Partial fulfillment of Post Graduation Diploma in Data Science and AI

Savitribai Phule Pune University

For the Academic Year

2023-2024

UNDER THE GUIDANCE OF

Prof. Tausif Shaikh

Prof. Poonam Bhawke.

Department of Technology Savitribai Phule

Pune University, Ganeshkhind, Pune-411007

SAVITRIBAI PHULE PUNE UNIVERSITY

A

PROJECT REPORT

ON

”Brain Stroke Prediction using Machine Learning

BY

Mr. Jibraan Attar

Mr. Tushar Jadhav

Submitted in Partial fulfillment of

Post Graduation Diploma in Data Science and AI

Savitribai Phule Pune University

For the Academic Year

2023-2024

UNDER THE GUIDANCE OF

Prof. Tausif Shaikh

Prof. Poonam Bhawke

Department of Technology
Savitribai Phule Pune University,
Ganeshkhind, Pune-411007

DEPARTMENT OF TECHNOLOGY

CERTIFICATE

This is to certify that **Jibraan Attar** and

Tushar Jadhav has successfully completed her project on

"Brain Stroke Prediction Using Machine

Learning"

in partial fulfillment of

2nd Semester work for his Post Graduation Diploma in Data Science and AI

under Savitribai Phule Pune University, for the academic year 2023-2024.

Prof…Tausif Shaikh

Dr. Manisha Bharati

Dr. Aditya Abhyankar

Prof. Poonam Bhawke
(Project Guide)

(Course Coordinator)

(H.O.D)

Signed By
(External Examiner)

Place : Pune
Date : / /

Students Declaration

We undersigned a student of the Department of Technology, Savitribai Phule Pune University, Pune PGD Data Science and AI - 2nd semester, declare that the summer internship project “Brain Stroke Prediction using Machine Learning” is a result of our own work and ours indebtedness to other work publications, references, if any, have been duly acknowledged. If we are found guilty of copying any other report or published information and showing it as our original work, We understand that we shall be liable and punishable by the Institute or University, which may include Fail in the examination, repeat study and resubmission of the report or any other punishment that Institute or University may decide.

Name of Student: Jibraan Attar
Enrolment Number:
Signature:

Name of Student:

Enrolment Number:

Signature:

ACKNOWLEDGEMENT

We are deeply grateful to all those who have played a pivotal role in shaping our internship project and enriching our learning experience. We extend my heartfelt gratitude to:

Prof. Tausif Shaikh
Prof. Poonam Bhawke

(Professor and Project Guide from the Department of Technology, SPPU) For their scholarly guidance, insightful suggestions, and continuous encouragement throughout the course of my project.

The entire faculty of the Department of Technology, Savitribai Phule Pune University, for fostering an environment of academic excellence and nurturing my curiosity.

We are profoundly thankful to everyone mentioned above for their selfless contributions to my journey. Their mentorship and support have been instrumental in shaping my skills and fostering personal growth.
Thank You

Mr. Jibraan Attar
Mr. Tushar Jadhav

INDEX

4.	4. Research Methodology 4.1 Rationale of the study 4.2 Statement of problem 4.3 Significance of the Problem 4.4 Research Objectives 4.5 Scope of the study 4.6 Research hypothesis 4.7 Research design (Research Type) 4.8 Data Collection Instrument (for e.g. Questionnaire) 4.9 Outline of analysis 4.10 Limitations of the Project	
5.	5.Images	

CHAPTER 1

INTRODUCTION AND RATIONALE OF THE STUDY

1.1 Introduction to the title

Brain stroke prediction is an important area of research in healthcare, as strokes can have severe consequences, including disability and death. In recent years, machine learning algorithms have been increasingly used to predict the likelihood of a patient experiencing a stroke, based on various demographic, medical, and lifestyle factors. This paper presents a review of recent research on brain stroke prediction using machine learning, with a focus on the use of Python programming language. The paper discusses the different machine learning algorithms that have been used for stroke prediction, including logistic regression, decision trees, and neural networks. It also discusses the datasets that have been used for training and testing these models, and the performance metrics that have been used to evaluate their accuracy. The paper concludes with a discussion of the challenges and opportunities for future research in this area, including the need for larger and more diverse datasets, and the potential for combining machine learning with other types of medical

data analysis, such as imaging and genomics.

1.2 Significance of the study

Brain stroke prediction involves assessing an individual's risk factors such as age, high blood pressure, smoking, and diabetes, among others. Machine learning models can be trained on large datasets to predict the likelihood of a stroke occurring in a particular individual based on their risk factors. Early detection and management of risk factors can help prevent strokes.

Python is a popular programming language used for machine learning applications, including brain stroke prediction projects. In such projects, machine learning algorithms are trained on large datasets of patient information, including medical history, demographic data, and test results, to learn patterns and make predictions about the likelihood of a patient experiencing a stroke.

CHAPTER 2

THEORETICAL FRAMEWORK (PROJECT) / REVIEW OF LITERATURE

2.1 Conceptualizing Brain Stroke Prediction

Understanding the Domain

Brain strokes are medical emergencies that occur when the blood supply to part of the brain

is interrupted or reduced. Predicting strokes involves identifying individuals at high risk based on various factors like demographics, medical history, lifestyle, and biological markers**.**

Defining the problem

The goal is to develop a model that can predict the likelihood of a stroke occurring in a patient within a specific time frame. This is typically a binary classification problem where the outcome is either a stroke (1) or no stroke (0).

Data Collection

Collect relevant data from various sources, including medical records, patient surveys, and online databases. The data should include demographic information, lifestyle factors, and medical history.

Data Preprocessing

Clean the data to remove any missing values or outliers. Also, perform feature engineering to extract meaningful features from the data.

Feature Selection

Identify the most relevant features that have the most significant impact on stroke prediction.

Model Training

Split the data into training and testing sets. Train a machine learning algorithm, such as logistic regression, random forest, or support vector machine, using the training data.

Hyperparameter Tuning

Optimize the model by tuning hyperparameters using techniques like Grid Search or Random Search.

Model Optimization

Optimize the model by fine-tuning hyperparameters, selecting a different algorithm, or performing feature selection again.

Deployment

Once the model is optimized and performs well, deploy it in a web or mobile application to help individuals assess their risk of having a stroke.

CHAPTER 3

OBJECTIVES AND SCOPE OF PROJECT

3.1 Objectives:

Develop a Predictive Model for Stroke Risk:

Create a machine learning model that accurately predicts the likelihood of an individual experiencing a stroke based on various risk factors such as demographics, medical history, lifestyle, and biological markers.

Improve Early Detection:

Enhance the early detection of high-risk individuals, enabling timely medical intervention and potentially reducing the incidence and severity of strokes.

Integrate with Healthcare Systems:

Develop a system that can be seamlessly integrated into existing healthcare infrastructures, providing healthcare professionals with an easy-to-use tool for assessing stroke risk.

Enhance Patient Outcomes:

By identifying high-risk individuals early, aim to improve patient outcomes through preventive measures, targeted treatments, and lifestyle modifications.

3.2 Scope

Data Collection and Preparation:

Gather and preprocess data from various sources, including electronic health records (EHRs), patient surveys, clinical trials, and other relevant databases.

Ensure data quality by handling missing values, correcting errors, and standardizing data formats.

Feature Selection and Engineering:

Identify and select relevant features that significantly contribute to stroke prediction.

Create new features through feature engineering to enhance model performance.

Model Development:

Develop multiple machine learning models (e.g., logistic regression,GaussianNB, boosting, XGB boost) to predict stroke risk.

Compare and evaluate these models to identify the best-performing algorithm.

Model Training and Validation:

Split the data into training and testing sets to train the models and evaluate their performance.

Use cross-validation techniques to ensure the models generalize well to unseen data.

Evaluation Metrics:

Assess model performance using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

Conduct residual analysis and other diagnostic checks to validate model assumptions and performance.

Chapter 4

Research Methodology

4.1 Rational for the study

4.2 Statement of problem

4.1 Rational for the study

1. Public Health Importance

Stroke is a leading cause of disability and death worldwide. Early identification of individuals at high risk of stroke can significantly reduce morbidity and mortality rates through timely medical interventions and lifestyle modifications. Developing a predictive model for stroke risk serves as a proactive measure to address this critical public health issue.

2. High Prevalence and Burden

The prevalence of stroke and its associated burden on individuals, families, and healthcare systems is substantial. Strokes often result in long-term physical and cognitive impairments, leading to reduced quality of life and increased healthcare costs. By predicting and preventing strokes, the model can help alleviate this burden.

3. Advancements in Data Analytics and Machine Learning

The availability of large datasets from electronic health records (EHRs), combined with advancements in machine learning algorithms, provides an unprecedented opportunity to develop accurate predictive models. Leveraging these technologies can enhance the early detection of stroke

risk and support data-driven healthcare decisions.

4.2 Statement of the problem

Stroke is a major public health concern, being one of the leading causes of death and long-term disability globally. Despite advances in medical research and healthcare practices, the early detection and prevention of strokes remain challenging due to the complex interplay of various risk factors. Traditional risk assessment methods often fall short in accurately predicting stroke occurrence, leading to missed opportunities for early intervention and prevention.

Key Issues:

High Incidence and Mortality:

Strokes affect millions of people worldwide annually, leading to significant mortality and morbidity. The inability to accurately predict and prevent strokes contributes to the high burden on individuals, families, and healthcare systems.

Complex Risk Factor Interaction:

Stroke risk is influenced by a multitude of factors, including age, hypertension, diabetes, lifestyle behaviors (such as smoking and physical inactivity), and genetic predispositions. Traditional risk assessment tools often struggle to capture the complexity and interdependence

of these factors.

Inadequate Early Detection**:**

Current clinical practices for stroke risk assessment, such as standardized risk scores and guidelines, are often insufficiently tailored to individual patients, resulting in suboptimal early detection and preventive measures.

Healthcare System Burden:

The management and treatment of stroke patients place a considerable strain on healthcare resources. Effective prediction and prevention strategies could alleviate this burden by reducing the incidence of strokes.

4.3 Significance of problem

The problem of accurately predicting and preventing strokes is significant due to the profound impact strokes have on individuals, healthcare systems, and society at large. Addressing this issue can lead to numerous benefits across multiple dimensions:

1. Public Health Impact

High Mortality and Morbidity:

Strokes are a leading cause of death and long-term disability worldwide. Early prediction and prevention can reduce the incidence of strokes, thereby lowering mortality rates and improving the quality of life for millions of individuals.

Chronic Health Issues:

Survivors often face chronic health problems, including physical disabilities, cognitive impairments, and emotional challenges. Effective prediction and prevention strategies can mitigate these long-term consequences.

2. Economic Burden

Healthcare Costs:

Treating and managing stroke patients is expensive, involving hospital stays, rehabilitation, long-term care, and loss of productivity. By preventing strokes, healthcare systems can save substantial costs.

Workforce Productivity:

Strokes often affect individuals in their productive years, leading to significant economic losses due to disability and premature death. Reducing stroke incidence can enhance workforce productivity and economic stability.

4.4 Research Objectives

Objective: Create a robust machine learning-based model to predict the likelihood of stroke in individuals based on a comprehensive set of risk factors.

Identify Key Risk Factors:

Objective: Analyze and identify the most significant risk factors contributing to stroke occurrence.
Understanding these factors can help in designing targeted prevention programs and personalized treatment plans.
Enhance Early Detection:

Objective: Develop a model that can detect high-risk individuals early, allowing for timely medical intervention.
Rationale: Early detection is critical in preventing strokes and reducing their severity, leading to better health outcomes.

Integrate Predictive Model with Healthcare Systems:

Objective: Design a user-friendly tool that can be easily integrated into existing electronic health record (EHR) systems and clinical workflows.
Rationale: Seamless integration ensures that healthcare providers can efficiently use the predictive model in routine practice to assess and manage stroke risk.

4.5 Research Hypothesis

In the context of developing a predictive model for stroke risk, the research hypothesis sets the foundation for the investigation. The hypothesis should be clear, testable, and related to the specific objectives of the study. Here is a well-defined hypothesis for your study:

Hypothesis

Primary Hypothesis:

H1: A machine learning-based predictive model incorporating a comprehensive set of risk factors (such as age, gender, hypertension, diabetes, smoking status, cholesterol levels, and lifestyle factors) will predict the risk of stroke with greater accuracy and reliability than traditional risk assessment methods.

Secondary Hypotheses:

H2: The machine learning-based model will identify significant interactions between risk factors that are not captured by traditional models.

4.6 Research Design

The research design outlines the methodological approach and procedures for developing and validating a predictive model for stroke risk using machine learning. This design ensures that the study is structured, systematic, and capable of addressing the research hypotheses.

1. Study Type

Type: Quantitative, observational study.

Approach: Retrospective cohort study for model development and validation, followed by a prospective pilot implementation study.

2. Population and Sample

Population: Individuals with varying risk factors for stroke (e.g., age, gender, hypertension, diabetes, smoking status).

Sample Size: The dataset should include a large number of records to ensure the robustness of the model. For instance, a minimum of 10,000 records is recommended, with a balanced representation of those who have had strokes and those who have not.

Sampling Method: Use existing health records or datasets such as public health databases, clinical trial data, or electronic health records (EHR).

3. Data Collection

Public health datasets (e.g., National Health and Nutrition Examination Survey (NHANES)).

Clinical databases (e.g., hospital EHR systems).

Research datasets (e.g., clinical trial data).

4. Data Variables:

Demographic information: age, gender.
Medical history: hypertension, heart disease,glucose level.
Lifestyle factors: smoking status.
Clinical measurements: BMI.
Outcomes: incidence of stroke (binary outcome: yes/no)

5. Data Cleaning:

Handle missing values using imputation methods or by removing incomplete records.
Normalize numerical features to standardize the scale.
Convert categorical variables to factors or dummy variables.

4.7 Data Sources

Clinical Trial Databases:

Clinical trial databases, such as ClinicalTrials.gov, contain information on ongoing and completed clinical trials related to stroke prevention, treatment, and management. Analyzing trial data can provide insights into the effectiveness of interventions and risk factors associated with stroke.

4.8 Data Collection Instrument

Stroke Risk Assessment Questionnaires:

Standardized questionnaires designed to assess an individual's risk factors for stroke, including demographics (age, gender), medical history (hypertension, diabetes, heart disease), lifestyle factors (smoking, alcohol consumption), and family history of stroke.

Health Behavior Surveys:

Surveys that gather information on lifestyle factors associated with stroke risk, such as physical activity, dietary habits, stress levels, and sleep patterns.

Quality of Life Surveys:

Surveys assessing the impact of stroke risk factors and related health conditions on an individual's quality of life, including physical, mental, and social well-being.

4.9 Outline of analysis

Data Exploration and Preprocessing

Data Loading: **Import the dataset containing relevant features and the target variable (stroke).**
Data Inspection: Check for missing values, data types, and distribution of variables.
Data Cleaning: Handle missing values, outliers, and inconsistencies in the dataset.
Data Visualization: Explore the relationships between variables using histograms, scatter plots, and correlation matrices.

2. Feature Selection and Engineering

Feature Identification: Identify key features related to stroke risk based on domain knowledge and exploratory data analysis.

Dimensionality Reduction: Use techniques such as PCA to reduce the number of features while retaining relevant information.
Feature Engineering: Create new features or transformations that may improve the model's predictive performance.

3. Model Development
Model Selection: Choose appropriate machine learning algorithms for classification (e.g., logistic regression, decision trees, random forests, gradient boosting).
Model Training: Split the dataset into training and testing sets. Train the models on the training data.
Model Evaluation: Evaluate model performance using metrics such as accuracy,

precision, recall, F1-score, and ROC-AUC.

Hyperparameter Tuning: Optimize model hyperparameters using techniques like grid search or random search to improve performance.

4. Model Interpretation

Feature Importance: Determine the importance of features in predicting stroke risk using techniques like permutation importance (SHapley Additive exPlanations).

Partial Dependence Plots: Visualize the effect of individual features on the model's predictions while keeping other features constant.

Interpretability: Ensure the model is interpretable and understandable by clinicians and stakeholders.

5. Model Validation

Internal Validation: Validate the model's performance using cross-validation techniques (e.g., k-fold cross-validation) on the training dataset.

External Validation: Validate the model on an independent dataset to assess generalizability.

Sensitivity Analysis: Assess the model's robustness to changes in parameters or assumptions.

6. Model Deployment and Implementation

Scalability: Ensure the model can be deployed at scale and handle large volumes of data.

Integration: Integrate the model into healthcare systems or decision support tools for clinical use.

Usability Testing: Conduct usability testing with healthcare providers to ensure the model meets their needs and is easy to use.

Monitoring and Maintenance: Establish procedures for monitoring model performance

over time and updating the model as needed with new data or changes in clinical

4.10 Limitations of project

Data Quality and Availability:

Limited availability of high-quality data may constrain the model's predictive performance.
Incomplete or biased datasets may lead to model inaccuracies or generalization issues.

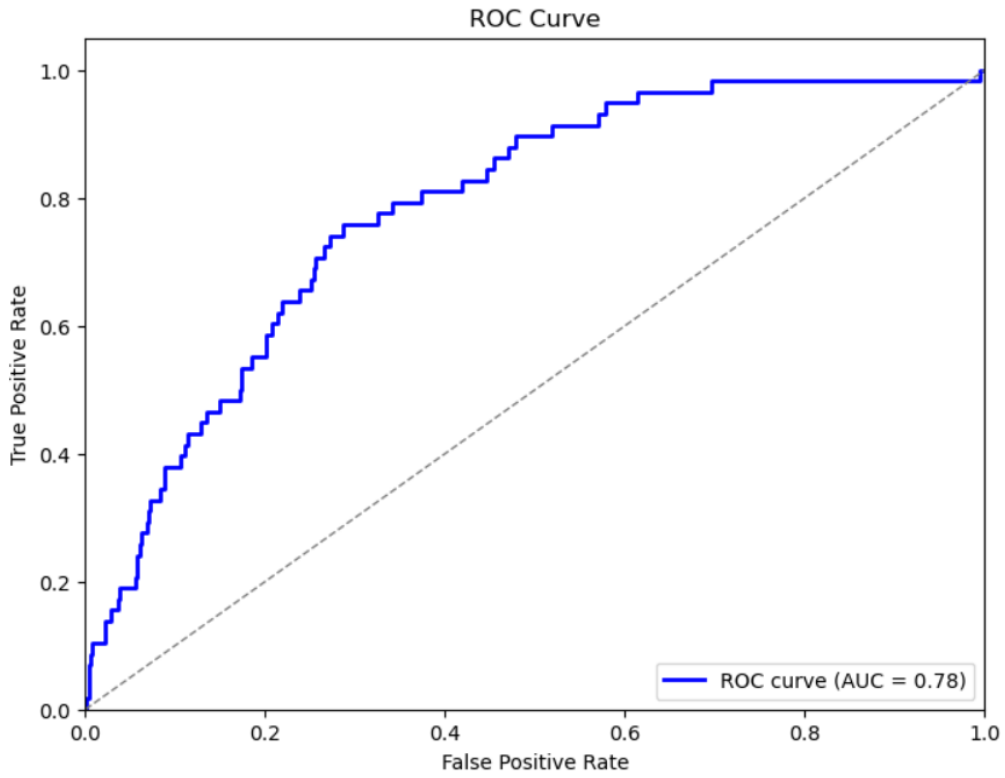
Imbalanced Data:

Imbalance between positive (stroke) and negative (non-stroke) cases in the dataset may affect the model's ability to learn and generalize.
Strategies such as oversampling, undersampling, or using class weights may be needed to address imbalance.

Chapter 5

Images

ROC- Curve (Receiver operating characteristic Curve)



The above ROC-curve is for the fitted logistic regression model .The graph shows a AUC value of 0.78 which is considered as good predictor model.

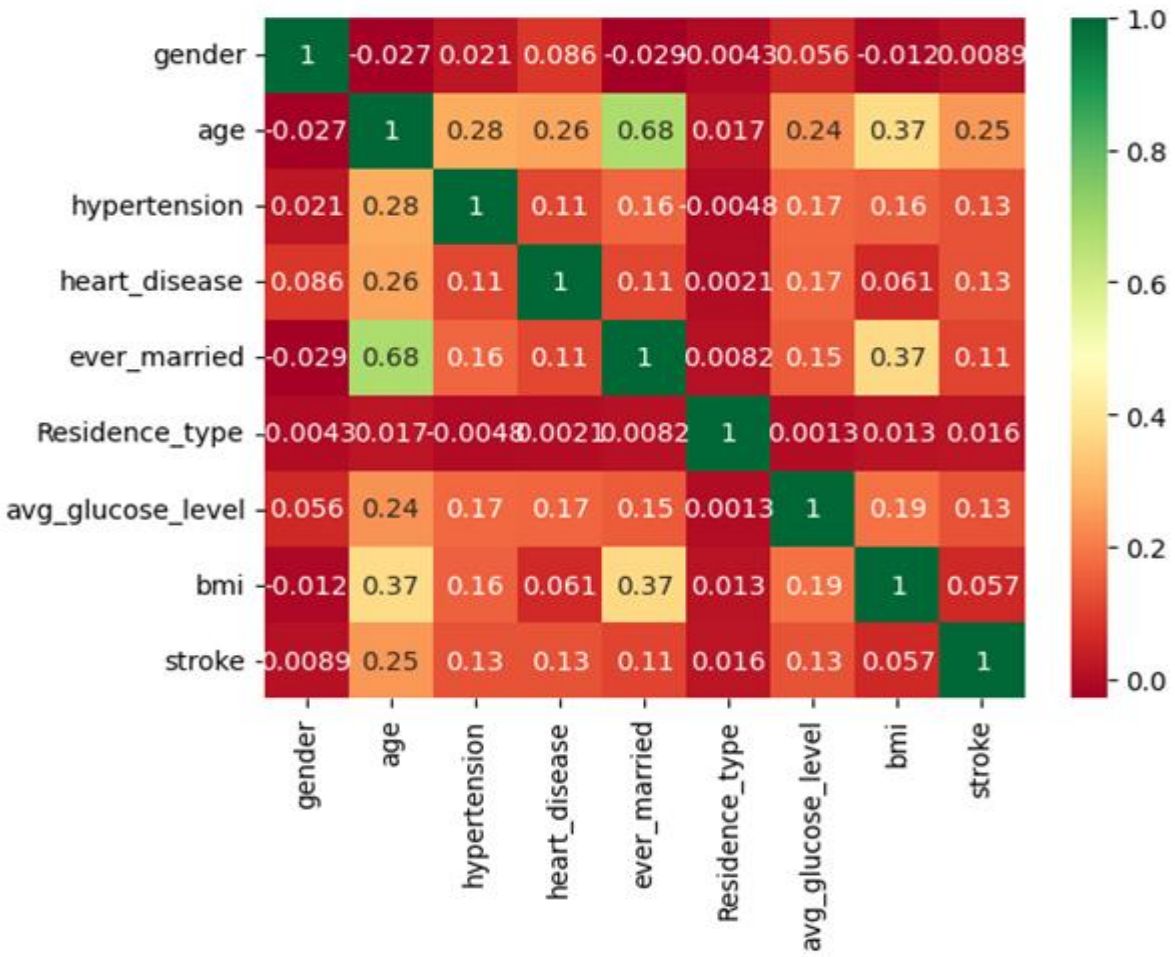
It plots the true positive rate (TPR), also known as sensitivity, against the false positive rate (FPR), which is 1-specificity, at various threshold settings. An area under the ROC curve (AUC) value quantifies the overall performance of the model, with higher values indicating better discrimination between positive and negative cases.

In the context of stroke risk prediction, an AUC value of 0.78 for the ROC curve suggests that the model has relatively good discriminatory power in distinguishing between individuals at high risk of stroke and those at low risk.

Libraries used

1. **matplotlib**
2. **pandas**
3. **seaborn**
4. **skicit learn**

Heatmap

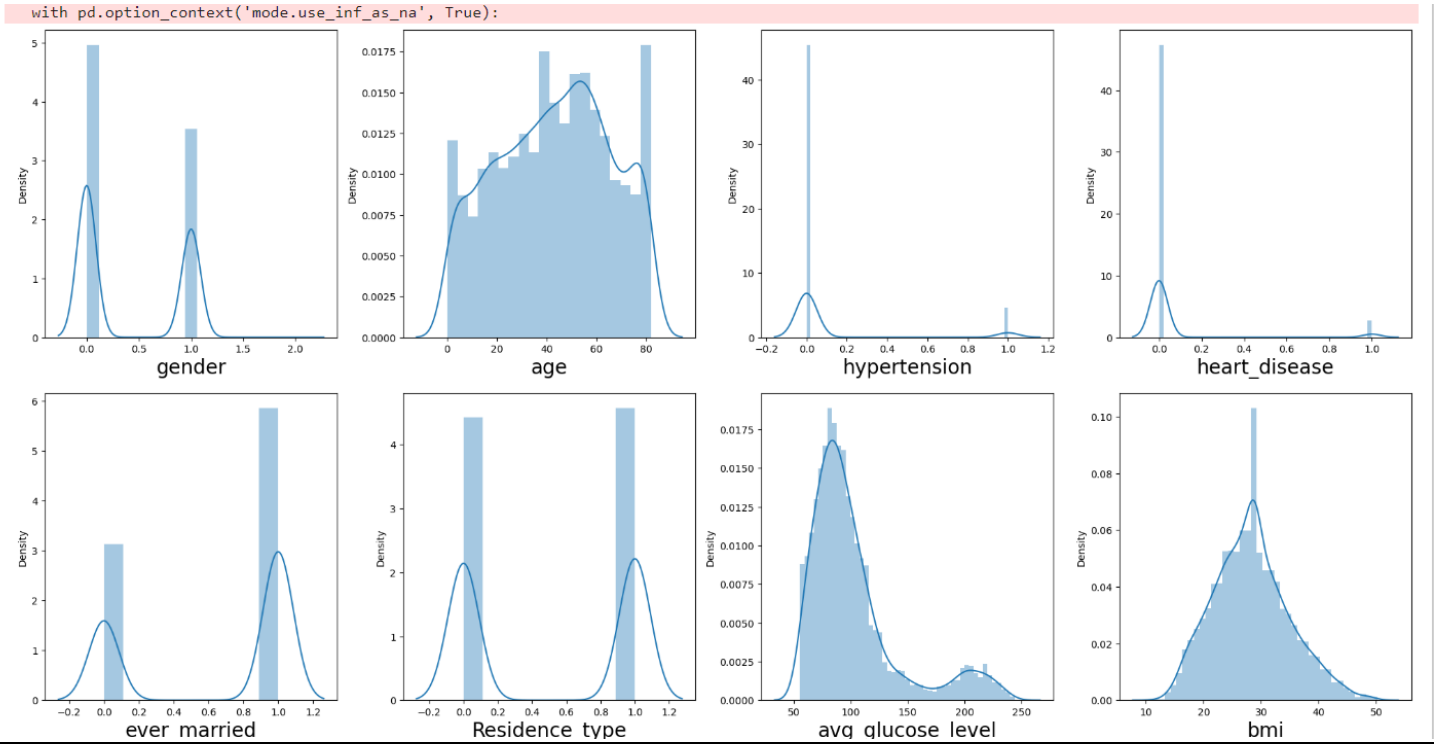


The above heatmap shows the correlation between the key variables present in the dataset .
By coinciding the one variable with the other one we can interpret their correlation .

We can see that among the variables rather than the same ones like (age vs age) we can see that the correlation between age and marital status is relatively high as compared to the other variables.

Displots for the key variables

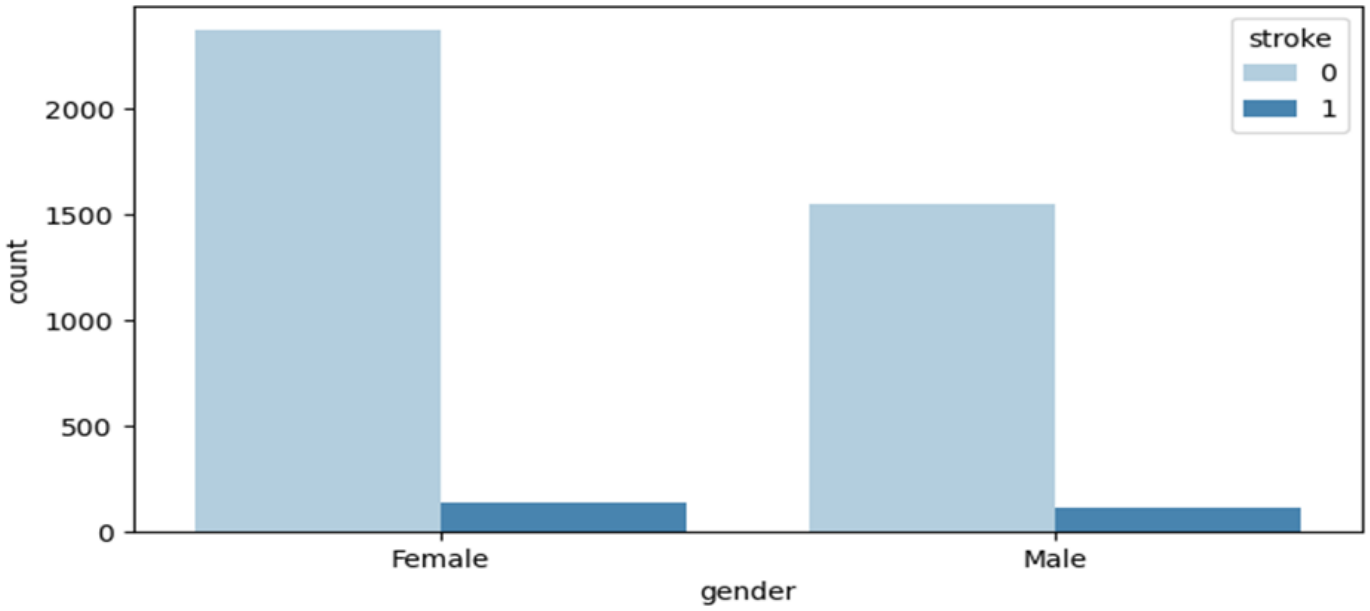
```
with pd.option_context('mode.use_inf_as_na', True):
```



As we can see from the above subplots which are plotted with the most key features in the dataset .

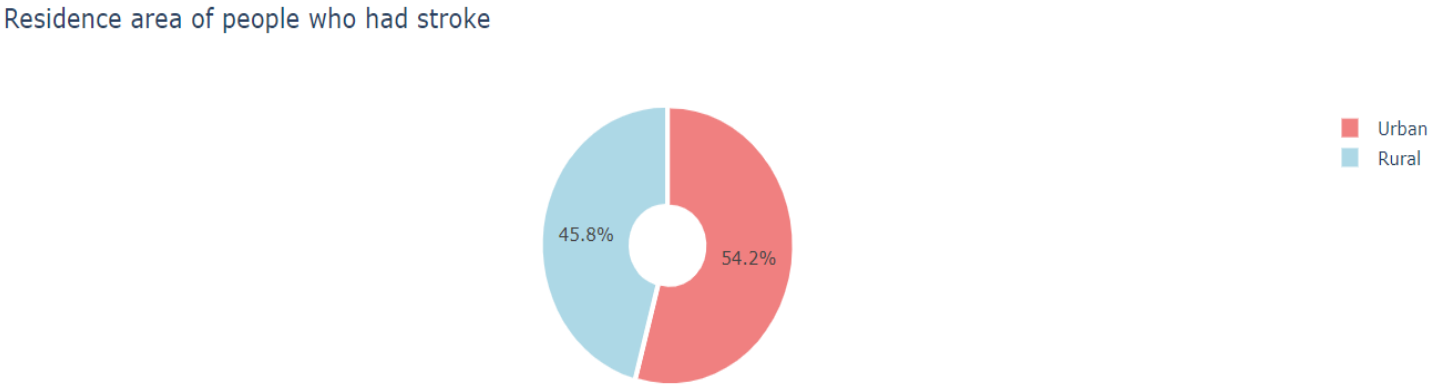
Most of the variables in the dataset are positively skewed , and their nature is like the Gaussian Distribution.

Stroke vs Non Stroke



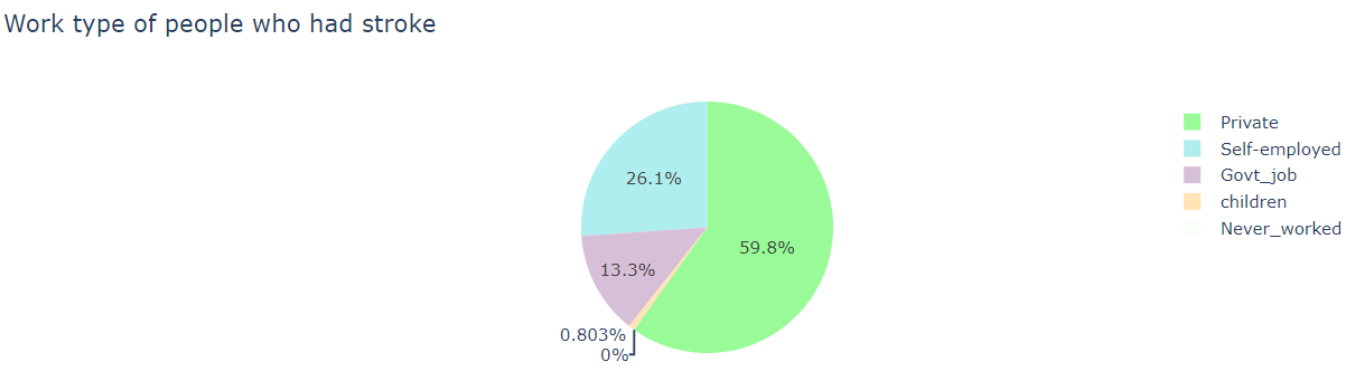
From the above bar charts it is evident that the proportion among men and women for stroke is very low. Therefore, we can say that the majority of the population is leaving without any Brain Stroke.

Resident type and stroke



From above graph we can say that people living in urban areas are most likely to have stroke .
The percent of stroke among them is 54.2% whereas for population in rural areas is 45.8%.

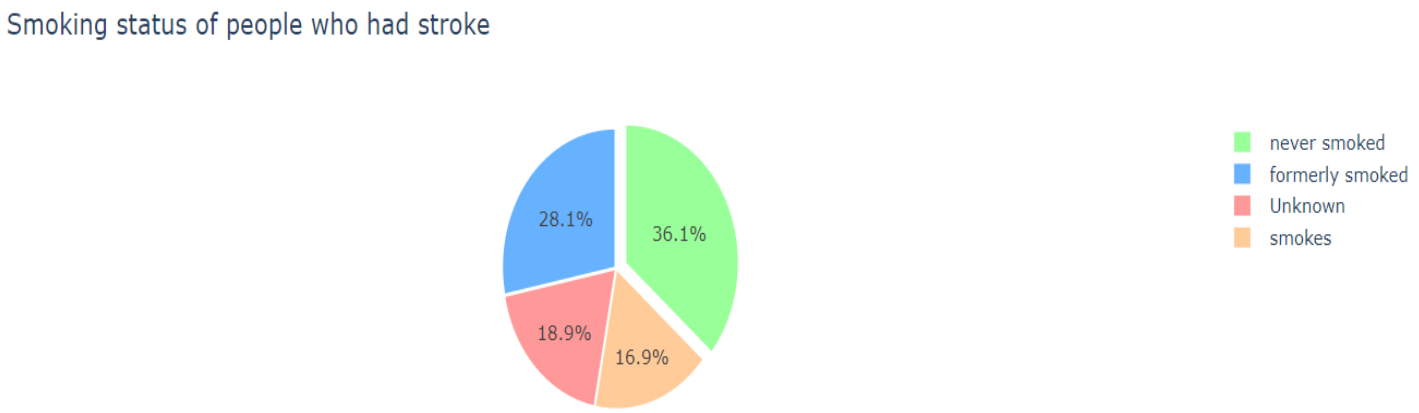
Work type and Stroke



The above pie chart shows the percent of stroke among the people working in different sectors.

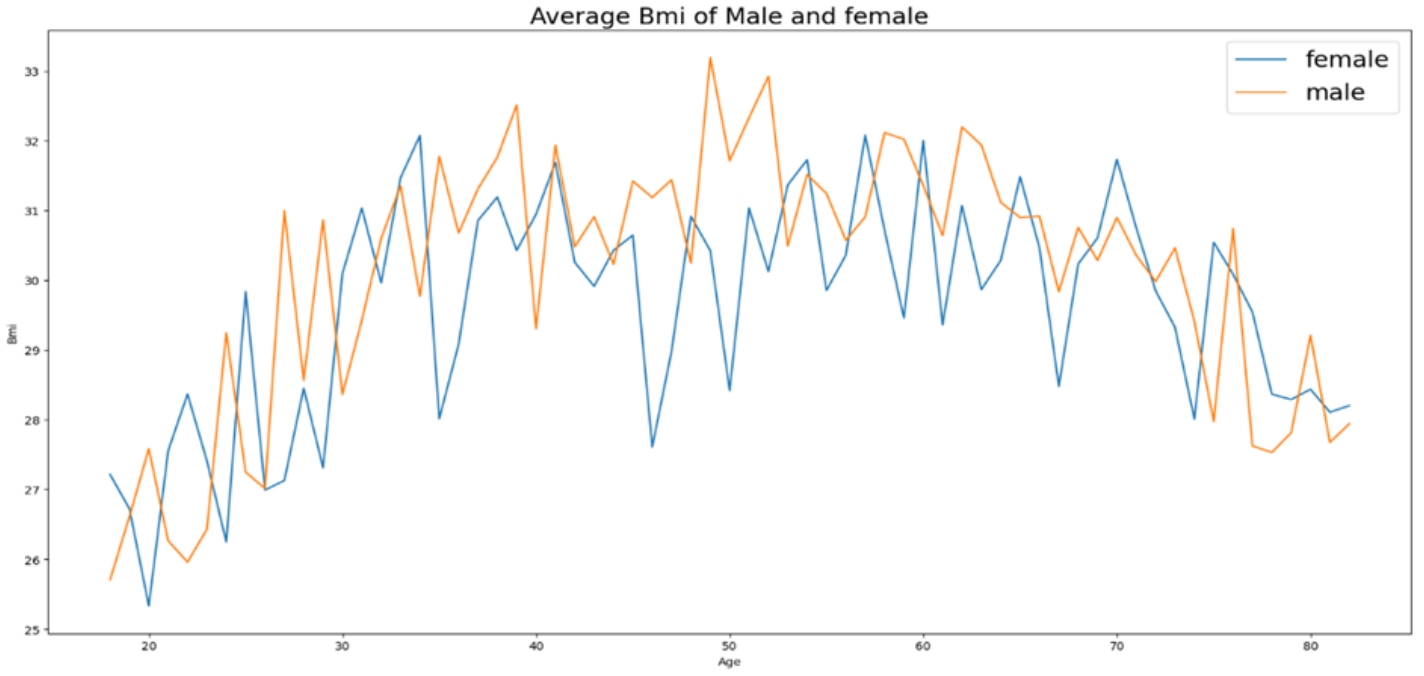
As we can see that the percent of stroke is higher for people working in the private sector with a stroke percent of 59.8%

Smoking Status and Stroke



The above pie chart describes the percentage of stroke amongst the persons with different smoking status . We can see that the people who never smoke have a stroke percent of 36.1%.

Gender and BMI



From above graph we can conclude that bmi increases from age 20-30 then the bmi from age 30-75 ranges from 27.5-33 and the gradually decreases from age 75-above.

Chapter 6

6.1 Data Analysis

Dataset

Brain stroke prediction

```
1]: import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    import warnings
    warnings.filterwarnings("ignore")
```

```
2]: df=pd.read_csv(r'D:\OneDrive\Desktop\healthcare-dataset-stroke-data.csv');df
```

2]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
...	...	...	...	...	...	...	...	...	...	...	...	...
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	0
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

The above is the dataset for brain stroke which consists of 5109 rows and 12 columns. We can see that there are some categorical values present in it .There are many subcategories like work type (private,self employed, govt job , children) and smoking status(never smoked,smokes,formerly smoked,unknown).
The dataset is consisting of both types of data categorical and some numericals. So we need to do proper exploratory data analysis for it to obtain accurate results.

[5]:
```
df.describe()
```

[5]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

Above are the descriptive statistics of the dataset . We can see that most of the variables in the dataset are positively skewed i.e. mean>median>mode. Also the described statistics are only for the variables that are numerical not categorical.

[ ]:
```
```

[18]:
```
df.isnull().sum()
```

[18]:
```
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
dtype: int64
```

6.2 Exploratory Data Analysis

Handling missing values in Bmi

```
[ ]:
```

```
[18]: df.isnull().sum()
```

```
[18]: 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
      dtype: int64
```

The BMI column contains 201 missing values in it .Therefore, it becomes necessary to handle this missing values.

We can replace with these values by various methods like

1. mean
2. median
3. mode

But considering the data type of bmi column we will replace the missing values with mean

[20]:
```
df['bmi'].describe()
```

[20]:
```
count    4909.000000
mean       28.893237
std         7.854067
min        10.300000
25%        23.500000
50%        28.100000
75%        33.100000
max        97.600000
Name: bmi, dtype: float64
```

[21]:
```
df['bmi'].fillna(df['bmi'].mean(),inplace=True)
```

[22]:
```
df.isnull().sum()
```

[22]:
```
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
dtype: int64
```

Detecting Outliers

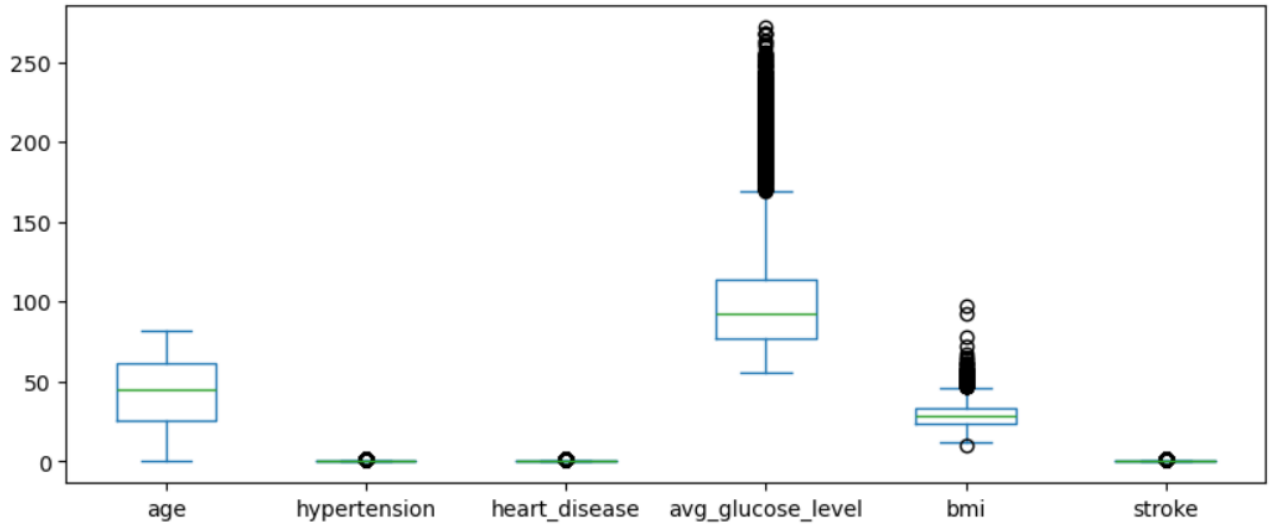
Before proceeding for further analysis it is necessary to detect the outliers in the dataset . To improve the models accuracy detecting outliers should be done .

Checking the outliers in dataset

Feature Selection

[23]:
```
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize']=(10,4)
plt.figure(figsize=(20,25))
df.plot(kind='box')
plt.show()
```

```
<Figure size 2000x2500 with 0 Axes>
```



We can see that every column in the dataset is having outliers , but glucose level and bmi are the columns with many outliers . Hence we can treat these outliers by following methods

1. IQR
2. Z score
3. Trimming
4. Isolation Forest

Handling Outliers

```
[24]: import pandas as pd
      from sklearn.ensemble import IsolationForest

      columns_of_interest = ['bmi', 'avg_glucose_level']  # Adjust as needed


      features = df[columns_of_interest]

      # Initialize the Isolation Forest model
      clf = IsolationForest(contamination=0.05, random_state=42)  # Adjust the contamination parameter as needed

      outliers = clf.fit_predict(features)
      df_cleaned = df[outliers == 1]
```

Defining Columns of Interest:

columns_of_interest = ['bmi', 'avg_glucose_level']: Specifies the columns of interest in the dataset that will be used as features for outlier detection. You can adjust this list to include other relevant features as needed.
Extracting Features:

features = df[columns_of_interest]: Creates a new DataFrame features containing only the columns specified in columns_of_interest. This DataFrame will be used as input to the Isolation Forest model for outlier detection.
Initializing Isolation Forest Model:

clf = IsolationForest(contamination=0.05, random_state=42): Initializes an instance of the IsolationForest class with parameters:
contamination: Specifies the expected proportion of outliers in the dataset. In this case, it's set to 0.05, indicating that 5% of the data is expected to be outliers. You can adjust this parameter based on your domain knowledge or exploration of the data.
random_state: Sets the random seed for reproducibility of results.
Outlier Detection:

outliers = clf.fit_predict(features): Fits the Isolation Forest model to the features data (features) and predicts the outlier labels for each sample. The fit_predict method returns a NumPy array containing the predicted

labels (-1 for outliers and 1 for inliers) for each sample.

Preprocessing Categorical Variables

Preprocessing categorical variables involves transforming categorical data into a numerical format suitable for machine learning models. This typically includes encoding categorical variables into binary or numerical representations. Common methods include one-hot encoding, which creates binary columns for each category, and label encoding, which assigns a unique numerical value to each category. Preprocessing categorical variables is essential for ensuring compatibility with machine learning algorithms, enabling them to effectively learn from categorical data and make accurate predictions. Additionally, it helps avoid bias towards certain categories and ensures that all features contribute equally to the model's performance. Careful consideration of encoding techniques and handling of categorical variables can significantly impact the quality and interpretability of machine learning models. Evaluating the distribution and cardinality of categorical variables, as well as considering domain-specific knowledge, are crucial steps in the preprocessing pipeline.

```
33]: df_cleaned['ever_married']=MS;df_cleaned
```

33]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.0	0	1	1	Private	1	228.69	36.600000	formerly smoked	1
1	0	61.0	0	0	1	Self-employed	0	202.21	28.893237	never smoked	1
2	1	80.0	0	1	1	Private	0	105.92	32.500000	never smoked	1
3	0	49.0	0	0	1	Private	1	171.23	34.400000	smokes	1
4	0	79.0	1	0	1	Self-employed	0	174.12	24.000000	never smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
5105	0	80.0	1	0	1	Private	1	83.75	28.893237	never smoked	0
5106	0	81.0	0	0	1	Self-employed	1	125.20	40.000000	never smoked	0
5107	0	35.0	0	0	1	Self-employed	0	82.99	30.600000	never smoked	0
5108	1	51.0	0	0	1	Private	0	166.29	25.600000	formerly smoked	0
5109	0	44.0	0	0	1	Govt_job	1	85.28	26.200000	Unknown	0

4954 rows × 11 columns

For preprocessing of the categorical variables like gender, ever_married,residence_type we have used label encoder . Label Encoder converts the categorical variables into the 1,0 form.

Also for smoking_status , work_type as there are many sub categories in it we have used One-hot encoder
Also for smoking_status , work_type as there are many sub categories in it we have used One-hot encoder. It converts all the subcategories into binary form and assigns unique columns for it.

```
[34]: from sklearn.preprocessing import OneHotEncoder
      enc = OneHotEncoder(handle_unknown='ignore')
      enc_data = pd.DataFrame(enc.fit_transform(df_cleaned[["work_type","smoking_status"]]).toarray())

[35]: enc.categories_

[35]: [array(['Govt_job', 'Never_worked', 'Private', 'Self-employed', 'children'],
            dtype=object),
       array(['Unknown', 'formerly smoked', 'never smoked', 'smokes'],
            dtype=object)]
```

```
7]: enc_data.columns = ['Govt_job','Never_worked', 'Private', 'Self-employed', 'children', 'Unknown', 'formerly smoked', 'never smoked', 'smokes']
    enc_data
```

7]:

	Govt_job	Never_worked	Private	Self-employed	children	Unknown	formerly smoked	never smoked	smokes
0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
4	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...
4849	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
4850	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
4851	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
4852	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
4853	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0

4854 rows × 9 columns

Now , after this we need to merged this processed dataset with our original dataset.

```
[40]: DF= pd.concat([df_cleaned, enc_data], axis = 1)
      DF.head()
```

[40]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke	Govt_job	Never_worked
0	1	67.0	0	1	1	Private	1	228.69	36.600000	formerly smoked	1	0.0	0.0
1	0	61.0	0	0	1	Self-employed	0	202.21	28.893237	never smoked	1	0.0	0.0
2	1	80.0	0	1	1	Private	0	105.92	32.500000	never smoked	1	0.0	0.0
3	0	49.0	0	0	1	Private	1	171.23	34.400000	smokes	1	0.0	0.0
4	0	79.0	1	0	1	Self-employed	0	174.12	24.000000	never smoked	1	0.0	0.0

Now, after completing all the necessary preprocessing dataset has 4854 rows and 16 columns.

Dropping the work_type and smoking status columns

As we have done all the necessary preprocessing on the categorical columns of the dataset it becomes necessary to remove the previous ones to reduce the error while predicting .

```
[44]: DF.drop(columns=['smoking_status'],axis=1,inplace=True)
```

```
[45]: DF
```

[45]:

	gender	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_level	bmi	stroke	Govt_job	Never_worked	Private	Self-employed	chi
0	1	67.0	0	1	1	1	228.69	36.600000	1	0.0	0.0	1.0	0.0	
1	0	61.0	0	0	1	0	202.21	28.893237	1	0.0	0.0	0.0	1.0	
2	1	80.0	0	1	1	0	105.92	32.500000	1	0.0	0.0	1.0	0.0	
3	0	49.0	0	0	1	1	171.23	34.400000	1	0.0	0.0	1.0	0.0	
4	0	79.0	1	0	1	0	174.12	24.000000	1	0.0	0.0	0.0	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4849	0	80.0	1	0	1	1	83.75	28.893237	0	0.0	0.0	1.0	0.0	
4850	0	81.0	0	0	1	1	125.20	40.000000	0	0.0	0.0	0.0	1.0	
4851	0	35.0	0	0	1	0	82.99	30.600000	0	0.0	0.0	0.0	1.0	
4852	1	51.0	0	0	1	0	166.29	25.600000	0	0.0	0.0	1.0	0.0	
4853	0	44.0	0	0	1	1	85.28	26.200000	0	1.0	0.0	0.0	0.0	

```
42]: DF.drop(columns=['work_type'],axis=1,inplace=True)
```

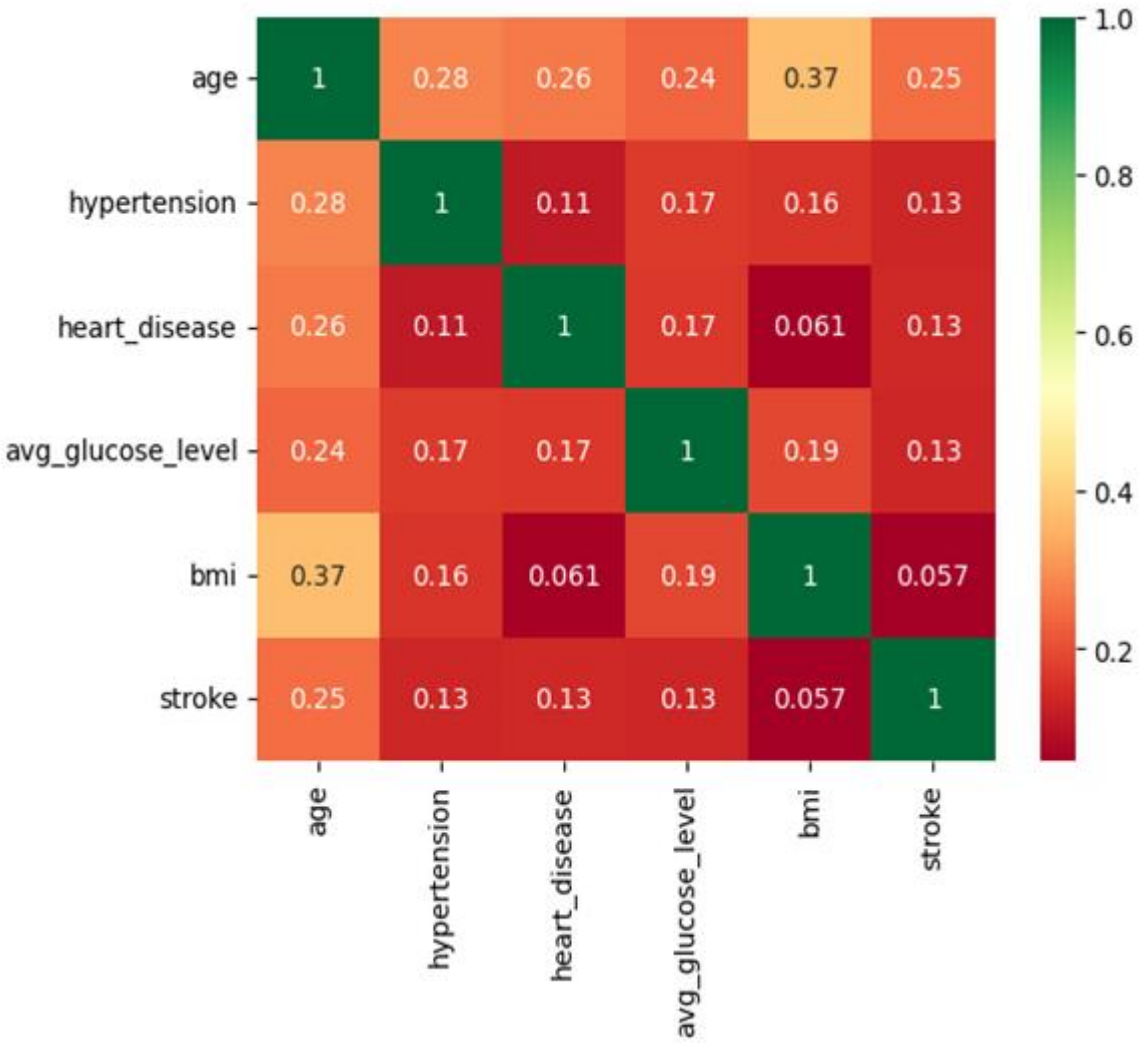
```
43]: DF
```

43]:

	gender	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_level	bmi	smoking_status	stroke	Govt_job	Never_worked	Private
0	1	67.0	0	1	1	1	228.69	36.600000	formerly smoked	1	0.0	0.0	1.0
1	0	61.0	0	0	1	0	202.21	28.893237	never smoked	1	0.0	0.0	0.0
2	1	80.0	0	1	1	0	105.92	32.500000	never smoked	1	0.0	0.0	1.0
3	0	49.0	0	0	1	1	171.23	34.400000	smokes	1	0.0	0.0	1.0
4	0	79.0	1	0	1	0	174.12	24.000000	never smoked	1	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4849	0	80.0	1	0	1	1	83.75	28.893237	never smoked	0	0.0	0.0	1.0
4850	0	81.0	0	0	1	1	125.20	40.000000	never smoked	0	0.0	0.0	0.0
4851	0	35.0	0	0	1	0	82.99	30.600000	never smoked	0	0.0	0.0	0.0
4852	1	51.0	0	0	1	0	166.29	25.600000	formerly smoked	0	0.0	0.0	1.0
4853	0	44.0	0	0	1	1	85.28	26.200000	Unknown	0	1.0	0.0	0.0

4854 rows × 19 columns

Heatmap for processed data



6.3 Model Training

Model training is the process of teaching a machine learning algorithm to recognize patterns and relationships in data so that it can make predictions or decisions when presented with new, unseen data.

Here, we divide the data in training and testing part using Skicit learn library

Libraries used

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import standard scaler
from sklearn.linear_model import LogisticRegression
```

Now we will separate the independent variables and the response variable(stroke)

```
[174]: from sklearn.model_selection import train_test_split
```

```
[175]: X=DF.drop('stroke',axis=1);X
```

[175]:

	gender	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_level	bmi	Govt_job	Private	Self-employed	children	Unknown	formerly smoke
0	1	67.0	0	1	1	1	228.69	36.600000	0.0	1.0	0.0	0.0	0.0	1.
1	0	61.0	0	0	1	0	202.21	28.893237	0.0	0.0	1.0	0.0	0.0	0.
2	1	80.0	0	1	1	0	105.92	32.500000	0.0	1.0	0.0	0.0	0.0	0.
3	0	49.0	0	0	1	1	171.23	34.400000	0.0	1.0	0.0	0.0	0.0	0.
4	0	79.0	1	0	1	0	174.12	24.000000	0.0	0.0	1.0	0.0	0.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
4849	0	80.0	1	0	1	1	83.75	28.893237	0.0	1.0	0.0	0.0	0.0	0.
4850	0	81.0	0	0	1	1	125.20	40.000000	0.0	0.0	1.0	0.0	0.0	0.
4851	0	35.0	0	0	1	0	82.99	30.600000	0.0	0.0	1.0	0.0	0.0	0.
4852	1	51.0	0	0	1	0	166.29	25.600000	0.0	1.0	0.0	0.0	0.0	1.
4853	0	44.0	0	0	1	1	85.28	26.200000	1.0	0.0	0.0	0.0	1.0	0.

4854 rows × 15 columns

The above are the independent variables stored in X.

```
[176]: Y=DF['stroke'];Y
```

```
[176]: 0       1
       1       1
       2       1
       3       1
       4       1
              ..
       4849    0
       4850    0
       4851    0
       4852    0
       4853    0
       Name: stroke, Length: 4854, dtype: int64
```

Separating the predictor variable(stroke) from the dataset.

As the dataset is about predicting that whether a person is going to have stroke based on the independent variables , it becomes easier to to understand that we will use Logistic Regression fot such case.

How logistic Regression can be implemented on Brain stroke data?

Logistic regression is a statistical method used to model the relationship between a binary outcome variable (e.g., presence or absence of stroke) and one or more predictor variables (e.g., age, hypertension status, BMI) by estimating the probability of the outcome occurring. Unlike linear regression, which predicts continuous outcomes, logistic regression predicts the probability of a categorical outcome using a logistic function.

In the context of brain stroke data, logistic regression can be implemented by:

Selecting relevant predictor variables (risk factors) from the dataset.
Encoding categorical variables and scaling numerical variables as necessary for model compatibility.
Fitting a logistic regression model to the data to estimate the relationship between the predictor variables and the likelihood of stroke occurrence.
Interpreting the coefficients of the logistic regression model to understand the impact of each predictor variable on the probability of stroke.

Performing Logistic Regression

```
[77]: import pandas as pd
      from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LogisticRegression
      from sklearn.preprocessing import StandardScaler
      from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


      # Spliting the dataset into train and test sets
      X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

      # Standardizing the features (optional but recommended for logistic regression)
      scaler = StandardScaler()
      X_train_scaled = scaler.fit_transform(X_train)
      X_test_scaled = scaler.transform(X_test)

      # Using Logistic Regression
      model = LogisticRegression()

      # Fitting the model to the training data
      model.fit(X_train_scaled, y_train)

      # Predicting on the test data
      y_pred = model.predict(X_test_scaled)

      # Evaluating the model
      accuracy = accuracy_score(y_test, y_pred)
      print("Accuracy:", accuracy)

      # Print classification report and confusion matrix
      print("\nClassification Report:")
      print(classification_report(y_test, y_pred))
```

For better results while prediction Scaling of variables during model selection

The method used to scale the variables is Standard Scaler .

```
Accuracy: 0.9402677651905252

Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       913
           1       0.00      0.00      0.00        58

    accuracy                           0.94       971
   macro avg       0.47      0.50      0.48       971
weighted avg       0.88      0.94      0.91       971


Confusion Matrix:
[[913   0]
 [ 58   0]]
```

From the above we can see that the accuracy of the model is 94% but its precision ,recall and f1 scores are 0 which is not considered as good for a logistic regression model.

Also vif among the variables is higher.

False Positive Rate

[182]:
```
import pandas as pd
import numpy as np
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Calculate VIF for each feature
vif_data = pd.DataFrame()
vif_data["Feature"] = X_train.columns
vif_data["VIF"] = [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]

print(vif_data)
```

```
              Feature        VIF
0              gender   1.755684
1                 age  13.011664
2        hypertension   1.220237
3       heart_disease   1.186759
4        ever_married   5.698942
5      Residence_type   2.051901
6   avg_glucose_level   7.949672
7                 bmi  24.525505
8            Govt_job   5.721272
9             Private  20.746119
10      Self-employed   7.243564
11           children   4.270669
12            Unknown   2.344608
13    formerly smoked   1.521265
14             smokes   1.443617
```

from the above it is evident that there is multicollinearity among the variables . So it is necessary to remove multicollinearity among the variables for predictions.

Using Principal Component Analysis

PCA is a statistical technique which is used to reduce the multicollinearity , model complexity and overfitting in the model .

PCA uses a correlation structure of original variables and derives p-linear combinations of it.

PCA fitment on the dataset

Using PCA to reduce multicollinearity

63]:
```
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Define PCA and Logistic Regression model
pca = PCA(n_components=14)



logistic_regression = LogisticRegression()

# Create a pipeline with PCA and Logistic Regression
pipeline = Pipeline([('pca', pca), ('logistic', logistic_regression)])

# Fit the pipeline on training data
pipeline.fit(X_train_scaled, y_train)

# Predict probabilities on test data
y_probs1 = pipeline.predict_proba(X_test)[:, 1]

# Calculate ROC AUC score
roc_auc = roc_auc_score(y_test, y_probs)

# Plot ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
```

From the above code we see that PCA is performed on the 14 components which are present in the dataset. Also a pipeline is created between the logistic regression and PCA.

Using PCA will address the issue with the multicollinearity amongst the variables.

Output for PCA

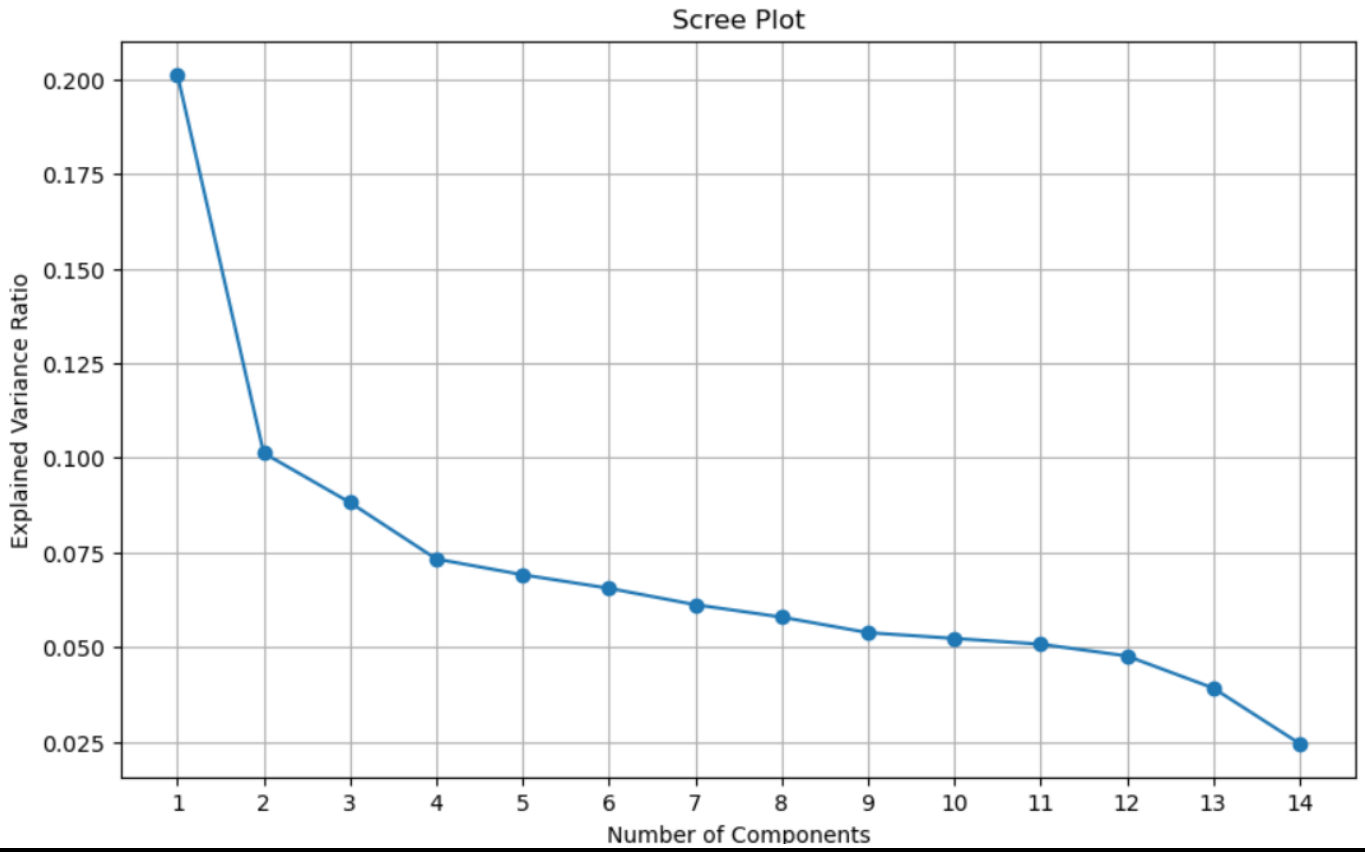
False Positive Rate

[64]:
```
# Fit PCA on the training data
pca.fit(X_train_scaled)

# Get the explained variance ratio
explained_variance_ratio = pca.explained_variance_ratio_

# Print the explained variance ratio for each component
for i, ratio in enumerate(explained_variance_ratio):
    print(f"Explained Variance Ratio for Component {i+1}: {ratio}")
```

```
Explained Variance Ratio for Component 1: 0.20129852948559226
Explained Variance Ratio for Component 2: 0.10126546949074061
Explained Variance Ratio for Component 3: 0.08817543762945976
Explained Variance Ratio for Component 4: 0.07326041131832313
Explained Variance Ratio for Component 5: 0.06909709276827386
Explained Variance Ratio for Component 6: 0.06552185869113678
Explained Variance Ratio for Component 7: 0.06118410701121933
Explained Variance Ratio for Component 8: 0.05790927990501233
Explained Variance Ratio for Component 9: 0.05379688974364739
Explained Variance Ratio for Component 10: 0.05226811040398311
Explained Variance Ratio for Component 11: 0.05075859601237715
Explained Variance Ratio for Component 12: 0.04764156676306083
Explained Variance Ratio for Component 13: 0.03915958004602245
Explained Variance Ratio for Component 14: 0.024503064685186142
```

The explained variance ratio for each component is now less suggesting that there is less multicollinearity among the variables.

Scree plot



The above fig shows that the variance among the component is decreasing after using PCA

Cross Validation

```
107]: from sklearn.model_selection import cross_val_score

# Define your logistic regression model
logistic_regression = LogisticRegression(max_iter=10000)  # Adjust parameters as needed

# Perform cross-validation
cv_scores = cross_val_score(logistic_regression, X_resampled, y_resampled, cv=5, scoring='accuracy')

# Print cross-validation scores
print("Cross-Validation Scores:", cv_scores)
print("Mean CV Score:", cv_scores.mean())
```

```
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
Cross-Validation Scores: [0.77658143 0.7973064  0.80740741 0.78787879 0.79191919]
Mean CV Score: 0.7922186422321011
```

Using XGA - Boost

Gradient Boosting: At its core, XGBoost is a type of ensemble method that relies on gradient boosting. This means it combines multiple weaker models (often decision trees) to create a stronger final model. Each tree learns from the errors of the previous one, focusing on areas where the prior Models struggled.

Extreme in Efficiency: XGBoost optimizes the gradient boosting process in several ways. It uses efficient algorithms for tree building and supports parallelization, allowing it to handle large datasets effectively.

Regularization for Better Generalization: XGBoost incorporates techniques to prevent overfitting, a common challenge in machine learning. This helps the model perform well not just on the training data but also on unseen data.

Output after XG boost

```
ResourceWarning: Enable tracemalloc to get the object alloca
```

```
import xgboost as xgb
from sklearn.metrics import accuracy_score

# Using XCB Classifier
xgb_classifier = xgb.XGBClassifier()

# Train the classifier
xgb_classifier.fit(X_train_pca, y_train)

# Make predictions on the test set
y_pred = xgb_classifier.predict(X_test_pca)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9382080329557158
```

Now the model is perfect fit .

6.4 Model Evaluation

Model evaluation is a crucial step in the machine learning process. It's essentially checking the model's homework to see if it learned its lesson well and can apply its knowledge to new situations.

Metrics for evaluation:

There are various metrics used for model evaluation, depending on the type of problem you're trying to solve. Here are some common ones:

- **Classification:** Accuracy, precision, recall, F1-score

```
version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

[127]:
```
from sklearn.metrics import confusion_matrix

# Generate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print the confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
```

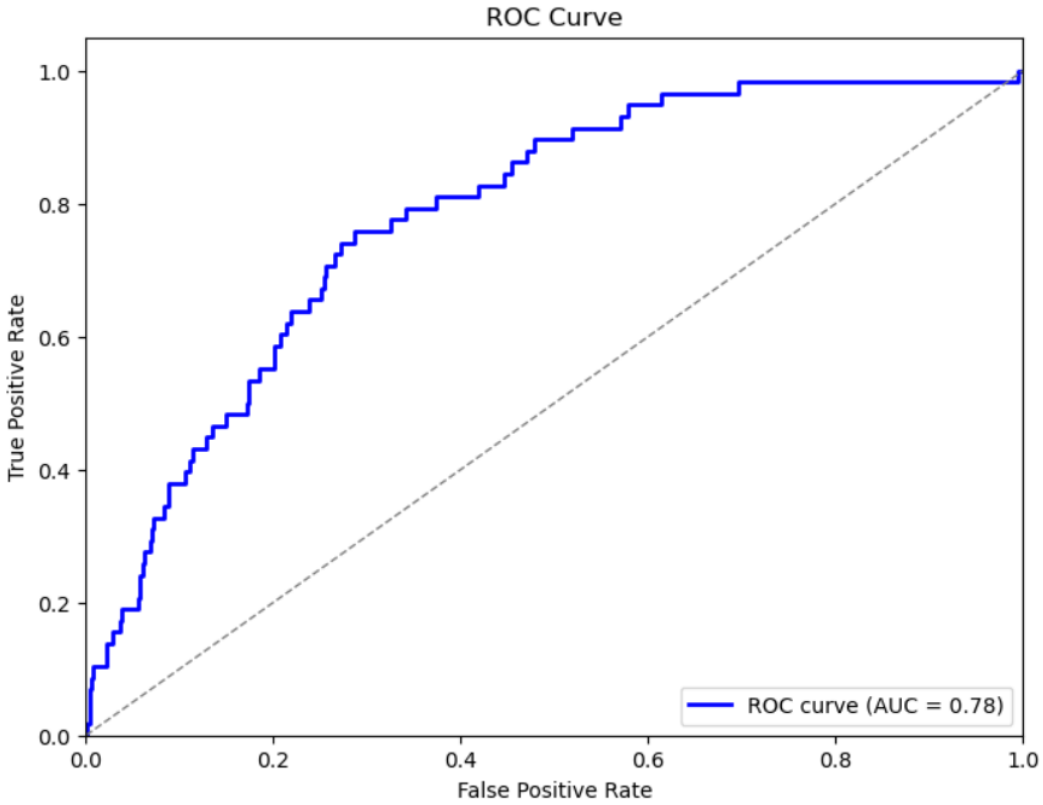
```
Confusion Matrix:
[[906   7]
 [ 53   5]]
```

[141]:
```
from sklearn.metrics import classification_report

# Generate a classification report
report = classification_report(y_test, y_pred)

# Print the classification report
print(report)
```

```
              precision    recall  f1-score   support

           0       0.94      0.99      0.96       913
           1       0.15      0.03      0.06        58

    accuracy                           0.93       971
   macro avg       0.55      0.51      0.51       971
weighted avg       0.89      0.93      0.91       971
```

ROC Curve

```
if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```



Comparison with Gaussian NB

```
print("accuracy",accuracy)
```

```
Precision: 0.14
Recall: 0.603448275862069
F1-score: 0.2272727272727273
accuracy 0.7548918640576725
```

The accuracy of Gaussian NB algorithm is very less as compared to the logistic regression model after XG boost.

Analysis on R

Filter

	gender	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_level	bmi	stroke	Gov
1	1	67	0	1	1	1	228.69	36.60000	1	
2	0	61	0	0	1	0	202.21	28.89324	1	
3	1	80	0	1	1	0	105.92	32.50000	1	
4	0	49	0	0	1	1	171.23	34.40000	1	
5	0	79	1	0	1	0	174.12	24.00000	1	
6	1	81	0	0	1	1	186.21	29.00000	1	
7	1	74	1	1	1	0	70.09	27.40000	1	
8	0	69	0	0	0	1	94.39	22.80000	1	
9	0	59	0	0	1	0	76.15	28.89324	1	
10	0	78	0	0	1	1	58.57	24.20000	1	
11	0	81	1	0	1	0	80.43	29.70000	1	
12	0	61	0	1	1	0	120.46	36.80000	1	

Showing 1 to 12 of 4,854 entries, 18 total columns

Console Terminal Background Jobs

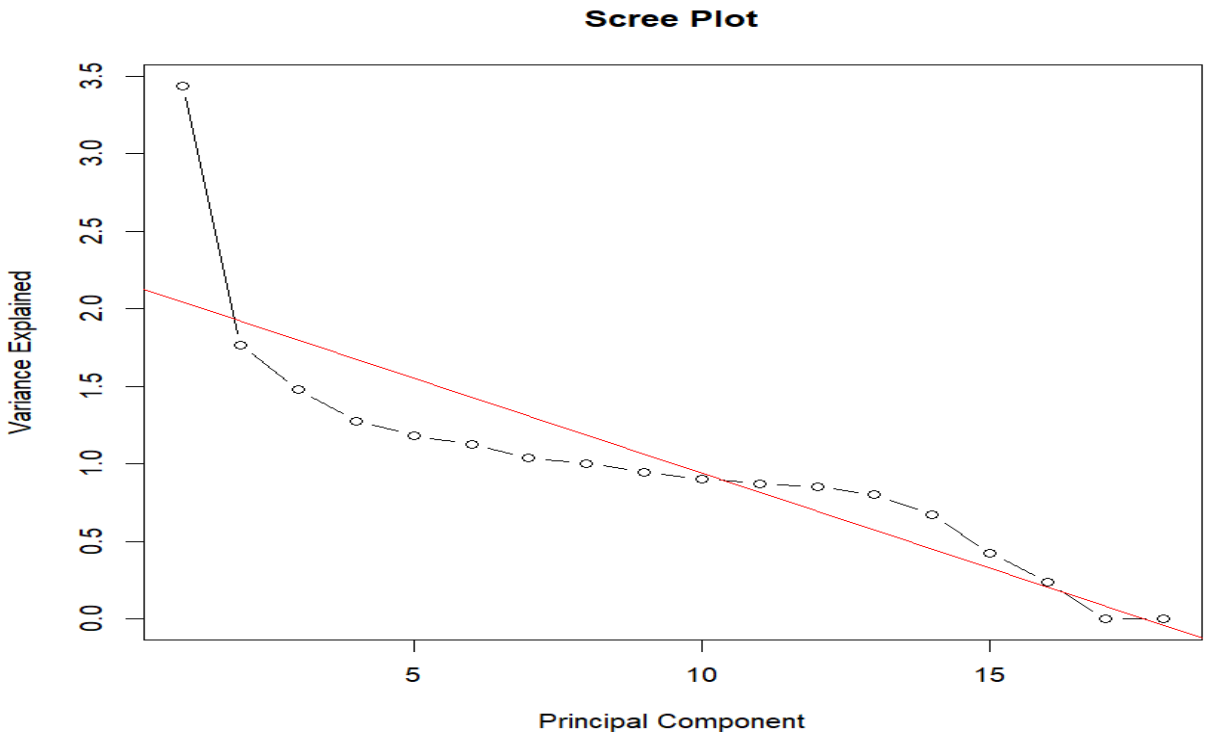
R 4.4.0 · D:/Project/

```
Call:
lm(formula = pca_result$sdev^2 ~ seq_along(pca_result$sdev))

Residuals:
     Min       1Q   Median       3Q      Max
-0.39802 -0.24882 -0.05875  0.08559  1.39526

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                2.16379    0.20307   10.65 1.13e-08 ***
seq_along(pca_result$sdev) -0.12250    0.01876   -6.53 6.94e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.413 on 16 degrees of freedom
Multiple R-squared:  0.7271,    Adjusted R-squared:  0.7101
F-statistic: 42.64 on 1 and 16 DF,  p-value: 6.937e-06
```

By analyzing the summary of we conclude that

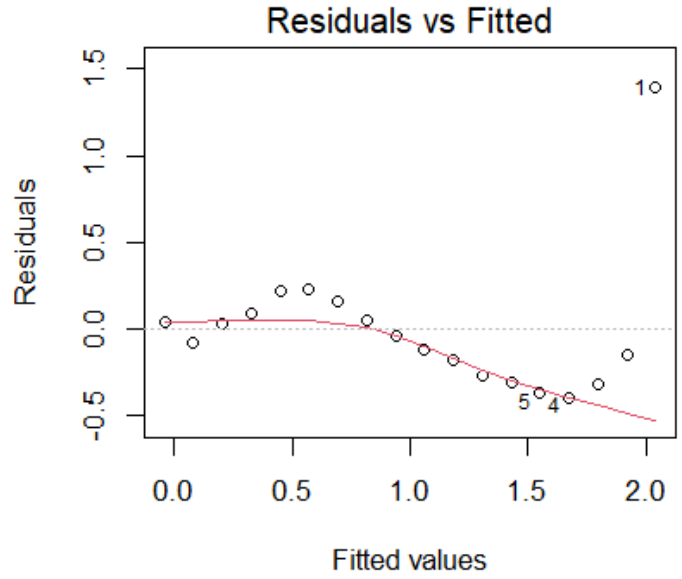
1. R-squared value is higher 72% which indicates the model is a good fit.
2. F-statistic with 16 degrees of freedom.
3. p- value is less than 0.05 .

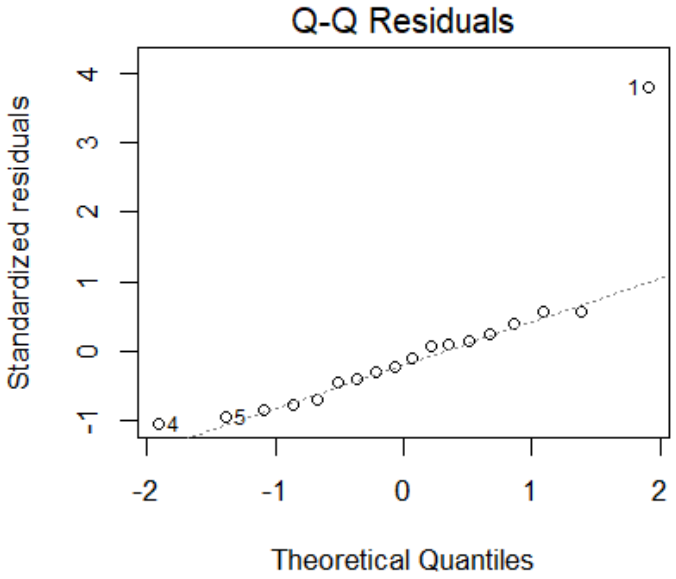
Line fitment

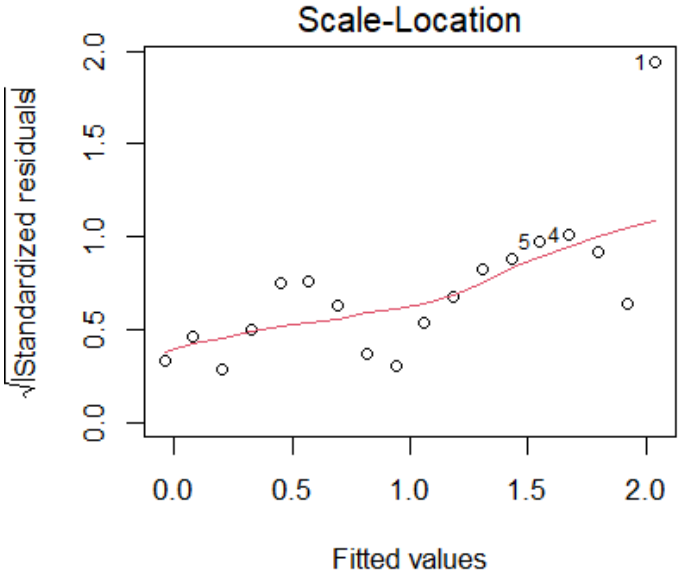


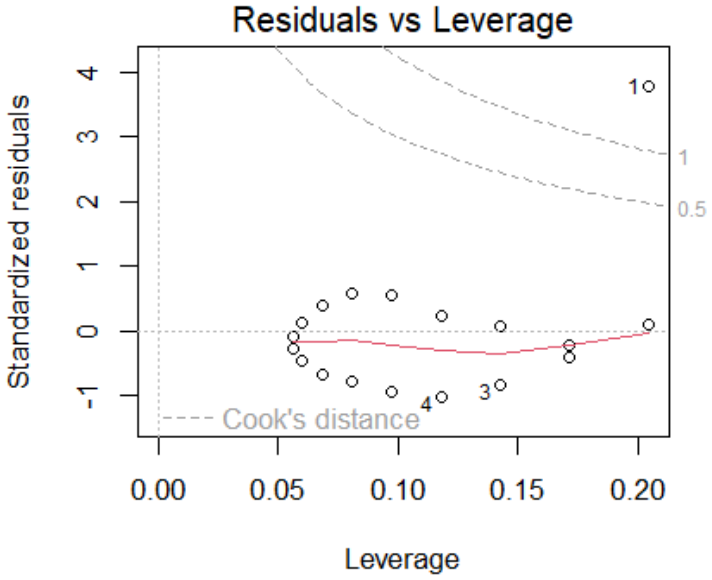
Equation of line

"Variance Explained = 2.16379 - 0.1225 * Principal Component Number"









Analysis of Variance Table

Response: pca_result$sdev^2

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
seq_along(pca_result$sdev)	1	7.2710	7.2710	42.637	6.937e-06	***
Residuals	16	2.7285	0.170			

Example on Random dataset

Performance on random dataset

[2]:
```
from sklearn.ensemble import RandomForestClassifier
```

119]:
```
# Initialize the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier
rf_classifier.fit(X_train, y_train)
```

119]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

[123]:
```
 gender=int(input("enter gender=",))
age=int(input("enter age=",))
glucose_level=int(input("enter your glucose_level=",))
hypertension=int(input("enter hypertension=",))
bmi=int(input("enter your bmi=",))
govt_job=int(input("enter govt_job=",))
never_worked=int(input("enter work=",))
ever_married=int(input("enter ms=",))
private=int(input("enter pri=",))
self_emlployeed=int(input("enter es=",))
children=int(input("enter child=",))
unknown=int(input("enter uk=",))
formerly=int(input("enter f=",))
never=int(input("enter ne=",))
smokes=int(input("enter sm=",))
heart_disease=int(input("enter hd=",))
residence=int(input("enter rt=",))
```

```
enter gender= 1
enter age= 45
enter your glucose_level= 70
enter hypertension= 0
enter your bmi= 30
enter govt_job= 1
enter work= 1
enter ms= 1
enter pri= 0
enter es= 0
enter child= 0
enter uk= 0
enter f= 1
enter ne= 0
enter sm= 1
enter hd= 1
enter rt= 0
```

[127]:
```
patients_data=[gender,age,glucose_level,hypertension,bmi,govt_job,never_worked,ever_married,private,self_emlployeed,children,unknown,formerly,never,smoke
```

[127]:
```
[1, 86, 75, 1, 32, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0]
```

[128]:
```
probs=rf_classifier.predict_proba([patients_data]);probs
brain_stroke_prob=probs[0][1]*100;brain_stroke_prob
```

[128]:
```
28.000000000000004
```

[129]:
```
print(f"probability of brain stroke in future={brain_stroke_prob}%")
```

```
probability of brain stroke in future=28.000000000000004%
```

CONCLUSION

We created a logistic regression model and also used a pipeline in order to over sample and under sample the data. We were able to successfully predict the values of brainstroke prediction upto 93% accuracy.

OUTPUT:

Accuracy = 93%

Also, key factors for a person to have stroke is Age as it is having positive correlation with stroke and also hypertension can be considered as a key factor.

Also after doing analysis on R - software we found that R-squared value is higher which suggests the model is good fit

CHAPTER 7

BIBLIOGRAPHY

Books:

1]. JEENA R S, " Stroke Prediction Using SVM" , 2016 International Conferecnce on Control Instrumentation, Communication and Computational Technologies (ICCICCT),978-1-5240- 076/$31.00@2016IEEE, https://ieeexplore.ieee.org/document/8079581

[2]. M. S. Singh and P. Choudhary, "Stroke prediction using artificial intelligence," 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), 2017, pp. 158- 161,doi: 10.1109/IEMECON.2017.8079581. https://ieeexplore.ieee.org/document/8079581

[3]. JAEHAK YU1 , SEJIN PARK 2 , SOON-HYUN KWON1 , KANG-HEE CHO3 , AND HANSUNG LEE 4 , " AI-Based Stroke Disease Prediction System Using Real-Time ElectromyographySignals",IEEEAccess,volume10,2022. . https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9761215

[4]. .H. Mcheick, H. Nasser, M. Dbouk and A. Nasser, "Stroke Prediction Context-Aware Health Care System," 2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2016, pp. 30-35, doi: 10.1109/CHASE.2016.49. https://ieeexplore.ieee.org/document/7545809

[5]. .P. A, N. G, V. K. R, P. P and S. R. R.V.T, "Stroke Prediction System Using Artificial Neural Network," 2021 6th International Conference on Communication and Electronics Systems (ICCES), 2021, pp. 1898- 1902, doi: 10.1109/ICCES51350.2021.9489055. https://ieeexplore.ieee.org/document/9489055

CHAPTER 8

GLOSSARY OF TERMS

Glossary of Machine Learning Terms:

- **Machine Learning:** A field of computer science that allows computers to learn from data without explicit programming.

- **Model:** A representation of the learned patterns from data that can be used to make predictions on new data.

- **Overfitting:** When a model memorizes the training data too well and fails to generalize to unseen data.

- **Gradient Boosting:** An ensemble machine learning technique that combines multiple weak models (like decision trees) into a stronger final model.

- **XGBoost (eXtreme Gradient Boosting):** A powerful machine learning algorithm known for its efficiency and accuracy in gradient boosting.

- **Regularization:** Techniques to prevent overfitting in machine learning models.

- **Model Evaluation:** The process of assessing how well a model performs on unseen data.

- **Training Set:** The data used to train the model and help it learn patterns.

- **Validation Set:** The data used to fine-tune the model's hyperparameters (settings that control the learning process).

- **Test Set:** Unseen data used for the final evaluation of the model's generalizability.

- **Metrics:** Measures used to evaluate a model's performance. Examples include accuracy, precision, recall, F1-score (for classification).