DIC Project Phase 1

Report

A Holistic Approach to Heart Stroke Prevention: Analyzing the Contributions of Clinical Variables and Risk Factors

Problem Statement

Heart disease, encompassing conditions such as coronary artery disease, arrhythmias, congenital defects, heart muscle diseases, and heart valve disorders, is a leading cause of morbidity and mortality worldwide. With the global population aging, heart disease has become an increasingly significant public health challenge, placing a substantial economic burden on healthcare systems. This problem statement aims to address the intricate relationship between clinical variables and risk factors in shaping cardiovascular health, specifically within the context of heart stroke occurrence.

Background of the Problem

Heart disease, in its various forms, is a pervasive health issue that affects millions of individuals and poses substantial challenges to healthcare systems. In the United States and many other countries, it ranks among the top causes of death. As the population ages, the prevalence of heart disease is on the rise, exacerbating its impact on individuals and society.

One of the critical aspects of heart disease is its complexity. It doesn't have a one-size-fits-all solution, as it encompasses numerous conditions with varying causes and risk factors. This complexity necessitates a deeper understanding of how clinical variables (e.g., age, genetics, medical history) and risk factors (e.g., hypertension, high cholesterol, smoking) intersect and contribute to the development and occurrence of heart strokes, which include heart attacks, arrhythmias, and heart failure.

Understanding this intricate relationship is crucial for several reasons:

A. Significance of the Problem

a. Public Health Impact: Heart disease is a leading cause of death worldwide, and it affects people of all ages and backgrounds. As the aging population grows, the burden of heart disease on healthcare systems and individuals is increasing.

b. Economic Burden: Treating heart disease, especially advanced cardiovascular conditions, places a significant financial strain on healthcare systems and individuals. Preventive measures can potentially reduce this economic burden.

c. Complexity of Heart Disease: The multifaceted nature of heart disease demands a deeper understanding of its various contributing factors to develop effective prevention and treatment strategies.

B. Potential Contribution of the Project

a. Targeted Prevention: A project that deciphers the interplay of clinical variables and risk factors can lead to highly targeted prevention strategies. These strategies can help identify individuals at higher risk and tailor interventions to their specific needs.
b. Personalized Medicine: By gaining insights into how individual characteristics influence heart stroke risk, this project can pave the way for personalized medicine approaches, optimizing treatment and outcomes.
c. Informed Healthcare Policy: The research outcomes can inform healthcare policies and guidelines, enabling policymakers to develop evidence-based strategies for cardiovascular disease prevention and management.
d. Economic Benefits: Better understanding and prevention can lead to substantial cost savings in healthcare, making the system more sustainable and accessible to all segments of society.

In conclusion, the project's goal of unraveling the relationship between clinical variables and risk factors in heart stroke occurrence is of paramount significance due to the growing burden of heart disease and its economic impact. This project has the potential to contribute by improving prevention, treatment, and healthcare policy, ultimately leading to better cardiovascular health outcomes and reduced societal costs.

Data Sources

The dataset used for this project was acquired from **Kaggle** and is accessible via the following link: Heart Disease Dataset.

Description of the Data

This dataset contains crucial information pertaining to heart disease and cardiovascular health. It encompasses a wide range of columns, each providing valuable insights into the factors that influence heart health and the occurrence of heart strokes. Importantly, this dataset comprises a substantial volume of data, consisting of approximately 4000 records (rows) and a number of columns.

Below is a comprehensive overview of the dataset, highlighting the significance of various columns for the analysis:

1. **Gender**: This column records the gender of individuals, which is fundamental for gender-based analysis of health-related data.
2. **Age**: Age is a critical demographic variable as it plays a pivotal role in categorizing individuals into different age groups, enabling age-specific health assessments and risk analysis.
3. **Education**: The education level of each individual is indicated in this column. Education can significantly impact health outcomes, as individuals with higher education levels may exhibit different health behaviors and have better access to healthcare.
4. **Current Smoker**: This binary column (0 or 1) indicates whether an individual is a current smoker. Smoking is a major risk factor for various health issues, including heart disease and stroke.
5. **Cigarettes Per Day** (cigsPerDay): For individuals who are smokers, this column quantifies their smoking intensity by specifying the number of cigarettes smoked per day.
6. **BPMeds**: This binary column indicates whether an individual is taking blood pressure medication. Identifying medication usage is essential for understanding how individuals manage hypertension.
7. **Prevalent Stroke**: This column informs us whether an individual has experienced a stroke in the past. Information regarding past strokes is vital for assessing stroke risk and implementing preventive measures.
8. **Prevalent Hypertension** (prevalentHyp): A binary column indicating whether an individual has prevalent hypertension (high blood pressure), a significant risk factor for heart disease.
9. **Diabetes**: This binary column indicates whether an individual has diabetes, a chronic condition that can profoundly impact cardiovascular health.
10. **Total Cholesterol** (totChol)**:** Total cholesterol levels are measured in this column. Elevated cholesterol levels can contribute to atherosclerosis and heart disease.

11. **Systolic Blood Pressure** (sysBP)**:** This column records the systolic blood pressure, which represents the pressure in the arteries when the heart beats. Elevated systolic blood pressure is associated with an increased risk of cardiovascular events.
12. **Diastolic Blood Pressure** (diaBP)**:** Diastolic blood pressure is recorded in this column, representing the pressure in the arteries when the heart is at rest between beats. Elevated diastolic blood pressure is another significant risk factor for heart disease.
13. **BMI** (Body Mass Index)**:** BMI is a measure of body fat based on an individual's height and weight. It serves as an essential indicator of overall health, particularly concerning obesity-related health risks.
14. **Heart Rate:** This column provides the heart rate of each individual. Heart rate can serve as an indicator of cardiovascular health and fitness.
15. **Glucose:** Glucose levels in the blood are measured in this column. Elevated glucose levels can indicate diabetes or prediabetes.
16. **Heart Stroke** (Heart_Stroke)**:** Similar to the "Prevalent Stroke" column, this binary column indicates whether an individual has experienced a heart stroke. Heart strokes are a specific type of stroke that are closely related to heart health.

In summary, this dataset, with its almost 4,000 records and comprehensive set of variables, has the potential to significantly contribute to the field of cardiovascular health by offering a deeper understanding of the factors influencing heart disease and heart strokes. This understanding can lead to more effective prevention, intervention, and policy decisions, ultimately improving the health and well-being of individuals and reducing the societal burden of cardiovascular diseases.

Data Cleaning/Processing

Step 1: Check for Datatypes

The output presented provides a summary of the data types of each column, and it is essential for ensuring that data is appropriately represented and ready for further processing and analysis.

Code:

```
# Check for data types
print(df.dtypes)

```

Output:

```
Gender              object
age                  int64
education           object
currentSmoker        int64
cigsPerDay         float64
BPMeds             float64
prevalentStroke     object
prevalentHyp         int64
diabetes             int64
totChol            float64
sysBP              float64
diaBP              float64
BMI                float64
heartRate          float64
glucose            float64
Heart_ stroke       object
dtype: object
```

Overall, the datatypes align with the nature of the data in each column, ensuring that the dataset is structured correctly for analysis.

Step 2: Remove Irrelevant Columns

During the data cleaning process, we reviewed the dataset and assessed the relevance of each column for our analysis. Since the dataset we initially obtained contained all the relevant columns for our analysis, we didn't remove any existing columns.
However, as a part of the data preparation, we introduced a new column named "Patient_id."
Code

```
# Generate unique patient IDs
patient_ids = range(1, len(df) + 1)

# Add the patient ID column to the DataFrame
df['Patient_ID'] = patient_ids

# Print the updated DataFrame with the new 'Patient_ID' column
print(df)
```

Output

```
      Gender  age      education  currentSmoker  cigsPerDay  BPMeds  \
0       Male   39   postgraduate              0         0.0     0.0
1     Female   46  primaryschool              0         0.0     0.0
2       Male   48     uneducated              1        20.0     0.0
3     Female   61       graduate              1        30.0     0.0
4     Female   46       graduate              1        23.0     0.0
...      ...  ...            ...            ...         ...     ...
4233    Male   50     uneducated              1         1.0     0.0
4234    Male   51       graduate              1        43.0     0.0
4235  Female   48  primaryschool              1        20.0     NaN
4236  Female   44     uneducated              1        15.0     0.0
4237  Female   52  primaryschool              0         0.0     0.0

     prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                 no             0         0    195.0  106.0   70.0  26.97
1                 no             0         0    250.0  121.0   81.0  28.73
2                 no             0         0    245.0  127.5   80.0  25.34
3                 no             1         0    225.0  150.0   95.0  28.58
4                 no             0         0    285.0  130.0   84.0  23.10
...              ...           ...       ...      ...    ...    ...    ...
4233              no             1         0    313.0  179.0   92.0  25.97
4234              no             0         0    207.0  126.5   80.0  19.71
4235              no             0         0    248.0  131.0   72.0  22.00
4236              no             0         0    210.0  126.5   87.0  19.16
4237              no             0         0    269.0  133.5   83.0  21.47

      heartRate  glucose Heart_ stroke  Patient_ID
0          80.0     77.0            No           1
1          95.0     76.0            No           2
2          75.0     70.0            No           3
3          65.0    103.0           yes           4
4          85.0     85.0            No           5
...         ...      ...           ...         ...
4233       66.0     86.0           yes        4234
4234       65.0     68.0            No        4235
4235       84.0     86.0            No        4236
4236       86.0      NaN            No        4237
4237       80.0    107.0            No        4238

[4238 rows x 17 columns]
```

This column was added for identification and tracking purposes and was not present in the original dataset. Subsequently, in later steps of data processing, this "Patient_id" column was removed. The introduction and removal of this column served as an internal operation to enhance data management and tracking while ensuring that only relevant columns were retained for our analysis.

Code

```
df=df.drop([ 'Patient_ID'], axis = 1)
print(df)
```

Output

```
      Gender  age      education  currentSmoker  cigsPerDay  BPMeds  \
0       Male   39   postgraduate              0         0.0     0.0
1     Female   46  primaryschool              0         0.0     0.0
2       Male   48     uneducated              1        20.0     0.0
3     Female   61       graduate              1        30.0     0.0
4     Female   46       graduate              1        23.0     0.0
...      ...  ...            ...            ...         ...     ...
4233    Male   50     uneducated              1         1.0     0.0
4234    Male   51       graduate              1        43.0     0.0
4235  Female   48  primaryschool              1        20.0     NaN
4236  Female   44     uneducated              1        15.0     0.0
4237  Female   52  primaryschool              0         0.0     0.0

     prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  \
0                 no             0         0    195.0  106.0   70.0  26.97
1                 no             0         0    250.0  121.0   81.0  28.73
2                 no             0         0    245.0  127.5   80.0  25.34
3                 no             1         0    225.0  150.0   95.0  28.58
4                 no             0         0    285.0  130.0   84.0  23.10
...              ...           ...       ...      ...    ...    ...    ...
4233              no             1         0    313.0  179.0   92.0  25.97
4234              no             0         0    207.0  126.5   80.0  19.71
4235              no             0         0    248.0  131.0   72.0  22.00
4236              no             0         0    210.0  126.5   87.0  19.16
4237              no             0         0    269.0  133.5   83.0  21.47

      heartRate  glucose Heart_ stroke
0          80.0     77.0            No
1          95.0     76.0            No
2          75.0     70.0            No
3          65.0    103.0           yes
4          85.0     85.0            No
...         ...      ...           ...
4233       66.0     86.0           yes
4234       65.0     68.0            No
4235       84.0     86.0            No
4236       86.0      NaN            No
4237       80.0    107.0            No

[4238 rows x 16 columns]
```

Step 3: Fill Null Values for Categorical Columns

```
# Check for missing values
print(df.isna().sum())
```

```
Gender               0
age                  0
education          105
currentSmoker        0
cigsPerDay          29
BPMeds              53
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
Heart_ stroke        0
Patient_ID           0
dtype: int64
```

Before the data cleaning process, the dataset contained varying numbers of null values in different columns, ranging from 0 to 388 null values per column.

```
## Null values in categorical variables
df.education = df.education.fillna(df.education.mode().iloc[0])  # Fill null values in 'education' with the mode
df.BPMeds = df.BPMeds.fillna(df.BPMeds.mode().iloc[0])  # Fill null values in 'BPMeds' with the mode
```

Filled null values in categorical columns using appropriate methods. This ensures that no categorical data is missing.

```
## Check whether null values have been removed or not, after treating them
null_values = pd.isna(df).sum()  # Count the remaining null values in the dataframe
null_values = null_values[null_values > 0]  # Select columns with remaining null values
print(null_values)  # Print the columns with remaining null values
```

```
Series([], dtype: int64)
```

Step 4: Fill Null Values for Numerical Columns

Similar to step 3, fill null values in numerical columns using suitable methods. This ensures that no numerical data is missing.

```
# Check for missing values
print(df.isna().sum())
```

```
Gender               0
age                  0
education          105
currentSmoker        0
cigsPerDay          29
BPMeds              53
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
Heart_ stroke        0
Patient_ID           0
dtype: int64
```

```
## Now we proceed with our original plan to remove null values

## Null values in numerical variables
df.glucose.fillna(df.glucose.median(), inplace=True)  # Fill null values in 'glucose' with the median
df.cigsPerDay.fillna(df.cigsPerDay.median(), inplace=True)  # Fill null values in 'cigsPerDay' with the median
df.totChol.fillna(df.totChol.median(), inplace=True)  # Fill null values in 'totChol' with the median
df.BMI.fillna(df.BMI.median(), inplace=True)  # Fill null values in 'BMI' with the median
df.heartRate.fillna(df.heartRate.median(), inplace=True)  # Fill null values in 'heartRate' with the median
```

```
## Check whether null values have been removed or not, after treating them
null_values = pd.isna(df).sum()  # Count the remaining null values in the dataframe
null_values = null_values[null_values > 0]  # Select columns with remaining null values
print(null_values)  # Print the columns with remaining null values
```

```
Series([], dtype: int64)
```

After the data cleaning and null value imputation steps, all null values have been successfully addressed, and the dataset is now free of missing data, making it complete for analysis.

Step 5: Remove Rows with Missing Values

```
# Check for missing values
print(df.isna().sum())
```

```
Gender             0
age                0
education          0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          0
glucose            0
Heart_ stroke      0
Heart_Stroke       0
dtype: int64
```

We do not have any rows with missing values, as shown above as we have already ensure that in step 3 and step 4.

Step 6: Remove Duplicate Rows

Checked for and removed any duplicate rows to ensure each record is unique.

```
# Remove any duplicate rows
df = df.drop_duplicates()
```

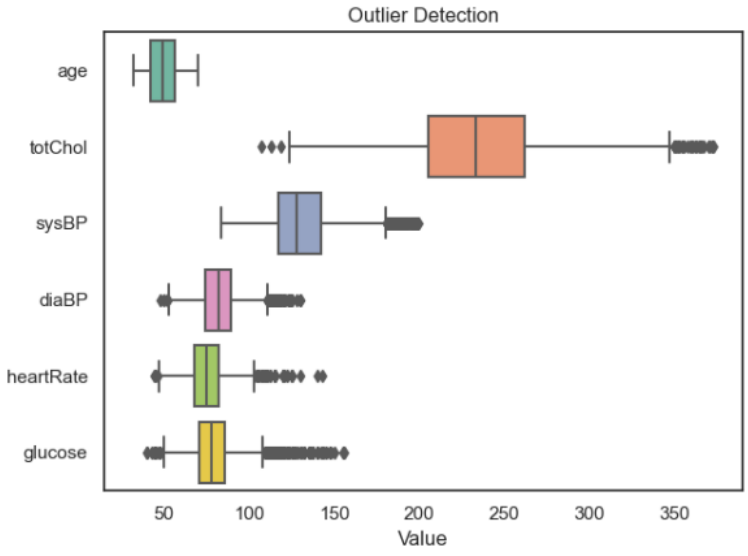
Step 7: Remove Outliers

Conducted outlier detection and removal for numerical features to maintain data integrity.

The code utilizes a filter mechanism to remove rows where certain numerical variables exceed predefined threshold values. Specifically, it removes rows where "totChol" (total cholesterol) is greater than or equal to 380, "sysBP" (systolic blood pressure) is greater than or equal to 200, "diaBP" (diastolic blood pressure) is greater than or equal to 170, "heartRate" is greater than or equal to 150, and "glucose" is greater than or equal to 160.

```
1
2  # Remove any outliers
3  df = df[~((df["totChol"] >= 380) | (df["sysBP"] >= 200) | (df["diaBP"] >= 170) | (df["heartRate"] >= 150) | (df["glucose"] >
4
5  # Check for outliers
6  columns_to_check = ["age", "totChol", "sysBP", "diaBP", "heartRate", "glucose"]
7
8  # Create a boxplot with Seaborn
9  sns.boxplot(data=df[columns_to_check], orient="h", palette="Set2")
10 plt.title("Outlier Detection")
11 plt.xlabel("Value")
12 plt.show()
```

The purpose of this operation is to eliminate data points that are considered outliers based on these threshold values. Outliers can have a significant impact on statistical analyses and machine learning models, and their removal helps ensure the robustness and reliability of the analysis.
After executing this code, the dataset will contain only the data points that fall within the specified ranges for the mentioned numerical features, removing any observations that exceed these threshold values.



Step 8: Convert Categorical Data to Dummies

Converted categorical data into dummy variables to facilitate analysis**.**

```
# Convert categorical columns to dummy variables
df1 = pd.get_dummies(df, columns=["Gender"])
print(df1)
```

The "Gender" column was selected for conversion into dummy variables, which represents gender as binary values: "Gender_Female" and "Gender_Male".

```
      age      education  currentSmoker  cigsPerDay  BPMeds prevalentStroke  \
0      39   postgraduate              0         0.0     0.0              no
1      46  primaryschool              0         0.0     0.0              no
2      48     uneducated              1        20.0     0.0              no
3      61       graduate              1        30.0     0.0              no
4      46       graduate              1        23.0     0.0              no
...   ...            ...            ...         ...     ...             ...
4233   50     uneducated              1         1.0     0.0              no
4234   51       graduate              1        43.0     0.0              no
4235   48  primaryschool              1        20.0     0.0              no
4236   44     uneducated              1        15.0     0.0              no
4237   52  primaryschool              0         0.0     0.0              no

      prevalentHyp  diabetes  totChol  sysBP  diaBP    BMI  heartRate  \
0                0         0    195.0  106.0   70.0  26.97       80.0
1                0         0    250.0  121.0   81.0  28.73       95.0
2                0         0    245.0  127.5   80.0  25.34       75.0
3                1         0    225.0  150.0   95.0  28.58       65.0
4                0         0    285.0  130.0   84.0  23.10       85.0
...            ...       ...      ...    ...    ...    ...        ...
4233             1         0    313.0  179.0   92.0  25.97       66.0
4234             0         0    207.0  126.5   80.0  19.71       65.0
4235             0         0    248.0  131.0   72.0  22.00       84.0
4236             0         0    210.0  126.5   87.0  19.16       86.0
4237             0         0    269.0  133.5   83.0  21.47       80.0

      glucose Heart_ stroke  Gender_Female  Gender_Male
0        77.0            No              0            1
1        76.0            No              1            0
2        70.0            No              0            1
3       103.0           yes              1            0
4        85.0            No              1            0
...       ...           ...            ...          ...
4233     86.0           yes              0            1
4234     68.0            No              0            1
4235     86.0            No              1            0
4236     78.0            No              1            0
4237    107.0            No              1            0

[4131 rows x 17 columns]
```

Step 9: Scale Numerical Features

Scaled numerical features to standardize the data and improve model performance.

```
# Scale numerical features
scaler = StandardScaler()
df[["age", "totChol", "sysBP", "diaBP", "heartRate", "glucose"]] = scaler.fit_transform(df[["age", "totChol", "sysBP", "diaB
```

Step 10: Make Data Consistent

```
df['Gender'] = df['Gender'].str.lower()
df['Heart_ stroke'] = df['Heart_ stroke'].str.lower()

```

```
## Let's check the labels in categorical features

for col in df.columns:
    if df[col].dtype=='object':
        print()
        print(col)
        print(df[col].unique())
```

The data cleaning step ensures consistency in the text data by converting all characters to lowercase. This standardization can help in subsequent data processing and analysis by making text data more uniform and eliminating potential discrepancies due to variations in capitalization.


```
df['Gender'] = df['Gender'].str.lower()
df['Heart_ stroke'] = df['Heart_ stroke'].str.lower()

```

```
## Let's check the labels in categorical features

for col in df.columns:
    if df[col].dtype=='object':
        print()
        print(col)
        print(df[col].unique())
```


```
Gender
['male' 'female']

education
['postgraduate' 'primaryschool' 'uneducated' 'graduate']

prevalentStroke
['no' 'yes']

Heart_ stroke
['no' 'yes']
```

Step 11: Save Data and Split the Dataset

Saved the cleaned dataset and split it into appropriate subsets for further analysis.

```
# Split the data into training and testing sets
X = df.drop(['Heart_ stroke'], axis = 1)
y = df['Heart_ stroke']
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df.drop("Heart_ stroke", axis=1), df["Heart_ stroke"], test_size=0.25, r
print("X_train number of rows:", X_train.shape[0])
print("X_test number of rows:", X_test.shape[0])
print("y_train number of rows:", y_train.shape[0])
print("y_test number of rows:", y_test.shape[0])
```


```
# Split the data into training and testing sets
X = df.drop(['Heart_ stroke'], axis = 1)
y = df['Heart_ stroke']
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df.drop("Heart_ stroke", axis=1), df["Heart_ stroke"], test_size=0.25, r
print("X_train number of rows:", X_train.shape[0])
print("X_test number of rows:", X_test.shape[0])
print("y_train number of rows:", y_train.shape[0])
print("y_test number of rows:", y_test.shape[0])
```


```
X_train number of rows: 3098
X_test number of rows: 1033
y_train number of rows: 3098
y_test number of rows: 1033
```

The output summarizes the number of rows in the training and testing datasets for both the feature matrix (X) and the target variable (y):

Training data (X_train): 3,098 rows.
Testing data (X_test): 1,033 rows.
Training target (y_train): 3,098 rows.
Testing target (y_test): 1,033 rows.

This information is crucial for understanding the dataset's split between training and testing sets, which is fundamental in machine learning for model development and evaluation.

Summary:

The dataset was thoroughly cleaned and processed to ensure data quality and relevance for analysis. The steps involved in this process were documented with comments and markup, and the dataset is now ready for further analysis and modeling.

Exploratory Data Analysis (EDA)

Step 1: Check for Datatypes

Code:

```
# Check for data types
print(df.dtypes)

```

Output:

```
Gender              object
age                  int64
education           object
currentSmoker        int64
cigsPerDay         float64
BPMeds             float64
prevalentStroke     object
prevalentHyp         int64
diabetes             int64
totChol            float64
sysBP              float64
diaBP              float64
BMI                float64
heartRate          float64
glucose            float64
Heart_ stroke       object
dtype: object
```

Checked the datatypes of all columns to ensure they are suitable for their respective data. Ensuring that data is correctly represented is crucial for analysis and modeling.

Step 2: Check the shape of the DataFrame

Checked the shape of the DataFrame to understand the dimensions, i.e., the number of rows and columns. This provides an overview of the dataset's size.

Code

```
print(f'\033[94mNumber of records (rows) in the dataset are: {df.shape[0]}')
print(f'\033[94mNumber of features (columns) in the dataset are: {df.shape[1]}')
print(f'\033[94mNumber of values in the dataset are: {df.count().sum()}')
print(f'\033[94mNumber missing values in the dataset are: {sum(df.isna().sum())}')
```

Output

```
Number of records (rows) in the dataset are: 4238
Number of features (columns) in the dataset are: 16
Number of values in the dataset are: 67163
Number missing values in the dataset are: 645
```

Understanding the dataset's size and the number of features is essential for planning data processing and analysis tasks.

Step 3: Generate summary statistics for the dataset

Generated summary statistics for the dataset to get an initial understanding of the data distribution, central tendencies, and spread of numerical columns.
Code

```
df.describe(include='all').T
```

Output

```
df.describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Gender	4238	2	Female	2419	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	4238.0	NaN	NaN	NaN	49.584946	8.57216	32.0	42.0	49.0	56.0	70.0
education	4238	4	uneducated	1825	NaN	NaN	NaN	NaN	NaN	NaN	NaN
currentSmoker	4238.0	NaN	NaN	NaN	0.494101	0.500024	0.0	0.0	0.0	1.0	1.0
cigsPerDay	4238.0	NaN	NaN	NaN	8.941482	11.902399	0.0	0.0	0.0	20.0	70.0
BPMeds	4238.0	NaN	NaN	NaN	0.029259	0.168552	0.0	0.0	0.0	0.0	1.0
prevalentStroke	4238	2	no	4213	NaN	NaN	NaN	NaN	NaN	NaN	NaN
prevalentHyp	4238.0	NaN	NaN	NaN	0.310524	0.462763	0.0	0.0	0.0	1.0	1.0
diabetes	4238.0	NaN	NaN	NaN	0.02572	0.158316	0.0	0.0	0.0	0.0	1.0
totChol	4238.0	NaN	NaN	NaN	236.689476	44.327427	107.0	206.0	234.0	262.0	696.0
sysBP	4238.0	NaN	NaN	NaN	132.352407	22.038097	83.5	117.0	128.0	144.0	295.0
diaBP	4238.0	NaN	NaN	NaN	82.893464	11.91085	48.0	75.0	82.0	89.875	142.5
BMI	4238.0	NaN	NaN	NaN	25.800205	4.071041	15.54	23.08	25.4	28.0375	56.8
heartRate	4238.0	NaN	NaN	NaN	75.878716	12.025185	44.0	68.0	75.0	83.0	143.0
glucose	4238.0	NaN	NaN	NaN	81.603587	22.865246	40.0	72.0	78.0	85.0	394.0
Heart_ stroke	4238	2	No	3594	NaN	NaN	NaN	NaN	NaN	NaN	NaN

These summary statistics provide initial insights into the data distribution and characteristics, which can inform subsequent data processing and modeling steps. Further analysis and visualization will be necessary to explore relationships between variables and identify patterns.

Step 4: Check Labels in Categorical Features

Inspected the unique labels or categories within categorical features to verify data integrity and identify any inconsistencies or anomalies.

Code

```
## Let's check the labels in categorical features

for col in df.columns:
    if df[col].dtype=='object':
        print()
        print(col)
        print(df[col].unique())
```

Output

```
Gender
['Male' 'Female']

education
['postgraduate' 'primaryschool' 'uneducated' 'graduate']

prevalentStroke
['no' 'yes']

Heart_ stroke
['No' 'yes']
```

Overall, the categorical features exhibit well-defined labels, which are essential for accurate data analysis and modeling. In the case of "Heart_Stroke," addressing the case sensitivity issue is recommended for data consistency and we addressed that in the data cleaning process.

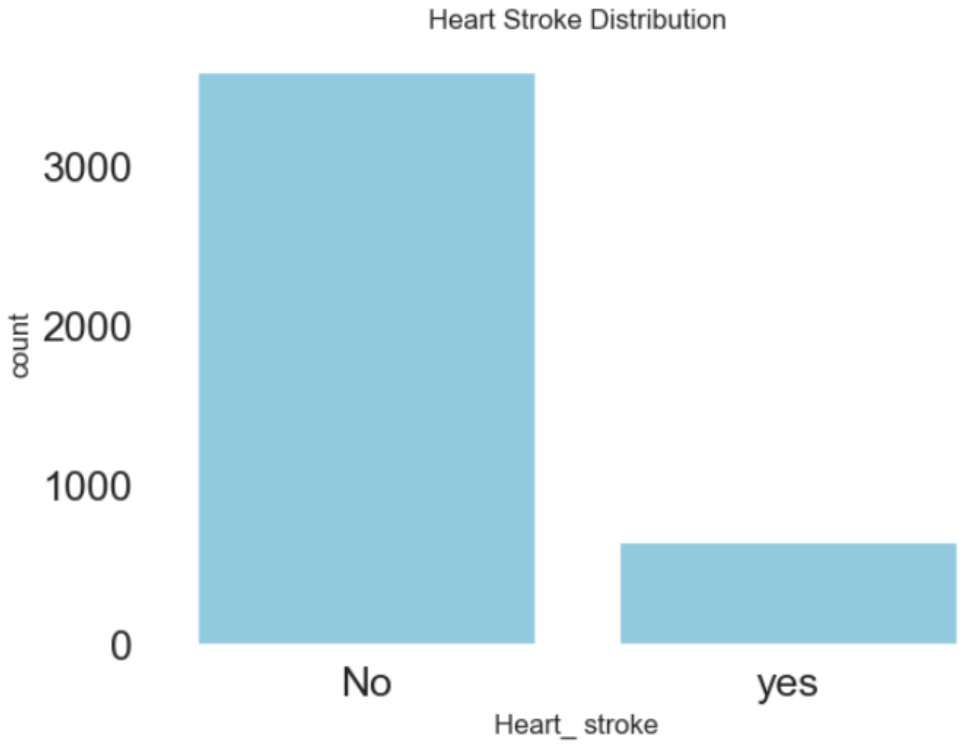
Step 5: Distribution of Target Variables

Analyzed the distribution of target variables, which helps in understanding class imbalances and potential issues in classification tasks.

Code

```
plot_color = 'skyblue'
sns.countplot(x='Heart_ stroke', data=df, color=plot_color)
plt.title("Heart Stroke Distribution")
plt.show()
```

Output



The countplot reveals that the majority of individuals in the dataset (approximately 3,000) fall into the "No" category, meaning they have not had a heart stroke.
On the other hand, a smaller group of individuals (almost 500) falls into the "Yes" category, indicating that they have experienced a heart stroke. This means that the data is imbalanced.

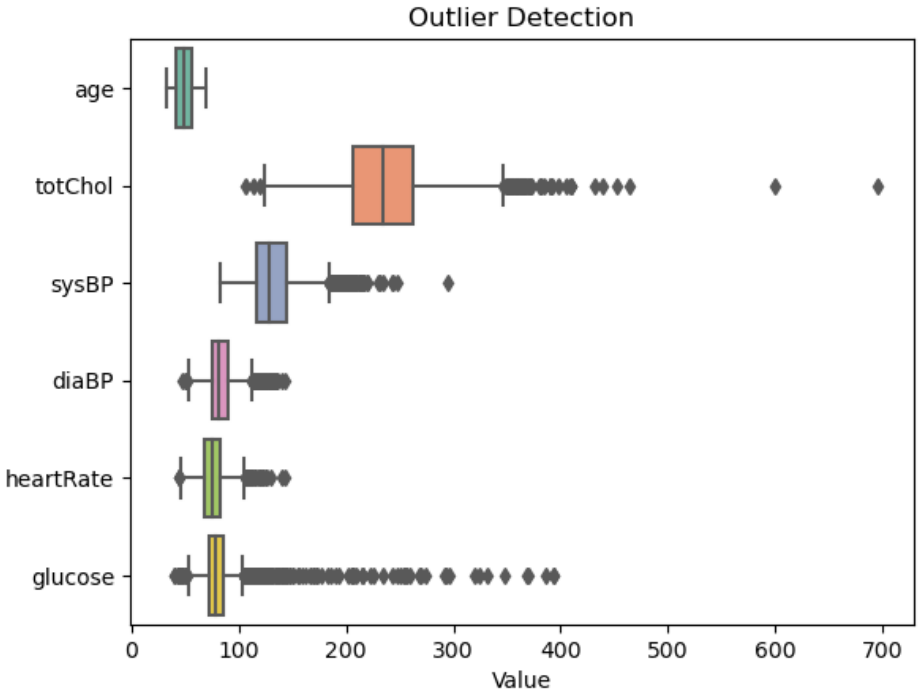
Step 6: Check for Outliers

Conducted outlier detection to identify extreme values in numerical features that might affect model performance. This step is essential for maintaining data quality.

Code

```
# Check for outliers
columns_to_check = ["age", "totChol", "sysBP", "diaBP", "heartRate", "glucose"]

# Create a boxplot with Seaborn
sns.boxplot(data=df[columns_to_check], orient="h", palette="Set2")
plt.title("Outlier Detection")
plt.xlabel("Value")
plt.show()
```

Output



Like in a typical patient health records dataset, we can see several (in fact, almost all the) features with outliers. Let's retain the outliers as-is, assuming that they are real values of some patients. Glucose, Total Cholesterol, Systolic BP, and BMI features have most number of outliers.

Step 7: Anova Analysis of Categorical Features

Performed analysis of variance (ANOVA) on categorical features to assess the variability between different categories and their potential impact on the target variable.

Output

```python
from scipy.stats import f_oneway

# Select the numerical features
numerical_features = df.select_dtypes(include=np.number)

# Calculate the ANOVA F-statistic and p-value for each feature
f_stats = []
p_vals = []

for feature in numerical_features.columns:
    groups = [group[feature].values for name, group in df.groupby("Heart_ stroke")]
    f_stat, p_val = f_oneway(*groups)
    f_stats.append(f_stat)
    p_vals.append(p_val)

# Print the results
for i, feature in enumerate(numerical_features.columns):
    print(f"{feature}: F-statistic: {f_stats[i]:.3f}, P-value: {p_vals[i]:.3f}")
```

```
age: F-statistic: 204.225, P-value: 0.000
currentSmoker: F-statistic: 3.056, P-value: 0.080
cigsPerDay: F-statistic: 16.594, P-value: 0.000
BPMeds: F-statistic: 16.514, P-value: 0.000
prevalentHyp: F-statistic: 113.136, P-value: 0.000
diabetes: F-statistic: 5.175, P-value: 0.023
totChol: F-statistic: 18.586, P-value: 0.000
sysBP: F-statistic: 158.957, P-value: 0.000
diaBP: F-statistic: 67.376, P-value: 0.000
BMI: F-statistic: 18.425, P-value: 0.000
heartRate: F-statistic: 0.678, P-value: 0.410
glucose: F-statistic: 7.811, P-value: 0.005
```

The ANOVA analysis indicates:
Age: Significant differences between individuals who have and haven't had a heart stroke, with age playing a significant role.
Current Smoker: Some differences in smoking status, but not highly significant.
Other Features: Features with missing values need preprocessing.
Prevalent Hypertension and Diabetes: Highly significant differences between the two groups.
Blood Pressure: Highly significant differences in blood pressure between the two groups.

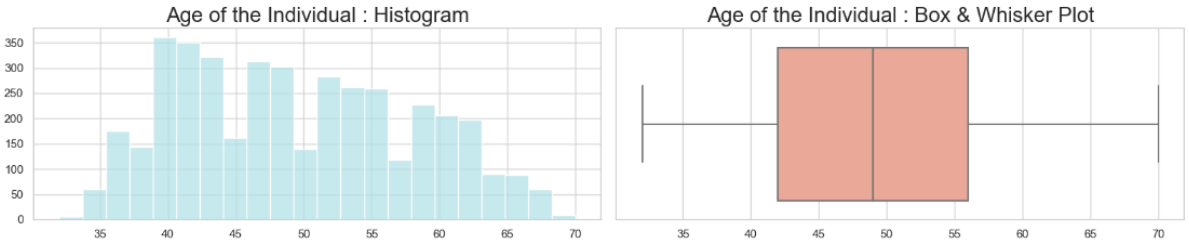
Step 8: Distribution of Numerical Values (Histograms and Boxplots)

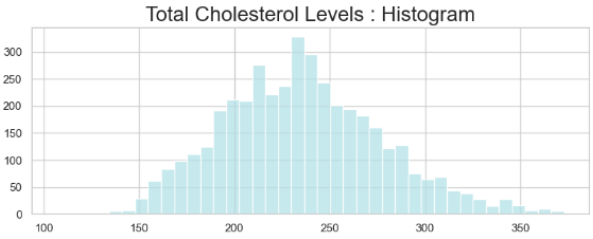
Visualized the distribution of numerical values using histograms and boxplots to identify data skewness, central tendencies, and the presence of outliers.

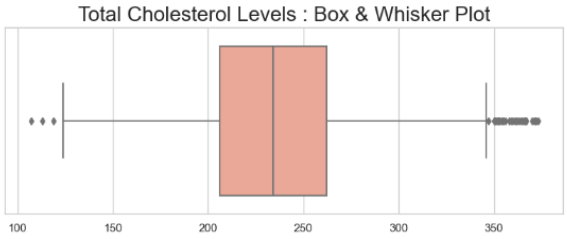
Code

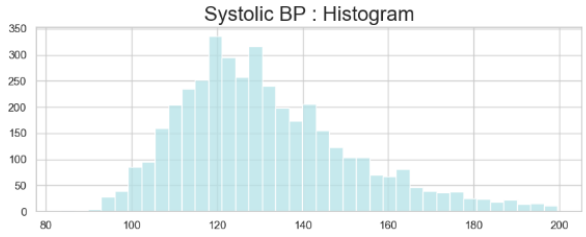
```
# Set the style and figure size
sns.set(style='whitegrid')
plt.figure(figsize=(16, 26))

# Define light colors
hist_color = '#B4E2E7'
box_color = '#F7A08A'

# Define numerical features
numerical_features = [
    ('Age of the Individual', df.age),
    ('Total Cholesterol Levels', df.totChol),
    ('Systolic BP', df.sysBP),
    ('Diastolic BP', df.diaBP),
    ('BMI', df.BMI),
    ('Heart Rate', df.heartRate),
    ('Glucose', df.glucose)
]

# Create subplots for each feature
for i, (title, data) in enumerate(numerical_features):
    # Histogram
    plt.subplot(8, 2, i * 2 + 1)
    plt.title(f'{title} : Histogram', fontsize=20)
    sns.histplot(data, color=hist_color, kde_kws={'linewidth': 2, 'color': 'r'})
    plt.ylabel(None)
    plt.xlabel(None)

    # Box & Whisker Plot
    plt.subplot(8, 2, i * 2 + 2)
    plt.title(f'{title} : Box & Whisker Plot', fontsize=20)
    sns.boxplot(data, orient='h', color=box_color)
    plt.yticks([])

# Adjust the layout and display the plot
plt.tight_layout()
plt.show()
```

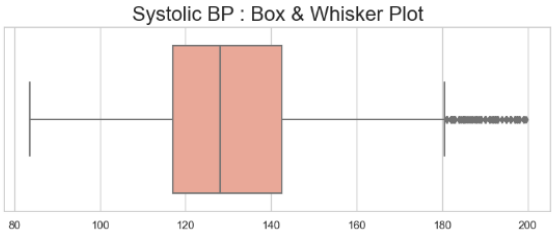
Output

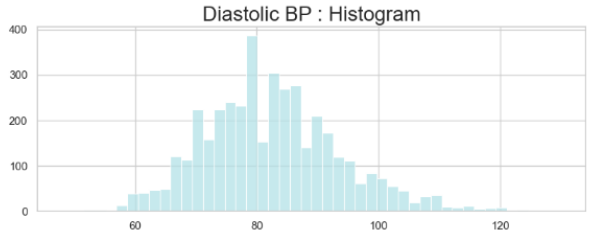


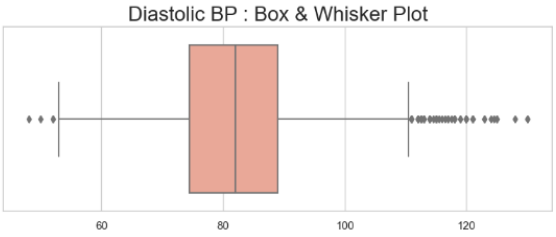

Total Cholesterol Levels : Histogram

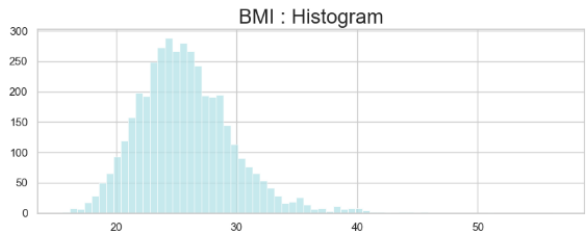

Total Cholesterol Levels : Box & Whisker Plot

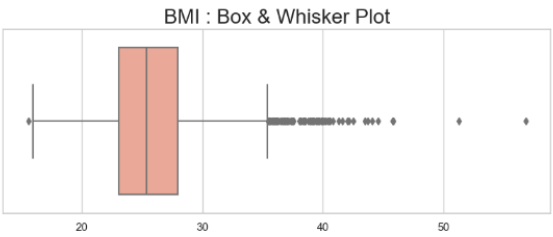

Systolic BP : Histogram

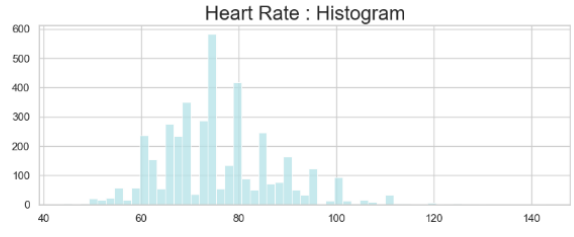

Systolic BP : Box & Whisker Plot

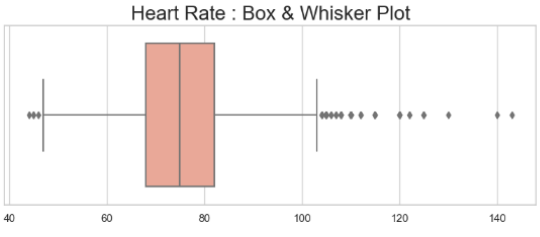

Diastolic BP : Histogram

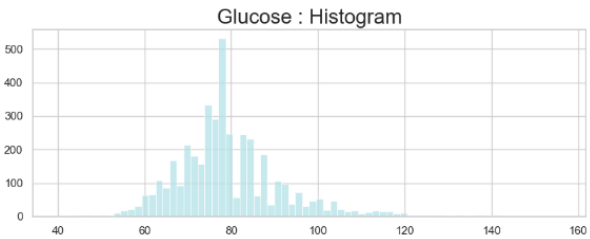

Diastolic BP : Box & Whisker Plot

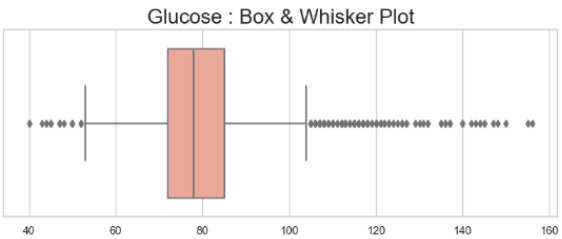

BMI : Histogram


BMI : Box & Whisker Plot


Heart Rate : Histogram


Heart Rate : Box & Whisker Plot


Glucose : Histogram


Glucose : Box & Whisker Plot

Key Observations:

The dataset exhibits outliers in several features, which is common in patient health records. These outliers are retained as genuine data points representing specific patients.
Notably, features such as Glucose, Total Cholesterol, Systolic Blood Pressure (BP), and Body Mass Index (BMI) have a higher prevalence of outliers, suggesting the existence of extreme values in these health parameters.
Among the features, Total Cholesterol, BMI, and Glucose demonstrate bell-shaped (normal) distribution patterns, which are characteristic of typical data distributions in patient health records.

Step 9: Distribution of Categorical Values

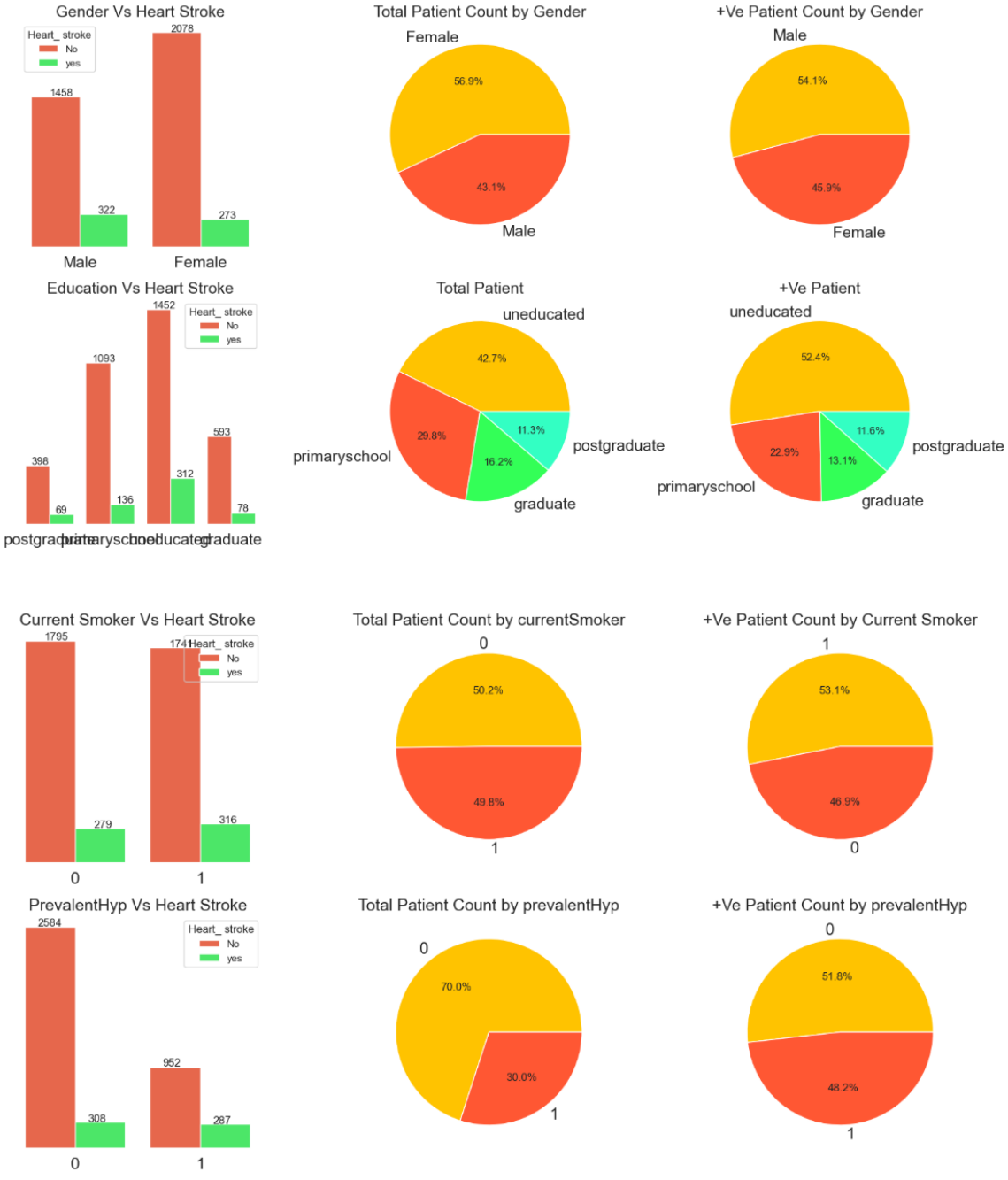
Visualized the distribution of categorical values using appropriate plots, aiding in understanding category frequencies and patterns.

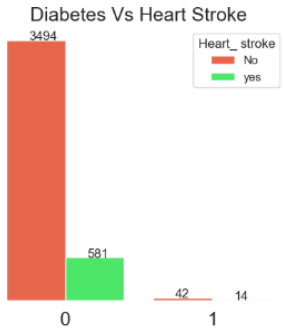
Code

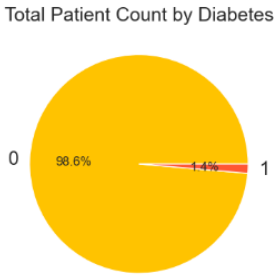
```
## Now let's do some analysis on categorical variables and their impact on Heart Stroke

sns.set(rc={'axes.facecolor':'none','axes.grid':False,'xtick.labelsize':18,'ytick.labelsize':18, 'figure.autolayout':True})
my_col = ('#FF5733', '#33FF57')
my_pal = ('#FFC300', '#FF5733', '#33FF57', '#33FFC3', '#C333FF', '#FF33C3')
plt.subplots(figsize=(16,26))

##  Gender

plt.subplot(6,3,1)
plt.title('Gender Vs Heart Stroke',fontsize=18)
ax = sns.countplot(x='Gender', hue='Heart_ stroke', palette=my_col, data=df)
for p in ax.patches:
   ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.15, p.get_height()+8))
plt.ylabel(None), plt.yticks([]), plt.xlabel(None)
plt.subplot(6,3,2)
plt.title('Total Patient Count by Gender', fontsize=18)
df['Gender'].value_counts().plot(kind='pie', subplots=True, colors =my_pal, legend=None, ylabel='', autopct='%1.1f%%')
plt.subplot(6,3,3)
plt.title('+Ve Patient Count by Gender', fontsize=18)
df[df['Heart_ stroke'] == "yes"]['Gender'].value_counts().plot(kind='pie', subplots=True, colors = my_pal, legend=None, ylab

## education

plt.subplot(6,3,4)
plt.title('Education Vs Heart Stroke', fontsize=18)
ax = sns.countplot(x='education', hue='Heart_ stroke', palette=my_col, data=df)
for p in ax.patches:
   ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.10, p.get_height()+8))
plt.ylabel(None), plt.yticks([]), plt.xlabel(None)
plt.subplot(6,3,5)
plt.title('Total Patient', fontsize=18)
df['education'].value_counts().plot(kind='pie', subplots=True, colors = my_pal, legend=None, ylabel='', autopct='%1.1f%%'),
plt.subplot(6,3,6)
plt.title('+Ve Patient', fontsize=18)
df[df['Heart_ stroke'] == "yes"]['education'].value_counts().plot(kind='pie', subplots=True, colors = my_pal, legend=None, y

## Current Smoker

plt.subplot(6,3,7)
plt.title('Current Smoker Vs Heart Stroke', fontsize=18)
ax = sns.countplot(x='currentSmoker', hue='Heart_ stroke', palette=my_col, data=df)
for p in ax.patches:
   ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.15, p.get_height()+8))
plt.ylabel(None), plt.yticks([]), plt.xlabel(None)
plt.subplot(6,3,8)
plt.title('Total Patient Count by currentSmoker', fontsize=18)
df['currentSmoker'].value_counts().plot(kind='pie', subplots=True, colors = my_pal, legend=None, ylabel='', autopct='%1.1f%%
plt.subplot(6,3,9)
plt.title('+Ve Patient Count by Current Smoker', fontsize=18)
df[df['Heart_ stroke'] == "yes"]['currentSmoker'].value_counts().plot(kind='pie', subplots=True, colors = my_pal, legend=Non

## Hypertension

plt.subplot(6,3,10)
plt.title('PrevalentHyp Vs Heart Stroke', fontsize=18)
ax = sns.countplot(x='prevalentHyp', hue='Heart_ stroke', palette=my_col, data=df)
for p in ax.patches:
   ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.10, p.get_height()+8))
plt.ylabel(None), plt.yticks([]), plt.xlabel(None)
plt.subplot(6,3,11)
plt.title('Total Patient Count by prevalentHyp', fontsize=18)
df['prevalentHyp'].value_counts().plot(kind='pie', subplots=True, colors = my_pal, legend=None, ylabel='', autopct='%1.1f%%'
plt.subplot(6,3,12)
plt.title('+Ve Patient Count by prevalentHyp', fontsize=18)
df[df['Heart_ stroke'] == "yes"]['prevalentHyp'].value_counts().plot(kind='pie', subplots=True, colors = my_pal, legend=None

## diabetes

plt.subplot(6,3,13)
plt.title('Diabetes Vs Heart Stroke', fontsize=18)
ax = sns.countplot(x='diabetes', hue='Heart_ stroke', palette=my_col, data=df)
```

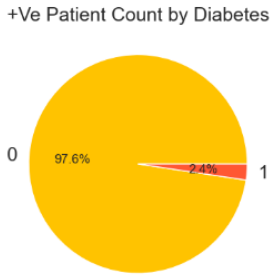
```
ax = sns.countplot(x='diabetes', hue='Heart_ stroke', palette=my_col, data=df)
for p in ax.patches:
   ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.15, p.get_height()+8))
plt.ylabel(None), plt.yticks([]), plt.xlabel(None)
plt.subplot(6,3,14)
plt.title('Total Patient Count by Diabetes', fontsize=18)
df['diabetes'].value_counts().plot(kind='pie', subplots=True, colors = my_pal, legend=None, ylabel='', autopct='%1.1f%%')
plt.subplot(6,3,15)
plt.title('+Ve Patient Count by Diabetes', fontsize=18)
df[df['Heart_ stroke'] == "yes"]['diabetes'].value_counts().plot(kind='pie', subplots=True, colors = my_pal, legend=None, yl

plt.show()

```

Output









Key Observations:

1. Female patients constitute the majority, accounting for approximately 56% of the total patient population in the dataset.
2. Among patients who have tested positive (+ve) for certain health conditions, male patients represent the majority, with a share of about 52%.
3. In terms of education levels, the uneducated patient group exhibits a higher count of positive patients, indicating that a lack of formal education may be associated with a higher risk of certain health conditions.

Step 10: Checking the impact of the combination of features

The analysis reveals that there is minimal correlation between the majority of numerical features and the target variable (heart stroke). Most of the features do not display a strong relationship with the occurrence of heart stroke.

Code

```
# Set the Seaborn style to a light theme
sns.set(style="whitegrid")
# Create a custom color palette
my_pal = ["#F5A623", "#F8E71C"]  # Change these colors to your desired palette

# Create a figure with subplots
fig, axes = plt.subplots(5, 2, figsize=(16, 20))

# Flatten the axes array for easy indexing
axes = axes.flatten()

# Titles for each subplot
titles = [
    'Heart Stroke by Age & Cigarettes/Day',
    'Heart Stroke by Age & Cholesterol',
    'Heart Stroke by Age & Systolic BP',
    'Heart Stroke by Age & Diastolic BP',
    'Heart Stroke by Age & BMI',
    'Heart Stroke by Age & Glucose',
    'Heart Stroke by Age & Heart Rate',
    'Heart Stroke by Cholesterol & Cigarettes/Day',
    'Heart Stroke by Systolic BP & Diastolic BP'
]
# Plot each subplot
for i in range(9):
    axes[i].set_title(titles[i], fontsize=18)
    sns.scatterplot(x='age', y='cigsPerDay' if i == 0 else
                               'totChol' if i == 1 else
                               'sysBP' if i == 2 else
                               'diaBP' if i == 3 else
                               'BMI' if i == 4 else
                               'glucose' if i == 5 else
                               'heartRate' if i == 6 else
                               'cigsPerDay' if i == 7 else
                               'diaBP' if i == 8 else '',
                    hue='Heart_Stroke', palette=my_pal, data=df, ax=axes[i])
    axes[i].legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0)
    axes[i].set_xlabel(None)
# Hide the last subplot
axes[-1].axis('off')

# Adjust layout
plt.tight_layout()
plt.show()
```

Output

Key Observations

Blood Pressure Correlation: Notably, diastolic blood pressure and systolic blood pressure exhibit some level of correlation. These two blood pressure measurements show a modest relationship with heart stroke occurrence.

Absence of Expected Correlation: While it is conventionally believed that cholesterol levels and smoking habits have a direct correlation with heart disease and heart stroke, the current dataset does not clearly exhibit such a robust correlation. The relationship between these factors and heart stroke is not pronounced based on the available data.

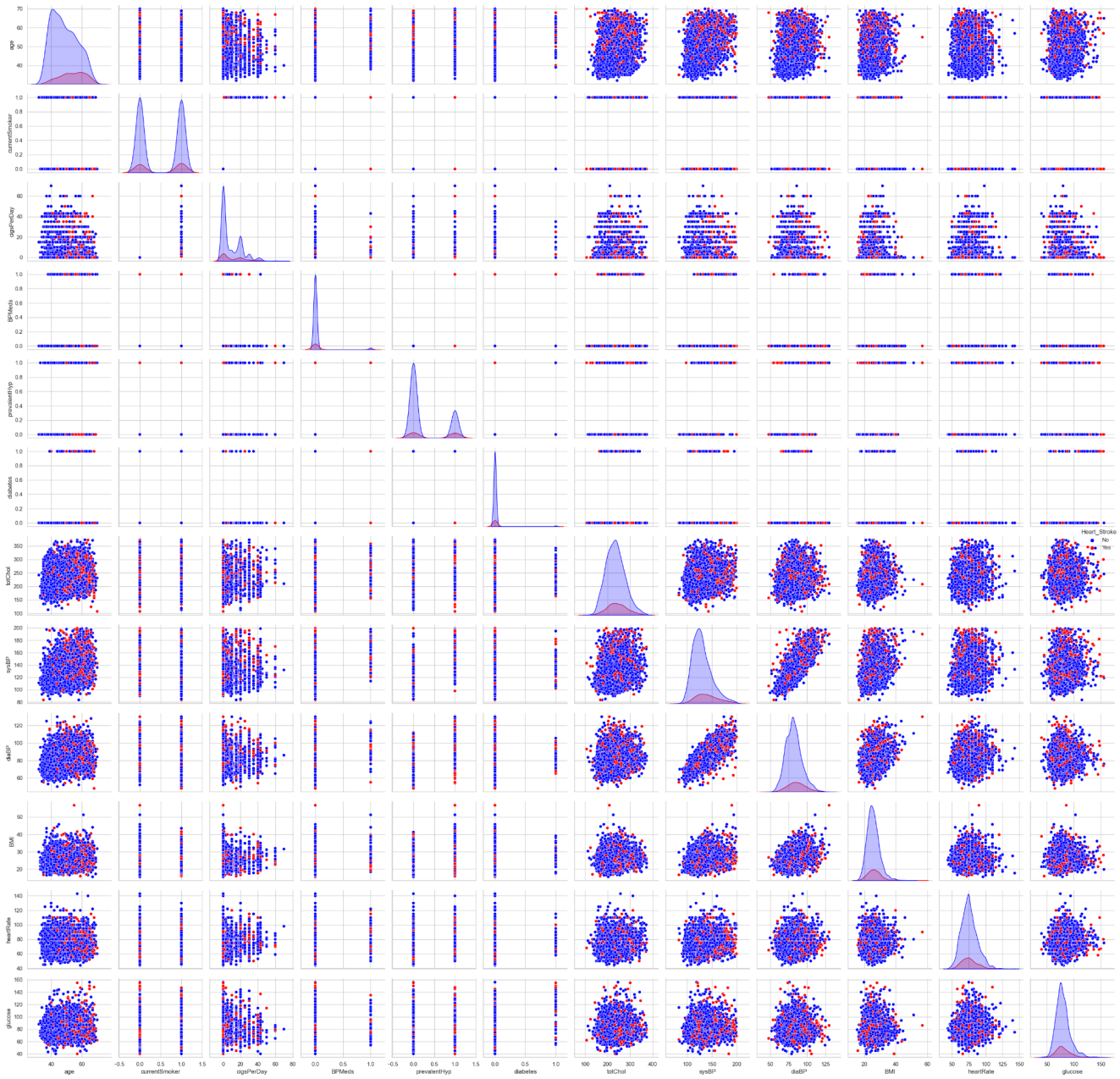
Step 11: Pair Plot

Generated a pair plot to visualize pairwise relationships between numerical variables, which can provide insights into correlations and dependencies.

Code

```
# Assuming df is your DataFrame

# Define a custom color palette
custom_palette = {'No': 'blue', 'Yes': 'red'}  # Change the colors as needed

# Convert the 'Heart_Stroke' column values to title case to match the dictionary
df['Heart_Stroke'] = df['Heart_ stroke'].str.title()

# Create the pairplot with the custom color palette
sns.pairplot(df, hue='Heart_Stroke', diag_kind='kde', palette=custom_palette)

# Show the plot
plt.show()
```

Output



From the above pairplot we can infer that there is some positive correlation between age and systolic blood pressure. There is also a correlation between cholesterol and systolic blood pressure, cholesterol with diastolic blood pressure and very positive skewness between systolic blood pressure and diastolic blood pressure. There is somewhat correlation with BMI and systolic blood pressure, glucose, diastolic blood pressure.

Conclusion

In this Phase 1 report, we have embarked on a comprehensive exploration of a critical issue in public health - heart disease, with a particular focus on heart stroke prevention. We have taken a holistic approach to understand the intricate relationship between clinical variables and risk factors in shaping cardiovascular health. The findings and data processing steps outlined in this report are integral to the larger mission of mitigating the impact of heart disease on individuals and healthcare systems.

Data Sources and Description

The dataset used for this project, obtained from Kaggle, is a valuable resource containing a wide range of variables that shed light on heart disease and cardiovascular health. It encompasses information on demographics, lifestyle, medical history, and clinical measurements, providing a comprehensive view of factors affecting heart health.

Data Cleaning and Processing

Data cleaning and processing are crucial for ensuring the quality and integrity of the dataset. We followed a systematic approach to:

- Check and validate data types.
- Remove irrelevant columns while adding and subsequently removing an identification column.
- Address missing values through imputation techniques.
- Handle outliers by retaining genuine data points while eliminating extreme values.
- Convert categorical data into a suitable format for analysis.
- Standardize text data for consistency.
- Split the dataset into training and testing subsets for machine learning.

Exploratory Data Analysis (EDA)

Our EDA phase provided essential insights into the dataset, helping us understand its characteristics, distribution, and relationships between variables. Some key findings from EDA include:

- Identification of class imbalances in the target variable.
- Detection of outliers in numerical features.
- Assessment of the impact of categorical features through ANOVA.
- Visualizing the distribution of numerical and categorical data.
- Observing correlations between numerical features via pair plots.
- The insights from EDA serve as a foundation for feature selection, engineering, and model development in subsequent project phases.

In conclusion, this Phase 1 report sets the stage for a deeper exploration into heart stroke prevention and cardiovascular health. By understanding the complexities of clinical variables and risk factors, we aim to make a significant contribution to reducing the burden of heart disease and improving the well-being of individuals and society. The project is now poised to enter the modeling and analysis phase, armed with a clean and well-understood dataset.