Name : Atharva Phalke
Class : D15C
Batch : C
Rollno: 65

DMBI - 3

Aim: To perform Exploratory Data Analysis and Visualization using python.

Theory:
Exploratory Data Analysis (EDA) is the process of exploring, summarizing, and visualizing data to understand its main characteristics before applying statistical models or machine learning. It helps researchers **detect underlying structures, spot anomalies, identify relationships, and test hypotheses.**

The major steps include:

1. **Descriptive Statistics**
   ○ Provides numerical summaries such as mean, median, variance, min/max values, and standard deviation.
   ○ Helps detect skewness, outliers, and unusual distributions

2. **Target Variable Analysis**
   ○ The dependent variable (here: heart_disease) is visualized with count plots to check class balance.
   ○ If the dataset is highly imbalanced, it may affect classification performance.

3. **Correlation Analysis**
   ○ Pearson's correlation coefficient is calculated between numeric features.
   ○ A heatmap helps identify strong positive/negative correlations (e.g., thalach vs. age, chol vs. bmi).
   ○ Useful for detecting multicollinearity or redundant features.

4. **Feature Distribution Analysis**
   ○ Histograms/KDE plots show how features like age, cholesterol, and bmi are distributed (normal, skewed, multimodal).
   ○ Boxplots grouped by target show how continuous features vary between patients with and without heart disease.

5. **Categorical Feature Analysis**
   ○ Countplots and bar charts show how categorical features (e.g., sex, cp, thal) are distributed across target classes.
   ○ This helps assess the predictive power of categorical variables (e.g., chest pain type has strong association with heart disease).

6. **Multivariate Visualization**
   ○ Pairplots allow simultaneous visualization of multiple variables, highlighting clusters and class separation.
   ○ Useful to see which combinations of features separate patients with vs. without heart disease.

Importance:

- Provides deeper insight into dataset structure.
- Helps select features that are most relevant for prediction.
- Reveals outliers or errors that might need special treatment.
- Builds intuition about how independent variables influence the target outcome.

Conclusion:

● The Stroke Prediction dataset provides valuable features for building predictive models. However, proper handling of class imbalance, missing values, and outliers will be crucial for developing robust and reliable machine learning solutions.

Code and Output:

Exp3.py × healthcare-dataset-stroke-data.csv

Exp3.py > ...

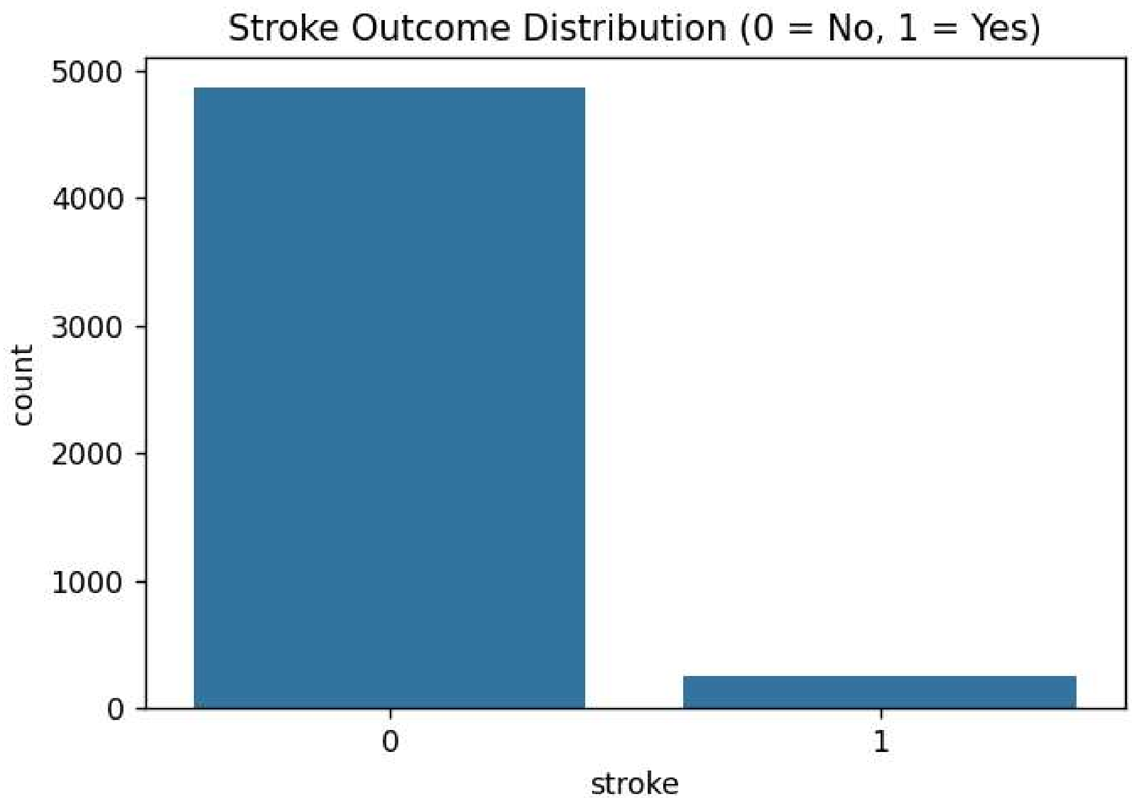
```python
# Exploratory Data Analysis (EDA) for Stroke Prediction Dataset

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Load the dataset (update the path if needed)
df = pd.read_csv('C:\\Users\\athar\\OneDrive\\Desktop\\DMBI\\archive\\healthcare-dataset-stroke-data.csv') # Change filename/path a

# 2. Initial Data Exploration
print("First 5 rows:")
print(df.head())
print("\nDataset info:")
print(df.info())
print("\nStatistical summary:")
print(df.describe(include='all'))
print("\nMissing values in each column:")
print(df.isnull().sum())
```

👇

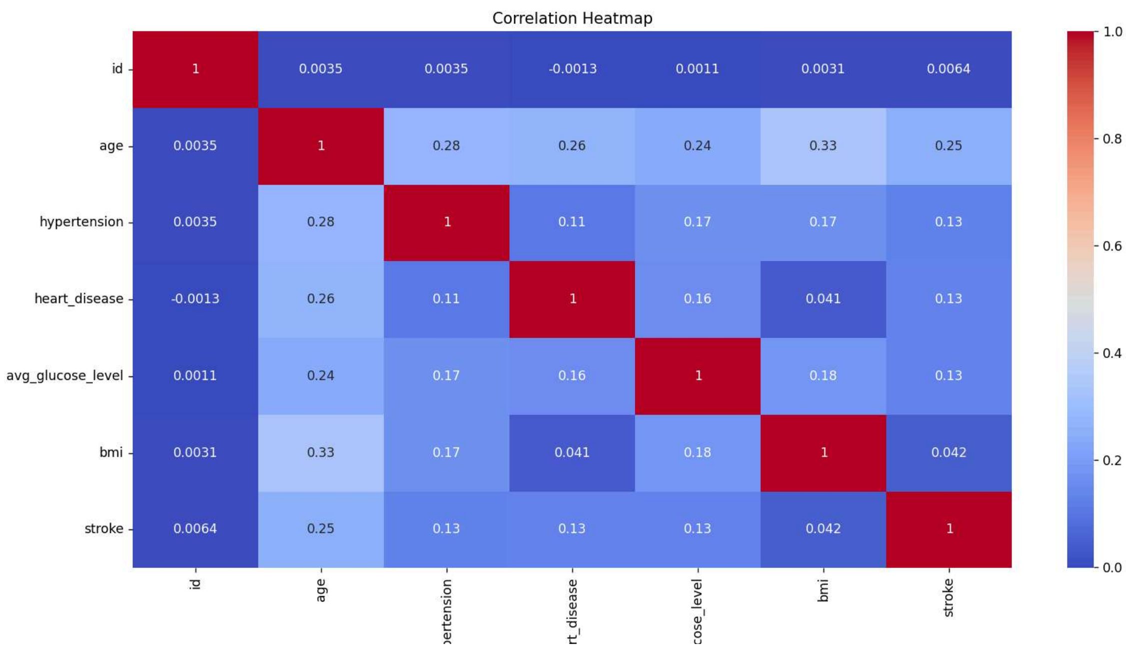
```
First 5 rows:
      id  gender   age  hypertension  heart_disease  ... Residence_type avg_glucose_level   bmi   smoking_status  stroke
0   9046    Male  67.0             0              1  ...          Urban            228.69  36.6  formerly smoked       1
1  51676  Female  61.0             0              0  ...          Rural            202.21   NaN     never smoked       1
2  31112    Male  80.0             0              1  ...          Rural            105.92  32.5     never smoked       1
3  60182  Female  49.0             0              0  ...          Urban            171.23  34.4           smokes       1
4   1665  Female  79.0             1              0  ...          Rural            174.12  24.0     never smoked       1

[5 rows x 12 columns]

Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 5110 non-null   int64  
 1   gender             5110 non-null   object 
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64  
 4   heart_disease      5110 non-null   int64  
 5   ever_married       5110 non-null   object 
 6   work_type          5110 non-null   object 
 7   Residence_type     5110 non-null   object 
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object 
 11  stroke             5110 non-null   int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
None
```

```
Statistical summary:
                  id  gender          age  hypertension  ...  avg_glucose_level          bmi smoking_status       stroke
count    5110.000000    5110  5110.000000   5110.000000  ...        5110.000000  4909.000000           5110  5110.000000
unique           NaN       3          NaN           NaN  ...                NaN          NaN              4          NaN
top              NaN  Female          NaN           NaN  ...                NaN          NaN   never smoked          NaN
freq             NaN    2994          NaN           NaN  ...                NaN          NaN           1892          NaN
mean    36517.829354     NaN    43.226614      0.097456  ...         106.147677    28.893237            NaN     0.048728
std     21161.721625     NaN    22.612647      0.296607  ...          45.283560     7.854067            NaN     0.215320
min        67.000000     NaN     0.080000      0.000000  ...          55.120000    10.300000            NaN     0.000000
25%     17741.250000     NaN    25.000000      0.000000  ...          77.245000    23.500000            NaN     0.000000
50%     36932.000000     NaN    45.000000      0.000000  ...          91.885000    28.100000            NaN     0.000000
75%     54682.000000     NaN    61.000000      0.000000  ...         114.090000    33.100000            NaN     0.000000
max     72940.000000     NaN    82.000000      1.000000  ...         271.740000    97.600000            NaN     1.000000

[11 rows x 12 columns]

Missing values in each column:
id                     0
gender                 0
age                    0
hypertension           0
heart_disease          0
ever_married           0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                  201
smoking_status         0
stroke                 0
dtype: int64
```

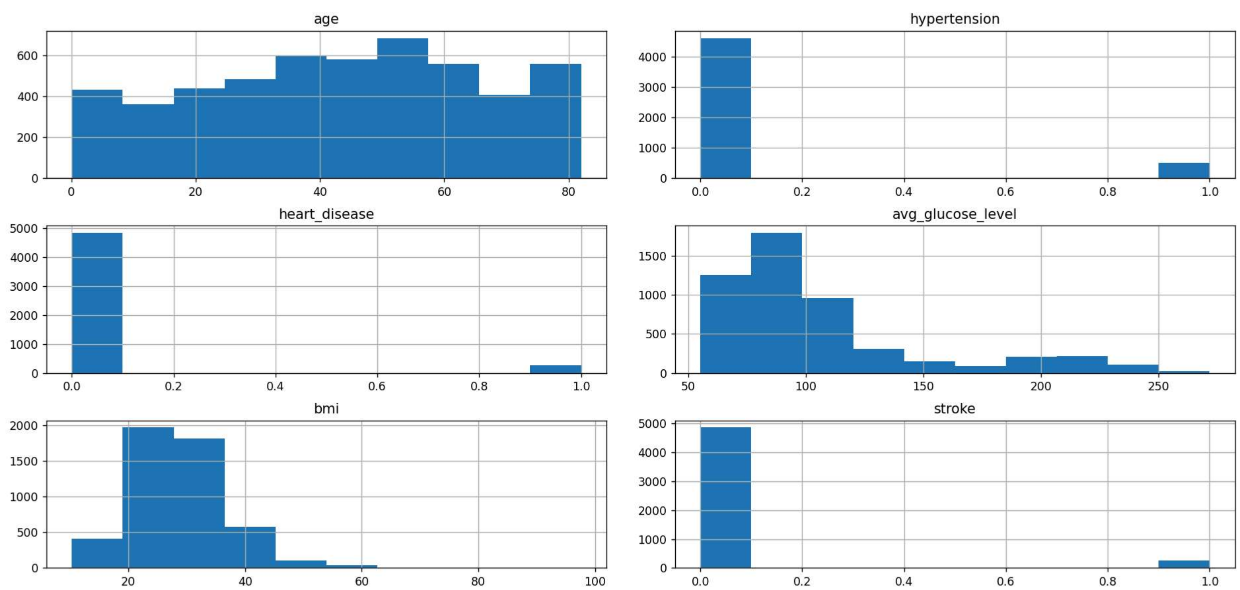
```
# 4. Target Variable Analysis
plt.figure(figsize=(6,4))
sns.countplot(x='stroke', data=df)
plt.title('Stroke Outcome Distribution (0 = No, 1 = Yes)')
plt.show()
```

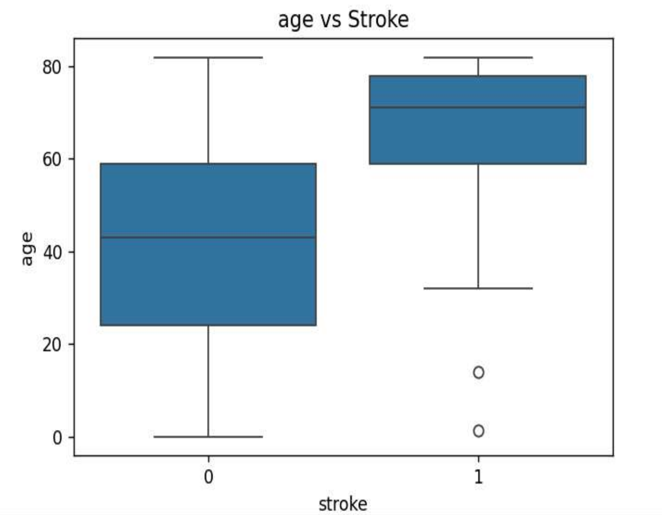

Stroke Outcome Distribution (0 = No, 1 = Yes)
count
5000
4000
3000
2000
1000
0
0
1
stroke

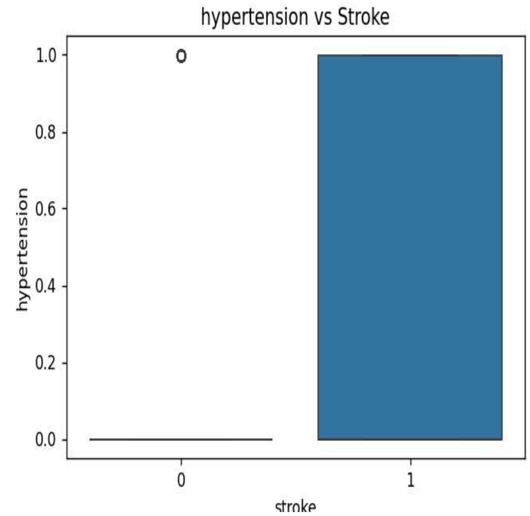
```python
# 5. Correlation Analysis (Numerical Features)
corr = df.corr(numeric_only=True)
plt.figure(figsize=(10,8))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

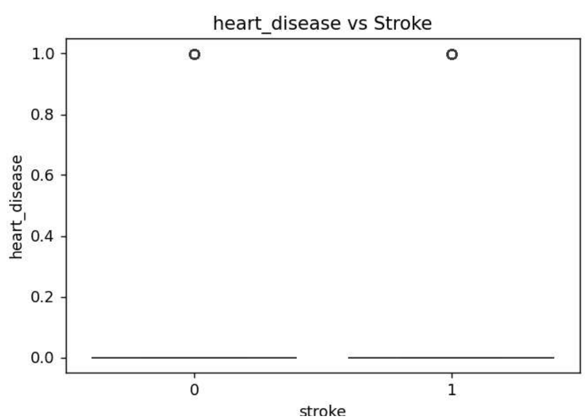


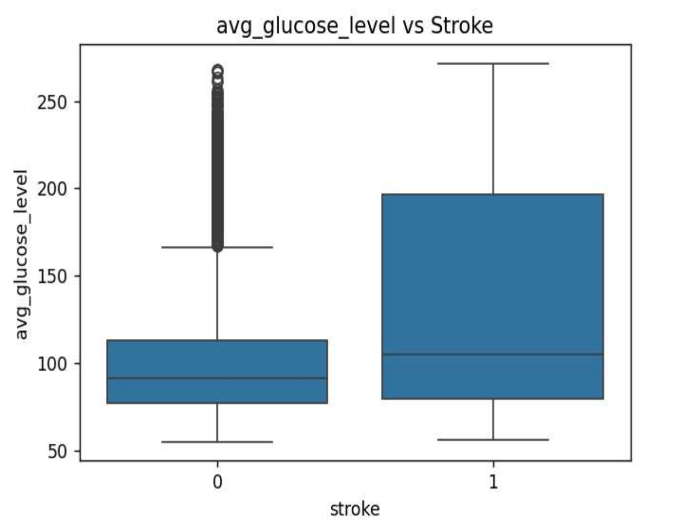
```
# 6. Feature Distribution Analysis
num_cols = df.select_dtypes(include=np.number).columns.tolist()
if 'id' in num_cols:
    num_cols.remove('id')  # Remove 'id' as it's not a feature

df[num_cols].hist(figsize=(12,8))
plt.suptitle("Histograms of Numerical Features", y=1.02)
plt.tight_layout()
plt.show()

# Boxplots by Target
for col in num_cols:
    if col != 'stroke':
        plt.figure(figsize=(6,4))
        sns.boxplot(x='stroke', y=col, data=df)
        plt.title(f'{col} vs Stroke')
        plt.show()
```

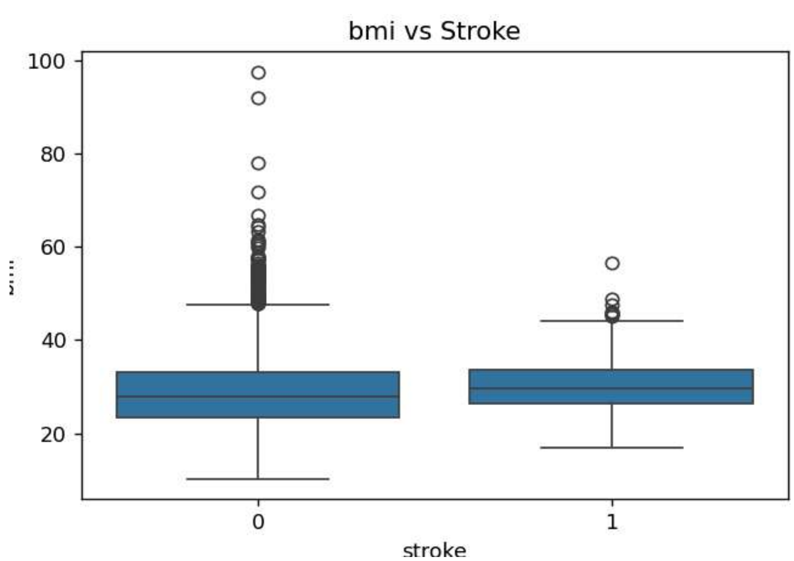



age vs Stroke
age
0
20
40
60
80
0
1
stroke

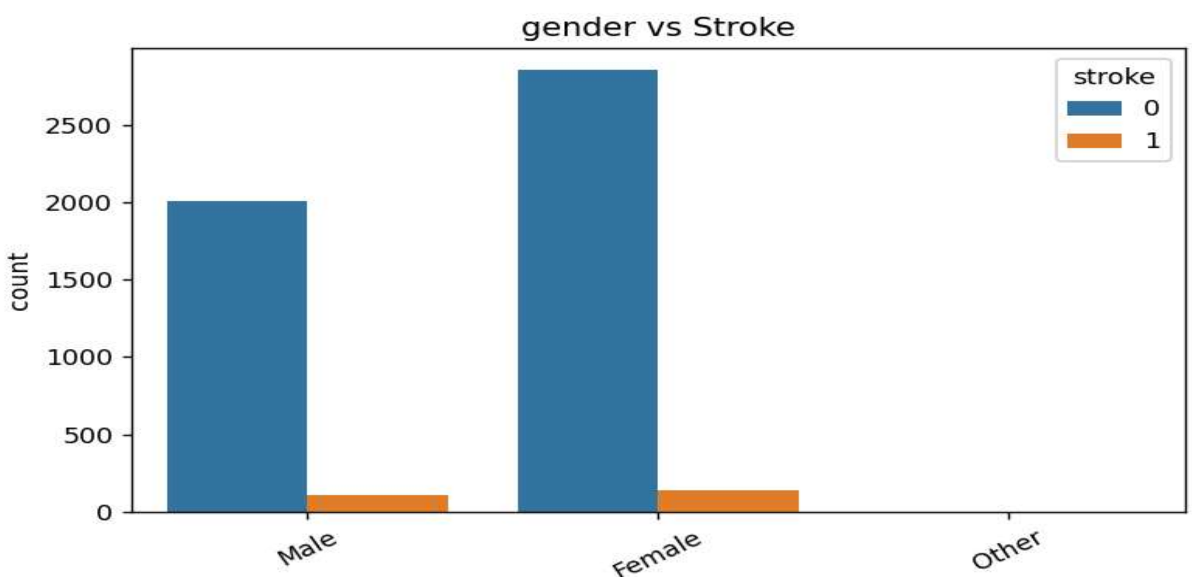

hypertension vs Stroke
hypertension
0.0
0.2
0.4
0.6
0.8
1.0
0
1
stroke

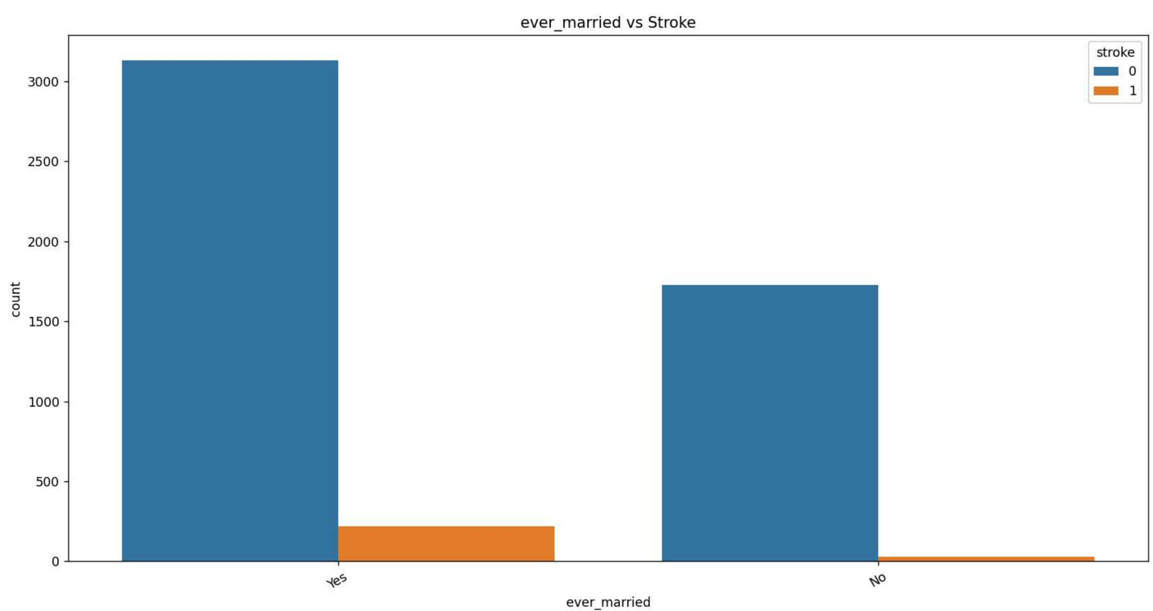

heart_disease vs Stroke
heart_disease
0.0
0.2
0.4
0.6
0.8
1.0
0
1
stroke

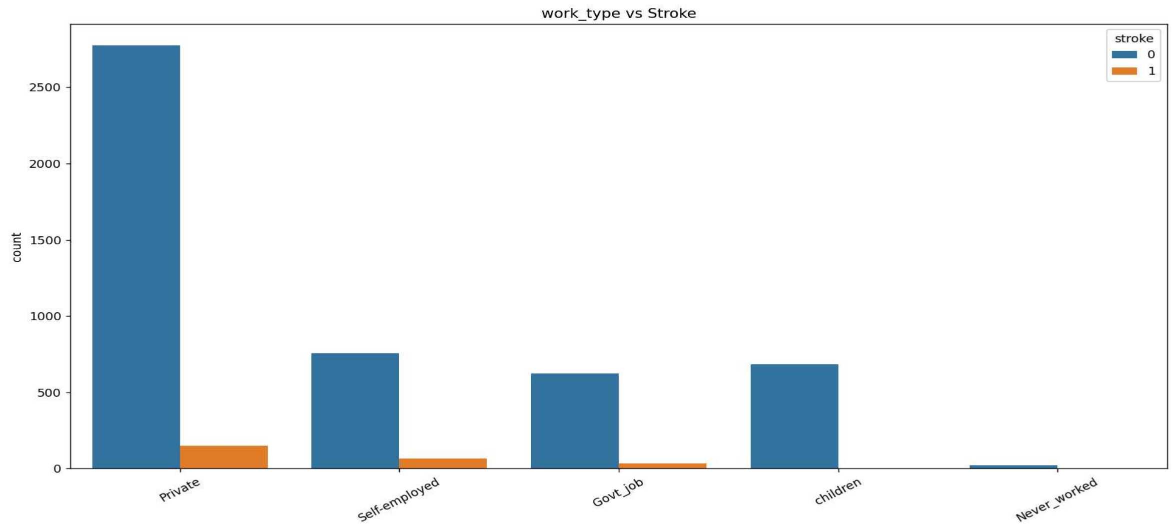

avg_glucose_level vs Stroke
avg_glucose_level
50
100
150
200
250
0
1
stroke

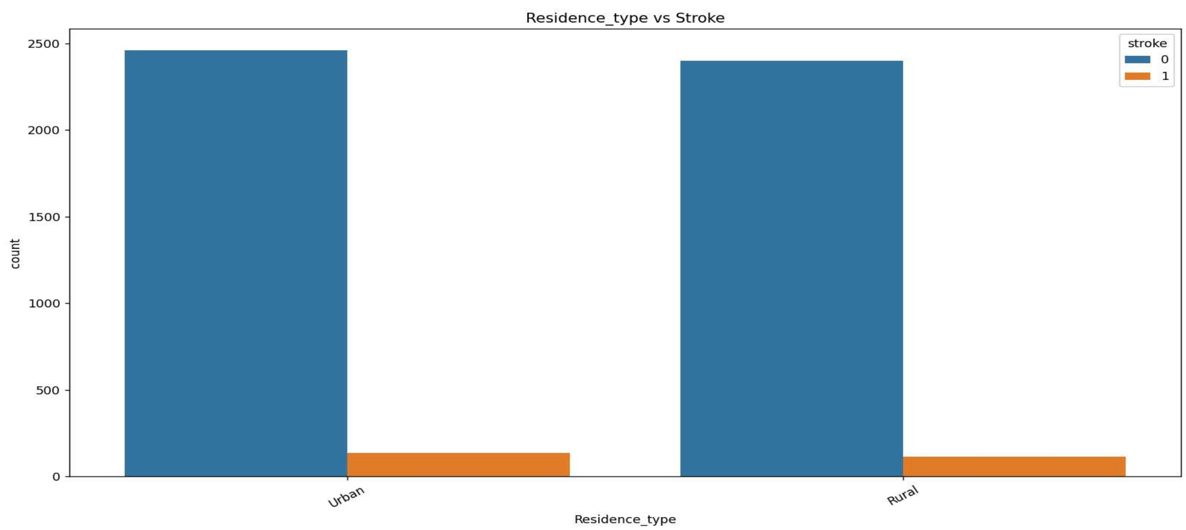

bmi vs Stroke
20
40
60
80
100
0
1
stroke

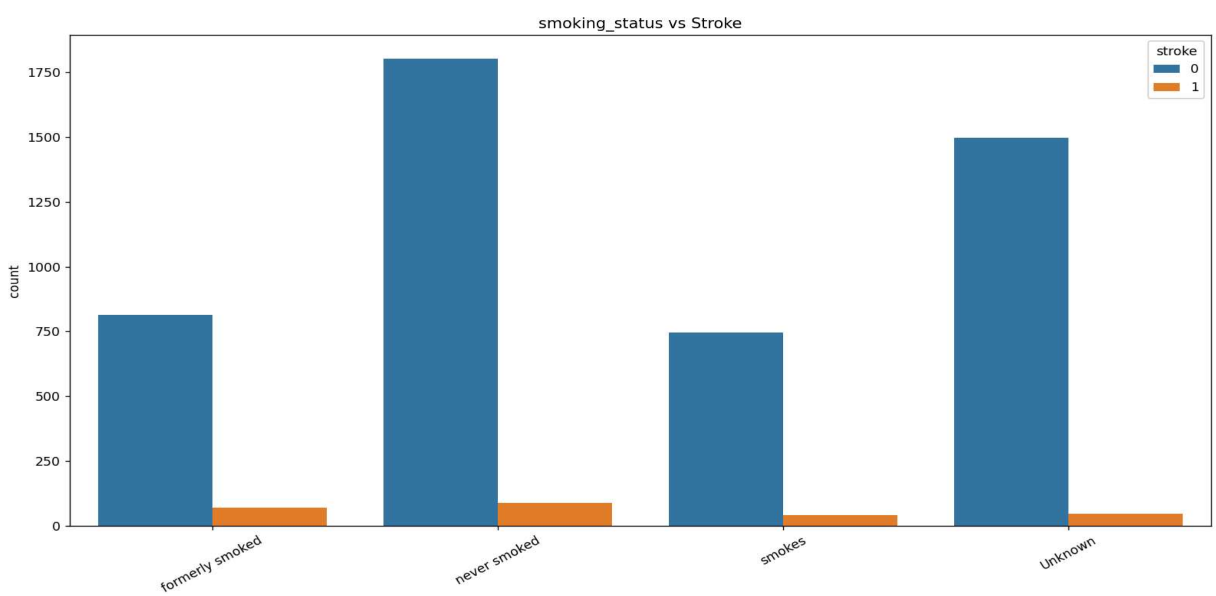
```python
# 7. Categorical Feature Analysis
cat_cols = [col for col in cat_cols if col != 'id']  # Exclude 'id' if present
for col in cat_cols:
    plt.figure(figsize=(7,4))
    sns.countplot(x=col, hue='stroke', data=df)
    plt.title(f'{col} vs Stroke')
    plt.xticks(rotation=30)
    plt.show()
```






work_type vs Stroke
stroke
0
1
count
0
500
1000
1500
2000
2500
Private
Self-employed
Govt_job
children
Never_worked


Residence_type vs Stroke
stroke
0
1
count
0
500
1000
1500
2000
2500
Urban
Rural
Residence_type


smoking_status vs Stroke
stroke
0
1
count
0
250
500
750
1000
1250
1500
1750
formerly smoked
never smoked
smokes
Unknown

```python
# 8. Multivariate Visualization (Pairplot)
selected_features = ['age', 'avg_glucose_level', 'bmi', 'stroke']
sns.pairplot(df[selected_features].dropna(), hue='stroke', palette='Set2', diag_kind='kde')
plt.suptitle("Pairplot of Selected Features", y=1.02)
plt.show()

# 9. Example: Grouped Analysis (Age Groups)
df['age_group'] = pd.cut(df['age'], bins=[0,20,40,60,80,100,120], labels=['0-20','21-40','41-60','61-80','81-100','100+'])
plt.figure(figsize=(8,4))
sns.countplot(x='age_group', hue='stroke', data=df)
plt.title('Age Group vs Stroke')
plt.show()

# 10. Summary Statement (Edit or expand as needed)
print("""
Interpretation & Summary:
- Review the class balance for the target variable 'stroke'.
- Identify which features differ between stroke and non-stroke groups.
- Check for missing values and outliers (especially in 'bmi').
- Assess correlation strength between features and the target.
- Use these insights for preprocessing and modeling.
""")
```