HEART STROKE PREDICTION SYSTEM

In this machine learning project, the overall topic that will be resolved is in the field of stroke health, where it will try to predict the possibility of a stroke in a person with certain conditions based on several factors including: age, certain diseases (hypertension, heart disease), smoking, etc.

Install and import required libraries.

```
# library for data processing
import pandas as pd
from sklearn.preprocessing import LabelEncoder
#Library for data visualization
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
# library for modeling
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
# library for model evaluation
from sklearn.metrics import accuracy_score
```

Read data with pandas

```
df = pd.read_csv('healthcare-dataset-stroke-data.csv')
```

Explore Dataset information

```
df.tail()

         id  gender   age  hypertension  heart_disease ever_married  \
5105  18234  Female  80.0             1              0          Yes
5106  44873  Female  81.0             0              0          Yes
5107  19723  Female  35.0             0              0          Yes
5108  37544    Male  51.0             0              0          Yes
5109  44679  Female  44.0             0              0          Yes

          work_type Residence_type  avg_glucose_level   bmi
smoking_status  \
5105        Private          Urban              83.75   NaN      never
```

```
smoked
5106  Self-employed          Urban              125.20  40.0     never
smoked
5107  Self-employed          Rural               82.99  30.6     never
smoked
5108        Private          Rural              166.29  25.6  formerly
smoked
5109       Govt_job          Urban               85.28  26.2
Unknown

      stroke
5105       0
5106       0
5107       0
5108       0
5109       0
```

check dataset info

```
df.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
```

describe numeric column

```
df.describe()
```

```
                 id          age  hypertension  heart_disease  \
count   5110.000000  5110.000000   5110.000000    5110.000000
mean   36517.829354    43.226614      0.097456       0.054012
std    21161.721625    22.612647      0.296607       0.226063
min       67.000000     0.080000      0.000000       0.000000
25%    17741.250000    25.000000      0.000000       0.000000
50%    36932.000000    45.000000      0.000000       0.000000
75%    54682.000000    61.000000      0.000000       0.000000
max    72940.000000    82.000000      1.000000       1.000000

       avg_glucose_level          bmi       stroke
count        5110.000000  4909.000000  5110.000000
mean          106.147677    28.893237     0.048728
std            45.283560     7.854067     0.215320
min            55.120000    10.300000     0.000000
25%            77.245000    23.500000     0.000000
50%            91.885000    28.100000     0.000000
75%           114.090000    33.100000     0.000000
max           271.740000    97.600000     1.000000
```

Check value counts

```
df["stroke"].value_counts()

0    4861
1     249
Name: stroke, dtype: int64

def get_smoke_count(smoking_status):
    mask=df['smoking_status']==smoking_status
    return df[mask]

get_smoke_count('Unknown')

         id  gender   age  hypertension  heart_disease ever_married
work_type  \
8     27419  Female  59.0             0              0          Yes
Private
9     60491  Female  78.0             0              0          Yes
Private
13     8213    Male  78.0             0              1          Yes
Private
19    25226    Male  57.0             0              1           No
Govt_job
23    64778    Male  82.0             0              1          Yes
Private
...     ...     ...   ...           ...            ...          ...
...
5098    579    Male   9.0             0              0           No
```

```
children
5101  36901  Female  45.0             0              0          Yes
Private
5103  22127  Female  18.0             0              0           No
Private
5104  14180  Female  13.0             0              0           No
children
5109  44679  Female  44.0             0              0          Yes
Govt_job

     Residence_type  avg_glucose_level   bmi smoking_status  stroke
8             Rural              76.15   NaN        Unknown       1
9             Urban              58.57  24.2        Unknown       1
13            Urban             219.84   NaN        Unknown       1
19            Urban             217.08   NaN        Unknown       1
23            Rural             208.30  32.5        Unknown       1
...             ...                ...   ...            ...     ...
5098          Urban              71.88  17.5        Unknown       0
5101          Urban              97.95  24.5        Unknown       0
5103          Urban              82.85  46.9        Unknown       0
5104          Rural             103.08  18.6        Unknown       0
5109          Urban              85.28  26.2        Unknown       0

[1544 rows x 12 columns]
```

```
df["smoking_status"].value_counts()
```

```
never smoked       1892
Unknown            1544
formerly smoked     885
smokes              789
Name: smoking_status, dtype: int64
```

Check for null values

```
df.isnull().sum()
```

```
id                     0
gender                 0
age                    0
hypertension           0
heart_disease          0
ever_married           0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                  201
smoking_status         0
```

```
stroke                    0
dtype: int64

null_values = df['bmi'].isnull().sum()
null_values

201

null_percentage = df['bmi'].isnull().mean()*100
null_percentage

3.9334637964774952

column_mean = df['bmi'].mean()
df['bmi'].fillna(column_mean, inplace=True)

df.columns

Index(['id', 'gender', 'age', 'hypertension', 'heart_disease',
'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')

columns = ['gender','ever_married','work_type',
'Residence_type','smoking_status']

for col in columns:
    unique_values = df[col].unique()
    print(f"Unique values for{col}:{unique_values}")

Unique values forgender:['Male' 'Female' 'Other']
Unique values forever_married:['Yes' 'No']
Unique values forwork_type:['Private' 'Self-employed' 'Govt_job'
'children' 'Never_worked']
Unique values forResidence_type:['Urban' 'Rural']
Unique values forsmoking_status:['formerly smoked' 'never smoked'
'smokes' 'Unknown']

df['gender'].value_counts()

Female    2994
Male      2115
Other        1
Name: gender, dtype: int64

# In gender column there is only one value for other so we replace it
with male
df['gender'] = df['gender'].replace('Other', 'Male')

df['gender'].value_counts()
```

```
Female    2994
Male      2116
Name: gender, dtype: int64
```

```
mask1 = df[['gender','ever_married','work_type',
'Residence_type','smoking_status']].nunique()
mask1
```

```
gender            2
ever_married      2
work_type         5
Residence_type    2
smoking_status    4
dtype: int64
```

```
df.info()
```

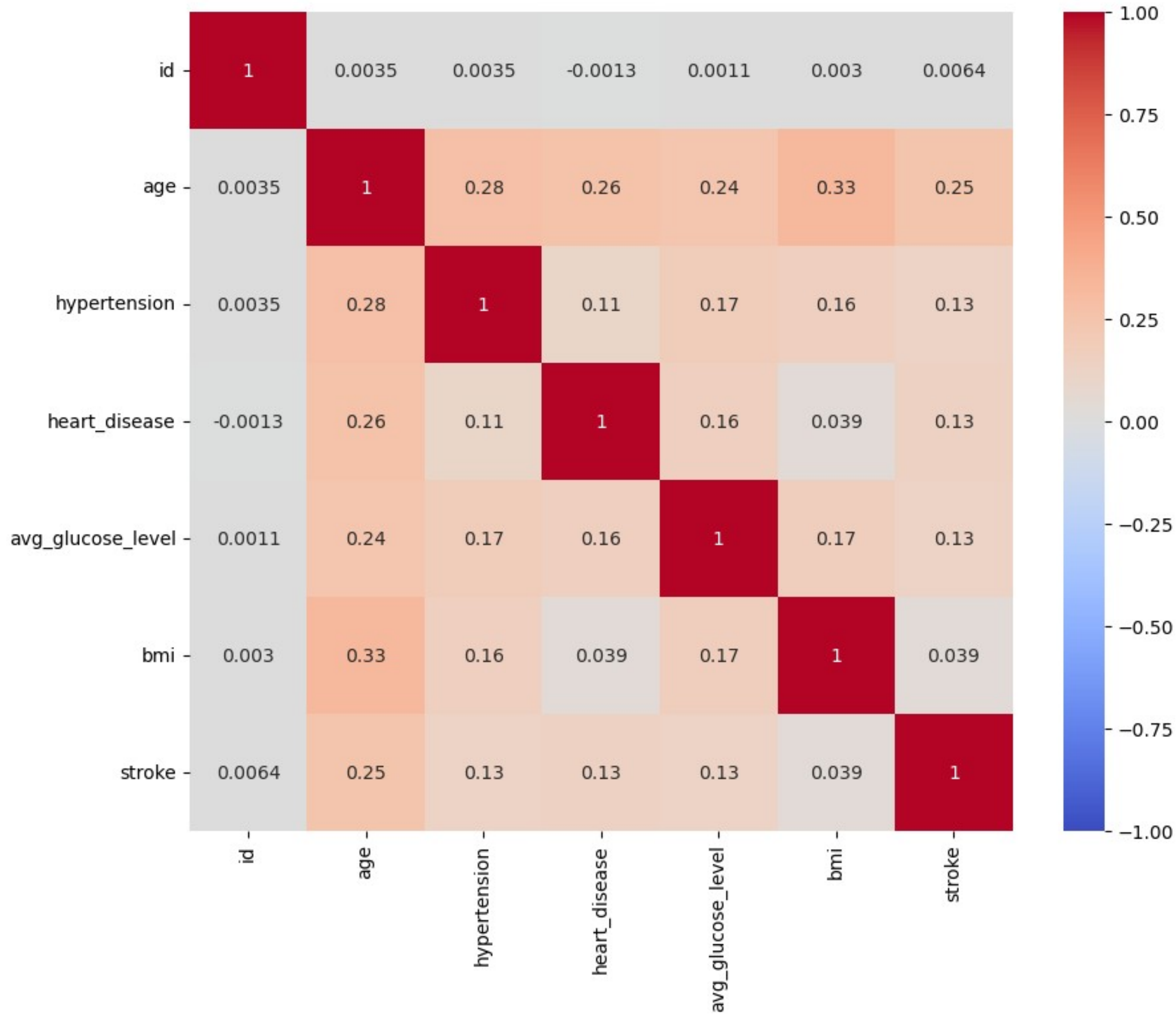
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   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                5110 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
```

Data Visualisation

```
corr_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True,vmin=-1, vmax=1, cmap='coolwarm')
plt.show()
```

```
C:\Users\prita\AppData\Local\Temp\ipykernel_17232\1618691362.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
```

```
  corr_matrix = df.corr()
C:\Users\prita\AppData\Local\Temp\ipykernel_17232\1618691362.py:3:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  sns.heatmap(df.corr(), annot=True,vmin=-1, vmax=1, cmap='coolwarm')
```



Feature Engineering

```
# Creating age group categories

print(f'maximum age variable: {df["age"].max()}')
print(f'minimum age variable: {df["age"].min()}')
print(f'Number of age variable: {df["age"].nunique()}')
```

```
maximum age variable: 82.0
minimum age variable: 0.08
Number of age variable: 104
```

```
# collapse age group categories
ranges = [0,13,18,45,60,100]
group_names = ['Children','Teens','Adults','Mid-adults','Elderly']
df['age_group'] = pd.cut(df['age'],bins=ranges,labels=group_names)
df['age_group'].unique()
```

```
['Elderly', 'Mid-adults', 'Adults', 'Children', 'Teens']
Categories (5, object): ['Children' < 'Teens' < 'Adults' < 'Mid-
adults' < 'Elderly']
```

```
# For BMI

print(f'maximum age variable: {df["bmi"].max()}')
print(f'minimum age variable: {df["bmi"].min()}')
print(f'Number of age variable: {df["bmi"].nunique()}')
```

```
maximum age variable: 97.6
minimum age variable: 10.3
Number of age variable: 419
```

```
# collapse bmi into fewer groups

ranges = [0,19,25,30,100]
group_names = ['Underweight', 'Normal', 'Overweight', 'Obesity']
df['bmi_group'] = pd.cut(df['bmi'],bins=ranges,labels=group_names)
df['bmi_group'].unique()
```

```
['Obesity', 'Overweight', 'Normal', 'Underweight']
Categories (4, object): ['Underweight' < 'Normal' < 'Overweight' <
'Obesity']
```

```
# for avg glucose level

print(f'maximum age variable: {df["avg_glucose_level"].max()}')
print(f'minimum age variable: {df["avg_glucose_level"].min()}')
print(f'Number of age variable: {df["avg_glucose_level"].nunique()}')
```

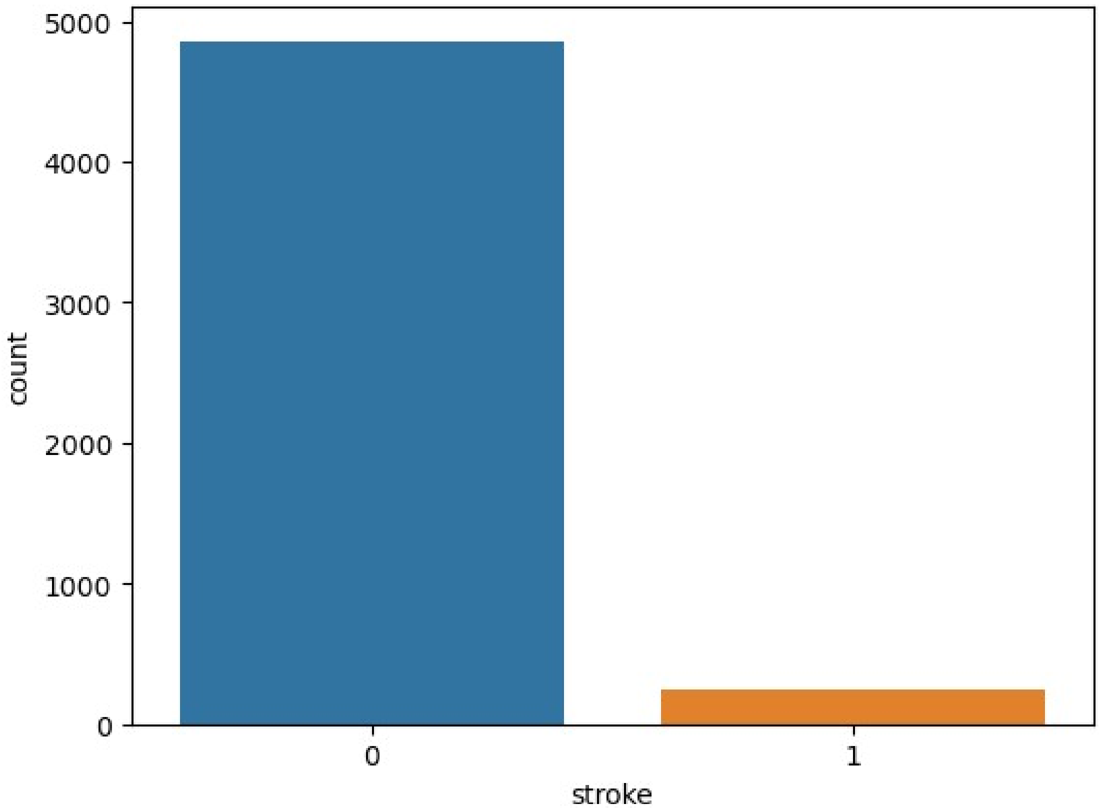
```
maximum age variable: 271.74
minimum age variable: 55.12
Number of age variable: 3979
```

```
ranges = [0, 70, 99, 125, 280]
group_names = ['Low', 'Normal', 'High', 'Very_high']
df['avg_glucose_level_group'] =
pd.cut(df['avg_glucose_level'],bins=ranges,labels=group_names)
df['avg_glucose_level_group'].unique()
```

```
['Very_high', 'High', 'Normal', 'Low']
Categories (4, object): ['Low' < 'Normal' < 'High' < 'Very_high']
```

Exploratory data analysis(EDA)

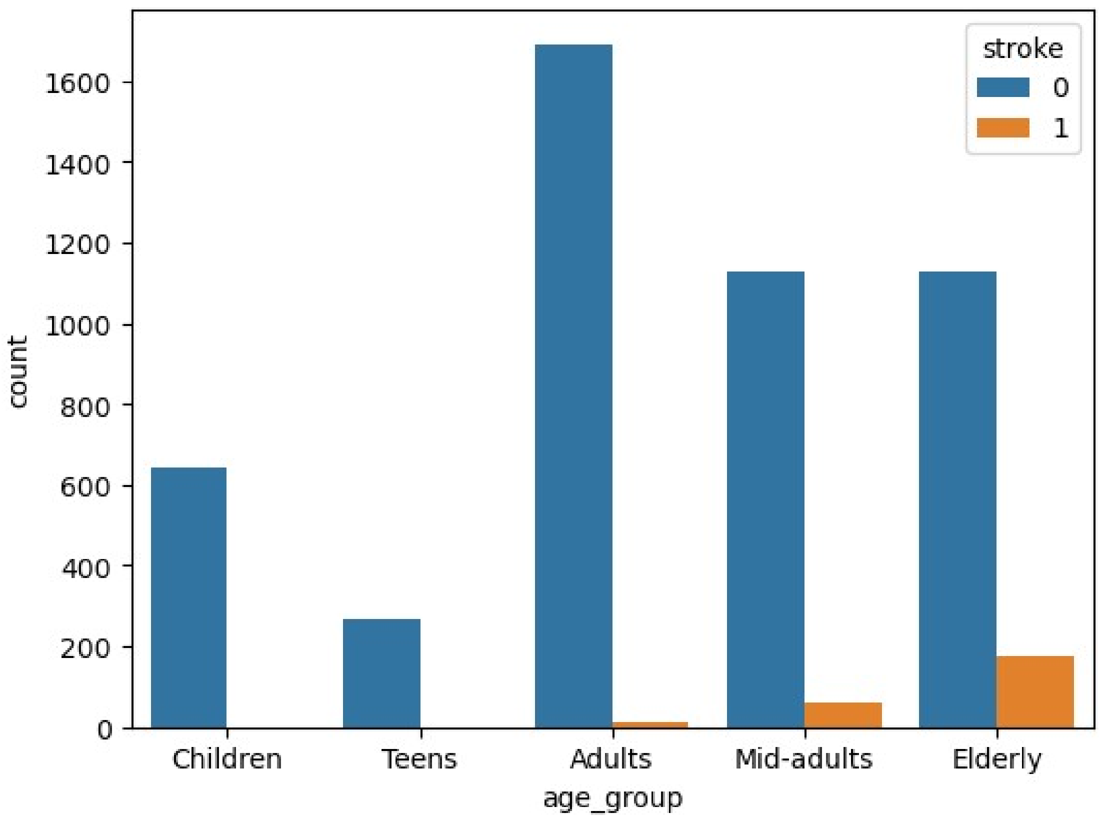
```
sns.countplot(x='stroke',data=df)
plt.show()
```



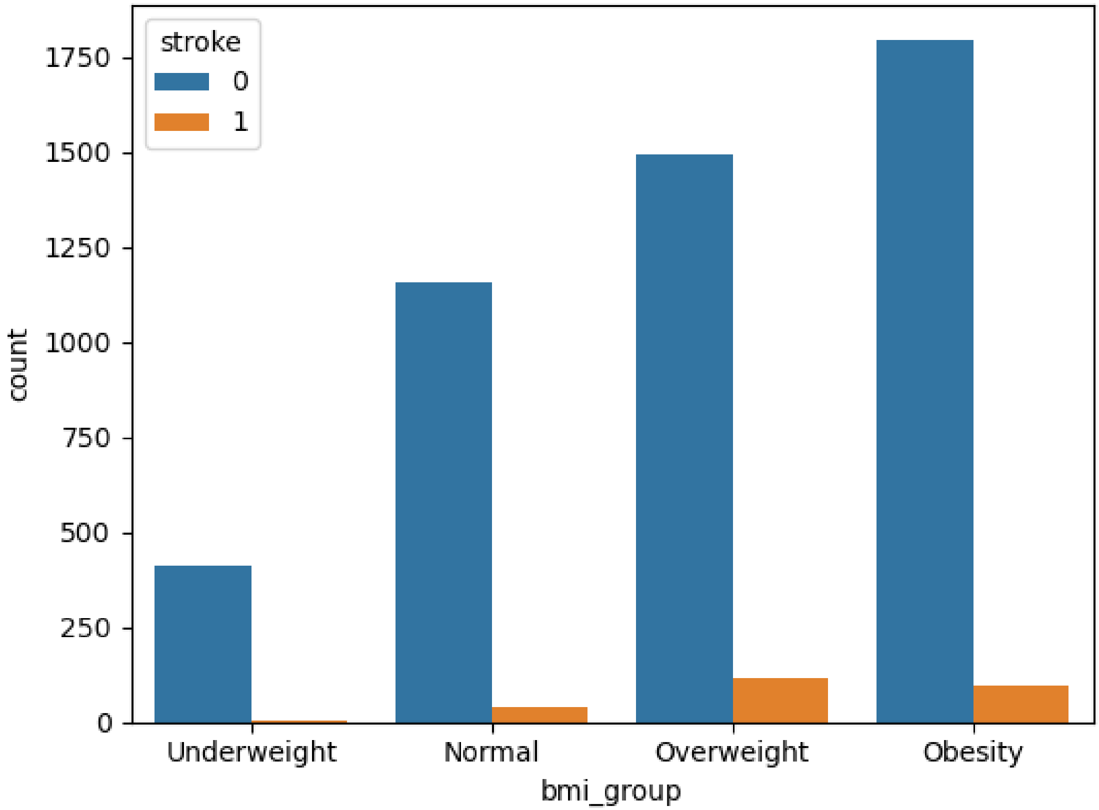

```
df.head()
```

```
      id  gender   age  hypertension  heart_disease ever_married  \
0   9046    Male  67.0             0              1          Yes
1  51676  Female  61.0             0              0          Yes
2  31112    Male  80.0             0              1          Yes
3  60182  Female  49.0             0              0          Yes
4   1665  Female  79.0             1              0          Yes

       work_type Residence_type  avg_glucose_level        bmi  \
0        Private          Urban             228.69  36.600000
1  Self-employed          Rural             202.21  28.893237
2        Private          Rural             105.92  32.500000
3        Private          Urban             171.23  34.400000
```

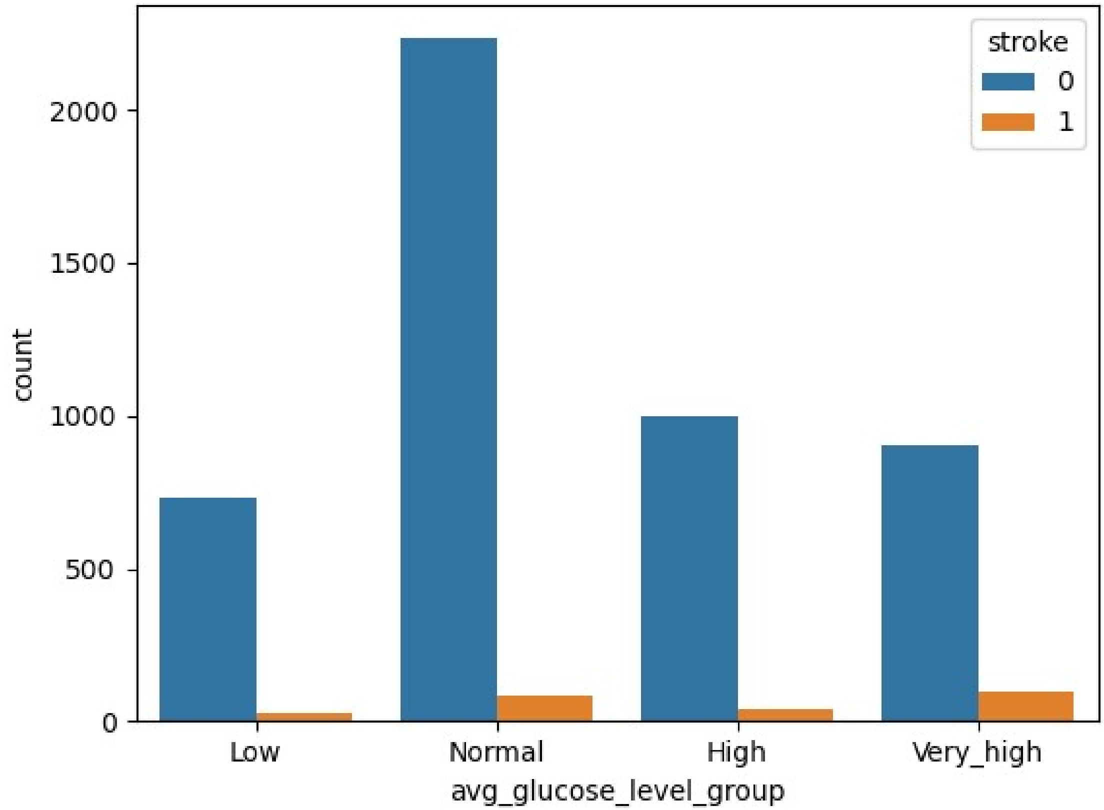
```
4  Self-employed          Rural            174.12  24.000000

    smoking_status  stroke   age_group   bmi_group
avg_glucose_level_group
0  formerly smoked       1     Elderly     Obesity
Very_high
1     never smoked       1     Elderly  Overweight
Very_high
2     never smoked       1     Elderly     Obesity
High
3           smokes       1  Mid-adults     Obesity
Very_high
4     never smoked       1     Elderly      Normal
Very_high

sns.countplot(x='age_group',hue='stroke',data=df)
plt.show()
```




```
sns.countplot(x='bmi_group',hue='stroke',data=df)
plt.show()
```

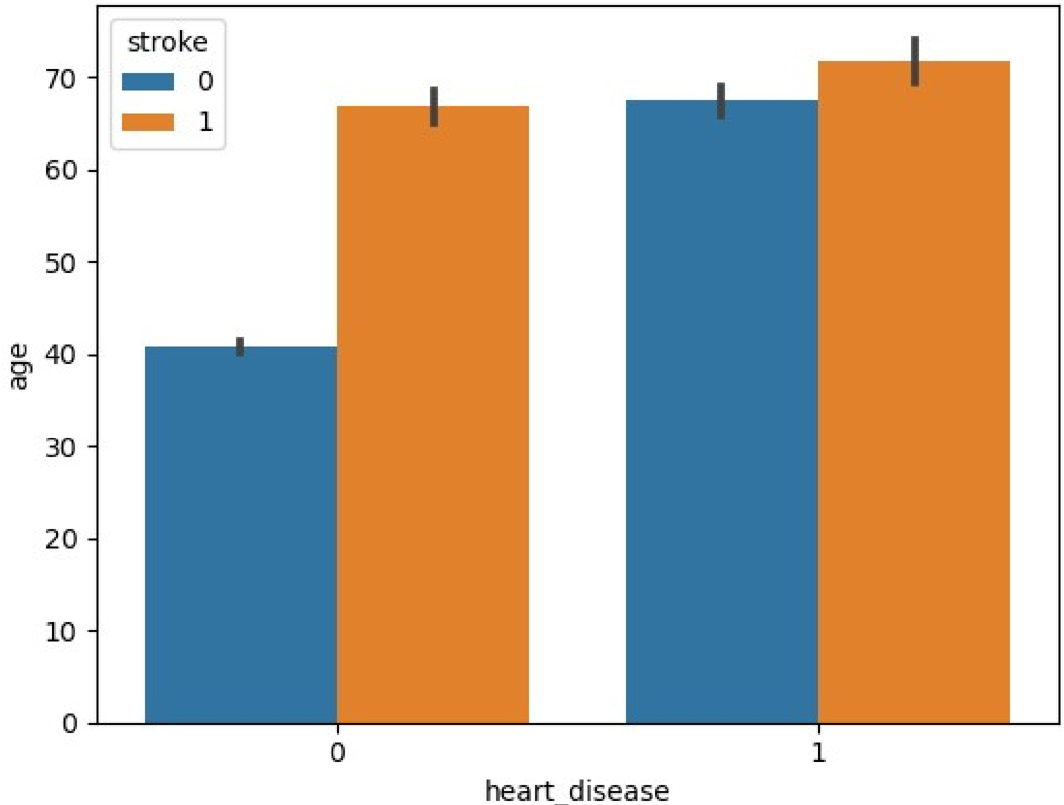



```
sns.countplot(x='avg_glucose_level_group',hue='stroke',data=df)
plt.show()
```




```
sns.barplot(x='heart_disease',y='age',data=df,hue='stroke')
```

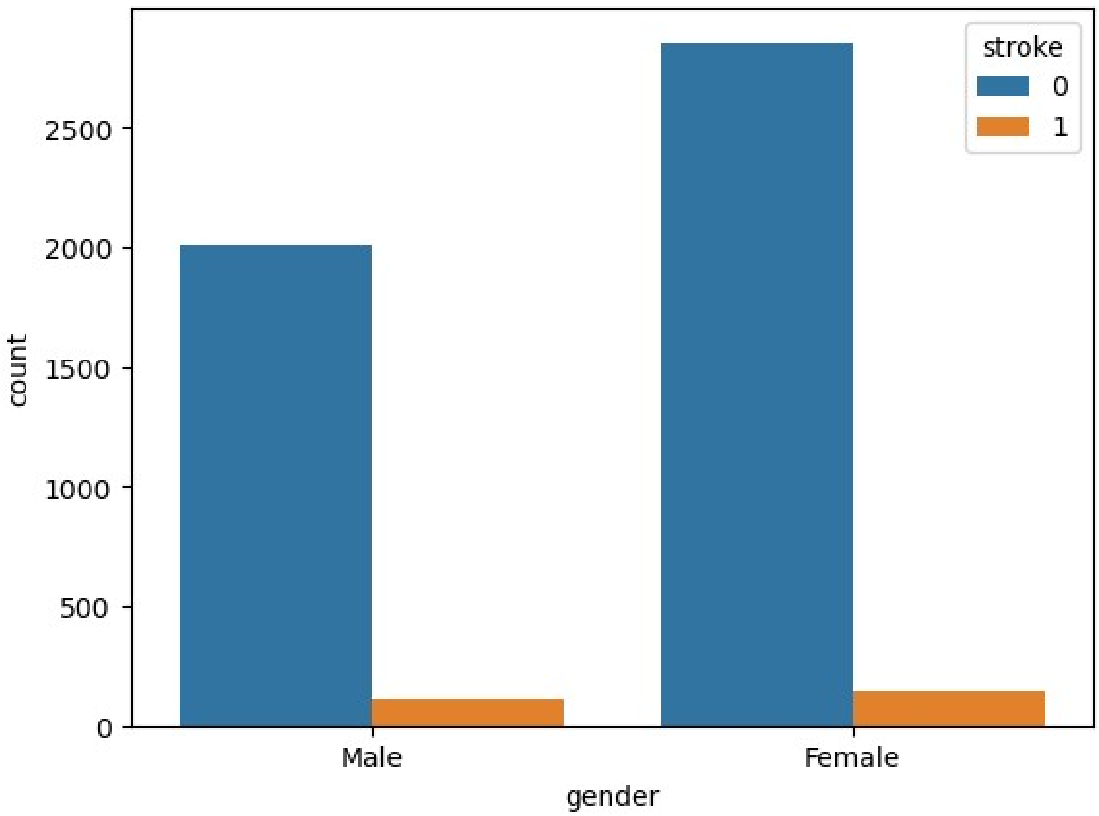
```
<Axes: xlabel='heart_disease', ylabel='age'>
```



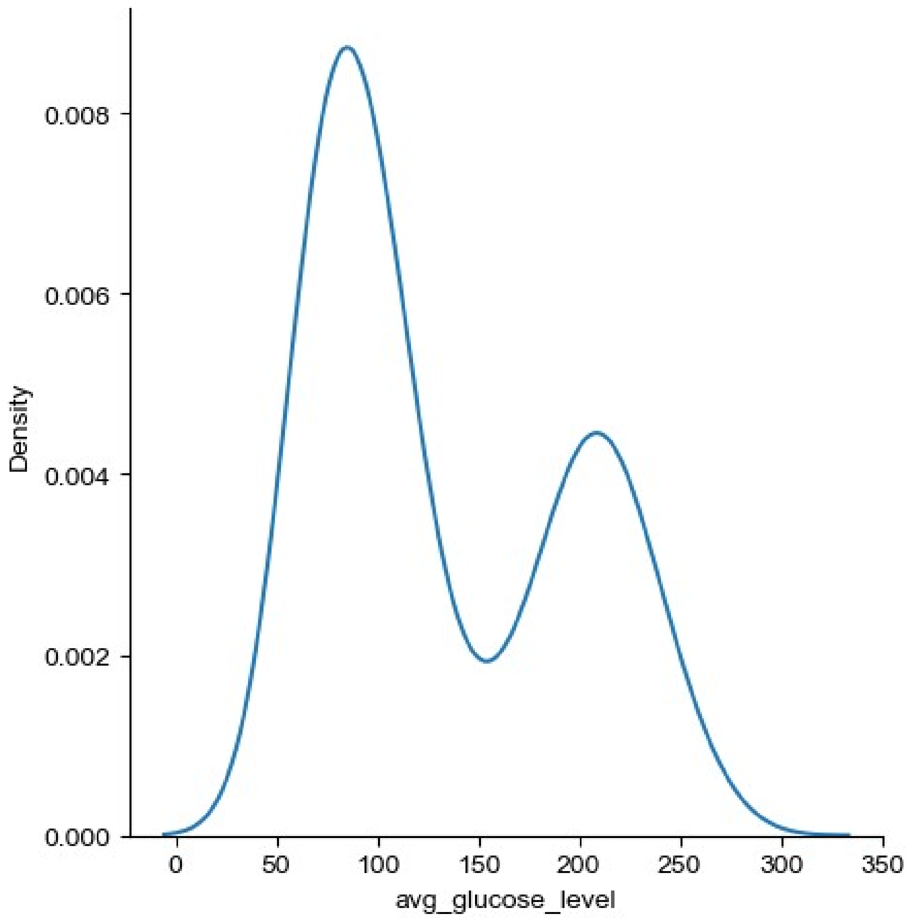

```
df
```

```
          id  gender   age  hypertension  heart_disease ever_married  \
0       9046    Male  67.0             0              1          Yes
1      51676  Female  61.0             0              0          Yes
2      31112    Male  80.0             0              1          Yes
3      60182  Female  49.0             0              0          Yes
4       1665  Female  79.0             1              0          Yes
...      ...     ...   ...           ...            ...          ...
5105   18234  Female  80.0             1              0          Yes
5106   44873  Female  81.0             0              0          Yes
5107   19723  Female  35.0             0              0          Yes
5108   37544    Male  51.0             0              0          Yes
5109   44679  Female  44.0             0              0          Yes

          work_type Residence_type  avg_glucose_level        bmi  \
0           Private          Urban             228.69  36.600000
1     Self-employed          Rural             202.21  28.893237
2           Private          Rural             105.92  32.500000
3           Private          Urban             171.23  34.400000
4     Self-employed          Rural             174.12  24.000000
...             ...            ...                ...        ...
5105        Private          Urban              83.75  28.893237
5106  Self-employed          Urban             125.20  40.000000
```

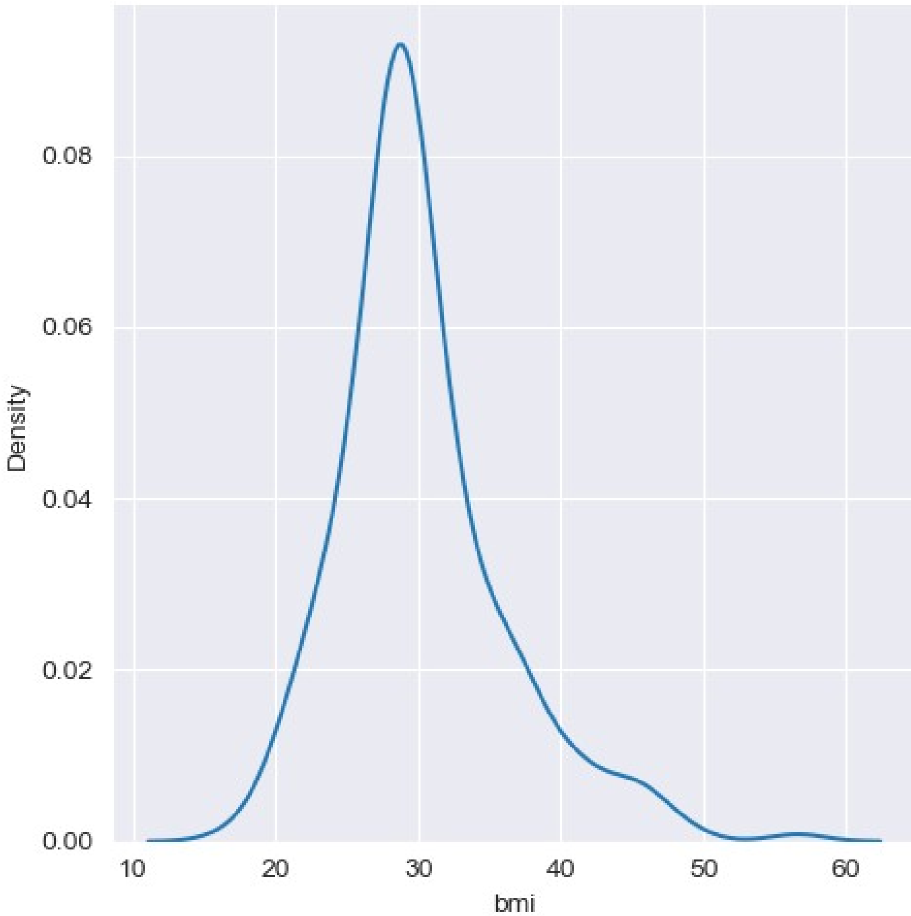
```
5107  Self-employ ed          Rural              82.99  30.600000
5108        Private          Rural             166.29  25.600000
5109       Govt_job          Urban              85.28  26.200000

       smoking_status  stroke   age_group   bmi_group
avg_glucose_level_group
0     formerly smoked       1     Elderly     Obesity
Very_high
1        never smoked       1     Elderly  Overweight
Very_high
2        never smoked       1     Elderly     Obesity
High
3              smokes       1  Mid-adults     Obesity
Very_high
4        never smoked       1     Elderly      Normal
Very_high
...               ...     ...         ...         ...
...
5105     never smoked       0     Elderly  Overweight
Normal
5106     never smoked       0     Elderly     Obesity
Very_high
5107     never smoked       0      Adults     Obesity
Normal
5108  formerly smoked       0  Mid-adults  Overweight
Very_high
5109          Unknown       0      Adults  Overweight
Normal

[5110 rows x 15 columns]

sns.countplot(x='gender',hue='stroke',data=df)
plt.show()
```




```
stroke = df[df['stroke']==1]
sns.displot(stroke['avg_glucose_level'], kind='kde')
sns.set_style('darkgrid')
plt.show()
```




```
stroke = df[df['stroke']==1]
sns.displot(stroke['bmi'], kind='kde')
sns.set_style('darkgrid')
plt.show()
```



Preprocessing

Binary Encoding

```
#Instantiate LabelEncoder
labelencoder = LabelEncoder()

#Binary Encoding(encoding object columns with 2 unique values)
binary_cols =  ['ever_married', 'Residence_type', 'gender']
for col in binary_cols:
    df[col]=labelencoder.fit_transform(df[col])
```

Label Encoding

```
categorical_cols = ['age_group', 'bmi_group',
'avg_glucose_level_group']

label_encoder = LabelEncoder()
for col in categorical_cols:
    df[col] = label_encoder.fit_transform(df[col])
```

One hot encoding

```
#Encode object columns that more than 2 unique values
df = pd.get_dummies(df, columns=['work_type', 'smoking_status'],
drop_first=True)

df

         id  gender   age  hypertension  heart_disease
ever_married  \
0      9046       1  67.0             0              1             1

1     51676       0  61.0             0              0             1

2     31112       1  80.0             0              1             1

3     60182       0  49.0             0              0             1

4      1665       0  79.0             1              0             1

...     ...     ...   ...           ...            ...           ...

5105  18234       0  80.0             1              0             1

5106  44873       0  81.0             0              0             1

5107  19723       0  35.0             0              0             1

5108  37544       1  51.0             0              0             1

5109  44679       0  44.0             0              0             1


      Residence_type  avg_glucose_level        bmi  stroke  age_group
\
0                  1             228.69  36.600000       1          2

1                  0             202.21  28.893237       1          2

2                  0             105.92  32.500000       1          2

3                  1             171.23  34.400000       1          3

4                  0             174.12  24.000000       1          2

...              ...                ...        ...     ...        ...

5105               1              83.75  28.893237       0          2

5106               1             125.20  40.000000       0          2

5107               0              82.99  30.600000       0          0
```

```
5108              0             166.29  25.600000       0          3

5109              1              85.28  26.200000       0          0


      bmi_group  avg_glucose_level_group  work_type_Never_worked  \
0             1                        3                       0
1             2                        3                       0
2             1                        0                       0
3             1                        3                       0
4             0                        3                       0
...         ...                      ...                     ...
5105          2                        2                       0
5106          1                        3                       0
5107          1                        2                       0
5108          2                        3                       0
5109          2                        2                       0

      work_type_Private  work_type_Self-employed
work_type_children  \
0                     1                        0                    0

1                     0                        1                    0

2                     1                        0                    0

3                     1                        0                    0

4                     0                        1                    0

...                 ...                      ...                  ...

5105                  1                        0                    0

5106                  0                        1                    0

5107                  0                        1                    0

5108                  1                        0                    0

5109                  0                        0                    0


      smoking_status_formerly smoked  smoking_status_never smoked  \
0                                  1                            0
1                                  0                            1
2                                  0                            1
3                                  0                            0
4                                  0                            1
...                              ...                          ...
```

```
5105                                    0                                    1
5106                                    0                                    1
5107                                    0                                    1
5108                                    1                                    0
5109                                    0                                    0

      smoking_status_smokes
0                         0
1                         0
2                         0
3                         1
4                         0
...                     ...
5105                      0
5106                      0
5107                      0
5108                      0
5109                      0

[5110 rows x 20 columns]
```

Pre Modeling Steps

```
# separate feature and target
X = df.drop('stroke', axis=1)
y = df['stroke']

# using SMOTE Techniqe
# sm = SMOTE(random_state=111)
# X_sm , y_sm = sm.fit_resample(X,y)
#print(f'''Shape of X before SMOTE:{X.shape}
#Shape of X after SMOTE:{X_sm.shape}''',"\n\n")

# print(f'''Target Class distributuion before SMOTE:\
n{y.value_counts(normalize=True)}
# Target Class distributuion after SMOTE :\
n{y_sm.value_counts(normalize=True)}''')
```

Split train data and test data

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Machine Learning Modeling

1) Random forest Classifier

```
# Create simple model
rf_classifier = RandomForestClassifier(n_estimators=100,
random_state=42)

rf_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
# Test model with test data
y_pred = rf_classifier.predict(X_test)

# Simple model report
rf_accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {rf_accuracy}")
```

```
Accuracy: 0.9393346379647749
```

2) Logistic Regression

```
logistic_reg = LogisticRegression()
```

```
<IPython.core.display.Javascript object>
```

```
logistic_reg.fit(X_train, y_train)
```

```
LogisticRegression()
```

```
# Test model with test data
y_pred = logistic_reg.predict(X_test)

# Simple model report
log_reg_accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {log_reg_accuracy}")
```

```
Accuracy: 0.9373776908023483
```

3) Decision Tree

```
# Create the Decision Tree Classifier
decision_tree_classifier = DecisionTreeClassifier()

# Train the classifier on the training data
decision_tree_classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```python
# Make predictions on the test set
y_pred = decision_tree_classifier.predict(X_test)

# Calculate the accuracy of the classifier
dt_accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {dt_accuracy}")
```

```
Accuracy: 0.9158512720156555
```