```python
import subprocess

process = subprocess.Popen(["python", "-c", "import imbalanced_learn;
print(imbalanced_learn.__file__)"], stdout=subprocess.PIPE)
output, err = process.communicate()

if err:
    print("Error:", err)
else:
    print("imbalanced_learn.__file__:", output.decode("utf-
8").strip())
```

```
imbalanced_learn.__file__:
```

```
rm -rf <path_to_imblearn_dir>
```

```
/bin/bash: -c: line 1: syntax error near unexpected token `newline'
/bin/bash: -c: line 1: `rm -rf <path_to_imblearn_dir>'
```

```python
import subprocess

process = subprocess.Popen(["python", "-c", "import sklearn;
print(sklearn.__file__)"], stdout=subprocess.PIPE)
output, err = process.communicate()

if err:
    print("Error:", err)
else:
    print("sklearn.__file__:", output.decode("utf-8").strip())
```

```
sklearn.__file__:
/usr/local/lib/python3.10/dist-packages/sklearn/__init__.py
```

```
rm -rf <path_to_sklearn_dir>
```

```
/bin/bash: -c: line 1: syntax error near unexpected token `newline'
/bin/bash: -c: line 1: `rm -rf <path_to_sklearn_dir>'
```

```
!pip cache purge
```

```
Files removed: 4
```

```
!pip install scikit-learn
!pip install imblearn
```

```
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.10/dist-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.5.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in
```

```
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
Requirement already satisfied: imblearn in
/usr/local/lib/python3.10/dist-packages (0.0)
Requirement already satisfied: imbalanced-learn in
/usr/local/lib/python3.10/dist-packages (from imblearn) (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn-
>imblearn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn-
>imblearn) (1.11.3)
Requirement already satisfied: scikit-learn>=1.0.2 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn-
>imblearn) (1.3.2)
Requirement already satisfied: joblib>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn-
>imblearn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn-
>imblearn) (3.2.0)
```

```
cd /content/drive/MyDrive/StrokeProject
```

```
/content/drive/MyDrive/StrokeProject
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as ms
import sklearn
```

```
train = pd.read_csv('train_2v.csv')
test = pd.read_csv('test_2v.csv')
train.head()
```

```
      id  gender   age  hypertension  heart_disease ever_married  \
0  30669    Male   3.0             0              0           No
1  30468    Male  58.0             1              0          Yes
2  16523  Female   8.0             0              0           No
3  56543  Female  70.0             0              0          Yes
4  46136    Male  14.0             0              0           No

      work_type Residence_type  avg_glucose_level   bmi
smoking_status  \
0      children          Rural              95.12  18.0
NaN
1       Private          Urban              87.96  39.2     never
```

```
smoked
2       Private          Urban                 110.89  17.6
NaN
3       Private          Rural                  69.04  35.9  formerly
smoked
4  Never_worked          Rural                 161.28  19.1
NaN

   stroke
0       0
1       0
2       0
3       0
4       0
```

In the dataset we have 12 columns where 11 contains the features and the last one contains the result

```
test.head()

      id  gender   age  hypertension  heart_disease ever_married  \
0  36306    Male  80.0             0              0          Yes
1  61829  Female  74.0             0              1          Yes
2  14152  Female  14.0             0              0           No
3  12997    Male  28.0             0              0           No
4  40801  Female  63.0             0              0          Yes

       work_type Residence_type  avg_glucose_level   bmi
smoking_status
0        Private          Urban              83.84  21.1  formerly
smoked
1  Self-employed          Rural             179.50  26.0  formerly
smoked
2       children          Rural              95.16  21.2
NaN
3        Private          Urban              94.76  23.4
NaN
4       Govt_job          Rural              83.57  27.6     never
smoked

train.shape

(43400, 12)

test.shape

(18601, 11)
```

Data Cleaning Identifing missing *attributes*

```
train_missing_values=train.isnull().sum()
train_missing_values
```

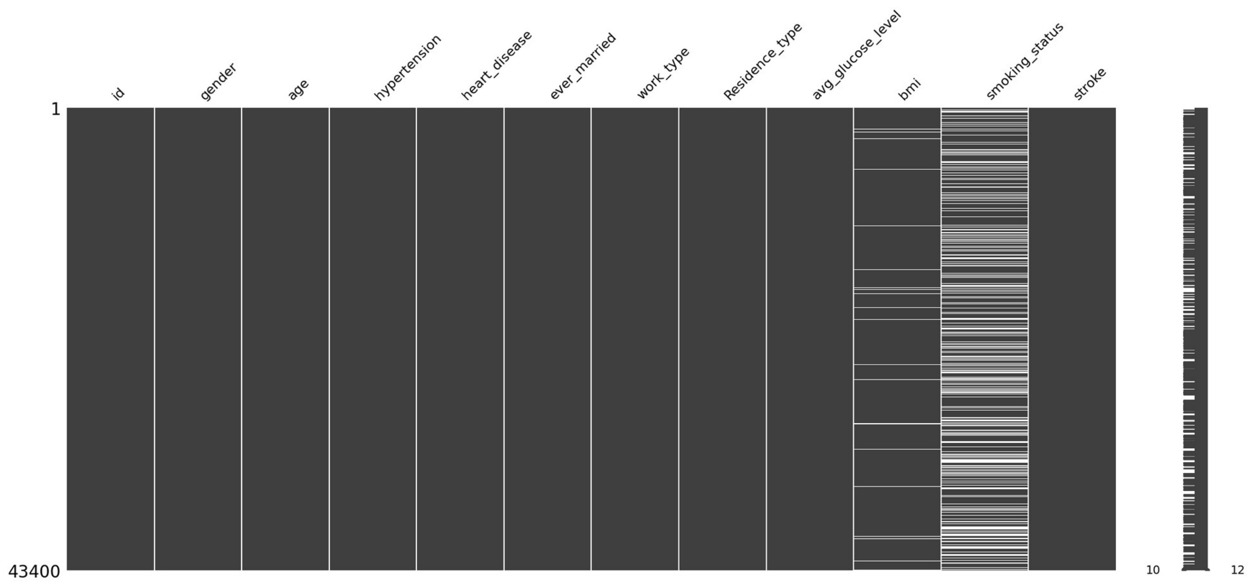
```
id                       0
gender                   0
age                      0
hypertension             0
heart_disease            0
ever_married             0
work_type                0
Residence_type           0
avg_glucose_level        0
bmi                   1462
smoking_status       13292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

```
test_missing_values=test.isnull().sum()
test_missing_values
```

```
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                   591
smoking_status       5751
dtype: int64
```

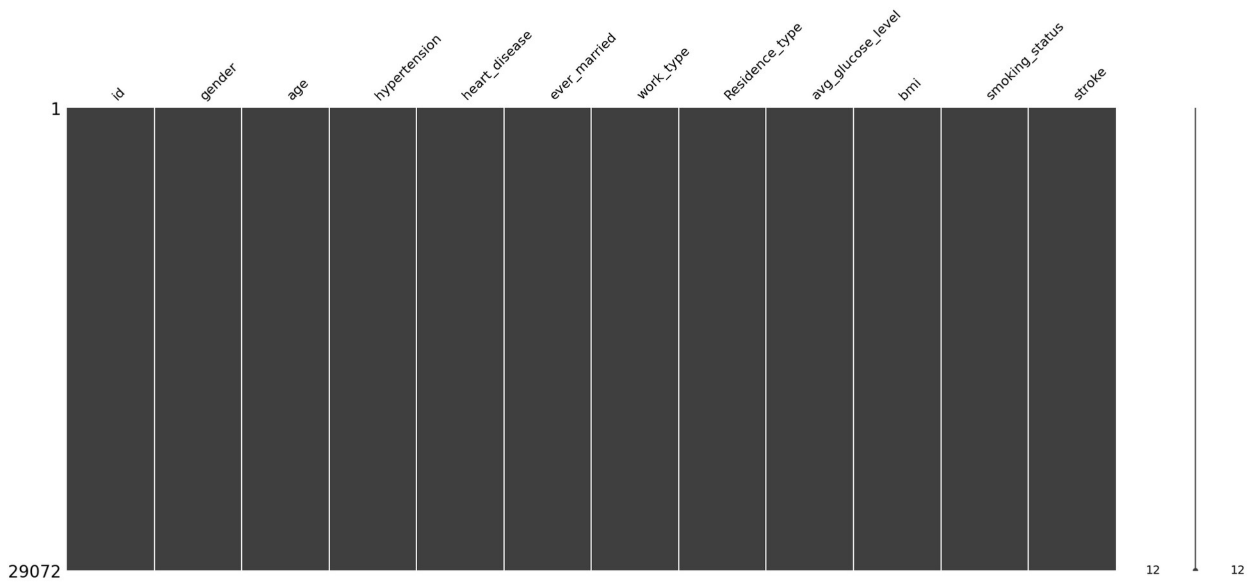
```
ms.matrix(train)
```

```
<Axes: >
```

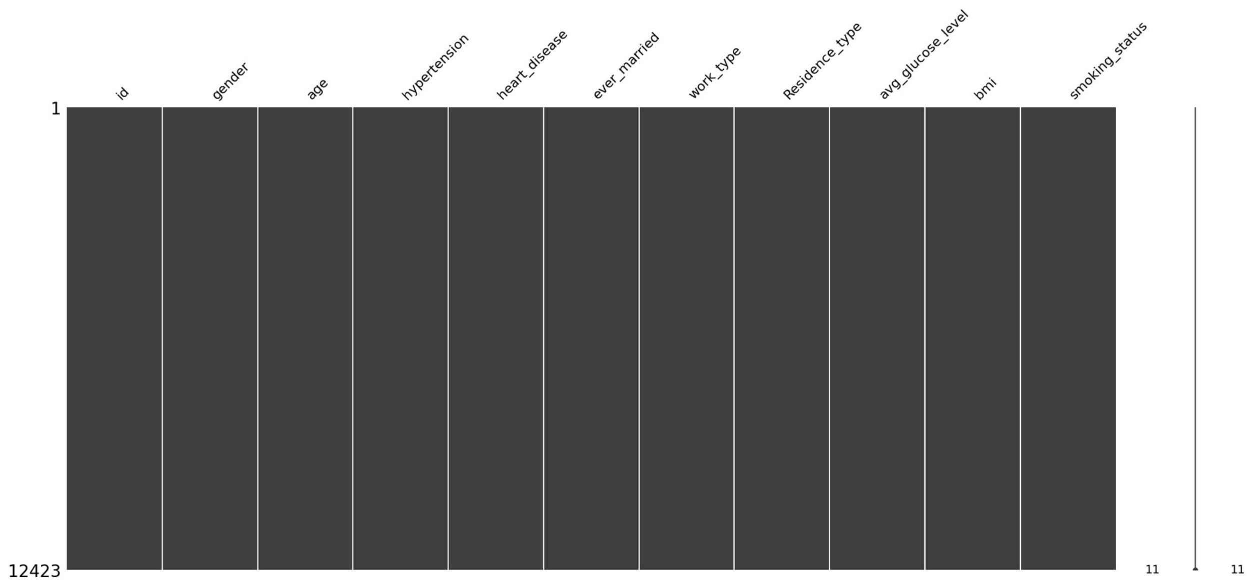


Removing missing value

```
train_data=train.dropna(axis=0,how="any")
test_data=test.dropna(axis=0,how="any")
print('train data shape: {}' .format(train_data.shape))
print('test data shape: {}' .format(test_data.shape))

train data shape: (29072, 12)
test data shape: (12423, 11)

ms.matrix(train_data)

<Axes: >
```

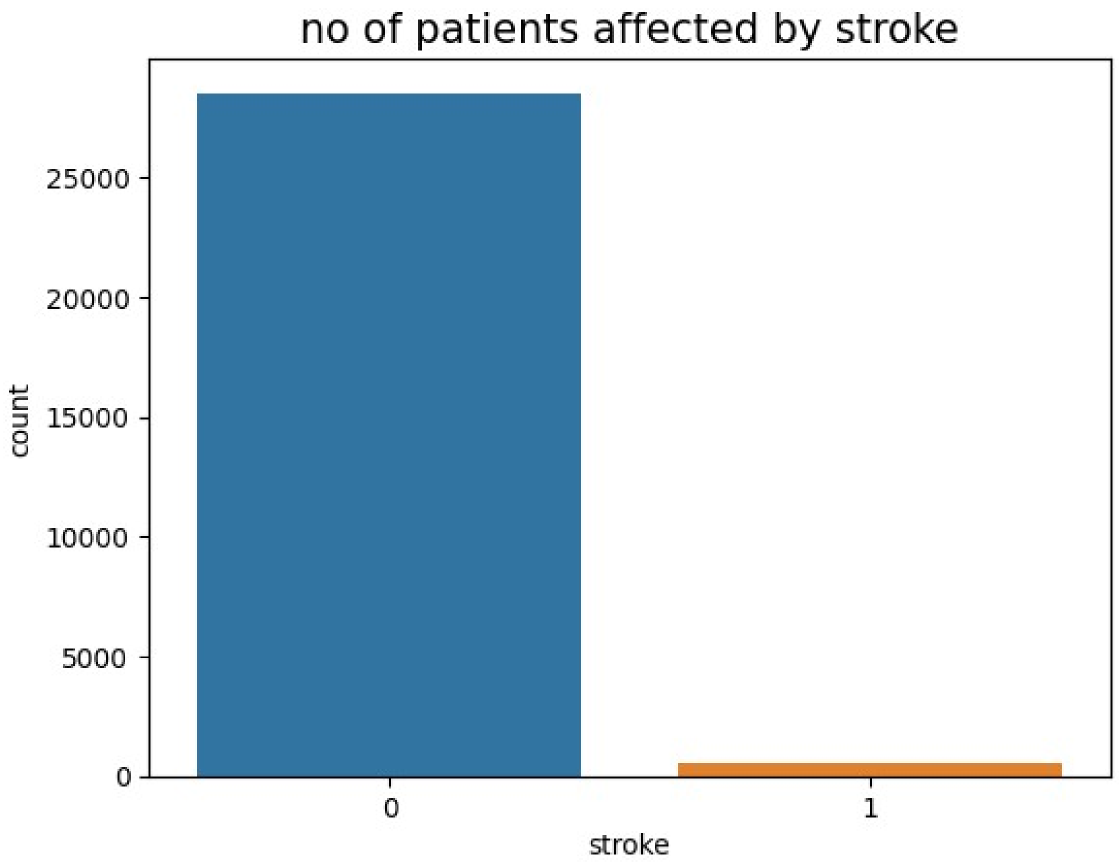


```
ms.matrix(test_data)

<Axes: >
```

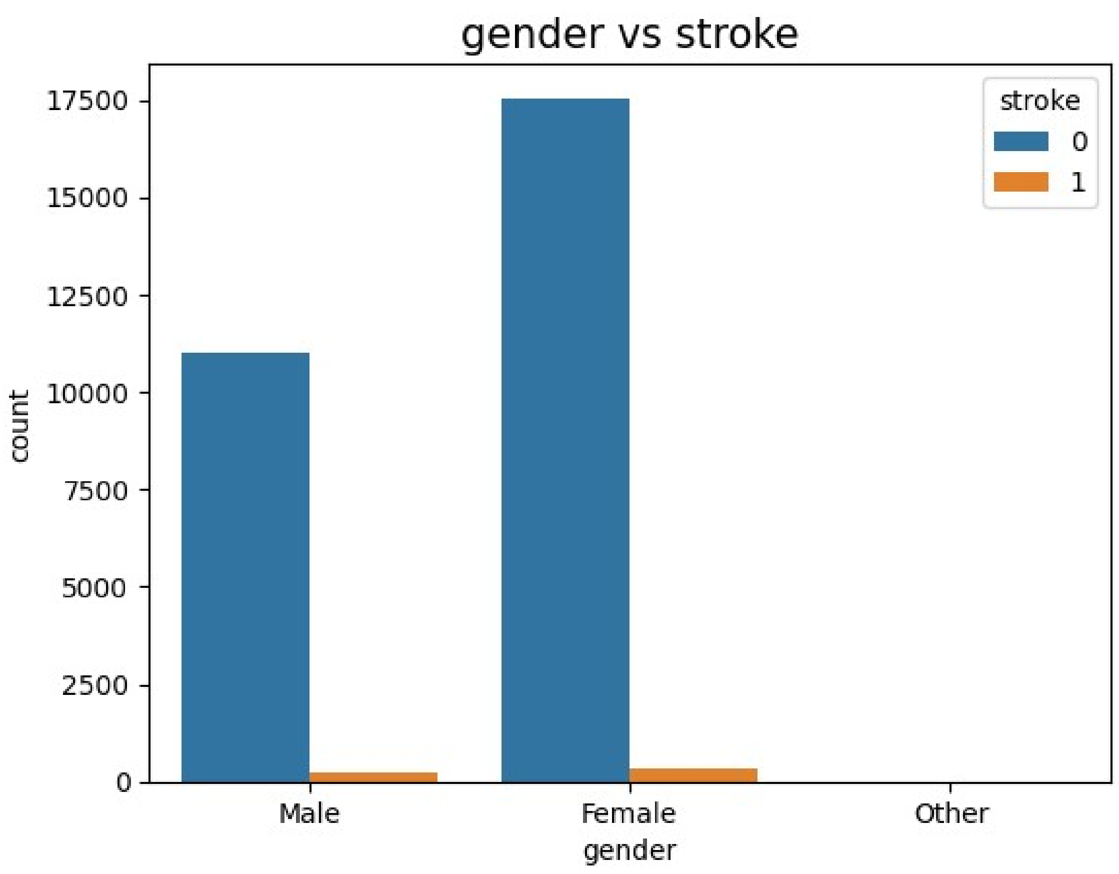


Pattern Recognition

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

0    28524
1      548
Name: stroke, dtype: int64

sns.countplot(x=train_data["stroke"])
plt.title("no of patients affected by stroke", fontsize=15)
plt.show()
```



```
sns.countplot(x=train_data["gender"], hue=train_data["stroke"])
plt.title("gender vs stroke", fontsize=15)
plt.show()
```



```
train_data.groupby(["gender"])["stroke"].value_counts()
```

```
gender  stroke
Female  0         17539
        1           313
Male    0         10978
        1           235
Other   0             7
Name: stroke, dtype: int64
```

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

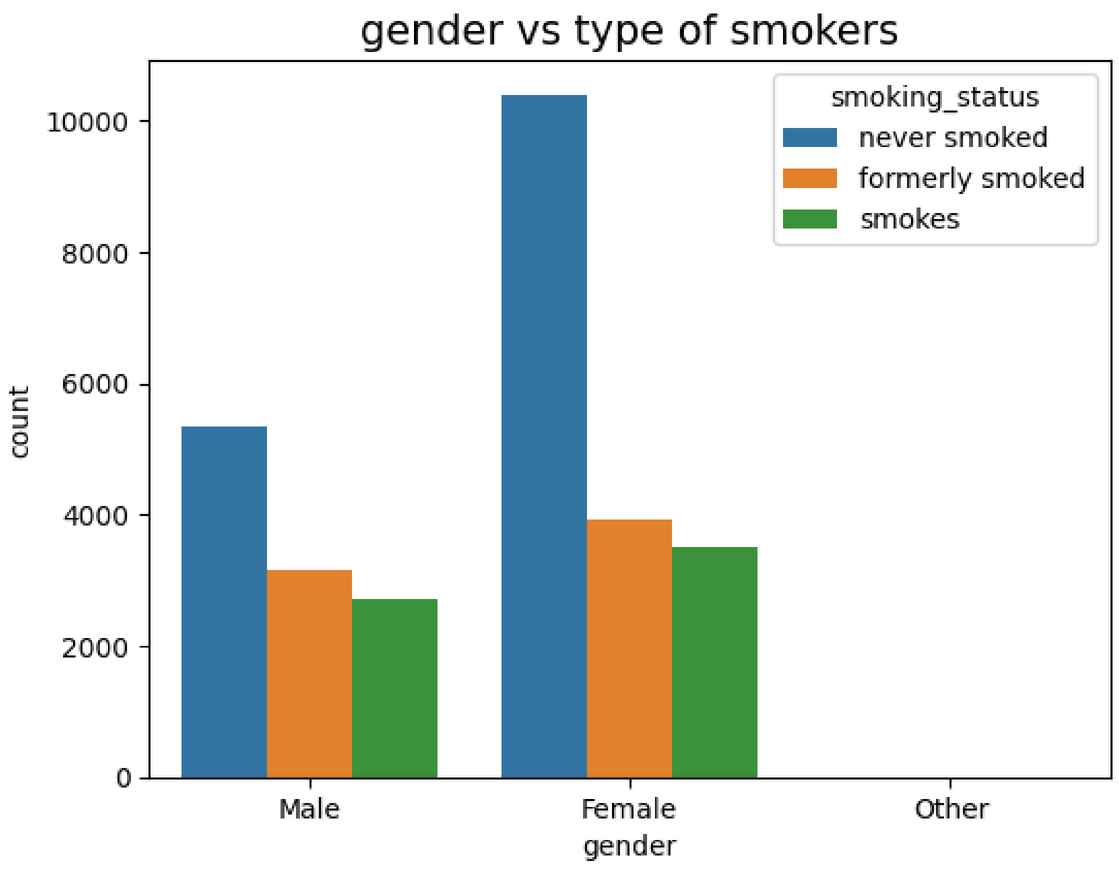
```
never smoked       15747
formerly smoked     7099
smokes              6226
Name: smoking_status, dtype: int64
```

```
train_data.groupby(["smoking_status"])["stroke"].value_counts()
```

```
smoking_status   stroke
formerly smoked  0         6919
                 1          180
never smoked     0        15491
                 1          256
```

```
smokes            0          6114
                  1           112
Name: stroke, dtype: int64
```

```
sns.countplot(x=train_data["gender"],
hue=train_data["smoking_status"])
plt.title("gender vs type of smokers", fontsize=15)
plt.show()
```



Encoding data

```
str_data=train_data.select_dtypes(include=['object'])
str_dt=test_data.select_dtypes(include=['object'])

int_data=train_data.select_dtypes(include=['integer', 'float'])
int_dt=test_data.select_dtypes(include=['integer', 'float'])

from sklearn.preprocessing import LabelEncoder

label=LabelEncoder()
features=str_data.apply(label.fit_transform)
features=features.join(int_data)
features.head()
```

```
   gender  ever_married  work_type  Residence_type  smoking_status
id  \
1       1             1          2               1               1
30468
3       0             1          2               0               0
56543
6       0             1          2               1               0
52800
7       0             1          3               0               1
41413
8       0             1          2               0               2
15266

     age  hypertension  heart_disease  avg_glucose_level   bmi  stroke

1   58.0             1              0              87.96  39.2       0

3   70.0             0              0              69.04  35.9       0

6   52.0             0              0              77.59  17.7       0

7   75.0             0              1             243.53  27.0       0

8   32.0             0              0              77.67  32.3       0

test1=str_dt.apply(label.fit_transform)
Test=test1.join(int_dt)
Test.head()

   gender  ever_married  work_type  Residence_type  smoking_status
id  \
0       1             1          2               1               0
36306
1       0             1          3               0               0
61829
4       0             1          0               0               1
40801
5       0             1          2               1               1
9348
7       1             1          0               1               1
60512

     age  hypertension  heart_disease  avg_glucose_level   bmi
0   80.0             0              0              83.84  21.1
1   74.0             0              1             179.50  26.0
4   63.0             0              0              83.57  27.6
5   66.0             1              0             219.98  32.2
7   46.0             0              0             120.80  32.5
```

Modeling & predicting the **data**

```
xtrain=features.drop(["stroke"],axis=1)
xtrain.shape
```

```
(29072, 11)
```

```
ytrain=features["stroke"]
ytrain.head()
ytrain.shape
```

```
(29072,)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(xtrain, ytrain)
```

```
x_test.shape
```

```
(7268, 11)
```

```
y_test.shape
```

```
(7268,)
```

```
x_train.head()
```

```
       gender  ever_married  work_type  Residence_type  smoking_status
id  \
14628       0             0          2               0               1
33280
17895       1             1          3               1               2
70939
10614       1             1          2               0               0
48669
34892       1             1          2               1               2
51685
28149       0             1          3               0               0
40489

       age  hypertension  heart_disease  avg_glucose_level   bmi
14628  37.0             0              0              93.80  24.6
17895  61.0             0              0              75.28  19.6
10614  71.0             0              0              66.58  27.5
34892  44.0             0              0             228.40  36.1
28149  59.0             0              0              82.92  34.9
```

```
y_train.head()
```

```
14628    0
17895    0
10614    0
34892    0
```

```
28149    0
Name: stroke, dtype: int64
```

Naive Bayes

```
x_test.head()
```

```
       gender  ever_married  work_type  Residence_type  smoking_status
id  \
805         1             0          2               0               1
58037
31952       1             0          2               1               0
62918
5083        1             1          3               0               0
3820
12863       0             1          2               0               0
14684
21818       1             1          2               0               1
63056

        age  hypertension  heart_disease  avg_glucose_level   bmi
805    21.0             0              0              78.52  27.2
31952  40.0             0              0              63.29  29.7
5083   77.0             0              0              88.75  31.9
12863  65.0             0              0             244.95  36.1
21818  78.0             1              0              90.04  25.6
```

```
y_test.head()
```

```
805      0
31952    0
5083     0
12863    0
21818    0
Name: stroke, dtype: int64
```

```
from sklearn.naive_bayes import GaussianNB
```

```
model=GaussianNB()
model.fit(x_train, y_train)
```

```
GaussianNB()
```

```
predict=model.predict(x_test)
predict
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
test_score=model.score(x_test, y_test)
print("NBtest_score:", test_score)
```

```
NBtest_score: 0.9744083654375344
```

```
nb_conf_mtr=pd.crosstab(y_test, predict)
nb_conf_mtr
```

```
col_0      0   1
stroke
0       7077  61
1        125   5
```

```
from sklearn.metrics import classification_report
```

```
nbreport=classification_report(y_test, predict)
print(nbreport)
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      7138
           1       0.08      0.04      0.05       130

    accuracy                           0.97      7268
   macro avg       0.53      0.51      0.52      7268
weighted avg       0.97      0.97      0.97      7268
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt_mod=DecisionTreeClassifier(max_depth=8)
dt_mod.fit(x_train, y_train)
```

```
DecisionTreeClassifier(max_depth=8)
```

```
y_predict=dt_mod.predict(x_test)
y_predict
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
ts_dt_score=dt_mod.score(x_test, y_test)
print("Decision tree test score:", ts_dt_score)
```

```
Decision tree test score: 0.9794991744634012
```

```
dectree_report=classification_report(y_test, y_predict)
print(dectree_report)
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      7138
           1       0.12      0.02      0.04       130
```

```
    accuracy                           0.98      7268
   macro avg       0.55      0.51      0.51      7268
weighted avg       0.97      0.98      0.97      7268


dt_conf_mtr=pd.crosstab(y_test, y_predict)
dt_conf_mtr

col_0      0   1
stroke
0       7116  22
1        127   3
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators= 100)

rfc.fit(x_train,y_train)

RandomForestClassifier()

y_pred_rfc = rfc.predict(x_test)

print(classification_report(y_test,y_pred_rfc))

              precision    recall  f1-score   support

           0       0.98      1.00      0.99      7138
           1       0.00      0.00      0.00       130

    accuracy                           0.98      7268
   macro avg       0.49      0.50      0.50      7268
weighted avg       0.96      0.98      0.97      7268


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1471: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1471: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1471: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
frst_cnf= pd.crosstab(y_test,y_pred_rfc)
print(frst_cnf)
```

```
col_0      0
stroke
0       7138
1        130
```

Multi-Layer Perceptron Classifier

```
from sklearn.neural_network import MLPClassifier

mlp=MLPClassifier()

mlp.fit(x_train,y_train)

y_pred_mlp = mlp.predict(x_test)

mlp.score(x_test,y_test)
```

```
0.9821133736929004
```

```
from sklearn.model_selection import cross_val_score
cross_val_score(model,xtrain,ytrain,cv = 20,
scoring='accuracy').mean()
```

```
0.9757842475511938
```

```
cross_val_score(dt_mod,xtrain,ytrain,cv = 20,
scoring='accuracy').mean()
```

```
0.9790521152934069
```

```
cross_val_score(rfc,xtrain,ytrain,cv = 20, scoring='accuracy').mean()
```

```
0.9811503212534707
```

```
cross_val_score(mlp,xtrain,ytrain,cv = 20, scoring='accuracy').mean()
```

```
0.9392994241388353
```

Applying PCA

```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
principalComponents = pca.fit_transform(xtrain)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(xtrain, ytrain)

model_2=GaussianNB()
model_2.fit(x_train, y_train)
```

```
GaussianNB()

test_score=model_2.score(x_test, y_test)
print("NBtest_score:", test_score)

NBtest_score: 0.9753714914694551

dt_mod=DecisionTreeClassifier()
dt_mod.fit(x_train, y_train)

DecisionTreeClassifier()

ts_dt_score=dt_mod.score(x_test, y_test)
print("Decision tree test score:", ts_dt_score)

Decision tree test score: 0.9595487066593286

rfc.fit(x_train,y_train)

RandomForestClassifier()

y_pred_rfc = rfc.predict(x_test)

print(pd.crosstab(y_test,y_pred_rfc))
print(classification_report(y_test,y_pred_rfc))

col_0      0
stroke
0       7132
1        136
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      7132
           1       0.00      0.00      0.00       136

    accuracy                           0.98      7268
   macro avg       0.49      0.50      0.50      7268
weighted avg       0.96      0.98      0.97      7268


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1471: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1471: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1471: UndefinedMetricWarning: Precision and F-score are ill-
```

```
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
rfc.score(x_test,y_test)
```

```
0.9812878370941112
```

```
mlp=MLPClassifier()
```

```
mlp.fit(x_train,y_train)
```

```
y_pred_mlp = mlp.predict(x_test)
```

```
mlp.score(x_test,y_test)
```

```
0.9812878370941112
```

```
cross_val_score(model_2,xtrain,ytrain,cv = 20,
scoring='accuracy').mean()
```

```
0.9757842475511938
```

```
cross_val_score(dt_mod,xtrain,ytrain,cv = 20,
scoring='accuracy').mean()
```

```
0.9581380978121441
```

```
cross_val_score(rfc,xtrain,ytrain,cv = 20, scoring='accuracy').mean()
```

```
0.9811503212534707
```

```
cross_val_score(mlp,xtrain,ytrain,cv = 20, scoring='accuracy').mean()
```

```
0.9810127696716275
```