NAME: HRISHIKESH SHIVPUTRA KAMBLE

PROJECT:-

SOLVING CLASSIFICATION PREDICTION FOR "BRAIN STROKE" DATASET USING "LOGISTIC REGRESSION, NAIVES BAYES CLASSIFICATION,SUPPORT VECTOR CLASSIFIER,K NEAREST NEIGHBOUR, DESICION TREE CLASSIFIER".

DATA:-

1. GENDER: "MALE", "FEMALE" OR "OTHER"
2. AGE: AGE OF THE PATIENT
3. HYPERTENSION: 0 IF THE PATIENT DOESN'T HAVE HYPERTENSION, 1 IF THE PATIENT HAS HYPERTENSION
4. HEART DISEASE: 0 IF THE PATIENT DOESN'T HAVE ANY HEART DISEASES, 1 IF THE PATIENT HAS A HEART DISEASE
5. EVER-MARRIED: "NO" OR "YES"
6. WORK TYPE: "CHILDREN", "GOVTJOV", "NEVER WORKED", "PRIVATE" OR "SELF-EMPLOYED"
7. RESIDENCETYPE: "RURAL" OR "URBAN"
8. AVG GLUCOSE LEVEL: AVERAGE GLUCOSE LEVEL IN BLOOD
9. BMI: BODY MASS INDEX
10. SMOKING_STATUS: "FORMERLY SMOKED", "NEVER SMOKED", "SMOKES" OR "UNKNOWN"
11. STROKE: 1 IF THE PATIENT HAD A STROKE OR 0 IF NOT

APPROACH:

1.LOAD THE REQUIRED LIBRARIES SUCH AS PANDAS,MATPLOTLIB,SEABORN ALONG WITH GIVEN DATASET.

2.PERFORM EDA ON THE GIVEN DATASET.

3.CONVERT ALL THE REQUIRED COLUMNS INTO NUMERIAL COLUMNS USING GET DUMMIES FUNCTION FROM PANDAS LIBRARY.

4.CONVERTING ALL REQUIRED FEATURES IN NUMERIAL , CHECK FOR CORRELATION BETWEEN FEATURES AND TARGET AND CONSIDER THE ONLY FEATURES WITH CORRELATION HIGHER THAN 30.

5.IMPORT "LOGISTIC REGRESSION, NAIVES BAYES CLASSIFICATION,SUPPORT VECTOR CLASSIFIER,K NEAREST NEIGHBOUR", AND SPLIT THE GIVEN DATASET INTO TRAINING AND TESTING DATA USING TRAIN_TEST_SPLIT FUNCTION.THEN CALUCLATE ACCURACY SCORE USING SKLEARN LIBRARY BY IMPORTING METRICS.

6.ONCE WE GET ACCURACY SCORE OF ALL MODELS FOR BOTH TRAING AND TESTING DATA, CREATE A DATAFRAME AND LOAD ALL THE ACCURACY OF ALL MODEL.

7.VISUALIZATION: ONCE THE DATASET IS CREATED PLOT THE ACCURACIES OF ALL THE MODELS USING BARPLOT USING

In [1]:

```
import pandas as pd                             # LOADING ALL THE REQUIRED LIBRARIES.
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

```
C:\Users\Hrishikesh\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

In [2]:

```
data=pd.read_csv(r"C:\Users\Hrishikesh\Desktop\DATA SCIENCE\brain_stroke.csv")   # LOADING THE GIVEN DATASET
```

In [3]:
```
data
```

Out[3]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4976	Male	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	Male	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	Female	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	Male	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0
4980	Female	80.0	1	0	Yes	Private	Urban	83.75	29.1	never smoked	0

4981 rows × 11 columns

In [4]:
```
data.columns
```

Out[4]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

In [5]:
```
data
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4976	Male	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	Male	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	Female	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	Male	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0
4980	Female	80.0	1	0	Yes	Private	Urban	83.75	29.1	never smoked	0

In [6]:
```
data.isna().sum()     # CHECK NULL VALUES
```

Out[6]:
```
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

In [7]:
```
data.info()                   # SHOWS ALL INFORMATION REGARDING THE DATA SUCH AS NULL VALUE, COLUMNS,DATATYPES
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4981 entries, 0 to 4980
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             4981 non-null   object 
 1   age                4981 non-null   float64
 2   hypertension       4981 non-null   int64  
 3   heart_disease      4981 non-null   int64  
 4   ever_married       4981 non-null   object 
 5   work_type          4981 non-null   object 
 6   Residence_type     4981 non-null   object 
 7   avg_glucose_level  4981 non-null   float64
 8   bmi                4981 non-null   float64
 9   smoking_status     4981 non-null   object 
 10  stroke             4981 non-null   int64  
dtypes: float64(3), int64(3), object(5)
memory usage: 428.2+ KB
```

In [8]:
```
data.describe()            # SHOWS THE ALL DETAILS REGARDING ALL NUMERICAL COLUMNS
```

Out[8]:

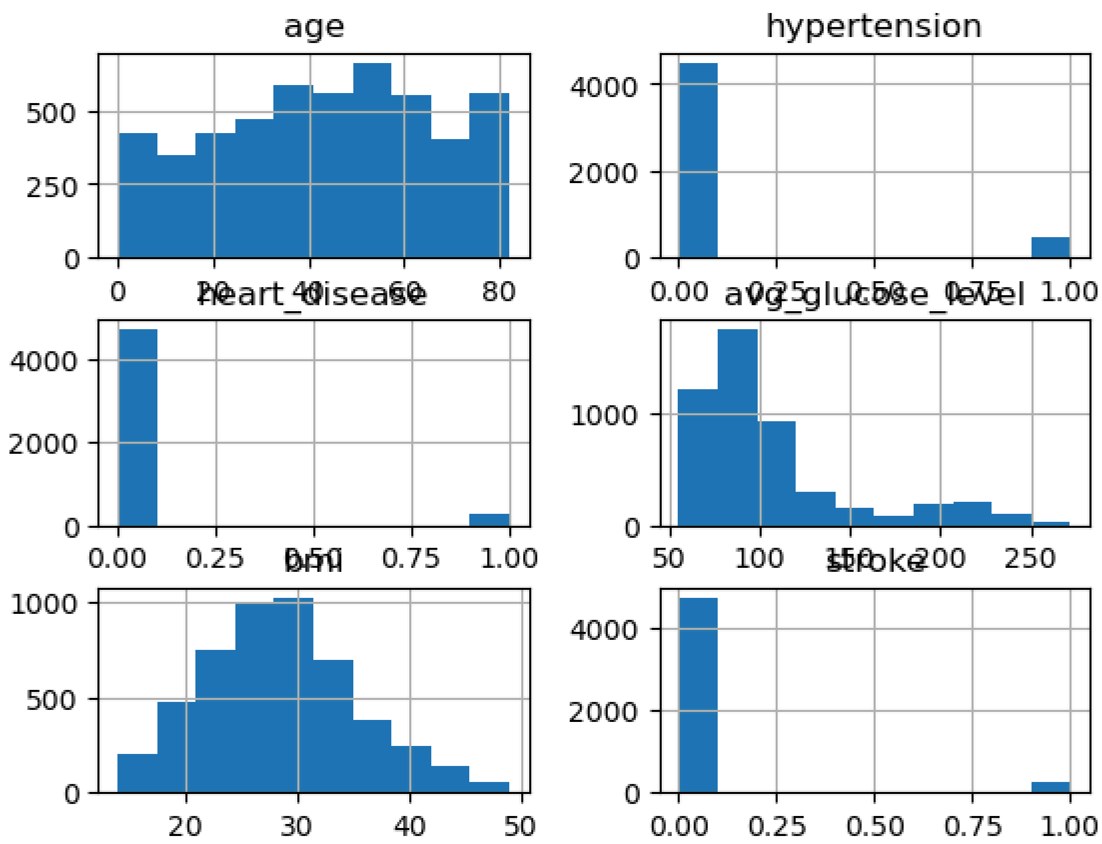
	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	4981.000000	4981.000000	4981.000000	4981.000000	4981.000000	4981.000000
mean	43.419859	0.096165	0.055210	105.943562	28.498173	0.049789
std	22.662755	0.294848	0.228412	45.075373	6.790464	0.217531
min	0.080000	0.000000	0.000000	55.120000	14.000000	0.000000
25%	25.000000	0.000000	0.000000	77.230000	23.700000	0.000000
50%	45.000000	0.000000	0.000000	91.850000	28.100000	0.000000
75%	61.000000	0.000000	0.000000	113.860000	32.600000	0.000000
max	82.000000	1.000000	1.000000	271.740000	48.900000	1.000000

In [9]:
```
data.shape                          # SHOWS THE NUMBER OF ROWS AND COLUMNS
```

Out[9]:
```
(4981, 11)
```

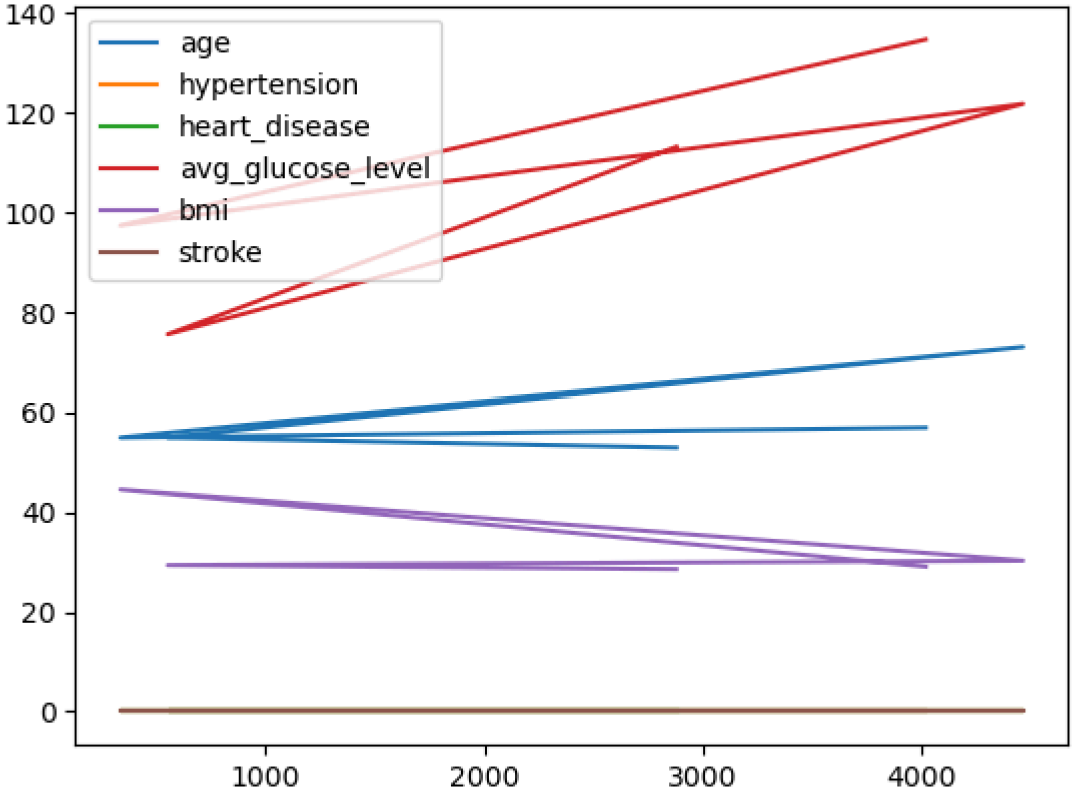
In [10]:
```
plt.figure(figsize=(16,7))        # PLOT HISTPLOT TO SEE DATA DISTRIBUTION
data.hist()
plt.show()
```

```
<Figure size 1600x700 with 0 Axes>
```



In [11]:
```
data.sample(5).plot()                    # PLOT SAMPLE DATA
```

Out[11]: <Axes: >



In [12]:
```
data
```

Out[12]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4976	Male	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	Male	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	Female	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	Male	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0
4980	Female	80.0	1	0	Yes	Private	Urban	83.75	29.1	never smoked	0

4981 rows × 11 columns

In [ ]:

In [13]:

```
x=pd.get_dummies(data[["smoking_status","gender","ever_married","work_type","Residence_type"]],drop_first=["smoking_st
x          # CONVERT CATEGORICAL COLUMNS INTO NUMERICAL USING DUMMIES
```

Out[13]:

	smoking_status_formerly smoked	smoking_status_never smoked	smoking_status_smokes	gender_Male	ever_married_Yes	work_type_Private	work_type_Self employec
0	1	0	0	1	1	1	(
1	0	1	0	1	1	1	(
2	0	0	1	0	1	1	(
3	0	1	0	0	1	0	1
4	1	0	0	1	1	1	(
...	...	...	...	...	...	...	..
4976	1	0	0	1	0	1	(
4977	0	0	1	1	1	1	(
4978	0	0	1	0	1	0	(
4979	0	0	1	1	1	1	(
4980	0	1	0	0	1	1	(

4981 rows × 9 columns

In [14]:
```
data
```

Out[14]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
2	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
4	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...
4976	Male	41.0	0	0	No	Private	Rural	70.15	29.8	formerly smoked	0
4977	Male	40.0	0	0	Yes	Private	Urban	191.15	31.1	smokes	0
4978	Female	45.0	1	0	Yes	Govt_job	Rural	95.02	31.8	smokes	0
4979	Male	40.0	0	0	Yes	Private	Rural	83.94	30.0	smokes	0
4980	Female	80.0	1	0	Yes	Private	Urban	83.75	29.1	never smoked	0

4981 rows × 11 columns

In [15]:
```
data=pd.concat([data,x],axis=1).drop(columns=["smoking_status","ever_married","work_type","Residence_type","gender"])
data                      # CONCATE DUMMIES DATA WITH ORIGINAL DATASET
```

Out[15]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke	smoking_status_formerly smoked	smoking_status_never smoked	smoking_status_smokes
0	67.0	0	1	228.69	36.6	1	1	0	0
1	80.0	0	1	105.92	32.5	1	0	1	0
2	49.0	0	0	171.23	34.4	1	0	0	1
3	79.0	1	0	174.12	24.0	1	0	1	0
4	81.0	0	0	186.21	29.0	1	1	0	0
...	...	...	...	...	...	...	...	...	...
4976	41.0	0	0	70.15	29.8	0	1	0	0
4977	40.0	0	0	191.15	31.1	0	0	0	1
4978	45.0	1	0	95.02	31.8	0	0	0	1
4979	40.0	0	0	83.94	30.0	0	0	0	1
4980	80.0	1	0	83.75	29.1	0	0	1	0

4981 rows × 15 columns

In [16]:
```
data=data[['age', 'hypertension', 'heart_disease', 'avg_glucose_level', 'bmi', 'smoking_status_formerly smoked',
       'smoking_status_never smoked', 'smoking_status_smokes', 'gender_Male',
       'ever_married_Yes', 'work_type_Private', 'work_type_Self-employed',
       'work_type_children', 'Residence_type_Urban',
       'stroke']]
```

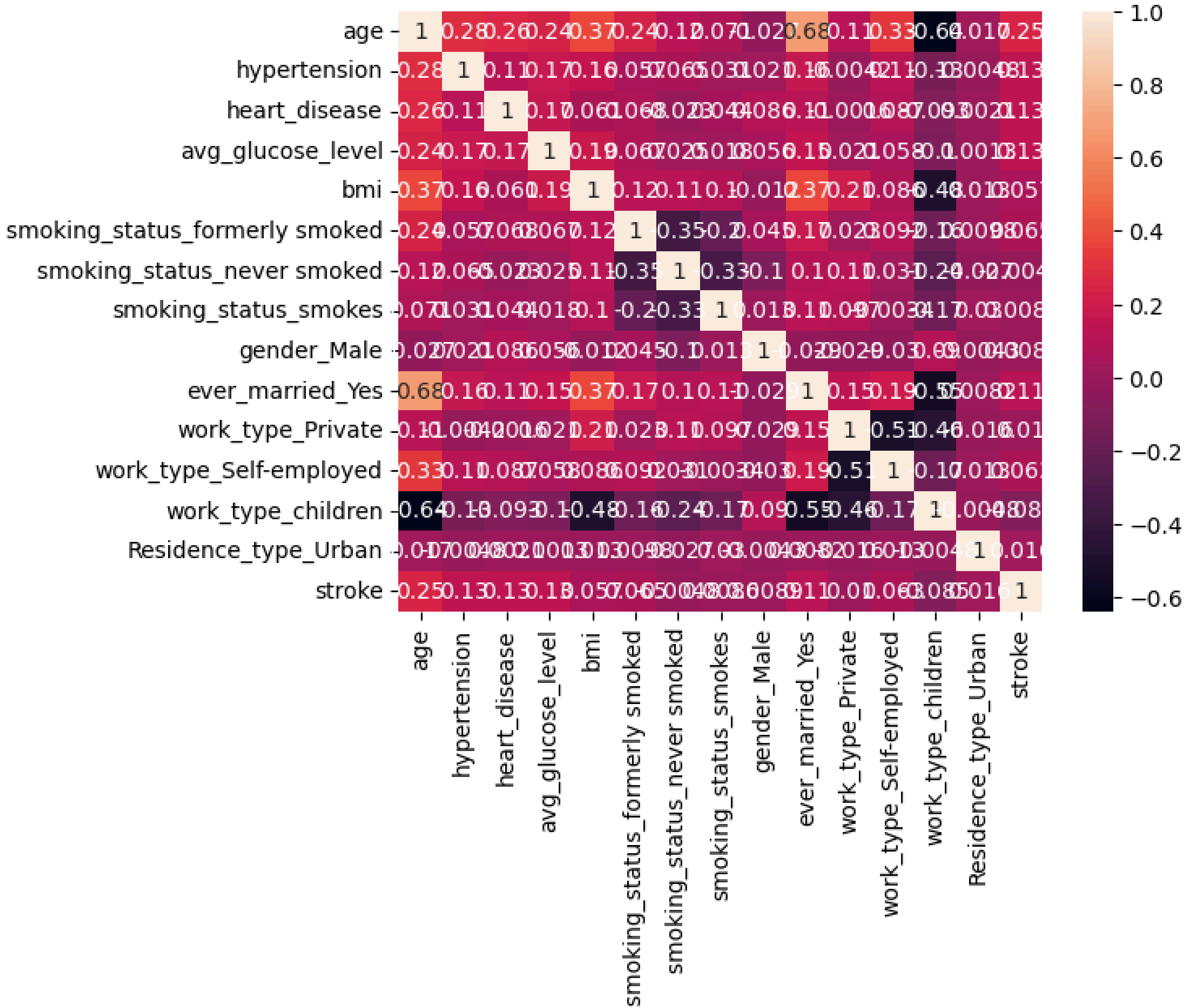
In [17]:
```
data.corr()*100
```

Out[17]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	smoking_status_formerly smoked	smoking_status_never smoked
age	100.000000	27.811956	26.485169	23.676268	37.370310	23.550811	12.261730
hypertension	27.811956	100.000000	11.197364	17.002767	15.876244	5.679747	6.526698
heart_disease	26.485169	11.197364	100.000000	16.684657	6.092647	6.754129	-2.272694
avg_glucose_level	23.676268	17.002767	16.684657	100.000000	18.634817	6.698903	2.472661
bmi	37.370310	15.876244	6.092647	18.634817	100.000000	12.015566	10.932215
smoking_status_formerly smoked	23.550811	5.679747	6.754129	6.698903	12.015566	100.000000	-35.105727
smoking_status_never smoked	12.261730	6.526698	-2.272694	2.472661	10.932215	-35.105727	100.000000
smoking_status_smokes	7.089943	3.074894	4.401079	1.787274	10.071028	-19.720833	-32.850987
gender_Male	-2.653843	2.148476	8.647553	5.579594	-1.209292	4.510887	-10.238666
ever_married_Yes	67.713657	16.453409	11.476489	15.072374	37.169006	17.203936	10.411985
work_type_Private	11.102048	-0.417718	-0.160001	2.076356	21.182000	2.268483	10.993599
work_type_Self-employed	32.683483	11.046797	8.747408	5.841942	8.558153	9.218600	3.089779
work_type_children	-63.686595	-12.892427	-9.297401	-10.196024	-48.425704	-16.130989	-23.652935
Residence_type_Urban	1.715450	-0.475503	0.212545	0.134561	1.318494	0.982495	-2.689246
stroke	24.647787	13.196524	13.461031	13.322733	5.692566	6.532000	-0.480609

In [18]:
```
sns.heatmap(data.corr(),annot=True)
```

Out[18]:
```
<Axes: >
```


age
hypertension
heart_disease
avg_glucose_level
bmi
smoking_status_formerly smoked
smoking_status_never smoked
smoking_status_smokes
gender_Male
ever_married_Yes
work_type_Private
work_type_Self-employed
work_type_children
Residence_type_Urban
stroke
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

In [19]:
```
data
```

Out[19]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	smoking_status_formerly smoked	smoking_status_never smoked	smoking_status_smokes	gender
0	67.0	0	1	228.69	36.6	1	0	0	
1	80.0	0	1	105.92	32.5	0	1	0	
2	49.0	0	0	171.23	34.4	0	0	1	
3	79.0	1	0	174.12	24.0	0	1	0	
4	81.0	0	0	186.21	29.0	1	0	0	
...	...	...	...	...	...	...	...	...	
4976	41.0	0	0	70.15	29.8	1	0	0	
4977	40.0	0	0	191.15	31.1	0	0	1	
4978	45.0	1	0	95.02	31.8	0	0	1	
4979	40.0	0	0	83.94	30.0	0	0	1	
4980	80.0	1	0	83.75	29.1	0	1	0	

4981 rows × 15 columns

In [20]:
```
data.columns
```

Out[20]:
```
Index(['age', 'hypertension', 'heart_disease', 'avg_glucose_level', 'bmi',
       'smoking_status_formerly smoked', 'smoking_status_never smoked',
       'smoking_status_smokes', 'gender_Male', 'ever_married_Yes',
       'work_type_Private', 'work_type_Self-employed', 'work_type_children',
       'Residence_type_Urban', 'stroke'],
      dtype='object')
```

In [21]:
```
F=data[[ 'age', 'hypertension', 'heart_disease', 'avg_glucose_level', 'bmi',
       'smoking_status_formerly smoked', 'smoking_status_never smoked',
       'smoking_status_smokes', 'gender_Male', 'ever_married_Yes',
       'work_type_Private', 'work_type_Self-employed', 'work_type_children',
       'Residence_type_Urban']]
T=data["stroke"]                                                    # STORE DATA INTO FEATURES AND TARGET ACCORDINGLY
```

In [22]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(F,T)
                                                                   # SPLIT THE DATASET INTO TRAIN AND TESTING DATA
```

In [23]:
```
from sklearn.preprocessing import MinMaxScaler
M=MinMaxScaler()                                                # IMPORT STANDARD SCALER FOR STANDARDIZATION
```

In [24]:
```
x_train
```

Out[24]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	smoking_status_formerly smoked	smoking_status_never smoked	smoking_status_smokes	gender
710	2.0	0	0	93.88	17.4	0	0	0	
54	76.0	0	0	104.47	20.3	0	0	0	
523	72.0	0	0	215.64	26.7	1	0	0	
3967	12.0	0	0	116.06	25.9	0	0	0	
903	61.0	1	1	148.24	32.2	1	0	0	
...	...	...	...	...	...	...	...	...	
1383	38.0	0	0	86.86	36.5	0	0	0	
1063	45.0	0	0	55.67	23.1	0	0	1	
1605	39.0	0	0	92.32	43.0	0	1	0	
957	19.0	0	0	75.08	21.7	0	0	0	
4207	2.0	0	0	112.75	25.1	0	0	0	

3735 rows × 14 columns

In [25]:
```
x_train[['age','avg_glucose_level','bmi']]=M.fit_transform(x_train[['age','avg_glucose_level','bmi']])
x_test[['age','avg_glucose_level','bmi']]=M.transform(x_test[['age','avg_glucose_level','bmi']])
    # FIT THE DATA INTO MODEL AND DO THE STANDARDIZATION
```

In [26]:
```
x_train
```

Out[26]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	smoking_status_formerly smoked	smoking_status_never smoked	smoking_status_smokes
710	0.023438	0	0	0.178514	0.097421	0	0	0
54	0.926758	0	0	0.227426	0.180516	0	0	0
523	0.877930	0	0	0.740890	0.363897	1	0	0
3967	0.145508	0	0	0.280957	0.340974	0	0	0
903	0.743652	1	1	0.429588	0.521490	1	0	0
...	...	...	...	...	...	...	...	..
1383	0.462891	0	0	0.146090	0.644699	0	0	0
1063	0.548340	0	0	0.002032	0.260745	0	0	1
1605	0.475098	0	0	0.171308	0.830946	0	1	0
957	0.230957	0	0	0.091682	0.220630	0	0	0
4207	0.023438	0	0	0.265669	0.318052	0	0	0

3735 rows × 14 columns

In [27]:
```
x_test
```

Out[27]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	smoking_status_formerly smoked	smoking_status_never smoked	smoking_status_smokes
739	0.499512	0	0	0.239019	0.389685	0	1	(
4024	0.792480	0	0	0.574754	0.478510	1	0	(
3496	0.340820	0	0	0.242529	0.702006	0	1	(
33	0.597168	0	0	0.026234	0.455587	0	1	(
785	0.523926	0	0	0.371900	0.627507	0	0	1
...	...	...	...	...	...	...	...	..
1811	0.060059	0	0	0.250196	0.171920	0	0	(
4653	0.218750	0	0	0.215787	0.916905	1	0	(
1552	0.389648	0	0	0.118101	0.882521	0	1	(
456	0.267578	0	0	0.308854	0.275072	0	0	1
4842	0.035645	0	0	0.112697	0.140401	0	0	(

1246 rows × 14 columns

LOGISTIC REGRESSION:-

In [28]:
```
from sklearn.linear_model import LogisticRegression
L=LogisticRegression()                                   #IMPORT LOGISTIC REGRESSION AND FIT
L.fit(x_train,y_train)
```

Out[28]:

```
▾ LogisticRegression
LogisticRegression()
```

In [29]:
```
L1=L.score(x_train,y_train)*100
L1                                #TRAINING ACCURACY
```

Out[29]: 94.99330655957162

In [30]:
```
L2=L.score(x_test,y_test)*100
L2                                #TESTING ACCURACY
```

Out[30]: 95.18459069020867

SVC:-

In [31]:
```
from sklearn.svm import SVC
S=SVC()                                   #IMPORT SVC AND FIT
S.fit(x_train,y_train)
```

Out[31]:
```
▾ SVC
SVC()
```

In [32]:
```
S1=S.score(x_train,y_train)*100
S1                                #TRAINING ACCURACY
```

Out[32]: 94.99330655957162

In [33]:
```
S2=S.score(x_test,y_test)*100
S2                                #TESTING ACCURACY
```

Out[33]: 95.10433386837882

NAIVES BAYES:-

In [34]:
```
from sklearn.naive_bayes import GaussianNB,ComplementNB,MultinomialNB,BernoulliNB
G=GaussianNB()
C=ComplementNB()
M=MultinomialNB()
B=BernoulliNB()                                       #IMPORT NAIVES BAYES AND FIT
```

GaussianNB:

In [35]:
```
G.fit(x_train,y_train)
```

Out[35]:
```
▾ GaussianNB
GaussianNB()
```

In [36]:
```
G1=G.score(x_train,y_train)*100
G1                              #TRAINING ACCURACY
```

Out[36]: 82.65060240963855

In [37]:
```
G2=G.score(x_test,y_test)*100    #TESTING ACCURACY
G2
```

Out[37]: 84.02889245585875

BernoulliNB:

In [38]:
```
B.fit(x_train,y_train)
```

Out[38]:
```
▾ BernoulliNB
BernoulliNB()
```

In [39]:
```
B1=B.score(x_train,y_train)*100     #TRAINING ACCURACY
B1
```

Out[39]: 94.56492637215528

In [40]:
```
B2=B.score(x_test,y_test)*100       #TESTING ACCURACY
B2
```

Out[40]: 95.02407704654897

In [ ]:
```

```

ComplementNB:-

In [41]:
```
C.fit(x_train,y_train)
```

Out[41]:
```
▾ ComplementNB
ComplementNB()
```

In [42]:
```
C1=C.score(x_train,y_train)*100
C1
```

Out[42]: 67.81793842034806

In [43]:
```
C2=C.score(x_test,y_test)*100
C2
```

Out[43]: 70.14446227929373

In [ ]:

MultinomialNB:-

In [44]:
```
M.fit(x_train,y_train)
```

Out[44]:
```
▾ MultinomialNB
MultinomialNB()
```

In [45]:
```
M1=M.score(x_train,y_train)*100
M1
```

Out[45]: 94.99330655957162

In [46]:
```
M2=M.score(x_train,y_train)*100
M2
```

Out[46]: 94.99330655957162

K NEAREST NEIGHBOUR:-

In [47]:
```
from sklearn.neighbors import KNeighborsClassifier
K=KNeighborsClassifier()                                  #IMPORT K NEAREST NEIGHBOUR AND FIT
```

In [48]:
```
K.fit(x_train,y_train)
```

Out[48]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [49]:
```
K1=K.score(x_train,y_train)*100     #TRAINING ACCURACY
K1
```

Out[49]: 95.20749665327979

In [50]:
```
K2=K.score(x_test,y_test)*100
K2                                 #TESTING ACCURACY
```

Out[50]: 95.02407704654897

In [ ]:

DECISION TREE CLASSIFIER:-

In [51]:
```
from sklearn.tree import DecisionTreeClassifier
D=DecisionTreeClassifier()                        #IMPORT DECISION TREE CLASSIFIER AND FIT
```

In [52]:
```
D.fit(x_train,y_train)
```

Out[52]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [53]:
```
DT1=D.score(x_train,y_train)*100        #TRAINING ACCURACY
DT1
```

Out[53]: 100.0

In [54]:
```
DT2=D.score(x_test,y_test)*100
DT2                                        #TESTING ACCURACY
```

Out[54]: 91.01123595505618

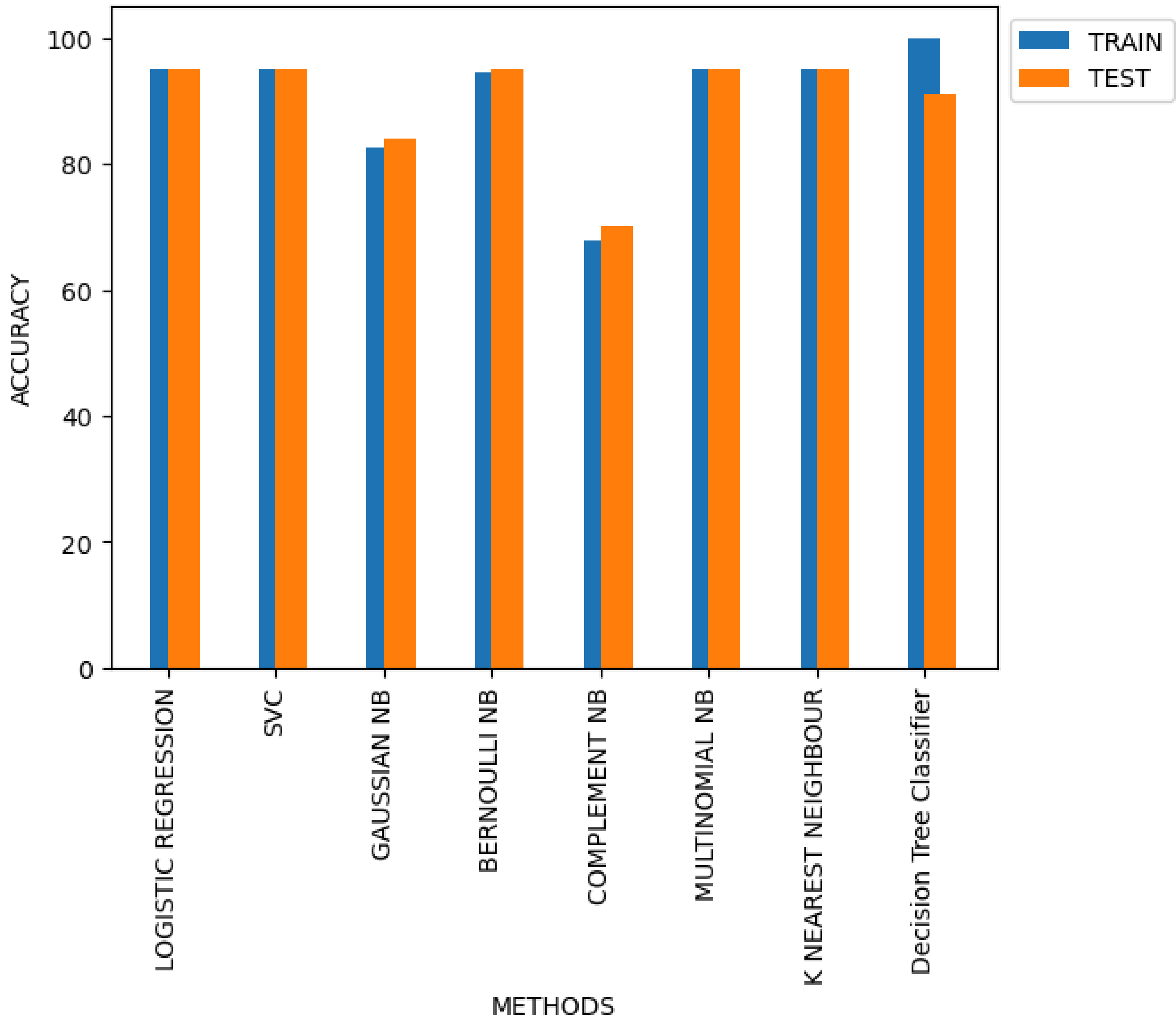
ACCURACY GRAPH:-

In [55]:
```
A={"METHODS":["LOGISTIC REGRESSION","SVC","GAUSSIAN NB","BERNOULLI NB","COMPLEMENT NB","MULTINOMIAL NB","K NEAREST NEI
A=pd.DataFrame(A)
A=np.around(A,2)
A
```

Out[55]:

	METHODS	TRAIN ACCURACY	TEST ACCURACY
0	LOGISTIC REGRESSION	94.99	95.18
1	SVC	94.99	95.10
2	GAUSSIAN NB	82.65	84.03
3	BERNOULLI NB	94.56	95.02
4	COMPLEMENT NB	67.82	70.14
5	MULTINOMIAL NB	94.99	94.99
6	K NEAREST NEIGHBOUR	95.21	95.02
7	Decision Tree Classifier	100.00	91.01

In [56]:
```
plt.bar(A["METHODS"],A["TRAIN ACCURACY"],width=0.3,label="TRAIN")
plt.bar(A["METHODS"],A["TEST ACCURACY"],align="edge",width=0.3,label="TEST")
plt.legend(bbox_to_anchor=[1,0,0,1])
plt.xlabel("METHODS")
plt.ylabel("ACCURACY")
plt.xticks(rotation=90)
plt.show()                   # PLOT THE ACCURACY CHART BETWEEN ALL MODELS ACCURACIES
```


100
80
60
40
20
0
ACCURACY
TRAIN
TEST
LOGISTIC REGRESSION
SVC
GAUSSIAN NB
BERNOULLI NB
COMPLEMENT NB
MULTINOMIAL NB
K NEAREST NEIGHBOUR
Decision Tree Classifier
METHODS

CONCLUSION:

FROM THE ABOVE BAR CHART IT IS CLEAR THAT LOGISTIC REGRESSION ,BERNOULLINB ,MULTINOMIAL NB, SVC,K NEAREST NEIGHBOUR ARE BEST FOR CLASSIFICATION FOR THIS DATASET .

In [ ]:

```

```

In [ ]:

```

```