stroke-prediction

[2]:
```
import numpy as np
import pandas as pd
df=pd.read_csv('/content/healthcare-dataset-stroke-data.csv')
df
```

[2]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
…	…	…	…	…	…	…	…	…	…	…	…	…
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	1
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	1
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	1
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	1
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	1

```
…      …
5105       0
5106       0
5107       0
5108       0
5109       0

[5110 rows x 12 columns]
```

[3]:
```
df.head()
```

[3]:
```
      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   smoking_status  \
0        Private          Urban             228.69  36.6  formerly smoked
1  Self-employed          Rural             202.21   NaN     never smoked
2        Private          Rural             105.92  32.5     never smoked
3        Private          Urban             171.23  34.4           smokes
4  Self-employed          Rural             174.12  24.0     never smoked

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

[4]:
```
df.tail()
```

[4]:
```
         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
```

[5]:
```
df.columns
```

[5]:
```
Index(['id', 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

[6]:
```
df.shape
```

[6]:
```
(5110, 12)
```

[7]:
```
#checking for missing values in columns

df.isna().sum()
```

[7]:
```
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
```

[8]:
```
#Filling the missing values

df['bmi']=df['bmi'].fillna(df['bmi'].mean())
df.isna().sum()
```

[8]:
```
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                  0
smoking_status       0
stroke               0
dtype: int64
```

[9]:
```
df.dtypes
```

[9]:
```
id                   int64
gender              object
age                float64
hypertension         int64
heart_disease        int64
ever_married        object
work_type           object
Residence_type      object
avg_glucose_level  float64
bmi                float64
smoking_status      object
stroke               int64
dtype: object
```

[10]:
```
#Encoding the object columns

from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
df['gender']=lb.fit_transform(df['gender'])
df['ever_married']=lb.fit_transform(df['ever_married'])
df['work_type']=lb.fit_transform(df['work_type'])
df['Residence_type']=lb.fit_transform(df['Residence_type'])
df['smoking_status']=lb.fit_transform(df['smoking_status'])
df.dtypes
```

[10]:
```
id                   int64
gender               int64
age                float64
hypertension         int64
heart_disease        int64
ever_married         int64
work_type            int64
Residence_type       int64
avg_glucose_level  float64
bmi                float64
smoking_status       int64
```

```
stroke                 int64
dtype: object
```

[11]:
```
#Removing the non-essential column

df.drop(['id'],axis=1,inplace=True)
df
```

[11]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.0	0	1	1	2	1	228.69	36.600000	1	1
1	0	61.0	0	0	1	3	0	202.21	28.893237	2	1
2	1	80.0	0	1	1	2	0	105.92	32.500000	2	1
3	0	49.0	0	0	1	2	1	171.23	34.400000	3	1
4	0	79.0	1	0	1	3	0	174.12	24.000000	2	1
…	…	…	…	…	…	…	…	…	…	…	…
5105	0	80.0	1	0	1	2	1	83.75	28.893237	2	0
5106	0	81.0	0	0	1	3	1	125.20	40.000000	2	0
5107	0	35.0	0	0	1	3	0	82.99	30.600000	2	0
5108	1	51.0	0	0	1	2	0	166.29	25.600000	1	0
5109	0	44.0	0	0	1	0	1	85.28	26.200000	0	0

```
[5110 rows x 11 columns]
```

[12]:
```
#Splitting data into input and output features

x=df.iloc[:,:-1].values
x
```

[12]:
```
array([[  1.        ,  67.        ,   0.        , …, 228.69      ,
         36.6       ,   1.        ],
       [  0.        ,  61.        ,   0.        , …, 202.21      ,
         28.89323691,   2.        ],
       [  1.        ,  80.        ,   0.        , …, 105.92      ,
         32.5       ,   2.        ],
```

```
       …,
       [  0.        ,  35.        ,   0.        , …,  82.99      ,
         30.6       ,   2.        ],
       [  1.        ,  51.        ,   0.        , …, 166.29      ,
         25.6       ,   1.        ],
       [  0.        ,  44.        ,   0.        , …,  85.28      ,
         26.2       ,   0.        ]])
```

[13]:
```
y=df.iloc[:,-1].values
y
```

[13]:
```
array([1, 1, 1, …, 0, 0, 0])
```

[14]:
```
#Separating features into training and testing datas

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪30,random_state=42)
x_train
```

[14]:
```
array([[  1.  ,   4.  ,   0.  , …,  90.42,  16.2 ,   0.  ],
       [  1.  ,  29.  ,   0.  , …, 207.58,  22.8 ,   3.  ],
       [  1.  ,  44.  ,   1.  , …,  91.28,  26.5 ,   2.  ],
       …,
       [  0.  ,   1.16,   0.  , …,  97.28,  17.8 ,   0.  ],
       [  1.  ,  80.  ,   0.  , …, 196.08,  31.  ,   1.  ],
       [  0.  ,  46.  ,   0.  , …, 100.15,  50.3 ,   3.  ]])
```

[15]:
```
x_test
```

[15]:
```
array([[ 1.  , 31.  ,  0.  , …, 64.85, 23.  ,  0.  ],
       [ 1.  , 40.  ,  0.  , …, 65.29, 28.3 ,  2.  ],
       [ 0.  ,  8.  ,  0.  , …, 74.42, 22.5 ,  0.  ],
       …,
       [ 1.  , 42.  ,  0.  , …, 93.79, 27.2 ,  2.  ],
       [ 0.  , 57.  ,  0.  , …, 69.4 , 24.  ,  0.  ],
       [ 0.  , 60.  ,  0.  , …, 73.04, 25.3 ,  2.  ]])
```

[16]:
```
#Normalization

from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
x_train
```

[16]:
```
array([[1.        , 0.04692082, 0.        , …, 0.16295818, 0.06758305,
        0.        ],
       [1.        , 0.35239492, 0.        , …, 0.70381313, 0.14318442,
        1.        ],
       [1.        , 0.53567937, 1.        , …, 0.16692826, 0.18556701,
        0.66666667],
       …,
       [0.        , 0.01221896, 0.        , …, 0.19462653, 0.08591065,
        0.        ],
       [1.        , 0.97556207, 0.        , …, 0.65072477, 0.2371134 ,
        0.33333333],
       [0.        , 0.5601173 , 0.        , …, 0.20787554, 0.45819015,
        1.        ]])
```

[17]:
```
#Models creation

from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.metrics import classification_report

knn=KNeighborsClassifier(n_neighbors=7)
nvb=BernoulliNB()
sv=SVC()
lst=[knn,nvb,sv]
```

[18]:
```
for i in lst:
  print("\n Model-:")
  print(i)
  print("-"*30)
  i.fit(x_train,y_train)
  print("Predicted value -")
  y_pred=i.predict(x_test)
  print(y_pred)
  print("\n Confusion matrix -")
  print(confusion_matrix(y_test,y_pred))
  print("\n Accuracy Score - ")
  print(accuracy_score(y_test,y_pred))
  print("\n Classification report - ")
  print(classification_report(y_test,y_pred))

  print('*'*50)
```

```
 Model-:
KNeighborsClassifier(n_neighbors=7)
```

```
------------------------------
Predicted value -
[0 0 0 … 0 0 0]

 Confusion matrix -
[[1443    1]
 [  88    1]]

 Accuracy Score -
0.9419439008480104

 Classification report -
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      1444
           1       0.50      0.01      0.02        89

    accuracy                           0.94      1533
   macro avg       0.72      0.51      0.50      1533
weighted avg       0.92      0.94      0.92      1533

******************************************************

 Model-:
BernoulliNB()
------------------------------
Predicted value -
[0 0 0 … 0 0 0]

 Confusion matrix -
[[1444    0]
 [  89    0]]

 Accuracy Score -
0.9419439008480104

 Classification report -
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      1444
           1       0.00      0.00      0.00        89

    accuracy                           0.94      1533
   macro avg       0.47      0.50      0.49      1533
weighted avg       0.89      0.94      0.91      1533

******************************************************
```

```
 Model-:
SVC()
------------------------------
```

Predicted value -

[0 0 0 ... 0 0 0]

Confusion matrix -

[[1444 0]
 [89 0]]

Accuracy Score -

0.9419439008480104

Classification report -

	precision	recall	f1-score	support
0	0.94	1.00	0.97	1444
1	0.00	0.00	0.00	89
accuracy			0.94	1533
macro avg	0.47	0.50	0.49	1533
weighted avg	0.89	0.94	0.91	1533

****