In [230]:

```
# Importing required packages
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn.svm import SVC

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plotPointer
import numpy as np
```

In [231]:

```
strokesData = read_csv('F:\ML\For Stroke Risk Dataset\healthcare-dataset-stroke-data.csv')
# strokesData.head()
strokesData
```

Out[231]:

	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	0
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

5110 rows × 12 columns

In [232]:

```
strokesData = strokesData[['hypertension', 'heart_disease', 'avg_glucose_level', 'bmi', 'smoking_status', 'stroke']]
# strokesData['smoking_status'] = strokesData['smoking_status'].replace('formerly smoked', 1)
# strokesData['smoking_status'] = strokesData['smoking_status'].replace('smokes', 2)
# strokesData['smoking_status'] = strokesData['smoking_status'].replace('never smoked', 0)
# strokesData['smoking_status'] = strokesData['smoking_status'].replace('Unknown', 0)
strokesData = strokesData.dropna()
```

In [233]:

```
if 'id' in strokesData :
    strokesData = strokesData.drop('id', axis=1)

strokesData
```

Out[233]:

	hypertension	heart_disease	avg_glucose_level	bmi	smoking_status	stroke
0	0	1	228.69	36.6	formerly smoked	1
2	0	1	105.92	32.5	never smoked	1
3	0	0	171.23	34.4	smokes	1
4	1	0	174.12	24.0	never smoked	1
5	0	0	186.21	29.0	formerly smoked	1
...	...	...	...	...	...	...
5104	0	0	103.08	18.6	Unknown	0
5106	0	0	125.20	40.0	never smoked	0
5107	0	0	82.99	30.6	never smoked	0
5108	0	0	166.29	25.6	formerly smoked	0
5109	0	0	85.28	26.2	Unknown	0

4909 rows × 6 columns

In [234]:

```
#  Spliting the data in training and testing subsets
D = strokesData.values
# iloc to select specefic rows
x = strokesData.iloc[:,:-1]
y = strokesData.iloc[:,-1]
# x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.50, random_state = 1)
```

In [235]:

```
transformingColoumn = ColumnTransformer(transformers = [('encoder', OneHotEncoder(), [0,4])], remainder='passthrough')
x = np.array(transformingColoumn.fit_transform(x))
x

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.50, random_state = 1)
```

In [236]:

```
sc = StandardScaler()
x_train_scaled = sc.fit_transform(x_train)
x_test_scaled = sc.fit_transform(x_test)
```

In [237]:

```
# SVM
modelSVM = SVC(random_state = 0, kernel = 'linear')
modelSVM.fit(x_train_scaled, y_train)
y_prediction_SVM = modelSVM.predict(x_test_scaled)

accOfSVM = accuracy_score(y_test, y_prediction_SVM)
accOfSVM

print("----------------------------------")
print('The accuracy of the SVM for Testing is: {}'.format(accOfSVM.round(4)))
print("----------------------------------")
# save the accuracy score
score = set()
score.add(('SVM', accOfSVM.round(4)))
```

```
----------------------------------
The accuracy of the SVM for Testing is: 0.9605
----------------------------------
```

In [238]:

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree Algoithm
modelDT = DecisionTreeClassifier(criterion = 'entropy')
modelDT.fit(x_train_scaled, y_train) #train the model with the training dataset
y_prediction_DT = modelDT.predict(x_test_scaled)

accOfDT = accuracy_score(y_test, y_prediction_DT)
accOfDT


print("----------------------------------")
print('The accuracy of the DT for Testing is: {}'.format(accOfDT.round(4)))
print("----------------------------------")
# save the accuracy score
score.add(('DT', accOfDT.round(4)))
```

```
----------------------------------
The accuracy of the DT for Testing is: 0.9214
----------------------------------
```

In [239]:

```
# Naive Bayes
from sklearn.naive_bayes import GaussianNB
modelNB = GaussianNB()
modelNB.fit(x_train_scaled, y_train) #train the model with the training dataset
y_prediction_NB = modelNB.predict(x_test_scaled)

accOfNB = accuracy_score(y_prediction_NB, y_test)
accOfNB


print("----------------------------------")
print('The accuracy of the NB for Testing is: {}'.format(accOfNB.round(4)))
print("----------------------------------")
# save the accuracy score
score.add(('NB', accOfNB.round(4)))
```

```
----------------------------------
The accuracy of the NB for Testing is: 0.8721
----------------------------------
```

In [240]:

```
# Linear Regression
from sklearn.linear_model import LogisticRegression # for Logistic Regression algorithm
model_LR = LogisticRegression()
model_LR.fit(x_train_scaled, y_train) #train the model with the training dataset
y_prediction_LR = model_LR.predict(x_test_scaled) #pass the testing data to the trained model
# checking the accuracy of the algorithm.
# by comparing predicted output by the model and the actual output
accOfLR = accuracy_score(y_prediction_LR, y_test)
accOfLR


print("----------------------------------")
print('The accuracy of the LR for Testing is: {}'.format(accOfLR.round(4)))
print("----------------------------------")
# save the accuracy score
score.add(('LR', accOfLR.round(4)))
```

```
----------------------------------
The accuracy of the LR for Testing is: 0.9605
----------------------------------
```

In [241]:

```
# KNN
from scipy import stats
from sklearn.neighbors import KNeighborsClassifier # for K nearest neighbours
#from sklearn.linear_model import LogisticRegression # for Logistic Regression algorithm
model_KNN = KNeighborsClassifier(n_neighbors=3) # 3 neighbours for putting the new data into a class
model_KNN.fit(x_train_scaled, y_train) #train the model with the training dataset
y_prediction_KNN = model_KNN.predict(x_test_scaled)

accOfKNN = accuracy_score(y_prediction_KNN, y_test)
accOfKNN

print("----------------------------------")
print('The accuracy of the KNN for Testing is: {}'.format(accOfKNN.round(4)))
print("----------------------------------")
# save the accuracy score
score.add(('KNN', accOfKNN.round(4)))
```

```
----------------------------------
The accuracy of the KNN for Testing is: 0.9523
----------------------------------

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction
functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.
11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is
taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warn
ing.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [242]:

```
print("The accuracy scores of different Models:")
print("----------------------------------------")
for s in score:
    print(s)
```

```
The accuracy scores of different Models:
----------------------------------------
('DT', 0.9214)
('LR', 0.9605)
('NB', 0.8721)
('KNN', 0.9523)
('SVM', 0.9605)
```

In [ ]: