```
Predict the Stroke in the given data set and apply all available kernels in SVC model
1- prepare the data set according to need (numeric)
2- let us know the which kernel is best for such application
```

In [31]:

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler , OneHotEncoder ,LabelEncoder ,MinMaxScaler
from sklearn.compose import ColumnTransformer
from sklearn.metrics import f1_score
```

In [2]:

```python
## 1- prepare the data set according to need (numeric)

data = pd.read_csv("../DataSets/healthcare-dataset-stroke-data.csv")
data.drop("id",axis=1,inplace=True)
data.head()
```

Out[2]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	strok
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	

In [3]:

```python
## check for missing values to Clean and preprocess data
```

In [4]:
```
print("Duplicated Values",data.duplicated().sum())
data.isna().sum()
```

```
Duplicated Values 0
```

Out[4]:
```
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
```

In [5]:
```
df = data.dropna()
#now check for duplicates
df.isna().sum()
```

Out[5]:
```
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 [6]:
```
df.head()
```

Out[6]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	strok
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	
5	Male	81.0	0	0	Yes	Private	Urban	186.21	29.0	formerly smoked	

In [41]:
```
df["stroke"].value_counts()
```

Out[41]:
```
0    4700
1     209
Name: stroke, dtype: int64
```

In [ ]:

In [69]:

```
##  Now do the preprocessing ( Label Encoding & StandardScaling)
X = df.drop("stroke",axis=1)
Y = df["stroke"]
scaling = StandardScaler()
encoder = OneHotEncoder()

category =["gender","ever_married","work_type","Residence_type","smoking_status"]
numerical = ["age","avg_glucose_level","bmi"]


transform = ColumnTransformer([("numerical",scaling,numerical),
                               ("category",encoder,category)],remainder="passthrough")

trans_x =  transform.fit_transform(X)
trans_x
```

Out[69]:

```
array([[ 1.07013796,  2.77769839,  0.98134488, ...,  0.        ,
         0.        ,  1.        ],
       [ 1.64656262,  0.0138418 ,  0.45926914, ...,  0.        ,
         0.        ,  1.        ],
       [ 0.27201152,  1.48413156,  0.70120668, ...,  1.        ,
         0.        ,  0.        ],
       ...,
       [-0.34875349, -0.50236926,  0.21733161, ...,  0.        ,
         0.        ,  0.        ],
       [ 0.36069224,  1.37291993, -0.41934612, ...,  0.        ,
         0.        ,  0.        ],
       [ 0.05030973, -0.45081569, -0.34294479, ...,  0.        ,
         0.        ,  0.        ]])
```

In [70]:

```
# train _test_split
np.random.seed(42)

x_train ,x_test ,y_train ,y_test = train_test_split(trans_x , Y ,test_size =0.2 ,random_state = 42)
```

In [71]:

```
## With first hyperparameter kernal =  "rbf"
```

In [72]:

```
svc1 = SVC(kernel='rbf')
svc1.fit(x_train ,y_train)
```

Out[72]:

```
SVC()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [73]:

```
print("svc1.score with Kernal = rbf",svc1.score(x_test , y_test))
y_preds = svc2.predict(x_test)
f1_score(y_test , y_preds)
```

```
svc1.score with Kernal = rbf 0.9460285132382892
```

Out[73]:

```
0.18761726078799248
```

In [74]:

```
## With 2nd hyperparameter kernal =  "linear"
```

In [75]:

```
svc2 = SVC(kernel='linear')
svc2.fit(x_train ,y_train)

print("svc2.score with Kernal = 'linear' ",svc2.score(x_test , y_test))
y_preds = svc2.predict(x_test)
y_preds
```

```
svc2.score with Kernal = 'linear'  0.9460285132382892
```

```
Out[75]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [76]:
```
## With 3nd hyperparameter kernal =  "poly'
```

In [77]:
```
svc3 = SVC(kernel='poly')
svc3.fit(x_train ,y_train)

print("svc2.score with Kernal = 'poly’' ",svc3.score(x_test , y_test))
```

```
svc2.score with Kernal = 'poly’'  0.945010183299389
```

In [78]:
```
## With 4nd hyperparameter kernal =  "sigmoid
```

In [79]:
```
svc4 = SVC(kernel='sigmoid')
svc4.fit(x_train ,y_train)

print("svc4.score with Kernal = 'sigmoid' ",svc4.score(x_test , y_test))
```

```
svc4.score with Kernal = 'sigmoid'  0.9215885947046843
```

In [80]:
```
## With 5nd hyperparameter kernal =  "precomputed"
```

In [ ]:
```
svc5 = SVC(kernel='precomputed')
svc5.fit(x_train ,y_train)

print("svc4.score with Kernal = 'sigmoid' ",svc5.score(x_test , y_test))
```

In [ ]: