Module 3: Performance measurements of Naive Bayes

```
# Import the Neccesary packages.
import pandas as pd
import numpy as np
```

In [ ]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```
data = pd.read_csv('brain_stroke.csv')
```

In [ ]:

```
data.head()
```

In [ ]:

```
df = data.dropna()
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

var = ['gender','smoking_status']

for i in var:
    df[i] = le.fit_transform(df[i]).astype(int)
```

In [ ]:

```
df.columns
```

In [ ]:

```
df.head()
```

In [ ]:

```
df['stroke'].unique()
```

In [ ]:

```
del df['ever_married']
del df['work_type']
del df['Residence_type']
```

In [ ]:

```
df.head()
```

In [ ]:

```
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='stroke', axis=1)
#Response variable
Y = df.loc[:,'stroke']
```

In [ ]:

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros =RandomOverSampler(random_state=1)
x_ros,y_ros=ros.fit_resample(X,Y)
print("OUR DATASET COUNT         : ", Counter(Y))
print("OVER SAMPLING DATA COUNT  : ", Counter(y_ros))
```

In [ ]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_ros, y_ros, test_size=0.30,
random_state=1, stratify=y_ros)
print("Number of training dataset : ", len(x_train))
print("Number of test dataset     : ", len(x_test))
print("Total number of dataset    : ", len(x_train)+len(x_test))
```

In [ ]:

```
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score,plot_confusion_matrix
from sklearn.naive_bayes import GaussianNB
```

In [ ]:

```
NB = GaussianNB()
NB.fit(x_train,y_train)
predictNb =NB.predict(x_test)
```

In [ ]:

```
accuracy = accuracy_score(y_test, predictNb)
print("Accuracy of naive bayes classifier:", accuracy*100)
```

In [ ]:

```
cr = classification_report(y_test, predictNb)
print("Classification report \n\n:", cr)
```

In [ ]:

```
cm = confusion_matrix(y_test, predictNb)
print("Confusion matrix:\n", cm)
```

In [ ]:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(6,6))
plot_confusion_matrix(NB, x_test, y_test, ax=ax)
plt.title('Confusion matrix of Mlp classifier')
plt.show()
```

In [ ]:

In [ ]: