4) LOGISTIC REGRESSION

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df=pd.read_csv('/content/drive/MyDrive/HCP/kyphosis.csv')
print(df.head())
from sklearn.model_selection import train_test_split

X=df.drop('Kyphosis',axis=1)
y=df['Kyphosis']

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)


#---Dataset represents the patients condition on kyphosis(situation on spinal condition)
#---After surgery whether the kyphosis condition was present or absent in the patient
#---Age=age is the person in months
#---Number- is the number of vetebrae involved in operation
#---start is the top most vertebare operated in the operation
```

```
  Kyphosis  Age  Number  Start
0   absent   71       3      5
1   absent  158       3     14
2  present  128       4      5
3   absent    2       5      1
4   absent    1       4     15
```

```
from sklearn.linear_model import LogisticRegression

logmodel=LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
LogisticRegression()
```

LogisticRegression(penalty='l2'(used to specify the penalty : {'l1', 'l2', 'elasticnet', 'none'}, default='l2'),

```
dual=False(---Dual or primal formulation. Dual formulation is only implemented for l2 penalty with liblinear solver. Prefer dual=False when n_samp
```

```
predictions=logmodel.predict(X_test)
print(predictions)
```

```
['present' 'absent' 'absent' 'present' 'absent' 'present' 'absent'
 'absent' 'absent' 'absent' 'absent' 'present' 'absent' 'absent' 'absent'
 'absent' 'absent' 'absent' 'present' 'absent' 'absent' 'absent' 'absent'
 'absent' 'absent']
```

Evaluating Metrics

```
random_state=None (---The seed of the pseudo random number generator to use when shuffling the data.),
```

Classification Report

```
solver='lbfgs' (---Algorithm to use in the optimization problem,solver={'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}each has its own uniquene
```

```
from  sklearn.metrics  import  classification_report

 print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

      absent       0.90      0.86      0.88        21
     present       0.40      0.50      0.44         4
```



```
l1_ratio=None (---its a combiation of l1 and l2),
```

```
              accuracy                           0.80        25
             macro avg       0.65      0.68      0.66        25
          weighted avg       0.82      0.80      0.81        25
```

Confusion Metrics

```
)
 1 from sklearn.metrics import confusion_matrix
 2
 3 confusion_matrix(y_test,predictions)
```

```
    array([[18,  3],
           [ 2,  2]])
```