Title of the Project -To create a classification filter (Using Logistics Regression & KNN Classification Algorithm) to predict Heart Failure. Compare the performance of the filters

Brief on the Project-

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide.

Heart failure is a common event caused by CVDs and this dataset contains 12 features that can be used to predict mortality by heart failure. Most cardiovascular diseases can be prevented by addressing behavioural risk factors such as tobacco use, unhealthy diet and obesity, physical inactivity and harmful use of alcohol using population-wide strategies. People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help.

Deliverables of the Project:

1. In this work, Machine Learning Models using Logistic Regression and KNN method are proposed to predict whether a person can die or not due to heart failure based on some input information such as age ,sex , blood pressure, smoke, diabetes,ejection fraction, creatinine phosphokinase, serum_creatinine, serum_sodium, time.
2. The performance of the proposed machine learning models will be evaluated and compared in terms of accuracy, errors, f1 score and R2 value.

Resources

#There are some factors that affects Death Event. This dataset contains person's information like age ,sex , blood pressure, smoke, diabetes,ejection fraction, creatinine phosphokinase, serum_creatinine, serum_sodium, time and we have to predict their DEATH EVENT.

Step-1- Firstly we have preprocessed our data taken from kaggle using various EDA approaches

#a) Libraries for Data Preprocessing- IN EDA firstly we have imported general libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

b)Reading Data file- we have brought the data file in our jupyter notebook using pandas

In [2]:
```
data=pd.read_csv('heart_failure_clinical_records.csv')
data.sample(5)
```

Out[2]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	tim
24	75.0	0	582	1	30	1	263358.03	1.83	134	0	0	2
253	70.0	0	88	1	35	1	236000.00	1.20	132	0	0	21
285	55.0	1	170	1	40	0	336000.00	1.20	135	1	0	25
190	80.0	0	582	1	35	0	350000.00	2.10	134	1	0	17
165	80.0	0	776	1	38	1	192000.00	1.30	135	0	0	13

In [3]:
```
data.shape
```

Out[3]:
```
(299, 13)
```

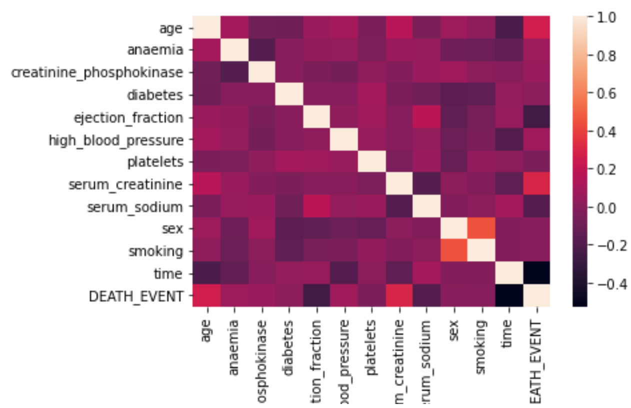
In [4]:
```
# checking the information of the data like data types, null count etc.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    int64
 2   creatinine_phosphokinase  299 non-null    int64
 3   diabetes                  299 non-null    int64
 4   ejection_fraction         299 non-null    int64
 5   high_blood_pressure       299 non-null    int64
 6   platelets                 299 non-null    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    int64
 9   sex                       299 non-null    int64
 10  smoking                   299 non-null    int64
 11  time                      299 non-null    int64
 12  DEATH_EVENT               299 non-null    int64
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

c) Finding Corelation between variables

In [5]:
```
sns.heatmap(data.corr())
```

Out[5]: <AxesSubplot:>



In [6]:
```
data.corr().T
```

Out[6]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_
age	1.000000	0.088006	-0.081584	-0.101012	0.060098	0.093289	-0.052354	0.159187	-(
anaemia	0.088006	1.000000	-0.190741	-0.012729	0.031557	0.038182	-0.043786	0.052174	(
creatinine_phosphokinase	-0.081584	-0.190741	1.000000	-0.009639	-0.044080	-0.070590	0.024463	-0.016408	(
diabetes	-0.101012	-0.012729	-0.009639	1.000000	-0.004850	-0.012732	0.092193	-0.046975	-(
ejection_fraction	0.060098	0.031557	-0.044080	-0.004850	1.000000	0.024445	0.072177	-0.011302	(
high_blood_pressure	0.093289	0.038182	-0.070590	-0.012732	0.024445	1.000000	0.049963	-0.004935	(
platelets	-0.052354	-0.043786	0.024463	0.092193	0.072177	0.049963	1.000000	-0.041198	(
serum_creatinine	0.159187	0.052174	-0.016408	-0.046975	-0.011302	-0.004935	-0.041198	1.000000	-(
serum_sodium	-0.045966	0.041882	0.059550	-0.089551	0.175902	0.037109	0.062125	-0.189095	1
sex	0.065430	-0.094769	0.079791	-0.157730	-0.148386	-0.104615	-0.125120	0.006970	-(
smoking	0.018668	-0.107290	0.002421	-0.147173	-0.067315	-0.055711	0.028234	-0.027414	(
time	-0.224068	-0.141414	-0.009346	0.033726	0.041729	-0.196439	0.010514	-0.149315	(
DEATH_EVENT	0.253729	0.066270	0.062728	-0.001943	-0.268603	0.079351	-0.049139	0.294278	-(

d)from the corelation figure we found the variable 'time' is having least relevance for precticting the death events due to heart failure in this data set. so we are deleting this variable as follows:

In [7]:
```
x=data.drop(['time'],axis=1)
```

e) Outlier detection and removal-

In [8]:
```
# checking the counts of outliers in each coloumn of our data set
q1=x.quantile(0.25)
q3=x.quantile(0.75)
iqr=q3-q1
((x<(q1-1.5*iqr)) | (x>(q3+1.5*iqr))).sum()
```

Out[8]:
```
age                          0
anaemia                      0
creatinine_phosphokinase    29
diabetes                     0
ejection_fraction            2
high_blood_pressure          0
platelets                   21
serum_creatinine            29
serum_sodium                 4
sex                          0
smoking                      0
DEATH_EVENT                  0
dtype: int64
```

In [9]:
```
# defining a function to see the various outliers in a coulumn
def find_outliers_IQR(df):
    q1=df.quantile(0.25)
    q3=df.quantile(0.75)
    IQR=q3-q1
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
    return outliers
```

Outlier treatment of reatinine_phosphokinase

In [10]:
```
# finding and replacing outliers of creatinine_phosphokinase
a=find_outliers_IQR(x.creatinine_phosphokinase)
```

In [11]:
```
# upper n lower bound of creatinine_phosphokinase
q1=x.creatinine_phosphokinase.quantile(0.25)
q3=x.creatinine_phosphokinase.quantile(0.75)
iqr=q3-q1
LB=q1-1.5*iqr
UB= q3+1.5*iqr
print(LB)
print(UB)
```

```
-581.75
1280.25
```

In [12]:
```
# calculating the mean value of creatinine_phosphokinase
b=x.creatinine_phosphokinase.mean()
b
```

Out[12]:
```
581.8394648829432
```

In [13]:
```
# replacing all outliers in creatinine_phosphokinase column with the calculated mean
for i in x.index:
    if (x.loc[i,'creatinine_phosphokinase']<LB) | (x.loc[i,'creatinine_phosphokinase']> UB):
        x.loc[i,'creatinine_phosphokinase']=b
```

Outlier treatment of ejection_fraction

In [14]:
```
# finding and replacing outliers of ejection_fraction
a=find_outliers_IQR(x.ejection_fraction)

# upper n lower bound of ejection_fraction
q1=x.ejection_fraction.quantile(0.25)
q3=x.ejection_fraction.quantile(0.75)
iqr=q3-q1
LB1=q1-1.5*iqr
UB1= q3+1.5*iqr
print(LB1)
print(UB1)

#finding mean of ejection_fraction
c=x.ejection_fraction.mean()
c

# replacing outliers in ejection_fraction
for i in x.index:
    if (x.loc[i,'ejection_fraction']<LB1) | (x.loc[i,'ejection_fraction']> UB1):
        x.loc[i,'ejection_fraction']=c
```

```
7.5
67.5
```

Outlier treatment of platelets

In [15]:
```
# finding and replacing outliers of platelets
a=find_outliers_IQR(x.platelets)

# upper n lower bound of platelets
q1=x.platelets.quantile(0.25)
q3=x.platelets.quantile(0.75)
iqr=q3-q1
LB2=q1-1.5*iqr
UB2= q3+1.5*iqr
print(LB2)
print(UB2)

#finding mean of platelets
d=x.platelets.mean()
print(d)

# replacing outliers in platelets
for i in x.index:
    if (x.loc[i,'platelets']<LB2) | (x.loc[i,'platelets']> UB2):
        x.loc[i,'platelets']=d
```

```
76000.0
440000.0
263358.02926421416
```

Outlier treatment of serum_creatinine

In [16]:
```
# finding and replacing outliers of serum_creatinine
a=find_outliers_IQR(x.serum_creatinine)

# upper n lower bound of serum_creatinine
q1=x.serum_creatinine.quantile(0.25)
q3=x.serum_creatinine.quantile(0.75)
iqr=q3-q1
LB3=q1-1.5*iqr
UB3= q3+1.5*iqr
print(LB3)
print(UB3)

#finding mean of serum_creatinine
e=x.serum_creatinine.mean()
e

# replacing outliers in serum_creatinine
for i in x.index:
    if (x.loc[i,'serum_creatinine']<LB3) | (x.loc[i,'serum_creatinine']> UB3):
        x.loc[i,'serum_creatinine']=e
```

```
0.15000000000000024
2.1499999999999995
```

Outlier treatment of serum_sodium

In [17]:
```
# finding and replacing outliers of serum_sodium
a=find_outliers_IQR(x.serum_sodium)

# upper n lower bound of serum_sodium
q1=x.serum_sodium.quantile (0.25)
q3=x.serum_sodium.quantile(0.75)
iqr=q3-q1
LB4=q1-1.5*iqr
UB4= q3+1.5*iqr
print(LB4)
print(UB4)

#finding mean of serum_sodium
f=x.serum_sodium.mean()
f

# replacing outliers in serum_sodium
for i in x.index:
    if (x.loc[i,'serum_sodium']<LB4) | (x.loc[i,'serum_sodium']> UB4):
        x.loc[i,'serum_sodium']=f
```

```
125.0
149.0
```

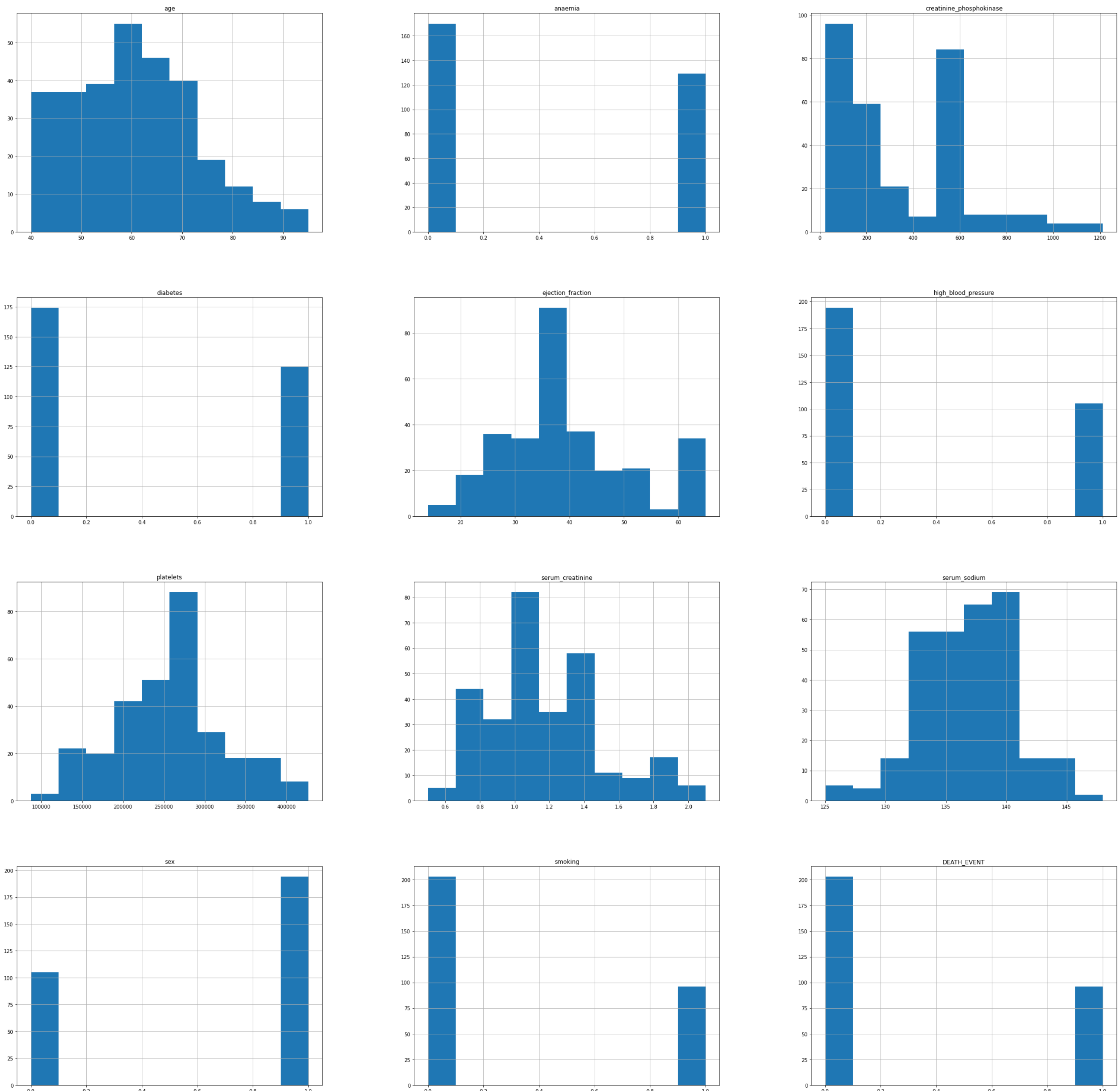
f) Finding Null values of each Variables

In [18]:
```
x.isnull().sum()
```

Out[18]:
```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
DEATH_EVENT                 0
dtype: int64
```

In [19]:
```
# To Find Distribution of each variable
x.hist(figsize=(40,40))
```

Out[19]:
```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'anaemia'}>,
        <AxesSubplot:title={'center':'creatinine_phosphokinase'}>],
       [<AxesSubplot:title={'center':'diabetes'}>,
        <AxesSubplot:title={'center':'ejection_fraction'}>,
        <AxesSubplot:title={'center':'high_blood_pressure'}>],
       [<AxesSubplot:title={'center':'platelets'}>,
        <AxesSubplot:title={'center':'serum_creatinine'}>,
        <AxesSubplot:title={'center':'serum_sodium'}>],
       [<AxesSubplot:title={'center':'sex'}>,
        <AxesSubplot:title={'center':'smoking'}>,
        <AxesSubplot:title={'center':'DEATH_EVENT'}>]], dtype=object)
```



2. Separating (independent)Input and Target(dependent) variables.

In [20]:
```
x_data=x.drop(columns=['DEATH_EVENT']) # Independent variables (input parameters)
y_data=x['DEATH_EVENT']                # Dependent variables (Target)
```

In [21]:
```
x_data.shape
```
Out[21]:
```
(299, 11)
```

In [22]:
```
y_data.shape
```
Out[22]:
```
(299,)
```

3. Splitting data for training and testing

In [23]:
```
from sklearn.model_selection import train_test_split
```

In [24]:
```
xTrain, xTest, yTrain, yTest = train_test_split(x_data, y_data, test_size= 0.25,random_state=0)
```

In [25]:
```
xTrain.shape
```
Out[25]:
```
(224, 11)
```

In [26]:
```
xTrain.head()
```
Out[26]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking
258	45.0	1	66.0	1	25.0	0	233000.00	0.80	135.0	1	0
37	82.0	1	855.0	1	50.0	1	321000.00	1.00	145.0	0	0
97	70.0	1	59.0	0	60.0	0	255000.00	1.10	136.0	0	0
191	64.0	1	62.0	0	60.0	0	309000.00	1.50	135.0	0	0
135	75.0	0	582.0	0	40.0	0	263358.03	1.18	137.0	1	0

In [27]:
```
yTrain.shape
```
Out[27]:
```
(224,)
```

4. Implementing standard scaling to scale the data set within equal range

Both the standard scaler and Minmax scaler have been studied for the same models and it is found that standard scaling is giving best performance and we selected standard scaling as the data set contains large variations.

In [28]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
xTrain = sc.fit_transform(xTrain)
xTest = sc.transform(xTest)
```

5. Logistic Regression Model Building-Training, Testing and Evaluation

In [29]:
```
# importing necessary packages
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.metrics import confusion_matrix
from sklearn import metrics
```

In [30]:
```
logModel = LogisticRegression()
```

In [31]:
```
# Hypertuning of Logistic Regression
param_grid = [
    {'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
    'C' : np.logspace(-4, 4, 20),
    'solver' : ['lbfgs','newton-cg','liblinear','sag','saga'],
    'max_iter' : [100, 1000,2500, 5000]
    }
]
```

In [32]:
```
from sklearn.model_selection import GridSearchCV
```

In [33]:
```
clf = GridSearchCV(logModel, param_grid = param_grid, cv = 3, verbose=True, n_jobs=-1)
```

In [34]:
```
best_clf = clf.fit(xTrain,yTrain)
```

```
Fitting 3 folds for each of 1600 candidates, totalling 4800 fits
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:372: FitFailedWarning:
2160 fits failed out of a total of 4800.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver newton-cg supports only 'l2' or 'none' penalties, got l1 penalty.

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver sag supports only 'l2' or 'none' penalties, got l1 penalty.

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got elasticnet penalty.

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver newton-cg supports only 'l2' or 'none' penalties, got elasticnet penalty.

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 457, in _check_solver
    raise ValueError(
ValueError: Only 'saga' solver supports elasticnet penalty, got solver=liblinear.

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver sag supports only 'l2' or 'none' penalties, got elasticnet penalty.

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
```

```
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1471, in fit
    raise ValueError(
ValueError: l1_ratio must be between 0 and 1; got (l1_ratio=None)

--------------------------------------------------------------------------------
240 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 464, in _check_solver
    raise ValueError("penalty='none' is not supported for the liblinear solver")
ValueError: penalty='none' is not supported for the liblinear solver

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\user\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:969: UserWarning: One or more of the test scores a
re non-finite: [       nan        nan 0.69195195 ...        nan 0.77651652 0.77651652]
  warnings.warn(
```

In [35]:
```
#finding the best hyper parameters
best_clf.best_estimator_
```

Out[35]:
```
LogisticRegression(C=0.615848211066026, penalty='l1', solver='liblinear')
```

In [36]:
```
# using best parameters calculating the train and test accuracy
print (f'Train Accuracy - : {best_clf.score(xTrain,yTrain):.3f}')
print (f'Test Accuracy - : {best_clf.score(xTest,yTest):.3f}')
```

```
Train Accuracy - : 0.790
Test Accuracy - : 0.680
```

In [ ]:
```
print('Mean Absolute Error (MAE) =', mean_absolute_error(yTest, pred))
```

In [54]:
```
pred_test  = best_clf.predict(xTest)
```

In [88]:
```
from sklearn.metrics import classification_report, precision_recall_fscore_support
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import r2_score

print(classification_report(yTest,pred_test))
print('f1 score is: ', f1_score(yTest, pred_test))
print('recall score is: ', recall_score(yTest, pred_test))
print('precision score is: ', precision_score(yTest, pred_test))
print('r2 score is: ', r2_score(yTest, pred_test))
print('MSE is: ', mean_squared_error(yTest, pred_test))
print('MAE is: ', mean_absolute_error(yTest, pred_test))
```

```
              precision    recall  f1-score   support

           0       0.70      0.88      0.78        48
           1       0.60      0.33      0.43        27

    accuracy                           0.68        75
   macro avg       0.65      0.60      0.60        75
weighted avg       0.66      0.68      0.65        75

f1 score is:  0.42857142857142855
recall score is:  0.3333333333333333
precision score is:  0.6
r2 score is:  -0.38888888888888906
MSE is:  0.32
MAE is:  0.32
```

Logistic Regression model without hyper parameter tuning

In [89]:
```
model = LogisticRegression()
model.fit(xTrain,yTrain)
```

Out[89]:
```
LogisticRegression()
```

In [90]:
```
from sklearn.metrics import confusion_matrix
from sklearn import metrics
```

In [91]:
```
pred = model.predict(xTest)
cm = confusion_matrix(yTest,pred)
print("Train set Accuracy: ", metrics.accuracy_score(yTrain, model.predict(xTrain)))
print("Test set Accuracy: ", metrics.accuracy_score(yTest, pred))
```

```
Train set Accuracy:  0.7857142857142857
Test set Accuracy:  0.68
```

In [92]:
```
print(classification_report(yTest,pred))
print('f1 score is: ', f1_score(yTest, pred))
print('recall score is: ', recall_score(yTest, pred))
print('precision score is: ', precision_score(yTest, pred))
print('r2 score is: ', r2_score(yTest, pred))
print('MSE is: ', mean_squared_error(yTest, pred))
print('MAE is: ', mean_absolute_error(yTest, pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.88      0.78        48
           1       0.60      0.33      0.43        27

    accuracy                           0.68        75
   macro avg       0.65      0.60      0.60        75
weighted avg       0.66      0.68      0.65        75

f1 score is:  0.42857142857142855
recall score is:  0.3333333333333333
precision score is:  0.6
r2 score is:  -0.38888888888888906
MSE is:  0.32
MAE is:  0.32
```

It is observed that logistic regression with and without hyperparameter tuning gives same results for this data set. In this data set the Death_event data is unbalanced but if we do balancing using smote then the accuracy and other performance parameters degrades as comapred to that of unbalanced data. hence we avoided to do balancing of data.

6. KNN- Training, Testing and performance Evaluation

One of the challenges in a k-NN algorithm is finding the best 'k' i.e. the number of neighbors to be used in the majority vote while deciding the class. Generally, it is advisable to test the accuracy of your model for different values of k and then select the best one from them.

In [93]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn
```

Out[93]:
```
KNeighborsClassifier()
```

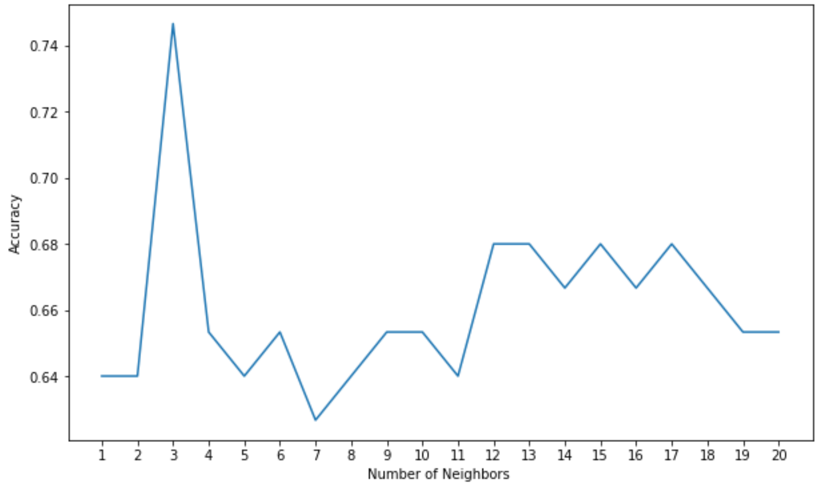
In [94]:
```
from sklearn import metrics
```

In [95]:
```
# calculating the accuracy of models with different values of k
mean_acc = np.zeros(20)
for i in range(1,21):
    #Train Model and Predict
    knn = KNeighborsClassifier(n_neighbors = i).fit(xTrain,yTrain)
    yhat= knn.predict(xTest)
    mean_acc[i-1] = metrics.accuracy_score(yTest, yhat)

mean_acc
```

Out[95]:
```
array([0.64      , 0.64      , 0.74666667, 0.65333333, 0.64      ,
       0.65333333, 0.62666667, 0.64      , 0.65333333, 0.65333333,
       0.64      , 0.68      , 0.68      , 0.66666667, 0.68      ,
       0.66666667, 0.68      , 0.66666667, 0.65333333, 0.65333333])
```

In [96]:
```
# plotting numberof neighbors vs accuracy
loc = np.arange(1,21,step=1.0)
plt.figure(figsize = (10, 6))
plt.plot(range(1,21), mean_acc)
plt.xticks(loc)
plt.xlabel('Number of Neighbors ')
plt.ylabel('Accuracy')
plt.show()
```



from above figure it is observed that the accuracy will be maximum when number of neighbors are 3 so we select n_neighbors as3 in following knn model

In [100]:
```
model1 = KNeighborsClassifier(n_neighbors=3)
model1.fit(xTrain,yTrain)
pred1 = model.predict(xTest)
cm = confusion_matrix(yTest,pred1
                     )
print("Train set Accuracy: ", metrics.accuracy_score(yTrain, model1.predict(xTrain)))
print("Test set Accuracy: ", metrics.accuracy_score(yTest, pred))
```

```
Train set Accuracy:  0.8526785714285714
Test set Accuracy:  0.68
```

In [101]:
```
print(classification_report(yTest,pred1))
print('f1 score is: ', f1_score(yTest, pred1))
print('recall score is: ', recall_score(yTest, pred1))
print('precision score is: ', precision_score(yTest, pred1))
print('r2 score is: ', r2_score(yTest, pred1))
print('MSE is: ', mean_squared_error(yTest, pred1))
print('MAE is: ', mean_absolute_error(yTest, pred1))
```

```
              precision    recall  f1-score   support

           0       0.74      0.94      0.83        48
           1       0.79      0.41      0.54        27

    accuracy                           0.75        75
   macro avg       0.76      0.67      0.68        75
weighted avg       0.75      0.75      0.72        75

f1 score is:  0.5365853658536585
recall score is:  0.4074074074074074
precision score is:  0.7857142857142857
r2 score is:  -0.0995370370370372
MSE is:  0.25333333333333335
MAE is:  0.25333333333333335
```

Performance Comparison

parameter	Logistic Regression	KNN Method
Train Accuracy	79%	85%
Test Accuracy	68%	68%
F1 Score	0.428	0.53
Recall	0.333	0.407
Precision	0.6	0.78
R2 Score	-0.388	-0.099
MSE	0.32	0.253
MAE	0.32	0.253

From the above comparison table , it is concluded that KNN model performs better than the logistic regression model for this data set to predict whether a person will die due to heart failure or not.

Submitted by- Dr. Ruchi, ruchi061179@gmail.com (mailto:ruchi061179@gmail.com), 9814796077