**IMPORT NECESSARY LIBRARIES**

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter as c
import missingno as msno
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
#from sklearn. linear_model import LogisticRegression
#import pickle
```

**READ THE DATASET**

In [ ]:

```
f1=pd.read_csv("chronickidneydisease.csv")
```

In [ ]:

```
f1.head()
```

Out[ ]:

| | id | age | bp | sg | al | su | rbc | pc | pcc | ba | ... | pcv | wc | rc | htn | dm | cad | appet | pe | ane | classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 48.0 | 80.0 | 1.020 | 1.0 | 0.0 | NaN | normal | notpresent | notpresent | ... | 44 | 7800 | 5.2 | yes | yes | no | good | no | no | ckd |
| **1** | 1 | 7.0 | 50.0 | 1.020 | 4.0 | 0.0 | NaN | normal | notpresent | notpresent | ... | 38 | 6000 | NaN | no | no | no | good | no | no | ckd |
| **2** | 2 | 62.0 | 80.0 | 1.010 | 2.0 | 3.0 | normal | normal | notpresent | notpresent | ... | 31 | 7500 | NaN | no | yes | no | poor | no | yes | ckd |
| **3** | 3 | 48.0 | 70.0 | 1.005 | 4.0 | 0.0 | normal | abnormal | present | notpresent | ... | 32 | 6700 | 3.9 | yes | no | no | poor | yes | yes | ckd |
| **4** | 4 | 51.0 | 80.0 | 1.010 | 2.0 | 0.0 | normal | normal | notpresent | notpresent | ... | 35 | 7300 | 4.6 | no | no | no | good | no | no | ckd |

5 rows × 26 columns

In [ ]:

```
f1.tail()
```

Out[ ]:

| | id | age | bp | sg | al | su | rbc | pc | pcc | ba | ... | pcv | wc | rc | htn | dm | cad | appet | pe | ane | classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **395** | 395 | 55.0 | 80.00 | 1.020 | 0.0 | 0.0 | normal | normal | notpresent | notpresent | ... | 47 | 6700 | 4.9 | no | no | no | good | no | no | notckd |
| **396** | 396 | 42.0 | 70.00 | 1.025 | 0.0 | 0.0 | normal | normal | notpresent | notpresent | ... | 54 | 7800 | 6.2 | no | no | no | good | no | no | notckd |
| **397** | 397 | 12.0 | 80.00 | 1.020 | 0.0 | 0.0 | normal | normal | notpresent | notpresent | ... | 49 | 6600 | 5.4 | no | no | no | good | no | no | notckd |
| **398** | 398 | 17.0 | 60.00 | 1.025 | 0.0 | 0.0 | normal | normal | notpresent | notpresent | ... | 51 | 7200 | 5.9 | no | no | no | good | no | no | notckd |
| **399** | 399 | 58.0 | 80.00 | 1.025 | 0.0 | 0.0 | normal | normal | notpresent | notpresent | ... | 53 | 6800 | 6.1 | no | no | no | good | no | no | notckd |

5 rows × 26 columns

In [ ]:

```
#dropping column 'id' as it is unnecessary
f1.drop(["id"], axis=1, inplace=True)
```

In [ ]:

```
f1.columns
```

Out[ ]:

```
Index(['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
       'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane', 'classification'],
      dtype='object')
```

In [ ]:

```
#rename column names
```

```
f1.columns=['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar',
'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
            'blood_glucose_random', 'blood_urea', 'serum_creatinine',
'sodium', 'potassium','hemoglobin', 'packed_cell_volume',
            'white_blood_cell_count', 'red_blood_cell_count', 'hypertension',
'diabetesmellitus', 'coronary_artery_disease', 'appetite',
            'pedal_edema' , 'anemia', 'class']
cols=f1.columns
cols
```

Out[ ]:

```
Index(['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar',
       'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
       'blood_glucose_random', 'blood_urea', 'serum_creatinine', 'sodium',
       'potassium', 'hemoglobin', 'packed_cell_volume',
       'white_blood_cell_count', 'red_blood_cell_count', 'hypertension',
       'diabetesmellitus', 'coronary_artery_disease', 'appetite',
       'pedal_edema', 'anemia', 'class'],
      dtype='object')
```

**UNDERSTANDING THE DATATYPE**

In [ ]:

```
#target column - find unique elements of the array/column
f1['class'].unique()
```

Out[ ]:

```
array(['ckd', 'ckd\t', 'notckd'], dtype=object)
```

In [ ]:

```
#rectifying the unknown class in the dataset
f1['class']=f1['class'].replace("ckd\t", "ckd")
f1['class'].unique()
```

Out[ ]:

```
array(['ckd', 'notckd'], dtype=object)
```

In [ ]:

```
#target column
f1['class'].value_counts()
```

Out[ ]:

```
ckd       250
notckd    150
Name: class, dtype: int64
```

In [ ]:

```
f1['coronary_artery_disease'].value_counts()
```

Out[ ]:

```
no      362
yes      34
\tno      2
Name: coronary_artery_disease, dtype: int64
```

In [ ]:

```
#rectifying column
f1['coronary_artery_disease']=f1['coronary_artery_disease'].replace("\tno",
"no")
f1['coronary_artery_disease'].value_counts()
```

Out[ ]:

```
no     364
yes     34
Name: coronary_artery_disease, dtype: int64
```

In [ ]:

```
f1['diabetesmellitus'].value_counts()
```

Out[ ]:

```
no       258
yes      134
\tno       3
\tyes      2
 yes       1
Name: diabetesmellitus, dtype: int64
```

In [ ]:

```
#rectifying column
f1['diabetesmellitus']=f1['diabetesmellitus'].replace("\tno", "no")
f1['diabetesmellitus']=f1['diabetesmellitus'].replace("\tyes", "yes")
f1['diabetesmellitus']=f1['diabetesmellitus'].replace(" yes", "yes")
f1['diabetesmellitus'].value_counts()
```

Out[ ]:

```
no     261
yes    137
Name: diabetesmellitus, dtype: int64
```

**CHECK CATEGORICAL AND NUMERICAL DATA**

In [ ]:

```
num_cols = f1.select_dtypes(include='number').columns
num_cols
```

Out[ ]:

```
Index(['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar',
       'blood_glucose_random', 'blood_urea', 'serum_creatinine', 'sodium',
       'potassium', 'hemoglobin'],
      dtype='object')
```

In [ ]:

```
cat_cols= f1.select_dtypes(include='object').columns
cat_cols
```

Out[ ]:

```
Index(['red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
       'packed_cell_volume', 'white_blood_cell_count', 'red_blood_cell_count'
,
       'hypertension', 'diabetesmellitus', 'coronary_artery_disease',
       'appetite', 'pedal_edema', 'anemia', 'class'],
      dtype='object')
```

**HANDLING THE MISSING VALUES**

In [ ]:

```
#conversion of numerical data(string format to numeric)
f1.packed_cell_volume = pd.to_numeric(f1.packed_cell_volume, errors="coerce")
f1.white_blood_cell_count = pd.to_numeric(f1.white_blood_cell_count,
errors="coerce")
```

```
f1.red_blood_cell_count = pd.to_numeric(f1.red_blood_cell_count,
errors="coerce")
```

In [ ]:

```
#check if any column is null
'''
True : null present
False: non-null
'''
f1.isnull().any()
```

Out[ ]:

```
age                          True
blood_pressure               True
specific_gravity             True
albumin                      True
sugar                        True
red_blood_cells              True
pus_cell                     True
pus_cell_clumps              True
bacteria                     True
blood_glucose_random         True
blood_urea                   True
serum_creatinine             True
sodium                       True
potassium                    True
hemoglobin                   True
packed_cell_volume           True
white_blood_cell_count       True
red_blood_cell_count         True
hypertension                 True
diabetesmellitus             True
coronary_artery_disease      True
appetite                     True
pedal_edema                  True
anemia                       True
class                       False
dtype: bool
```

In [ ]:

```
#handle missing values
f1[ "blood_pressure"].fillna(f1["blood_pressure"].mean(),inplace=True)
f1[
"blood_glucose_random"].fillna(f1["blood_glucose_random"].mean(),inplace=True
)
f1[ "blood_urea"].fillna(f1["blood_urea"].mean(),inplace=True)
f1[ "serum_creatinine"].fillna(f1["serum_creatinine"].mean(),inplace=True)
f1[ "sodium"].fillna(f1["sodium"].mean(),inplace=True)
f1[ "potassium"].fillna(f1["potassium"].mean(),inplace=True)
f1[ "hemoglobin"].fillna(f1["hemoglobin"].mean(),inplace=True)
f1[
"packed_cell_volume"].fillna(f1["packed_cell_volume"].mean(),inplace=True)
```

```
f1[
"white_blood_cell_count"].fillna(f1["white_blood_cell_count"].mean(),inplace=
True)
f1[
"red_blood_cell_count"].fillna(f1["red_blood_cell_count"].mean(),inplace=True
)
```

In [ ]:

```
#handle missing values
f1[ "age"].fillna(f1["age"].mode()[0],inplace=True)
f1[ "specific_gravity"].fillna(f1["specific_gravity"].mode()[0],inplace=True)
f1[ "albumin"].fillna(f1["albumin"].mode()[0],inplace=True)
f1[ "sugar"].fillna(f1["sugar"].mode()[0],inplace=True)
f1[ "red_blood_cells"].fillna(f1["red_blood_cells"].mode()[0],inplace=True)
f1[ "pus_cell"].fillna(f1["pus_cell"].mode()[0],inplace=True)
f1[ "pus_cell_clumps"].fillna(f1["pus_cell_clumps"].mode()[0],inplace=True)
f1[ "bacteria"].fillna(f1["bacteria"].mode()[0],inplace=True)
f1[ "diabetesmellitus"].fillna(f1["diabetesmellitus"].mode()[0],inplace=True)
f1[
"coronary_artery_disease"].fillna(f1["coronary_artery_disease"].mode()[0],inp
lace=True)
f1[ "appetite"].fillna(f1["appetite"].mode()[0],inplace=True)
f1[ "pedal_edema"].fillna(f1["pedal_edema"].mode()[0],inplace=True)
f1[ "anemia"].fillna(f1["anemia"].mode()[0],inplace=True)
f1[ "hypertension"].fillna(f1["hypertension"].mode()[0],inplace=True)
```

In [ ]:

```
f1.isnull().any()
```

Out[ ]:

```
age                         False
blood_pressure              False
specific_gravity            False
albumin                     False
sugar                       False
red_blood_cells             False
pus_cell                    False
pus_cell_clumps             False
bacteria                    False
blood_glucose_random        False
blood_urea                  False
serum_creatinine            False
sodium                      False
potassium                   False
hemoglobin                  False
packed_cell_volume          False
white_blood_cell_count      False
red_blood_cell_count        False
hypertension                False
diabetesmellitus            False
coronary_artery_disease     False
appetite                    False
pedal_edema                 False
anemia                      False
```

```
class                       False
dtype: bool
```

**LABEL ENCODING**

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
from collections import Counter
for i in cat_cols:
   print("LABEL ENCODING OF: ",i)
   encod=LabelEncoder()
   print(Counter(f1[i]))
   f1[i] = encod.fit_transform(f1[i])
   print(Counter(f1[i]))
   print("*"*100)
```

```
LABEL ENCODING OF:  red_blood_cells
Counter({'normal': 353, 'abnormal': 47})
Counter({1: 353, 0: 47})
****************************************************************************************************
LABEL ENCODING OF:  pus_cell
Counter({'normal': 324, 'abnormal': 76})
Counter({1: 324, 0: 76})
****************************************************************************************************
LABEL ENCODING OF:  pus_cell_clumps
Counter({'notpresent': 358, 'present': 42})
Counter({0: 358, 1: 42})
****************************************************************************************************
LABEL ENCODING OF:  bacteria
Counter({'notpresent': 378, 'present': 22})
Counter({0: 378, 1: 22})
****************************************************************************************************
LABEL ENCODING OF:  packed_cell_volume
Counter({38.88449848024316: 71, 52.0: 21, 41.0: 21, 44.0: 19, 48.0: 19, 40.0:
16, 43.0: 15, 45.0: 13, 42.0: 13, 32.0: 12, 36.0: 12, 33.0: 12, 28.0: 12, 50.
0: 12, 37.0: 11, 34.0: 11, 35.0: 9, 29.0: 9, 30.0: 9, 46.0: 9, 31.0: 8, 39.0:
7, 24.0: 7, 26.0: 6, 38.0: 5, 47.0: 4, 49.0: 4, 53.0: 4, 51.0: 4, 54.0: 4, 27
.0: 3, 22.0: 3, 25.0: 3, 23.0: 2, 19.0: 2, 16.0: 1, 14.0: 1, 18.0: 1, 17.0: 1
, 15.0: 1, 21.0: 1, 20.0: 1, 9.0: 1})
Counter({26: 71, 40: 21, 29: 21, 32: 19, 36: 19, 28: 16, 31: 15, 33: 13, 30:
13, 19: 12, 23: 12, 20: 12, 15: 12, 38: 12, 24: 11, 21: 11, 22: 9, 16: 9, 17:
9, 34: 9, 18: 8, 27: 7, 11: 7, 13: 6, 25: 5, 35: 4, 37: 4, 41: 4, 39: 4, 42:
4, 14: 3, 9: 3, 12: 3, 10: 2, 6: 2, 3: 1, 1: 1, 5: 1, 4: 1, 2: 1, 8: 1, 7: 1,
0: 1})
****************************************************************************************************
LABEL ENCODING OF:  white_blood_cell_count
Counter({8406.122448979591: 106, 9800.0: 11, 6700.0: 10, 9600.0: 9, 9200.0: 9
, 7200.0: 9, 6900.0: 8, 11000.0: 8, 5800.0: 8, 7800.0: 7, 9100.0: 7, 9400.0:
7, 7000.0: 7, 4300.0: 6, 6300.0: 6, 10700.0: 6, 10500.0: 6, 7500.0: 5, 6200.0
: 5, 8300.0: 5, 7900.0: 5, 8600.0: 5, 5600.0: 5, 10200.0: 5, 5000.0: 5, 8100.
```

```
0: 5, 9500.0: 5, 6000.0: 4, 8400.0: 4, 10300.0: 4, 7700.0: 4, 5500.0: 4, 1040
0.0: 4, 6800.0: 4, 6500.0: 4, 4700.0: 4, 7300.0: 3, 4500.0: 3, 6400.0: 3, 420
0.0: 3, 7400.0: 3, 8000.0: 3, 5400.0: 3, 3800.0: 2, 11400.0: 2, 5300.0: 2, 85
00.0: 2, 14600.0: 2, 7100.0: 2, 13200.0: 2, 9000.0: 2, 8200.0: 2, 15200.0: 2,
12400.0: 2, 12800.0: 2, 8800.0: 2, 5700.0: 2, 9300.0: 2, 6600.0: 2, 12100.0:
1, 12200.0: 1, 18900.0: 1, 21600.0: 1, 11300.0: 1, 11800.0: 1, 12500.0: 1, 11
900.0: 1, 12700.0: 1, 13600.0: 1, 14900.0: 1, 16300.0: 1, 10900.0: 1, 2200.0:
1, 11200.0: 1, 19100.0: 1, 12300.0: 1, 16700.0: 1, 2600.0: 1, 26400.0: 1, 490
0.0: 1, 12000.0: 1, 15700.0: 1, 4100.0: 1, 11500.0: 1, 10800.0: 1, 9900.0: 1,
5200.0: 1, 5900.0: 1, 9700.0: 1, 5100.0: 1})
Counter({42: 106, 54: 11, 25: 10, 52: 9, 48: 9, 30: 9, 27: 8, 63: 8, 17: 8, 3
5: 7, 47: 7, 50: 7, 28: 7, 5: 6, 21: 6, 60: 6, 59: 6, 33: 5, 20: 5, 40: 5, 36
: 5, 44: 5, 15: 5, 56: 5, 9: 5, 38: 5, 51: 5, 19: 4, 41: 4, 57: 4, 34: 4, 14:
4, 58: 4, 26: 4, 23: 4, 7: 4, 31: 3, 6: 3, 22: 3, 4: 3, 32: 3, 37: 3, 13: 3,
2: 2, 66: 2, 12: 2, 43: 2, 80: 2, 29: 2, 78: 2, 46: 2, 39: 2, 82: 2, 74: 2, 7
7: 2, 45: 2, 16: 2, 49: 2, 24: 2, 71: 1, 72: 1, 86: 1, 88: 1, 65: 1, 68: 1, 7
5: 1, 69: 1, 76: 1, 79: 1, 81: 1, 84: 1, 62: 1, 0: 1, 64: 1, 87: 1, 73: 1, 85
: 1, 1: 1, 89: 1, 8: 1, 70: 1, 83: 1, 3: 1, 67: 1, 61: 1, 55: 1, 11: 1, 18: 1
, 53: 1, 10: 1})
**************************************************************************
**********************
LABEL ENCODING OF:  red_blood_cell_count
Counter({4.707434944237917: 131, 5.2: 18, 4.5: 16, 4.9: 14, 4.7: 11, 3.9: 10,
5.0: 10, 4.8: 10, 4.6: 9, 3.4: 9, 3.7: 8, 6.1: 8, 5.5: 8, 5.9: 8, 3.8: 7, 5.4
: 7, 5.8: 7, 5.3: 7, 4.0: 6, 4.3: 6, 4.2: 6, 5.6: 6, 4.4: 5, 3.2: 5, 4.1: 5,
6.2: 5, 5.1: 5, 6.4: 5, 5.7: 5, 6.5: 5, 3.6: 4, 6.0: 4, 6.3: 4, 3.5: 3, 3.3:
3, 3.0: 3, 2.6: 2, 2.8: 2, 2.5: 2, 3.1: 2, 2.1: 2, 2.9: 2, 2.7: 2, 2.3: 1, 8.
0: 1, 2.4: 1})
Counter({26: 131, 31: 18, 23: 16, 28: 14, 25: 11, 17: 10, 29: 10, 27: 10, 24:
9, 12: 9, 15: 8, 40: 8, 34: 8, 38: 8, 16: 7, 33: 7, 37: 7, 32: 7, 18: 6, 21:
6, 20: 6, 35: 6, 22: 5, 10: 5, 19: 5, 41: 5, 30: 5, 43: 5, 36: 5, 44: 5, 14:
4, 39: 4, 42: 4, 13: 3, 11: 3, 8: 3, 4: 2, 6: 2, 3: 2, 9: 2, 0: 2, 7: 2, 5: 2
, 1: 1, 45: 1, 2: 1})
**************************************************************************
**********************
LABEL ENCODING OF:  hypertension
Counter({'no': 253, 'yes': 147})
Counter({0: 253, 1: 147})
**************************************************************************
**********************
LABEL ENCODING OF:  diabetesmellitus
Counter({'no': 263, 'yes': 137})
Counter({0: 263, 1: 137})
**************************************************************************
**********************
LABEL ENCODING OF:  coronary_artery_disease
Counter({'no': 366, 'yes': 34})
Counter({0: 366, 1: 34})
**************************************************************************
**********************
LABEL ENCODING OF:  appetite
Counter({'good': 318, 'poor': 82})
Counter({0: 318, 1: 82})
```

```
**************************************************************************
**********************
LABEL ENCODING OF:  pedal_edema
Counter({'no': 324, 'yes': 76})
Counter({0: 324, 1: 76})
**************************************************************************
**********************
LABEL ENCODING OF:  anemia
Counter({'no': 340, 'yes': 60})
Counter({0: 340, 1: 60})
**************************************************************************
**********************
LABEL ENCODING OF:  class
Counter({'ckd': 250, 'notckd': 150})
Counter({0: 250, 1: 150})
**************************************************************************
**********************
```

**SPLITTING THE DATA IN INDEPENDENT AND DEPENDENT VARIABLES**

In [ ]:

```
xcols=['age', 'blood_pressure', 'specific_gravity', 'albumin', 'sugar',
       'red_blood_cells', 'pus_cell', 'pus_cell_clumps', 'bacteria',
       'blood_glucose_random', 'blood_urea', 'serum_creatinine', 'sodium',
       'potassium', 'hemoglobin', 'packed_cell_volume',
       'white_blood_cell_count', 'red_blood_cell_count', 'hypertension',
       'diabetesmellitus', 'coronary_artery_disease',
'appetite','pedal_edema','anemia']

x=pd.DataFrame(f1, columns=xcols)
y=pd.DataFrame(f1, columns=['class'])
print(x.shape)
print(y.shape)
```

```
(400, 24)
(400, 1)
```

**SPLIT INTO TRAIN AND TEST SET**

In [ ]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,
random_state=2)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(280, 24)
(280, 1)
(120, 24)
(120, 1)
```

**MODEL BUILDING**

**a) LOGISTIC REGRESSION**

In [ ]:

```
from sklearn. linear_model import LogisticRegression
logreg= LogisticRegression()
logreg.fit(x_train, y_train.values.ravel())
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

Out[ ]:

```
LogisticRegression()
```

In [ ]:

```
x_test.head()
```

Out[ ]:

| | age | blood_pressure | specific_gravity | albumin | sugar | red_blood_cells | pus_cell | pus_cell_clumps | bacteria | blood_glucose_random | ... | hemoglobin | packed_cell_volume | white_blood_cell_count | red_blood_cell_count | hypertension | diabetesmellitus | coronary_artery_disease | appetite | pedal_edema | anemia |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **94** | 65.0 | 70.0 | 1.010 | 0.0 | 0.0 | 1 | 1 | 0 | 0 | 93.0 | ... | 11.6 | 23 | 69 | 17 | 0 | 1 | 0 | 0 | 0 | 0 |
| **32** | 61.0 | 90.0 | 1.010 | 1.0 | 1.0 | 1 | 1 | 0 | 0 | 159.0 | ... | 11.3 | 21 | 52 | 18 | 1 | 1 | 0 | 1 | 0 | 0 |
| **225** | 60.0 | 90.0 | 1.010 | 3.0 | 5.0 | 0 | 1 | 0 | 1 | 490.0 | ... | 11.5 | 22 | 70 | 23 | 1 | 1 | 0 | 0 | 0 | 0 |
| **157** | 62.0 | 70.0 | 1.025 | 3.0 | 0.0 | 1 | 0 | 0 | 0 | 122.0 | ... | 12.6 | 27 | 36 | 17 | 1 | 1 | 0 | 0 | 0 | 0 |

| | age | blood_pressure | specific_gravity | albumin | sugar | red_blood_cells | pus_cell | pus_cell_clumps | bacteria | blood_glucose_random | ... | hemoglobin | packed_cell_volume | white_blood_cell_count | red_blood_cell_count | hypertension | diabetesmellitus | coronary_artery_disease | appetite | pedal_edema | anemia |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **356** | 34.0 | 70.0 | 1.025 | 0.0 | 0.0 | 1 | 1 | 0 | 0 | 87.0 | ... | 17.1 | 35 | 32 | 40 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 24 columns

```
Warning: Total number of columns (24) exceeds max_columns (20) limiting to fi
rst (20) columns.
```

In [ ]:

```
x_test.iloc[4,-6:]
```

Out[ ]:

```
hypertension               0.0
diabetesmellitus           0.0
coronary_artery_disease    0.0
appetite                   0.0
pedal_edema                0.0
anemia                     0.0
Name: 356, dtype: float64
```

In [ ]:

```
logreg.predict(x_test)
```

Out[ ]:

```
array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 1, 1, 0])
```

In [ ]:

```
#training score
logreg.score(x_train,y_train)
```

Out[ ]:

```
0.9892857142857143
```

In [ ]:

```
#testing accuracy
logreg.score(x_test,y_test)
```

Out[ ]:

```
0.9583333333333334
```

In [ ]:

```
y_pred_log=logreg.predict(x_test)
```

In [ ]:

```
#accuracy
from sklearn.metrics import accuracy_score
log_acc=accuracy_score(y_test, y_pred_log)
print('Accuracy: {0:.3f}'.format(log_acc))
```

```
Accuracy: 0.958
```

**MODEL EVALUATION**

In [ ]:

```
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import roc_curve, auc
```

In [ ]:

```
mae_log=mean_absolute_error(y_test, y_pred_log)
```

In [ ]:

```
mse_log=mean_squared_error(y_test, y_pred_log, squared=False)
```

In [ ]:

```
clsrep_log=classification_report(y_test, y_pred_log)
```

In [ ]:

```
print("LOGISTIC REGRESSION:\n")
print('Accuracy             : {0:.3f}'.format(log_acc))
print("MAE                  : ",mae_log)
print("MSE                  : ",mse_log)
print("Classification Report: ", clsrep_log)
#print(classification_report(y_test, y_pred_log))
```

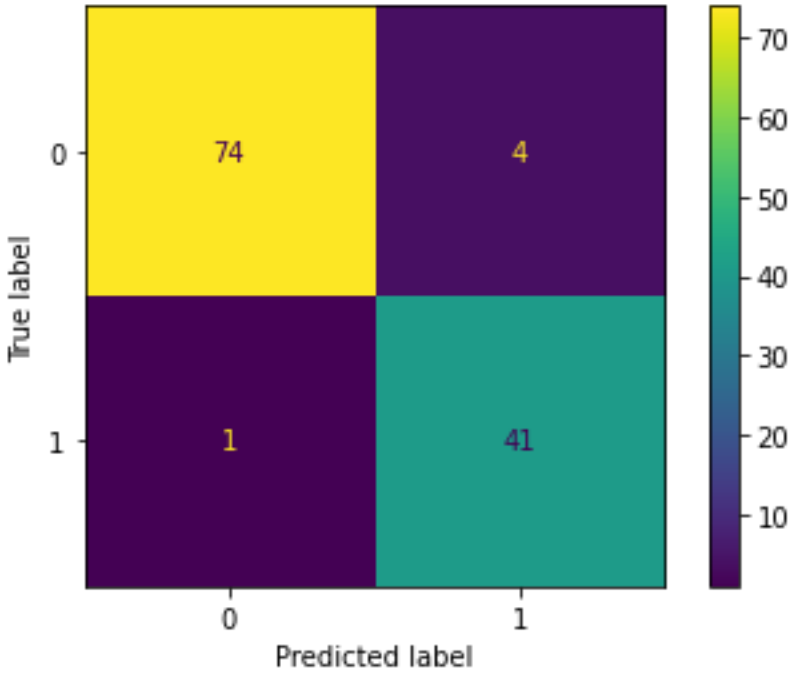
```
LOGISTIC REGRESSION:

Accuracy             : 0.958
MAE                  :  0.041666666666666664
MSE                  :  0.2041241452319315
Classification Report:                precision    recall  f1-score   support

           0       0.99      0.95      0.97        78
           1       0.91      0.98      0.94        42

    accuracy                           0.96       120
   macro avg       0.95      0.96      0.95       120
weighted avg       0.96      0.96      0.96       120
```

In [ ]:

```
#confusion matrix
cm_log= confusion_matrix(y_test, y_pred_log)
disp_log=
ConfusionMatrixDisplay(confusion_matrix=cm_log,display_labels=logreg.classes_
)
disp_log.plot()
print("Confusion Matrix for Logistic Regression: ")
```

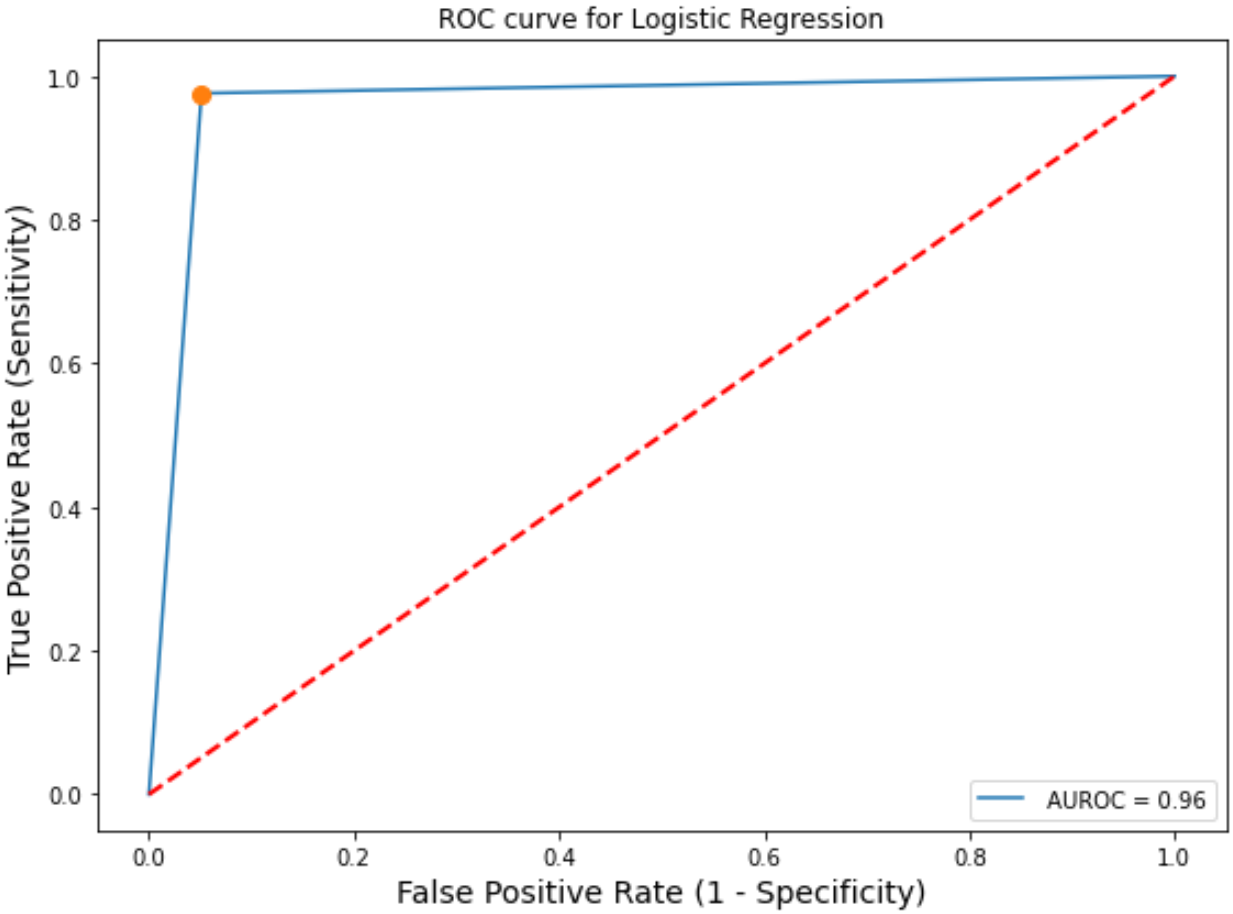
```
plt.show()
```

```
Confusion Matrix for Logistic Regression:
```




In [ ]:

```
#roc curve
fig, (ax2) = plt.subplots(figsize = (8,6))
        #roc-curve
fpr, tpr, thresholds_roc = roc_curve(y_test,y_pred_log)
roc_auc = auc(fpr,tpr)
ax2.plot(fpr,tpr, label = " AUROC = {:0.2f}".format(roc_auc))
ax2.plot([0,1], [0,1], 'r', linestyle = "--", lw = 2)
ax2.set_xlabel("False Positive Rate", fontsize = 14)
ax2.set_ylabel("True Positive Rate", fontsize = 14)
ax2.set_title("ROC Curve", fontsize = 18)
ax2.legend(loc = 'best')
plt.title('ROC curve for Logistic Regression')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
#find default threshold
close_default = np.argmin(np.abs(thresholds_roc - 0.5))
ax2.plot(fpr[close_default], tpr[close_default], 'o', markersize = 8)
plt.tight_layout()
```




In [ ]:

```
import pickle as pl
pl.dump(logreg,open('LinearReg.pkl','wb'))
```