In [1]:
```python
# KNN Classificationhttps://meet.google.com/hdp-wmjw-bmc
import pandas as pd #data manipulation
import numpy as np #numerical functions
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report ,accuracy_score
```

p1

data ----> tarin and test

p2 CV :

data ---> tarin and test

train_data-> data1,data2,data3,data4,data5

k=1

m1->data1,data2,data3,data4 :::: test-> data5 #91

m2->data1,data2,data3,data5 :::: test-> data4 #89

m3->data1,data2,data4,data5 :::: test-> data3 #80

m4->data1,data3,data4,data5 :::: test-> data2 #82

m5->data2,data3,data4,data5 :::: test-> data1 #87

overall =(91+89+80+82+87)/5

k=3

m1->data1,data2,data3,data4 :::: test-> data5 #91

m2->data1,data2,data3,data5 :::: test-> data4 #89

m3->data1,data2,data4,data5 :::: test-> data3 #80

m4->data1,data3,data4,data5 :::: test-> data2 #82

m5->data2,data3,data4,data5 :::: test-> data1 #87

overall =(91+89+80+82+87)/5

k=5

m1->data1,data2,data3,data4 :::: test-> data5 #91

m2->data1,data2,data3,data5 :::: test-> data4 #89

m3->data1,data2,data4,data5 :::: test-> data3 #80

m4->data1,data3,data4,data5 :::: test-> data2 #82

m5->data2,data3,data4,data5 :::: test-> data1 #87

overall =(91+89+80+82+87)/5

k=7

m1->data1,data2,data3,data4 :::: test-> data5 #91

m2->data1,data2,data3,data5 :::: test-> data4 #89

m3->data1,data2,data4,data5 :::: test-> data3 #80

m4->data1,data3,data4,data5 :::: test-> data2 #82

m5->data2,data3,data4,data5 :::: test-> data1 #87

overall =(91+89+80+82+87)/5

best k value is that one for which we get the best overall score and final model should be created for that value of k using entire training data

In [2]:

```python
from sklearn.datasets import load_diabetes
data=load_diabetes()
print(data.DESCR)
print(data)
```

```
.. _diabetes_dataset:

Diabetes dataset
----------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n =
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

:Number of Instances: 442

:Number of Attributes: First 10 columns are numeric predictive values

:Target: Column 11 is a quantitative measure of disease progression one year after baseline

:Attribute Information:
    - age     age in years
    - sex
    - bmi     body mass index
    - bp      average blood pressure
    - s1      tc, total serum cholesterol
    - s2      ldl, low-density lipoproteins
    - s3      hdl, high-density lipoproteins
    - s4      tch, total cholesterol / HDL
    - s5      ltg, possibly log of serum triglycerides level
    - s6      glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and scaled by the standard deviation times the
square root of `n_samples` (i.e. the sum of squares of each column totals 1).

Source URL:
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," Annals of St
atistics (with discussion), 407-499.
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)

{'data': array([[ 0.03807591,  0.05068012,  0.06169621, ..., -0.00259226,
         0.01990749, -0.01764613],
       [-0.00188202, -0.04464164, -0.05147406, ..., -0.03949338,
        -0.06833155, -0.09220405],
       [ 0.08529891,  0.05068012,  0.04445121, ..., -0.00259226,
         0.00286131, -0.02593034],
       ...,
       [ 0.04170844,  0.05068012, -0.01590626, ..., -0.01107952,
        -0.04688253,  0.01549073],
       [-0.04547248, -0.04464164,  0.03906215, ...,  0.02655962,
         0.04452873, -0.02593034],
       [-0.04547248, -0.04464164, -0.0730303 , ..., -0.03949338,
        -0.00422151,  0.00306441]]), 'target': array([151.,  75., 141., 206., 135.,  97., 138.,  63., 110., 310.
, 101.,
        69., 179., 185., 118., 171., 166., 144.,  97., 168.,  68.,  49.,
        68., 245., 184., 202., 137.,  85., 131., 283., 129.,  59., 341.,
        87.,  65., 102., 265., 276., 252.,  90., 100.,  55.,  61.,  92.,
       259.,  53., 190., 142.,  75., 142., 155., 225.,  59., 104., 182.,
       128.,  52.,  37., 170., 170.,  61., 144.,  52., 128.,  71., 163.,
       150.,  97., 160., 178.,  48., 270., 202., 111.,  85.,  42., 170.,
       200., 252., 113., 143.,  51.,  52., 210.,  65., 141.,  55., 134.,
        42., 111.,  98., 164.,  48.,  96.,  90., 162., 150., 279.,  92.,
        83., 128., 102., 302., 198.,  95.,  53., 134., 144., 232.,  81.,
       104.,  59., 246., 297., 258., 229., 275., 281., 179., 200., 200.,
       173., 180.,  84., 121., 161.,  99., 109., 115., 268., 274., 158.,
       107.,  83., 103., 272.,  85., 280., 336., 281., 118., 317., 235.,
        60., 174., 259., 178., 128.,  96., 126., 288.,  88., 292.,  71.,
       197., 186.,  25.,  84.,  96., 195.,  53., 217., 172., 131., 214.,
        59.,  70., 220., 268., 152.,  47.,  74., 295., 101., 151., 127.,
       237., 225.,  81., 151., 107.,  64., 138., 185., 265., 101., 137.,
       143., 141.,  79., 292., 178.,  91., 116.,  86., 122.,  72., 129.,
       142.,  90., 158.,  39., 196., 222., 277.,  99., 196., 202., 155.,
        77., 191.,  70.,  73.,  49.,  65., 263., 248., 296., 214., 185.,
        78.,  93., 252., 150.,  77., 208.,  77., 108., 160.,  53., 220.,
       154., 259.,  90., 246., 124.,  67.,  72., 257., 262., 275., 177.,
        71.,  47., 187., 125.,  78.,  51., 258., 215., 303., 243.,  91.,
       150., 310., 153., 346.,  63.,  89.,  50.,  39., 103., 308., 116.,
```

```
       145.,  74.,  45., 115., 264.,  87., 202., 127., 182., 241.,  66.,
        94., 283.,  64., 102., 200., 265.,  94., 230., 181., 156., 233.,
        60., 219.,  80.,  68., 332., 248.,  84., 200.,  55.,  85.,  89.,
        31., 129.,  83., 275.,  65., 198., 236., 253., 124.,  44., 172.,
       114., 142., 109., 180., 144., 163., 147.,  97., 220., 190., 109.,
       191., 122., 230., 242., 248., 249., 192., 131., 237.,  78., 135.,
       244., 199., 270., 164.,  72.,  96., 306.,  91., 214.,  95., 216.,
       263., 178., 113., 200., 139., 139.,  88., 148.,  88., 243.,  71.,
        77., 109., 272.,  60.,  54., 221.,  90., 311., 281., 182., 321.,
        58., 262., 206., 233., 242., 123., 167.,  63., 197.,  71., 168.,
       140., 217., 121., 235., 245.,  40.,  52., 104., 132.,  88.,  69.,
       219.,  72., 201., 110.,  51., 277.,  63., 118.,  69., 273., 258.,
        43., 198., 242., 232., 175.,  93., 168., 275., 293., 281.,  72.,
       140., 189., 181., 209., 136., 261., 113., 131., 174., 257.,  55.,
        84.,  42., 146., 212., 233.,  91., 111., 152., 120.,  67., 310.,
        94., 183.,  66., 173.,  72.,  49.,  64.,  48., 178., 104., 132.,
       220.,  57.]), 'frame': None, 'DESCR': '.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTen
baseline variables, age, sex, body mass index, average blood\npressure, and six blood serum measurements were ob
tained for each of n =\n442 diabetes patients, as well as the response of interest, a\nquantitative measure of d
isease progression one year after baseline.\n\n**Data Set Characteristics:**\n\n:Number of Instances: 442\n\n:Nu
mber of Attributes: First 10 columns are numeric predictive values\n\n:Target: Column 11 is a quantitative measu
re of disease progression one year after baseline\n\n:Attribute Information:\n    - age     age in years\n    -
sex\n    - bmi     body mass index\n    - bp      average blood pressure\n    - s1      tc, total serum choleste
rol\n    - s2      ldl, low-density lipoproteins\n    - s3      hdl, high-density lipoproteins\n    - s4      tc
h, total cholesterol / HDL\n    - s5      ltg, possibly log of serum triglycerides level\n    - s6      glu, blo
od sugar level\n\nNote: Each of these 10 feature variables have been mean centered and scaled by the standard de
viation times the square root of `n_samples` (i.e. the sum of squares of each column totals 1).\n\nSource URL:\n
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html\n\nFor more information see:\nBradley Efron, Trevor Ha
stie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," Annals of Statistics (with discussio
n), 407-499.\n(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)\n', 'feature_names': ['age', 's
ex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6'], 'data_filename': 'diabetes_data_raw.csv.gz', 'target_file
name': 'diabetes_target.csv.gz', 'data_module': 'sklearn.datasets.data'}
```


In [3]:
```
ls
```

```
 Volume in drive C is Windows-SSD
 Volume Serial Number is 08B4-FA6D

 Directory of C:\Users\Agnel Sharon Jerald\Machine learning

03-12-2025  09:43    <DIR>          .
24-11-2025  16:52    <DIR>          ..
03-12-2025  09:26    <DIR>          .ipynb_checkpoints
01-12-2025  09:16            29,822 claimants.csv
01-12-2025  10:02           128,594 Claimants_classification.ipynb
23-10-2025  10:52           252,200 Data Visualisations.ipynb
03-12-2025  09:25            10,523 KNN_updated.ipynb
24-11-2025  16:56            95,226 Linear_Regression.ipynb
18-11-2025  09:21            57,420 Machine Learning basics.ipynb
22-10-2025  17:51            26,046 Numpy.ipynb
03-12-2025  09:43            23,279 pima-indians-diabetes.csv
23-10-2025  16:20            54,750 Practice libraries.ipynb
               9 File(s)        677,860 bytes
               3 Dir(s)  113,203,838,976 bytes free
```

In [4]:
```
filename = 'pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pd.read_csv(filename, names=names)
dataframe
```

Out[4]:

	preg	plas	pres	skin	test	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [5]:
```
X = dataframe.iloc[:,0:-1]
```

```
Y = dataframe.iloc[:,-1]
```

In [6]:
```
dataframe["class"].value_counts()
```

Out[6]:
```
class
0    500
1    268
Name: count, dtype: int64
```

In [7]:
```
X=(X-X.min(axis=0))/(X.max(axis=0)-X.min(axis=0))
```

In [8]:
```
X.round(2)
```

Out[8]:

	preg	plas	pres	skin	test	mass	pedi	age
0	0.35	0.74	0.59	0.35	0.00	0.50	0.23	0.48
1	0.06	0.43	0.54	0.29	0.00	0.40	0.12	0.17
2	0.47	0.92	0.52	0.00	0.00	0.35	0.25	0.18
3	0.06	0.45	0.54	0.23	0.11	0.42	0.04	0.00
4	0.00	0.69	0.33	0.35	0.20	0.64	0.94	0.20
...	...	...	...	...	...	...	...	...
763	0.59	0.51	0.62	0.48	0.21	0.49	0.04	0.70
764	0.12	0.61	0.57	0.27	0.00	0.55	0.11	0.10
765	0.29	0.61	0.59	0.23	0.13	0.39	0.07	0.15
766	0.06	0.63	0.49	0.00	0.00	0.45	0.12	0.43
767	0.06	0.47	0.57	0.31	0.00	0.45	0.10	0.03

768 rows × 8 columns

In [9]:
```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3,random_state=19)
```

In [10]:
```
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
```

```
(537, 8) (231, 8) (537,) (231,)
```

In [11]:
```
# model of train data
model = KNeighborsClassifier(n_neighbors=5)
model.fit(x_train,y_train)
pred=model.predict(x_train)
print(classification_report(y_train,pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.90      0.87       346
           1       0.80      0.67      0.73       191

    accuracy                           0.82       537
   macro avg       0.81      0.79      0.80       537
weighted avg       0.82      0.82      0.82       537
```

In [12]:
```
# model of test data
model = KNeighborsClassifier(n_neighbors=5)
model.fit(x_train,y_train)
pred=model.predict(x_test)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.88      0.84       154
           1       0.69      0.56      0.62        77

    accuracy                           0.77       231
   macro avg       0.75      0.72      0.73       231
weighted avg       0.76      0.77      0.76       231
```

model of test data

model = KNeighborsClassifier(n_neighbors=15) model.fit(x_train,y_train) pred=model.predict(x_test) accuracy_score(y_test,pred) print(classification_report(y_test,pred))

In [13]:
```
n_neighbors=[2*i+1 for i in range(0,27)]
for i in n_neighbors:
```

```
    print(f"i value = {i}")
    model = KNeighborsClassifier(n_neighbors=i)
    model.fit(x_train,y_train)
    pred=model.predict(x_test)
    acc = accuracy_score(y_test,pred)
    print("acc=",acc)
    print(classification_report(y_test,pred))
```

```
i value = 1
acc= 0.7229437229437229
              precision    recall  f1-score   support

           0       0.78      0.82      0.80       154
           1       0.59      0.53      0.56        77

    accuracy                           0.72       231
   macro avg       0.69      0.68      0.68       231
weighted avg       0.72      0.72      0.72       231

i value = 3
acc= 0.7272727272727273
              precision    recall  f1-score   support

           0       0.78      0.83      0.80       154
           1       0.61      0.52      0.56        77

    accuracy                           0.73       231
   macro avg       0.69      0.68      0.68       231
weighted avg       0.72      0.73      0.72       231

i value = 5
acc= 0.7705627705627706
              precision    recall  f1-score   support

           0       0.80      0.88      0.84       154
           1       0.69      0.56      0.62        77

    accuracy                           0.77       231
   macro avg       0.75      0.72      0.73       231
weighted avg       0.76      0.77      0.76       231

i value = 7
acc= 0.7272727272727273
              precision    recall  f1-score   support

           0       0.77      0.84      0.80       154
           1       0.61      0.49      0.55        77

    accuracy                           0.73       231
   macro avg       0.69      0.67      0.68       231
weighted avg       0.72      0.73      0.72       231

i value = 9
acc= 0.7532467532467533
              precision    recall  f1-score   support

           0       0.79      0.86      0.82       154
           1       0.66      0.53      0.59        77

    accuracy                           0.75       231
   macro avg       0.72      0.70      0.71       231
weighted avg       0.75      0.75      0.75       231

i value = 11
acc= 0.7792207792207793
              precision    recall  f1-score   support

           0       0.79      0.90      0.84       154
           1       0.73      0.53      0.62        77

    accuracy                           0.78       231
   macro avg       0.76      0.72      0.73       231
weighted avg       0.77      0.78      0.77       231

i value = 13
acc= 0.7705627705627706
              precision    recall  f1-score   support

           0       0.80      0.88      0.84       154
           1       0.70      0.55      0.61        77

    accuracy                           0.77       231
   macro avg       0.75      0.71      0.73       231
weighted avg       0.76      0.77      0.76       231
```

```
i value = 15
acc= 0.7748917748917749
              precision    recall  f1-score   support

           0       0.80      0.89      0.84       154
           1       0.71      0.55      0.62        77

    accuracy                           0.77       231
   macro avg       0.75      0.72      0.73       231
weighted avg       0.77      0.77      0.77       231

i value = 17
acc= 0.7748917748917749
              precision    recall  f1-score   support

           0       0.79      0.90      0.84       154
           1       0.73      0.52      0.61        77

    accuracy                           0.77       231
   macro avg       0.76      0.71      0.72       231
weighted avg       0.77      0.77      0.76       231

i value = 19
acc= 0.7532467532467533
              precision    recall  f1-score   support

           0       0.76      0.92      0.83       154
           1       0.72      0.43      0.54        77

    accuracy                           0.75       231
   macro avg       0.74      0.67      0.68       231
weighted avg       0.75      0.75      0.73       231

i value = 21
acc= 0.7662337662337663
              precision    recall  f1-score   support

           0       0.77      0.93      0.84       154
           1       0.76      0.44      0.56        77

    accuracy                           0.77       231
   macro avg       0.76      0.69      0.70       231
weighted avg       0.76      0.77      0.75       231

i value = 23
acc= 0.7575757575757576
              precision    recall  f1-score   support

           0       0.76      0.92      0.84       154
           1       0.73      0.43      0.54        77

    accuracy                           0.76       231
   macro avg       0.75      0.68      0.69       231
weighted avg       0.75      0.76      0.74       231

i value = 25
acc= 0.7705627705627706
              precision    recall  f1-score   support

           0       0.76      0.95      0.85       154
           1       0.80      0.42      0.55        77

    accuracy                           0.77       231
   macro avg       0.78      0.68      0.70       231
weighted avg       0.78      0.77      0.75       231

i value = 27
acc= 0.7619047619047619
              precision    recall  f1-score   support

           0       0.76      0.94      0.84       154
           1       0.76      0.42      0.54        77

    accuracy                           0.76       231
   macro avg       0.76      0.68      0.69       231
weighted avg       0.76      0.76      0.74       231

i value = 29
acc= 0.7575757575757576
              precision    recall  f1-score   support

           0       0.76      0.93      0.84       154
```

```
           1       0.74      0.42      0.53        77

    accuracy                           0.76       231
   macro avg       0.75      0.67      0.68       231
weighted avg       0.76      0.76      0.74       231

i value = 31
acc= 0.7575757575757576
              precision    recall  f1-score   support

           0       0.75      0.95      0.84       154
           1       0.78      0.38      0.51        77

    accuracy                           0.76       231
   macro avg       0.77      0.66      0.67       231
weighted avg       0.76      0.76      0.73       231

i value = 33
acc= 0.7402597402597403
              precision    recall  f1-score   support

           0       0.74      0.94      0.83       154
           1       0.73      0.35      0.47        77

    accuracy                           0.74       231
   macro avg       0.74      0.64      0.65       231
weighted avg       0.74      0.74      0.71       231

i value = 35
acc= 0.7619047619047619
              precision    recall  f1-score   support

           0       0.75      0.95      0.84       154
           1       0.81      0.38      0.51        77

    accuracy                           0.76       231
   macro avg       0.78      0.67      0.68       231
weighted avg       0.77      0.76      0.73       231

i value = 37
acc= 0.7575757575757576
              precision    recall  f1-score   support

           0       0.75      0.95      0.84       154
           1       0.78      0.38      0.51        77

    accuracy                           0.76       231
   macro avg       0.77      0.66      0.67       231
weighted avg       0.76      0.76      0.73       231

i value = 39
acc= 0.7575757575757576
              precision    recall  f1-score   support

           0       0.75      0.95      0.84       154
           1       0.78      0.38      0.51        77

    accuracy                           0.76       231
   macro avg       0.77      0.66      0.67       231
weighted avg       0.76      0.76      0.73       231

i value = 41
acc= 0.7575757575757576
              precision    recall  f1-score   support

           0       0.75      0.95      0.84       154
           1       0.78      0.38      0.51        77

    accuracy                           0.76       231
   macro avg       0.77      0.66      0.67       231
weighted avg       0.76      0.76      0.73       231

i value = 43
acc= 0.7489177489177489
              precision    recall  f1-score   support

           0       0.75      0.94      0.83       154
           1       0.74      0.38      0.50        77

    accuracy                           0.75       231
   macro avg       0.75      0.66      0.67       231
weighted avg       0.75      0.75      0.72       231
```

```
i value = 45
acc= 0.7575757575757576
              precision    recall  f1-score   support

           0       0.75      0.95      0.84       154
           1       0.78      0.38      0.51        77

    accuracy                           0.76       231
   macro avg       0.77      0.66      0.67       231
weighted avg       0.76      0.76      0.73       231

i value = 47
acc= 0.7619047619047619
              precision    recall  f1-score   support

           0       0.76      0.95      0.84       154
           1       0.79      0.39      0.52        77

    accuracy                           0.76       231
   macro avg       0.77      0.67      0.68       231
weighted avg       0.77      0.76      0.73       231

i value = 49
acc= 0.7575757575757576
              precision    recall  f1-score   support

           0       0.76      0.94      0.84       154
           1       0.77      0.39      0.52        77

    accuracy                           0.76       231
   macro avg       0.76      0.67      0.68       231
weighted avg       0.76      0.76      0.73       231

i value = 51
acc= 0.7619047619047619
              precision    recall  f1-score   support

           0       0.76      0.95      0.84       154
           1       0.79      0.39      0.52        77

    accuracy                           0.76       231
   macro avg       0.77      0.67      0.68       231
weighted avg       0.77      0.76      0.73       231

i value = 53
acc= 0.7575757575757576
              precision    recall  f1-score   support

           0       0.75      0.95      0.84       154
           1       0.78      0.38      0.51        77

    accuracy                           0.76       231
   macro avg       0.77      0.66      0.67       231
weighted avg       0.76      0.76      0.73       231
```

In [14]:

```
(2*(0.76*0.94))/(0.76+0.94)
```

Out[14]:

```
0.8404705882352941
```

In [15]:

```
n_neighbors=[2*i+1 for i in range(0,27)]
n_neighbors
```

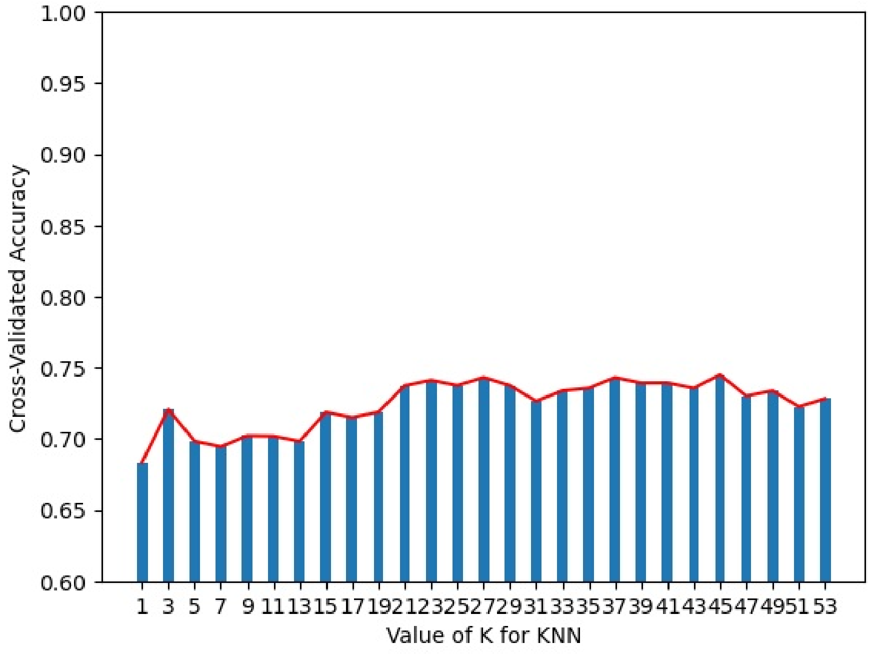
Out[15]:
```
[1,
 3,
 5,
 7,
 9,
 11,
 13,
 15,
 17,
 19,
 21,
 23,
 25,
 27,
 29,
 31,
 33,
 35,
 37,
 39,
 41,
 43,
 45,
 47,
 49,
 51,
 53]
```

In [ ]:
```
# print(grid.best_score_)
# print(grid.best_params_ )
```

Visualizing the CV results

In [16]:
```
import matplotlib.pyplot as plt
%matplotlib inline
# choose k between 1 to 55
k_range = [2*i+1 for i in range(0,27)]
k_scores = []
# use iteration to caclulator different k in models, then return the average accuracy based on the cross valida
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn,x_train , y_train, cv=10)
    k_scores.append(scores.mean())
# plot to see clearly
plt.bar(k_range, k_scores)
plt.plot(k_range, k_scores,color="red")

plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.xticks(k_range)
plt.ylim(0.6,1)
plt.show()
```



In [17]:
```
np.argmax(k_scores)
```

Out[17]: np.int64(22)

In [18]:
```
k_range[22]
```

Out[18]: 45

In [19]:
```
knn = KNeighborsClassifier(n_neighbors=45)
scores = cross_val_score(knn,x_train , y_train, cv=10)
scores.mean()
```

Out[19]: np.float64(0.7449336128581412)

In [20]:
```
from sklearn.linear_model import LogisticRegression
lrmodel = LogisticRegression()
lrmodel.fit(x_train,y_train)
pred = lrmodel.predict(x_test)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.90      0.83       154
           1       0.71      0.48      0.57        77

    accuracy                           0.76       231
   macro avg       0.74      0.69      0.70       231
weighted avg       0.75      0.76      0.75       231
```

In [21]:
```
k_scores
```

Out[21]:
```
[np.float64(0.6833682739343117),
 np.float64(0.7206149545772187),
 np.float64(0.698252969951083),
 np.float64(0.6944095038434661),
 np.float64(0.7019916142557652),
 np.float64(0.7017470300489168),
 np.float64(0.6981830887491264),
 np.float64(0.7186233403214536),
 np.float64(0.714884696016771 5),
 np.float64(0.7187281621243885),
 np.float64(0.7373165618448637),
 np.float64(0.7411250873515024),
 np.float64(0.737456324248777),
 np.float64(0.7430468204053109),
 np.float64(0.7374563242487772),
 np.float64(0.7263102725366877),
 np.float64(0.7337526205450733),
 np.float64(0.735569531795947),
 np.float64(0.7429769392033542),
 np.float64(0.7391684136967156),
 np.float64(0.7392732354996505),
 np.float64(0.735569531795947),
 np.float64(0.7449336128581412),
 np.float64(0.7300489168413697),
 np.float64(0.733717679944095),
 np.float64(0.7224668064290706),
 np.float64(0.7280573025856045)]
```