KNN

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sb

#regression
boston=pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Boston.csv')

#classification
iris=pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/IRIS.csv')

boston.head()
```

	CRIM	ZN	INDUS	CHAS	NX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LST
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.9
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.1
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.0
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.9
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.3

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa

```
boston.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

```
X=boston.drop(['MEDV'], axis=1)
```

```
X.head()
```

	CRIM	ZN	INDUS	CHAS	NX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LST
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.9
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.(
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.9
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.3

```
y=boston['MEDV']
```

```
iris.columns
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

```
X=iris.drop(['species'], axis=1)
```

```
y=iris['species']
```

```
y
```

```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
```

```
3         Iris-setosa
4         Iris-setosa
            ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: species, Length: 150, dtype: object
```

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

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=2529 )
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((105, 4), (45, 4), (105,), (45,))
```

```
X_train
```

	sepal_length	sepal_width	petal_length	petal_width
84	5.4	3.0	4.5	1.5
13	4.3	3.0	1.1	0.1

```
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
ss=StandardScaler()
```

```
X_train=ss.fit_transform(X_train)
```

```
X_train
```

```
array([[-0.97514097,  0.62399509, -1.31675446, -1.28941139],
       [-1.7587364 , -0.27972194, -1.31675446, -1.28941139],
       [ 0.72264911, -0.73158046,  0.77175337,  0.91650626],
       [-1.36693868,  0.39806584, -1.19741116, -1.28941139],
       [ 1.24504606,  0.17213658,  0.65241006,  0.5028967 ],
       [ 0.3308514 , -1.86122675,  0.2347085 , -0.18645256],
       [ 2.55103844, -0.05379268,  1.4878132 ,  1.60585552],
       [ 0.98384759, -0.05379268,  1.30879824,  1.46798567],
       [ 1.11444683, -0.05379268,  0.47339511,  0.36502685],
       [ 0.85324835, -0.27972194,  0.41372346,  0.227157  ],
       [ 0.06965293, -0.73158046,  0.89109667,  1.05437611],
       [-0.06094631, -0.5056512 ,  0.29438015,  0.227157  ],
       [ 1.89804225,  0.39806584,  1.42814154,  0.91650626],
       [ 1.24504606, -0.05379268,  0.95076833,  1.60585552],
       [ 0.3308514 , -1.86122675,  0.83142502,  0.5028967 ],
       [ 0.85324835,  0.17213658,  1.12978328,  0.91650626],
       [ 0.72264911, -1.63529749,  0.47339511,  0.227157  ],
       [-1.23633945,  0.84992435, -1.19741116, -1.28941139],
       [-0.97514097, -1.63529749, -0.18299307, -0.18645256],
       [-0.45274402,  1.97957065, -1.37642611, -1.01367168],
       [ 0.59204988, -0.5056512 ,  0.71208172,  0.91650626],
       [ 1.24504606,  0.17213658,  0.47339511,  0.36502685],
       [-0.19154555, -0.73158046,  0.3540518 ,  0.227157  ],
       [ 0.3308514 , -0.27972194,  0.53306676,  0.5028967 ],
       [-0.45274402,  1.97957065, -1.1377395 , -1.01367168],
       [ 0.46145064, -0.05379268,  0.77175337,  0.91650626],
       [ 1.3756453 , -0.05379268,  1.12978328,  1.33011582],
       [ 0.72264911, -0.5056512 ,  0.89109667,  0.5028967 ],
       [ 0.3308514 , -0.73158046,  0.89109667,  0.64076655],
       [-1.23633945,  0.17213658, -1.19741116, -1.28941139],
       [ 0.46145064, -0.5056512 ,  0.65241006,  0.08928714],
       [ 0.06965293,  2.2054999 , -1.43609776, -1.28941139],
       [-0.45274402,  0.84992435, -1.25708281, -1.01367168],
       [-0.32214479, -1.40936823,  0.05569354, -0.18645256],
       [-1.49753792,  1.30178287, -1.55544107, -1.28941139],
       [-1.62813716, -1.63529749, -1.37642611, -1.15154153],
       [ 0.72264911,  0.62399509,  0.65241006,  0.64076655],
```

```
[ 0.72264911,  0.62399509,  0.65241006,  0.64076655],
[-0.32214479,  1.07585361, -1.37642611, -1.28941139],
[ 0.06965293, -0.73158046,  0.29438015, -0.18645256],
[-1.7587364 ,  0.39806584, -1.37642611, -1.28941139],
[-1.36693868,  0.39806584, -1.37642611, -1.28941139],
[-0.97514097,  0.84992435, -1.25708281, -1.28941139],
[ 0.06965293, -0.95750971,  0.2347085 ,  0.08928714],
[ 0.72264911,  0.84992435,  1.18945493,  1.74372538],
[ 0.46145064, -0.5056512 ,  0.2347085 ,  0.227157  ],
[-0.06094631, -0.5056512 ,  0.53306676,  0.227157  ],
[-1.10574021, -1.18343897,  0.53306676,  0.77863641],
[-1.10574021,  0.17213658, -1.25708281, -1.42728124],
[-0.97514097,  1.07585361, -1.37642611, -1.15154153],
[ 1.11444683, -0.27972194,  0.59273841,  0.227157  ],
[-0.97514097,  1.07585361, -1.19741116, -0.73793197],
[ 0.3308514 ,  0.84992435,  0.53306676,  0.64076655],
[ 1.24504606,  0.62399509,  1.24912659,  1.33011582],
[ 0.72264911, -1.18343897,  0.77175337,  0.5028967 ],
[ 1.3756453 ,  0.39806584,  1.36846989,  1.60585552],
[-0.97514097, -2.31308526, -0.06364976, -0.18645256],
[ 1.63684377,  0.39806584,  0.65241006,  0.36502685],
[-0.06094631, -0.27972194,  0.3540518 ,  0.227157  ]])
```

```
X_test=ss.fit_transform(X_test)
```

```
X_test
```

```
array([[-0.84411839,  1.44663393, -1.38208209, -1.40129465],
       [-1.61260768,  0.69259694, -1.43496789, -1.27908873],
       [-0.40498166,  3.20605358, -1.38208209, -1.1568828 ],
       [ 0.03415508, -1.31816837,  0.78623548,  0.0651765 ],
       [ 1.68091784, -0.31278572,  1.31509342,  0.920618  ],
       [ 0.80264437, -0.81547704,  0.36314912,  0.0651765 ],
       [ 1.46134947, -0.81547704,  1.05066445,  0.67620614],
       [ 0.36350763,  0.18990561,  0.20449174,  0.18738242],
       [ 1.13199692, -0.31278572,  0.94489286,  0.920618  ],
       [-0.29519747, -0.81547704,  0.5218065 ,  1.28723579],
       [-0.18541329, -0.31278572,  0.5218065 ,  0.55400021],
       [-0.0756291 , -0.31278572,  0.36314912,  0.55400021],
       [ 0.91242855, -0.06144005,  0.5218065 ,  1.16502986],
       [ 2.01027039,  1.6979796 ,  1.20932183,  0.79841207],
       [-1.17347094,  0.18990561, -1.54073947, -1.40129465],
       [-0.29519747, -1.06682271,  0.5218065 ,  0.67620614],
       [ 0.03415508, -0.31278572,  0.25737753,  0.0651765 ],
       [ 1.35156529, -0.56413138,  1.15643604,  0.55400021],
       [-1.06368676,  1.6979796 , -1.3291963 , -1.40129465],
       [-1.28325513, -0.06144005, -1.38208209, -1.52350058],
       [-1.39303931, -0.31278572, -1.43496789, -1.52350058],
       [-1.28325513, -0.06144005, -1.38208209, -1.52350058],
       [ 0.69286018,  0.44125128,  0.83912127,  1.40944172],
       [ 0.91242855,  0.18990561,  0.83912127,  1.16502986],
       [-0.62455002, -1.31816837,  0.15160594, -0.17923536],
       [-0.62455002, -1.8208597 , -0.16570882, -0.30144129],
       [-0.95390258,  0.69259694, -1.43496789, -1.40129465],
```

```
       [ 0.91242855, -0.06144005,  0.68046389,  0.920618  ],
       [ 0.14393926,  0.69259694,  0.68046389,  1.16502986],
       [-1.28325513, -0.31278572, -1.43496789, -1.40129465],
       [-0.51476584, -0.81547704,  0.41603492,  0.79841207],
       [ 0.25372345,  0.44125128,  0.99777865,  1.40944172],
       [-1.06368676,  0.44125128, -1.2763105 , -1.03467687],
       [-1.06368676,  1.44663393, -1.38208209, -1.1568828 ],
       [ 1.79070202, -0.81547704,  1.36797921,  0.79841207],
       [ 0.36350763, -0.81547704,  0.78623548,  0.920618  ],
       [-0.51476584, -0.31278572, -0.00705144, -0.05702943],
       [-0.73433421,  1.44663393, -1.38208209, -1.40129465],
       [ 1.79070202,  1.6979796 ,  1.36797921,  1.04282393],
       [-1.17347094, -0.31278572, -1.3291963 , -1.40129465],
       [ 0.47329182,  0.18990561,  0.5218065 ,  0.79841207],
       [ 0.91242855, -0.06144005,  0.41603492,  0.18738242],
       [-0.18541329, -0.31278572,  0.04583436,  0.18738242],
       [-0.62455002, -2.07220536, -0.05993723, -0.05702943],
       [ 1.2417811 , -0.31278572,  0.89200706,  0.30958835]])
```

```
from sklearn.neighbors import KNeighborsRegressor
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
#knnreg=KNeighborsRegressor(n_neighbors=3)
```

```
knncla=KNeighborsClassifier()
```

```
knncla.fit(X_train, y_train)
```

```
     KNeighborsClassifier()
```

```
y_pred=knncla.predict(X_test)
```

```
#from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error
```

```
from sklearn.metrics import classification_report, confusion_matrix
```

```
#mean_absolute_error(y_test, y_pred)
```

```
#mean_absolute_percentage_error(y_test, y_pred)
```

```
confusion_matrix(y_test, y_pred)
```

```
     array([[14,  0,  0],
            [ 0,  9,  0],
            [ 0,  8, 14]])
```

```
print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00        14
               1       0.53      1.00      0.69         9
               2       1.00      0.64      0.78        22

        accuracy                           0.82        45
       macro avg       0.84      0.88      0.82        45
    weighted avg       0.91      0.82      0.83        45
```

```
error_rate=[]

for i in range(1,40):
  knn=KNeighborsRegressor(n_neighbors=i)
  knn.fit(X_train, y_train)
  pred_i=knn.predict(X_test)
  error_rate.append(mean_absolute_error(y_test,pred_i))
```

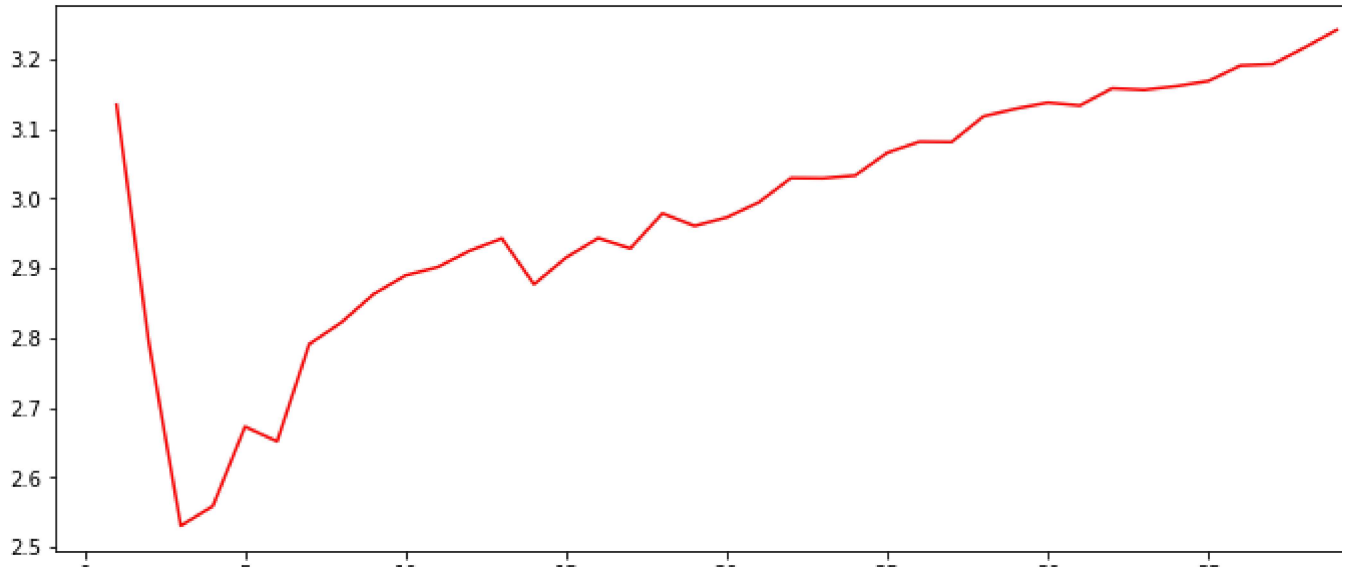
+ Code + Text

```
error_rate
```

```
[3.134210526315789,
 2.795065789473684,
 2.5309210526315793,
 2.5592105263157894,
 2.672894736842105,
 2.6517543859649124,
 2.7910714285714286,
 2.8222039473684206,
 2.8627192982456138,
 2.8897368421052634,
 2.9014952153110047,
 2.9251644736842106,
 2.9426619433198375,
 2.87687969924812,
 2.9153070175438596,
 2.943215460526315,
 2.928405572755418,
 2.9784722222222215,
 2.960560941828255,
 2.972763157894736,
 2.9944235588972425,
 3.029186602870813,
 3.02883295194508,
 3.0328673245614035,
 3.0655,
 3.0812500000000003,
 3.080750487329435,
 3.117763157894737,
 3.128493647912886,
```

```
        3.137214912280702,
        3.133000848896435,
        3.1576685855263156,
        3.1555223285486447,
        3.1607972136222906,
        3.1681766917293235,
        3.1903508771929823,
        3.192247510668563,
        3.2162049861495845,
        3.2415485829959514]
```

```
fig, ax=plt.subplots(figsize=(12,5))
ax.plot(range(1,40), error_rate, color='red')
plt.show()
```



✓ 0s completed at 10:48 AM