

```
In [3]: # # KNN CLASSIFIER
# # Vedanshu Patel
# # 20BCE0865
# importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
```

```
In [4]: # Load data set
dataset=pd.read_csv("processed.cleveland.data.csv",names=['age','sex','cp','trestbps',
dataset_mean=dataset
```

```
In [5]: dataset.tail()
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	output
298	45	1	1	110	264	0	0	132	0	1.2	2	0.0	7.0	
299	68	1	4	144	193	1	0	141	0	3.4	2	2.0	7.0	
300	57	1	4	130	131	0	0	115	1	1.2	2	1.0	7.0	
301	57	0	2	130	236	0	2	174	0	0.0	2	1.0	3.0	
302	38	1	3	138	175	0	0	173	0	0.0	1	NaN	3.0	

```
In [6]: dataset.columns
```

Out[6]: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'output'], dtype='object')

```
In [7]: dataset.shape
```

Out[7]: (303, 14)

```
In [8]: # filling missing values statistic measures
print("*\t Before fill missing values row 166,192,287,302 \t *")
print(dataset_mean.loc[287])
dataset1=dataset_mean
df1=pd.DataFrame(dataset1)
#print(df1)

print(" * \t Mean of column 11 'ca' \t *")
print(df1['ca'].mean())
df1.fillna(df1.mean(),inplace=True)
print("* \t After fill missing values row 16,192,287,302 \t *")
print(df1.loc[[166,192,287,302]])

print(" * \t Mean of column 12 'thal' \t *")
print(df1['thal'].mean())
df1.fillna(df1.mean(),inplace=True)
print("* \t After fill missing values row 87,266 \t *")
print(df1.loc[[87,266]])
```

```
*          Before fill missing values row 166,192,287,302          *
```

age	58.0
sex	1.0
cp	2.0
trestbps	125.0
chol	220.0
fbs	0.0
restecg	0.0
thalach	144.0
exang	0.0
oldpeak	0.4
slope	2.0
ca	NaN
thal	7.0
output	0.0

```
Name: 287, dtype: float64
*          Mean of column 11 'ca'          *
```

```
0.6722408026755853
```

```
*          After fill missing values row 16,192,287,302          *
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
166	52	1	3	138	223	0	0	169	0	0.0	
192	43	1	4	132	247	1	2	143	1	0.1	
287	58	1	2	125	220	0	0	144	0	0.4	
302	38	1	3	138	175	0	0	173	0	0.0	

  

	slope	ca	thal	output
166	1	0.672241	3.0	0
192	2	0.672241	7.0	1
287	2	0.672241	7.0	0
302	1	0.672241	3.0	0

```
*          Mean of column 12 'thal'          *
```

```
4.73421926910299
```

```
*          After fill missing values row 87,266          *
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
87	53	0	3	128	216	0	2	115	0	0.0	
266	52	1	4	128	204	1	0	156	1	1.0	

	slope	ca	thal	output
87	1	0.0	4.734219	0
266	2	0.0	4.734219	2

```
In [9]: # Extract feature columns
feature_cols=list(dataset.columns[0:13])

# Show the List of columns
print("Feature columns : \n{}".format(feature_cols))
```

Feature columns :

```
['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
```

```
In [10]: #Separating the data into feature and target
X= dataset[feature_cols]
y=dataset['output'].values

#showing the feature information by printing the first five rows
print("\nFeature Values:")
X.head()
```

Feature Values:

Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0

```
In [11]: #Splitting dataset into training and testing data
X_train,X_test, y_train,y_test = train_test_split(X , y ,test_size=0.3,random_state=42)
print(X_train)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
3	37	1	3	130	250	0	0	187	0	3.5	
55	54	1	4	124	266	0	2	109	1	2.2	
225	34	0	2	118	210	0	0	192	0	0.7	
224	63	0	4	108	269	0	0	169	1	1.8	
75	65	0	3	160	360	0	2	151	0	0.8	
..	...	...	..	...	...	...	...	...	...	...	
8	63	1	4	130	254	0	2	147	0	1.4	
73	65	1	4	110	248	0	2	158	0	0.6	
118	63	1	4	130	330	1	2	132	1	1.8	
189	69	1	3	140	254	0	2	146	0	2.0	
206	58	1	4	128	259	0	2	130	1	3.0	

  

	slope	ca	thal
3	3	0.0	3.0
55	2	1.0	7.0
225	1	0.0	3.0
224	2	2.0	3.0
75	1	0.0	3.0
..	...	...	...
8	2	1.0	7.0
73	1	2.0	6.0
118	1	3.0	7.0
189	2	3.0	7.0
206	2	2.0	7.0

[212 rows x 13 columns]

```
In [12]: #normalization
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

scaler.fit(X_train)
X_train = scaler.transform(X_train)
print(" \tAfter Z-score Normalization on X_train\t")
print(X_train)

scaler.fit(X_test)
X_test = scaler.transform(X_test)
print("\tAfter Z-score Normalization on X_test\t")
print(X_test)
```

```
After Z-score Normalization on X_train
[[-1.91736161  0.67975655 -0.16656264 ...  2.36151212 -0.68283167
  -0.93461042]
 [-0.06178394  0.67975655  0.8720044 ...  0.68151021  0.3635441
  1.13614677]
 [-2.24481649 -1.47111492 -1.20512967 ... -0.9984917  -0.68283167
  -0.93461042]
 ...
 [ 0.92058071  0.67975655  0.8720044 ... -0.9984917  2.45629564
  1.13614677]
 [ 1.57549048  0.67975655 -0.16656264 ...  0.68151021  2.45629564
  1.13614677]
 [ 0.37482257  0.67975655  0.8720044 ...  0.68151021  1.40991987
  1.13614677]]

After Z-score Normalization on X_test
[[-1.85828815  0.70128687 -0.16222142 ... -0.9335927  -0.05302469
  -0.81856114]
 [ 0.78936134  0.70128687 -2.2710999 ...  0.58349544  1.48316063
  -0.81856114]
 [-1.62805776  0.70128687  0.89221782 ... -0.9335927  -0.83079106
  1.26892886]
 ...
 [ 1.48005251  0.70128687  0.89221782 ...  0.58349544  1.48316063
  1.26892886]
 [ 0.78936134  0.70128687  0.89221782 ...  0.58349544  0.32618478
  -0.81856114]
 [ 0.44401575  0.70128687 -0.16222142 ... -0.9335927  1.48316063
  1.26892886]]
```

```
In [13]: print('kNN')
clf2= KNeighborsClassifier(n_neighbors=5)
clf2.fit(X_train,y_train)

y_predictions= clf2.predict(X_test)
cm1= confusion_matrix(y_test,y_predictions)

print("Accuracy=", accuracy_score(y_test,y_predictions))
```

```
kNN
Accuracy= 0.6923076923076923
```

In [ ]: