In [159…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
import statsmodels as stm

from sklearn.datasets import load_breast_cancer
from sklearn.neighbors import KDTree, BallTree, KNeighborsClassifier, RadiusNeighbor

%matplotlib inline
```

## Before we work with KNN and other distance dependent algorithms lets understand the various distance metrics. My motive in understanding the distances because there is no one metric that can be used for every problem.

## CASE-I: Entire variable as vectors

In [3]:
```
X = np.array([[1,2,3]])
Y = np.array([[3,4,5]])
X,Y
```

Out[3]:
```
(array([[1, 2, 3]]), array([[3, 4, 5]]))
```

In [4]:
```
X.shape, Y.shape, X.ndim, Y.ndim
```

Out[4]:
```
((1, 3), (1, 3), 2, 2)
```

In [5]:
```
eu_dist = DistanceMetric.get_metric('euclidean')
```

In [6]:
```
man_hat_dist = DistanceMetric.get_metric('manhattan')
```

In [7]:
```
eu_dist.pairwise(X,Y)
```

Out[7]:
```
array([[3.46410162]])
```

In [8]:
```
man_hat_dist.pairwise(X,Y)
```

Out[8]:
```
array([[6.]])
```

## CASE-II: Different shape vectors

In [9]:
```
XX = X.copy().reshape(3,1)
YY = Y.copy().reshape(3,1)
```

In [10]:
```
XX,YY
```

Out[10]:
```
(array([[1],
        [2],
        [3]]),
 array([[3],
        [4],
        [5]]))
```

In [12]:
```
eu_dist.pairwise(XX,YY)
```

Out[12]:
```
array([[2., 3., 4.],
       [1., 2., 3.],
       [0., 1., 2.]])
```

In [13]:
```
man_hat_dist.pairwise(XX,YY)
```

Out[13]:
```
array([[2., 3., 4.],
       [1., 2., 3.],
       [0., 1., 2.]])
```

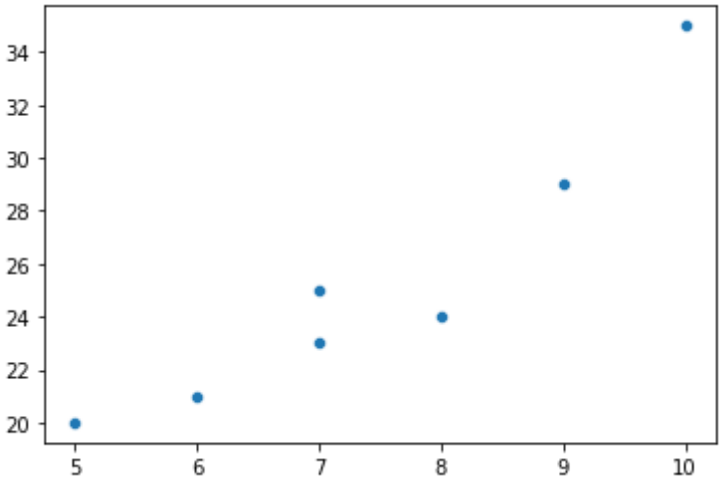
## CASE-III

In [23]:
```
shoe_size = np.array([7,8,9,10,6,5,7])
body_mass = np.array([23,24,29,35,21,20,25])
shoe_size, body_mass, shoe_size.shape, shoe_size.ndim, body_mass.shape, body_mass.nd
```

Out[23]:
```
(array([ 7,  8,  9, 10,  6,  5,  7]),
 array([23, 24, 29, 35, 21, 20, 25]),
 (7,),
 1,
 (7,),
 1)
```

In [26]:
```
sns.scatterplot(x=shoe_size,y=body_mass)
```

Out[26]:
```
<AxesSubplot:>
```




In [35]:
```
[shoe_size[5],body_mass[5]], [shoe_size[4],body_mass[4]]
```

Out[35]:
```
([5, 20], [6, 21])
```

In [37]:
```
eu_dist.pairwise([[shoe_size[5],body_mass[5]]], [[shoe_size[4],body_mass[4]]])
```

Out[37]:
```
array([[1.41421356]])
```

In [38]:
```
man_hat_dist.pairwise([[shoe_size[5],body_mass[5]]], [[shoe_size[4],body_mass[4]]])
```

Out[38]:
```
array([[2.]])
```

## CASE-IV

In [40]:
```
chby_dist = DistanceMetric.get_metric('chebyshev')
```

In [42]:
```
chby_dist.pairwise([[shoe_size[5],body_mass[5]]], [[shoe_size[4],body_mass[4]]])
```

Out[42]:
```
array([[1.]])
```

In [44]:
```
chby_dist.pairwise(shoe_size.reshape((1,7)),body_mass.reshape((1,7)))
```

Out[44]:
```
array([[25.]])
```

In [47]:
```
np.square(np.corrcoef(shoe_size,body_mass))
```

Out[47]:
```
array([[1.       , 0.8792535],
       [0.8792535, 1.       ]])
```

## CASE-V

In [327…
```
filepath = 'https://raw.githubusercontent.com/selva86/datasets/master/diamonds.csv'
df = pd.read_csv(filepath).iloc[0:500, [0,4,6]]
df.shape, df.ndim, df.head()
```

Out[327…
```
((500, 3),
 2,
    carat  depth  price
 0   0.23   61.5    326
 1   0.21   59.8    326
 2   0.23   56.9    327
 3   0.29   62.4    334
 4   0.31   63.3    335)
```

In [161…
```
def mahalanobis(x=None, data=None, cov=None):
    """Compute the Mahalanobis Distance between each row of x and the data
    x    : vector or matrix of data with, say, p columns.
    data : ndarray of the distribution from which Mahalanobis distance of each obser
    cov  : covariance matrix (p x p) of the distribution. If None, will be computed
    """
    x_minus_mu = x - np.mean(data)
    if not cov:
        cov = np.cov(data.values.T)
    inv_covmat = scipy.linalg.inv(cov)
    left_term = np.dot(x_minus_mu, inv_covmat)
    mahal = np.dot(left_term, x_minus_mu.T)
    return mahal.diagonal(), inv_covmat

df_x = df[['carat', 'depth', 'price']].head(500)
df_x['mahala'],VI = mahalanobis(x=df_x, data=df[['carat', 'depth', 'price']])
df_x.head()
```

Out[161…

| | carat | depth | price | mahala |
|---|---|---|---|---|
| **0** | 0.23 | 61.5 | 326 | 3.766898 |
| **1** | 0.21 | 59.8 | 326 | 5.040171 |
| **2** | 0.23 | 56.9 | 327 | 11.473766 |
| **3** | 0.29 | 62.4 | 334 | 3.876022 |
| **4** | 0.31 | 63.3 | 335 | 4.661105 |

### Understanding the above function

In [166…
```
x_minus_mu = df - np.mean(df)
cov = np.cov(df.values.T)
inv_covmat = scipy.linalg.inv(cov)
left_term = np.dot(x_minus_mu, inv_covmat)
mahal = np.dot(left_term, x_minus_mu.T)
```

### STEP-1

In [178…
```
np.mean(df)
```

Out[178…
```
carat       0.63612
depth      61.71280
```

```
price    2233.49800
dtype: float64
```

In [181…
```
x_minus_mu.head()
```

Out[181…

| | carat | depth | price |
|---|---|---|---|
| **0** | -0.40612 | -0.2128 | -1907.498 |
| **1** | -0.42612 | -1.9128 | -1907.498 |
| **2** | -0.40612 | -4.8128 | -1906.498 |
| **3** | -0.34612 | 0.6872 | -1899.498 |
| **4** | -0.32612 | 1.5872 | -1898.498 |

In [184…
```
pd.DataFrame(df - np.mean(df)).head()
```

Out[184…

| | carat | depth | price |
|---|---|---|---|
| **0** | -0.40612 | -0.2128 | -1907.498 |
| **1** | -0.42612 | -1.9128 | -1907.498 |
| **2** | -0.40612 | -4.8128 | -1906.498 |
| **3** | -0.34612 | 0.6872 | -1899.498 |
| **4** | -0.32612 | 1.5872 | -1898.498 |

### STEP-2

In [186…
```
pd.DataFrame(cov)
```

Out[186…

| | 0 | 1 | 2 |
|---|---|---|---|
| **0** | 0.046833 | 0.022599 | 197.305744 |
| **1** | 0.022599 | 2.983724 | -5.602980 |
| **2** | 197.305744 | -5.602980 | 993181.561118 |

In [187…
```
df.cov().values
```

Out[187…
```
array([[ 4.68330116e-02,  2.25988617e-02,  1.97305744e+02],
       [ 2.25988617e-02,  2.98372361e+00, -5.60298036e+00],
       [ 1.97305744e+02, -5.60298036e+00,  9.93181561e+05]])
```

In [173…
```
df.corr()
```

Out[173…

| | carat | depth | price |
|---|---|---|---|
| **carat** | 1.000000 | 0.060455 | 0.914849 |
| **depth** | 0.060455 | 1.000000 | -0.003255 |
| **price** | 0.914849 | -0.003255 | 1.000000 |

In [175…
```
np.corrcoef(df,rowvar=False)
```

Out[175…
```
array([[ 1.        ,  0.06045486,  0.91484888],
       [ 0.06045486,  1.        , -0.00325481],
       [ 0.91484888, -0.00325481,  1.        ]])
```

In [189…
```
scipy.linalg.inv(df.cov().values)
```

Out[189…
```
array([[ 1.34268725e+02, -1.06705836e+00, -2.66798846e-02],
       [-1.06705836e+00,  3.43635347e-01,  2.13920731e-04],
       [-2.66798846e-02,  2.13920731e-04,  6.30830586e-06]])
```

In [177…
```
inv_covmat
```

Out[177…
```
array([[ 1.34268725e+02, -1.06705836e+00, -2.66798846e-02],
       [-1.06705836e+00,  3.43635347e-01,  2.13920731e-04],
       [-2.66798846e-02,  2.13920731e-04,  6.30830586e-06]])
```

**STEP-3**

In [203…
```
pd.DataFrame(((df - np.mean(df)).values @ scipy.linalg.inv(df.cov().values)) @ ((df
```

Out[203…
```
(500, 500)
```

In [204…
```
pd.DataFrame(((df - np.mean(df)).values @ scipy.linalg.inv(df.cov().values)) @ ((df
```

Out[204…

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3.766898 | 3.916408 | 3.985651 | 3.509290 | 3.396797 | 3.658189 | 3.682101 | 3.631782 | 3.615154 |
| **1** | 3.916408 | 5.040171 | 6.724389 | 3.101321 | 2.465016 | 3.068993 | 3.374325 | 3.531883 | 1.749026 |
| **2** | 3.985651 | 6.724389 | 11.473766 | 2.590670 | 1.152378 | 1.861321 | 2.675488 | 3.354030 | -1.915498 |
| **3** | 3.509290 | 3.101321 | 2.590670 | 3.876022 | 4.122082 | 3.776873 | 3.677306 | 3.664491 | 4.162841 |
| **4** | 3.396797 | 2.465016 | 1.152378 | 4.122082 | 4.661105 | 4.052604 | 3.808965 | 3.714524 | 5.067003 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **495** | -1.030892 | -1.550898 | -2.805898 | -1.077694 | -0.864771 | -0.577531 | -0.770679 | -1.059940 | 0.449597 |
| **496** | -0.963937 | -0.695969 | -0.526231 | -1.356482 | -1.546962 | -1.124048 | -1.076651 | -1.143898 | -1.233603 |
| **497** | -0.877226 | 0.018579 | 1.250222 | -1.610250 | -2.134672 | -1.516238 | -1.285177 | -1.208840 | -2.467239 |
| **498** | -0.901139 | -0.286753 | 0.436055 | -1.510683 | -1.891033 | -1.321054 | -1.175901 | -1.178855 | -1.866096 |
| **499** | -1.741025 | -3.158284 | -4.643153 | -0.159136 | 0.751059 | -0.777156 | -1.092296 | -1.001465 | 0.323868 |

500 rows × 500 columns

**STEP-4**

In [171…
```
mahal.shape
```

Out[171…
```
(500, 500)
```

In [170…
```
pd.DataFrame(mahal)
```

Out[170…

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3.766898 | 3.916408 | 3.985651 | 3.509290 | 3.396797 | 3.658189 | 3.682101 | 3.631782 | 3.615154 |
| **1** | 3.916408 | 5.040171 | 6.724389 | 3.101321 | 2.465016 | 3.068993 | 3.374325 | 3.531883 | 1.749026 |
| **2** | 3.985651 | 6.724389 | 11.473766 | 2.590670 | 1.152378 | 1.861321 | 2.675488 | 3.354030 | -1.915498 |
| **3** | 3.509290 | 3.101321 | 2.590670 | 3.876022 | 4.122082 | 3.776873 | 3.677306 | 3.664491 | 4.162841 |
| **4** | 3.396797 | 2.465016 | 1.152378 | 4.122082 | 4.661105 | 4.052604 | 3.808965 | 3.714524 | 5.067003 |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | .. |
| **495** | -1.030892 | -1.550898 | -2.805898 | -1.077694 | -0.864771 | -0.577531 | -0.770679 | -1.059940 | 0.449597 |
| **496** | -0.963937 | -0.695969 | -0.526231 | -1.356482 | -1.546962 | -1.124048 | -1.076651 | -1.143898 | -1.233603 |
| **497** | -0.877226 | 0.018579 | 1.250222 | -1.610250 | -2.134672 | -1.516238 | -1.285177 | -1.208840 | -2.467239 |
| **498** | -0.901139 | -0.286753 | 0.436055 | -1.510683 | -1.891033 | -1.321054 | -1.175901 | -1.178855 | -1.866096 |
| **499** | -1.741025 | -3.158284 | -4.643153 | -0.159136 | 0.751059 | -0.777156 | -1.092296 | -1.001465 | 0.323868 |

500 rows × 500 columns

# So, good here as everything matched!!

### Critical Value

In [154...
```
from scipy.stats import chi2
chi2.ppf((1-0.01), df=2)
```

Out[154...
```
9.21034037197618
```

### P-Value

In [156...
```
1 - chi2.cdf(12.7,df=2)
```

Out[156...
```
0.0017467471362611064
```

## Applying Mahalanobis Distance Metric on Breast Cancer

In [222...
```
cancer = load_breast_cancer()
```

In [223...
```
cancer_df = pd.DataFrame(cancer.data,columns=cancer.feature_names)
cancer_df.head()
```

Out[223...

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | di |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.3001 | 0.14710 | 0.2419 | |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.0869 | 0.07017 | 0.1812 | |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.1974 | 0.12790 | 0.2069 | |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.2414 | 0.10520 | 0.2597 | |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.1980 | 0.10430 | 0.1809 | |

5 rows × 30 columns

In [224...
```
y = pd.DataFrame(cancer.target,columns=['Label'])
y.head()
```

Out[224...

| | Label |
|---|---|
| **0** | 0 |

| | Label |
|---|---|
| **1** | 0 |
| **2** | 0 |
| **3** | 0 |
| **4** | 0 |

In [226…
```
cancer_df = pd.concat([cancer_df,y],axis=1).copy(deep=True)
cancer_df
```

Out[226…

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.30010 | 0.14710 | 0.2419 |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.08690 | 0.07017 | 0.1812 |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.19740 | 0.12790 | 0.2069 |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.24140 | 0.10520 | 0.2597 |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.19800 | 0.10430 | 0.1809 |
| **…** | … | … | … | … | … | … | … | … | … |
| **564** | 21.56 | 22.39 | 142.00 | 1479.0 | 0.11100 | 0.11590 | 0.24390 | 0.13890 | 0.1726 |
| **565** | 20.13 | 28.25 | 131.20 | 1261.0 | 0.09780 | 0.10340 | 0.14400 | 0.09791 | 0.1752 |
| **566** | 16.60 | 28.08 | 108.30 | 858.1 | 0.08455 | 0.10230 | 0.09251 | 0.05302 | 0.1590 |
| **567** | 20.60 | 29.33 | 140.10 | 1265.0 | 0.11780 | 0.27700 | 0.35140 | 0.15200 | 0.2397 |
| **568** | 7.76 | 24.54 | 47.92 | 181.0 | 0.05263 | 0.04362 | 0.00000 | 0.00000 | 0.1587 |

569 rows × 31 columns

In [227…
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

In [230…
```
X_train, X_test, y_train, y_test = train_test_split(cancer_df.iloc[:, 0:-1],cancer_d
                                                    stratify=cancer_df['Label'])
```

In [231…
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[231… ((341, 30), (228, 30), (341,), (228,))

## Seggregating the positive and negative observations

### Only positivies

In [281…
```
train_df = pd.concat([X_train,y_train],axis=1).copy(deep=True)
train_df_pos = train_df[train_df['Label']==1].copy(deep=True)
train_df_pos.shape
```

Out[281… (214, 31)

In [282…
```
train_df_pos.head()
```

Out[282...

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|---|---|---|---|---|---|---|---|---|
| **525** | 8.571 | 13.10 | 54.53 | 221.3 | 0.10360 | 0.07632 | 0.025650 | 0.015100 | 0.1678 |
| **58** | 13.050 | 19.31 | 82.61 | 527.2 | 0.08060 | 0.03789 | 0.000692 | 0.004167 | 0.1819 |
| **342** | 11.060 | 14.96 | 71.49 | 373.9 | 0.10330 | 0.09097 | 0.053970 | 0.033410 | 0.1776 |
| **413** | 14.990 | 22.11 | 97.53 | 693.7 | 0.08515 | 0.10250 | 0.068590 | 0.038760 | 0.1944 |
| **559** | 11.510 | 23.93 | 74.52 | 403.5 | 0.09261 | 0.10210 | 0.111200 | 0.041050 | 0.1388 |

5 rows × 31 columns

### Only negatives

In [283...
```
train_df_neg = train_df[train_df['Label']==0].copy(deep=True)
train_df_neg.shape
```

Out[283... (127, 31)

In [284...
```
train_df_neg.head()
```

Out[284...

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|---|---|---|---|---|---|---|---|---|
| **39** | 13.48 | 20.82 | 88.40 | 559.2 | 0.10160 | 0.12550 | 0.1063 | 0.05439 | 0.1720 |
| **229** | 12.83 | 22.33 | 85.26 | 503.2 | 0.10880 | 0.17990 | 0.1695 | 0.06861 | 0.2123 |
| **40** | 13.44 | 21.58 | 86.18 | 563.0 | 0.08162 | 0.06031 | 0.0311 | 0.02031 | 0.1784 |
| **172** | 15.46 | 11.89 | 102.50 | 736.9 | 0.12570 | 0.15550 | 0.2032 | 0.10970 | 0.1966 |
| **213** | 17.42 | 25.56 | 114.50 | 948.0 | 0.10060 | 0.11460 | 0.1682 | 0.06597 | 0.1308 |

5 rows × 31 columns

### Distance calculation for positive observations

In [285...
```
train_df_pos_mh_dist = ((train_df_pos.iloc[:,0:-1] - np.mean(train_df_pos.iloc[:,0:-
                        @ scipy.linalg.inv(train_df_pos.iloc[:,0:-1].cov().values))\
@ ((train_df_pos.iloc[:,0:-1] - np.mean(train_df_pos.iloc[:,0:-1])).T).values
```

In [286...
```
train_df_pos['mahal_dist'] = train_df_pos_mh_dist.diagonal()
```

In [287...
```
train_df_pos.head()
```

Out[287...

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|---|---|---|---|---|---|---|---|---|
| **525** | 8.571 | 13.10 | 54.53 | 221.3 | 0.10360 | 0.07632 | 0.025650 | 0.015100 | 0.1678 |
| **58** | 13.050 | 19.31 | 82.61 | 527.2 | 0.08060 | 0.03789 | 0.000692 | 0.004167 | 0.1819 |
| **342** | 11.060 | 14.96 | 71.49 | 373.9 | 0.10330 | 0.09097 | 0.053970 | 0.033410 | 0.1776 |

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|---|---|---|---|---|---|---|---|---|
| **413** | 14.990 | 22.11 | 97.53 | 693.7 | 0.08515 | 0.10250 | 0.068590 | 0.038760 | 0.1944 |
| **559** | 11.510 | 23.93 | 74.52 | 403.5 | 0.09261 | 0.10210 | 0.111200 | 0.041050 | 0.1388 |

5 rows × 32 columns

**Distance calculation for negative observations**

In [288…
```
train_df_neg_mh_dist = ((train_df_neg.iloc[:,0:-1] - np.mean(train_df_neg.iloc[:,0:-
                        @ scipy.linalg.inv(train_df_neg.iloc[:,0:-1].cov().values))\
@ ((train_df_neg.iloc[:,0:-1] - np.mean(train_df_neg.iloc[:,0:-1])).T).values
```

In [289…
```
train_df_neg['mahal_dist'] = train_df_neg_mh_dist.diagonal()
```

In [290…
```
train_df_neg.head()
```

Out[290…

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|---|---|---|---|---|---|---|---|---|
| **39** | 13.48 | 20.82 | 88.40 | 559.2 | 0.10160 | 0.12550 | 0.1063 | 0.05439 | 0.1720 |
| **229** | 12.83 | 22.33 | 85.26 | 503.2 | 0.10880 | 0.17990 | 0.1695 | 0.06861 | 0.2123 |
| **40** | 13.44 | 21.58 | 86.18 | 563.0 | 0.08162 | 0.06031 | 0.0311 | 0.02031 | 0.1784 |
| **172** | 15.46 | 11.89 | 102.50 | 736.9 | 0.12570 | 0.15550 | 0.2032 | 0.10970 | 0.1966 |
| **213** | 17.42 | 25.56 | 114.50 | 948.0 | 0.10060 | 0.11460 | 0.1682 | 0.06597 | 0.1308 |

5 rows × 32 columns

# Hypothesis Testing

## CASE-I

### Here, first I'll perform the test on positive observations considering the `null hypothesis` as TRUE or POSITIVE and the `alternate hypothesis` as FALSE or negative.

In [295…
```
dof = train_df_pos.iloc[:,0:-1].shape[1]-1
dof
```

Out[295… 30

In [315…
```
c_val = chi2.ppf((1-0.01), df=dof)
c_val
```

Out[315… 50.89218131151707

In [299…
```
train_df_pos['p_val'] = 1 - chi2.cdf(train_df_pos['mahal_dist'], df=dof)
```

In [300…
```
train_df_pos.head()
```

Out[300...

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|---|---|---|---|---|---|---|---|---|
| **525** | 8.571 | 13.10 | 54.53 | 221.3 | 0.10360 | 0.07632 | 0.025650 | 0.015100 | 0.1678 |
| **58** | 13.050 | 19.31 | 82.61 | 527.2 | 0.08060 | 0.03789 | 0.000692 | 0.004167 | 0.1819 |
| **342** | 11.060 | 14.96 | 71.49 | 373.9 | 0.10330 | 0.09097 | 0.053970 | 0.033410 | 0.1776 |
| **413** | 14.990 | 22.11 | 97.53 | 693.7 | 0.08515 | 0.10250 | 0.068590 | 0.038760 | 0.1944 |
| **559** | 11.510 | 23.93 | 74.52 | 403.5 | 0.09261 | 0.10210 | 0.111200 | 0.041050 | 0.1388 |

5 rows × 33 columns

In [303...
```
train_df_pos.shape
```

Out[303... (214, 33)

In [304...
```
train_df_pos[train_df_pos['p_val'] < 0.01].shape
```

Out[304... (26, 33)

In [308...
```
rej_null_hyp = train_df_pos[train_df_pos['p_val'] < 0.01]['mahal_dist'].values
rej_null_hyp
```

Out[308...
```
array([ 81.02384852,  72.07114302,  52.08522329,  62.90183841,
        69.0939475 ,  66.10649304,  66.76169799,  76.36211933,
       118.79639287, 145.94310336,  59.52453926,  73.90722389,
        68.33222003,  96.34523381,  54.67357222, 154.13771741,
        76.20390925, 100.12061258,  74.13167215,  58.07212966,
       194.52697533,  54.58637054, 129.41505176,  60.14696292,
        89.70806506,  54.60903334])
```

In [311...
```
rej_null_hyp.min(), rej_null_hyp.max()
```

Out[311... (52.085223289182096, 194.5269753331549)

In [325...
```
train_df_pos[train_df_pos['p_val'] >= 0.01]['mahal_dist'].min(), train_df_pos[train_
```

Out[325... (6.167006044471601, 50.03579960395341)

## So, the null hypothesis got rejected based on the above 26 observations where the mahalanobis distance is greater than the critical value thus the distance ended up in the rejection region.

## CASE-II

## Now, I'll perform the test on negative observations considering the `null hypothesis` as FALSE or NEGATIVE and the `alternate hypothesis` as TRUE or POSITIVE.

In [317...
```
train_df_neg['p_val'] = 1 - chi2.cdf(train_df_neg['mahal_dist'], df=dof)
```

In [318...
```
train_df_neg.head()
```

Out[318...

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|---|---|---|---|---|---|---|---|---|

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry |
|---|---|---|---|---|---|---|---|---|---|
| **39** | 13.48 | 20.82 | 88.40 | 559.2 | 0.10160 | 0.12550 | 0.1063 | 0.05439 | 0.1720 |
| **229** | 12.83 | 22.33 | 85.26 | 503.2 | 0.10880 | 0.17990 | 0.1695 | 0.06861 | 0.2123 |
| **40** | 13.44 | 21.58 | 86.18 | 563.0 | 0.08162 | 0.06031 | 0.0311 | 0.02031 | 0.1784 |
| **172** | 15.46 | 11.89 | 102.50 | 736.9 | 0.12570 | 0.15550 | 0.2032 | 0.10970 | 0.1966 |
| **213** | 17.42 | 25.56 | 114.50 | 948.0 | 0.10060 | 0.11460 | 0.1682 | 0.06597 | 0.1308 |

5 rows × 33 columns

In [319…
```
train_df_neg.shape
```

Out[319… (127, 33)

In [320…
```
train_df_neg[train_df_neg['p_val'] < 0.01].shape
```

Out[320… (14, 33)

In [321…
```
rej_null_hyp_neg = train_df_neg[train_df_neg['p_val'] < 0.01]['mahal_dist'].values
rej_null_hyp_neg
```

Out[321…
```
array([104.13997248,  65.26593056,  57.53158832,  74.66179609,
        55.42755962,  79.93825875,  52.30118131,  67.52958184,
        58.82178158,  52.2838042 ,  61.00788787, 102.21571179,
        60.01309311,  70.31700251])
```

In [322…
```
rej_null_hyp_neg.min(), rej_null_hyp_neg.max()
```

Out[322… (52.28380420063802, 104.13997248029511)

In [324…
```
train_df_neg[train_df_neg['p_val'] >= 0.01]['mahal_dist'].min(), train_df_neg[train_
```

Out[324… (8.361142675252562, 49.68938504959755)

## So, the null hypothesis got rejected based on the above 14 observations where the mahalanobis distance is greater than the critical value thus the distance ended up in the rejection region.

## Euclidean Distance :: It performs the pairwise difference and not takes the distribution into account.

In [79]:
```
from scipy.spatial.distance import pdist, squareform
```

In [163…
```
distances = pdist(df.iloc[:,:], metric='euclidean')
dist_matrix = pd.DataFrame(squareform(distances))
```

In [164…
```
pd.DataFrame(dist_matrix.iloc[0:5,0])
```

Out[164…

| | 0 |
|---|---|

| | 0 |
|---|---|
| **0** | 0.000000 |
| **1** | 1.700118 |
| **2** | 4.707441 |
| **3** | 8.050689 |
| **4** | 9.178584 |

In [165...
```
df.head(5)
```

Out[165...

| | carat | depth | price |
|---|---|---|---|
| **0** | 0.23 | 61.5 | 326 |
| **1** | 0.21 | 59.8 | 326 |
| **2** | 0.23 | 56.9 | 327 |
| **3** | 0.29 | 62.4 | 334 |
| **4** | 0.31 | 63.3 | 335 |

## VIF Calculation

In [345...
```
from statsmodels.stats.outliers_influence import variance_inflation_factor as vif
```

In [346...
```
df.head()
```

Out[346...

| | carat | depth | price |
|---|---|---|---|
| **0** | 0.23 | 61.5 | 326 |
| **1** | 0.21 | 59.8 | 326 |
| **2** | 0.23 | 56.9 | 327 |
| **3** | 0.29 | 62.4 | 334 |
| **4** | 0.31 | 63.3 | 335 |

In [350...
```
vif_df = pd.DataFrame()
vif_df['Values'] = [vif(df.values,i) for i in range(len(df.columns))]
vif_df['Features'] = df.columns
```

In [351...
```
vif_df
```

Out[351...

| | Values | Features |
|---|---|---|
| **0** | 60.169063 | carat |
| **1** | 11.015691 | depth |
| **2** | 37.304660 | price |

## KNN

In [405...
```
knc = KNeighborsClassifier(n_neighbors=5,radius=0.7,weights='distance',algorithm='kd
```

In [406…
```
cancer_df.head()
```

Out[406…

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | di |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.3001 | 0.14710 | 0.2419 | |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.0869 | 0.07017 | 0.1812 | |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.1974 | 0.12790 | 0.2069 | |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.2414 | 0.10520 | 0.2597 | |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.1980 | 0.10430 | 0.1809 | |

5 rows × 31 columns

In [407…
```
knc.fit(cancer_df.iloc[:,0:-1],cancer_df.iloc[:,-1])
```

Out[407…
```
KNeighborsClassifier(algorithm='kd_tree', weights='distance')
```

In [408…
```
knc.predict(cancer_df.iloc[:,0:-1])
```

Out[408…
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 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, 0, 0, 0, 0, 0, 0, 1])
```