```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
student=pd.read_csv("/content/sample_data/student_scores.csv")
```

```
student
```

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
15	8.9	95
16	2.5	30
17	1.9	24
18	6.1	67
19	7.4	69
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

```
# someone put 9 hour what would be the predicted score
hours                      score
input                      output
x                           y
feature,                    label
feature                   target
independent              dependent
this problem is called regretion
superwise dataset
labeled dataset  (agar features aur label dono present ho use labeled dataset kahte hai)
supervised problem (features and label both)
unsupervised problem  (only features)
regerssion problem( prodict numbery2)
```

```
linear regession problem
  hours       score
  2.5hr       21
  5.1hr       47
  3.2hr       27
  9hr         ?
```

Double-click (or enter) to edit

```
student.shape
```

```
(25, 2)
```

```
student.ndim
```

```
2
```

```
student.size
```

```
50
```

```
student.dtypes
```

```
Hours      float64
Scores       int64
dtype: object
```

```
student.head(5)
```

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

```
student.tail(5)
```

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

```
student.head(-1)
```

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
15	8.9	95
16	2.5	30
17	1.9	24
18	6.1	67
19	7.4	69
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76

```
student.isnull()
```

	Hours	Scores
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
5	False	False
6	False	False
7	False	False
8	False	False
9	False	False
10	False	False
11	False	False
12	False	False
13	False	False
14	False	False
15	False	False
16	False	False
17	False	False
18	False	False
19	False	False
20	False	False
21	False	False
22	False	False
23	False	False
24	False	False

```
student.isnull().any(axis=0)
```

```
Hours     False
Scores    False
dtype: bool
```

```
student.isnull().any(axis=1)
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
dtype: bool
```

```
student.sum()
```

```
Hours      125.3
Scores    1287.0
dtype: float64
```

```
# saprate out the features and label
student['Hours']
```

```
0     2.5
1     5.1
2     3.2
3     8.5
4     3.5
5     1.5
6     9.2
```

```
7     5.5
8     8.3
9     2.7
10    7.7
11    5.9
12    4.5
13    3.3
14    1.1
15    8.9
16    2.5
17    1.9
18    6.1
19    7.4
20    2.7
21    4.8
22    3.8
23    6.9
24    7.8
Name: Hours, dtype: float64
```

```
features=student['Hours'].values
```

```
features #independent variavel/x/features/input
```

```
array([2.5, 5.1, 3.2, 8.5, 3.5, 1.5, 9.2, 5.5, 8.3, 2.7, 7.7, 5.9, 4.5,
       3.3, 1.1, 8.9, 2.5, 1.9, 6.1, 7.4, 2.7, 4.8, 3.8, 6.9, 7.8])
```

```
labels=student['Scores'].values
```

```
labels  # dependent/y/label/output
```

```
array([21, 47, 27, 75, 30, 20, 88, 60, 81, 25, 85, 62, 41, 42, 17, 95, 30,
       24, 67, 69, 30, 54, 35, 76, 86])
```

```
features.shape
```

```
(25,)
```

```
labels.shape
```

```
(25,)
```

```
features.ndim
```

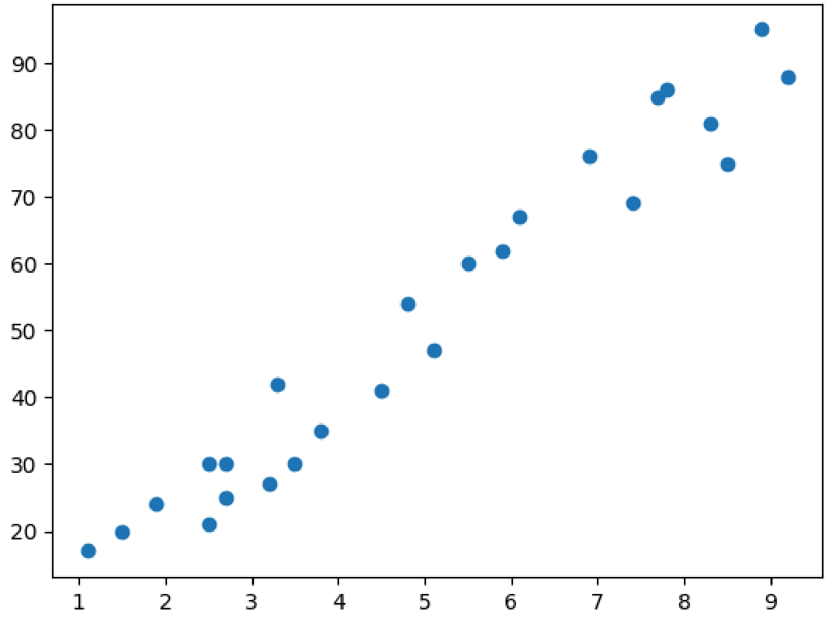
```
1
```

```
labels.ndim
```

```
1
```

```
plt.scatter(features,labels)   #positive core relative data
```

```
<matplotlib.collections.PathCollection at
0x7838a356d540>
```



```
x=[1,2,3,4,5]
y=[1,2,3,4,5]
y=mx+c
m=y2-y1  / x2-x1   #   this is called slope
m=4-2/4-2  =1
```

```
y=x                    #-----> # straightline  --------> linear regression (best fit line)
error                  # --->#  root mean equare error OR loss function OR cost function
```

```
#scikit learn ----> sklearn
```

- RMSE = sqrt [(Σ(Pi – Oi)²) / n]

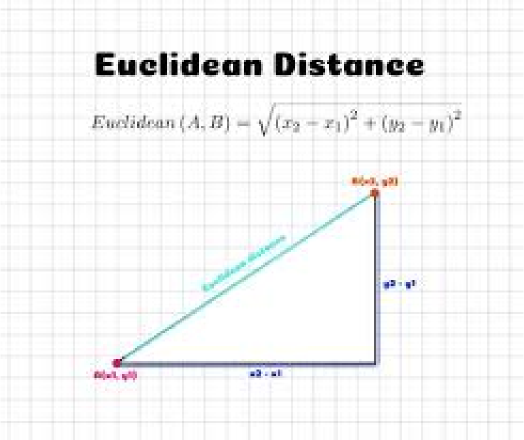
- The sum of the squared differences between the predicted and observed values is divided by the number of observations, and the square root of the result is taken to yield the RMSE

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} \left(x_i - \hat{x}_i\right)^2}{N}}$$

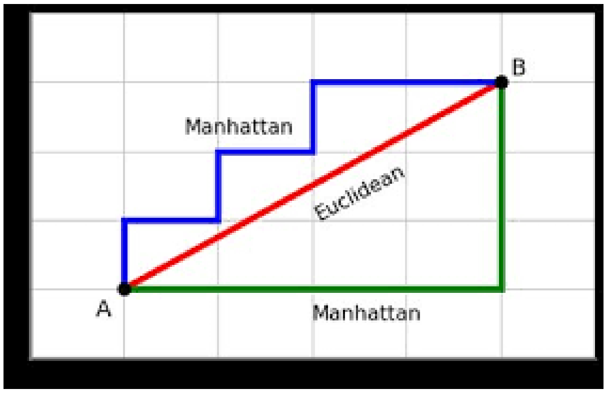
$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} \left(x_i - \hat{x}_i\right)^2}{N}}$$

```
distance machine learning-------.
euclidean distace---: used to represent the shortest distance between two points.

Manhattan distance :  often preferred over euclidean distance when the data has dimensionali


hamming distance: used to measure the distance between categorical variable.
```

Euclidean distance

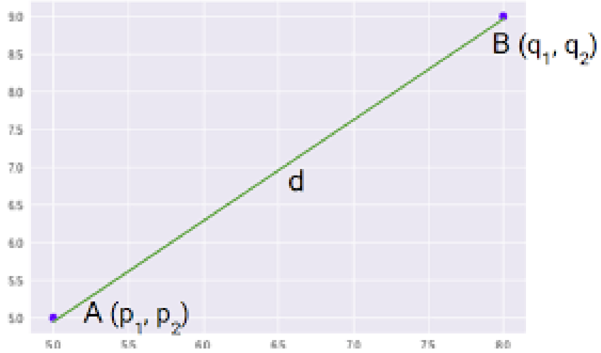


Manhattan distance



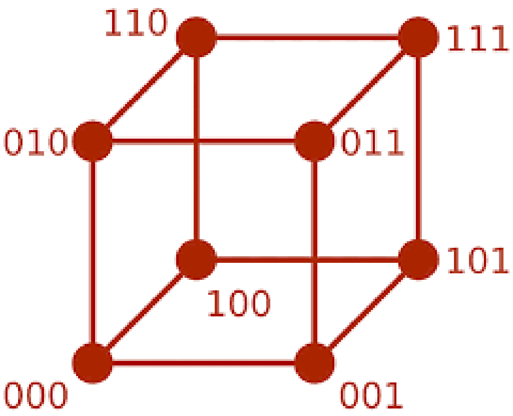
Start coding or generate with AI.

manhattan distance



Start coding or generate with AI.

Hamming Distance



```
################          day 4      #############
```

```
from sklearn.linear_model import LinearRegression  # best fit line
```

```
reg=LinearRegression()        # model      this model create best fir line given to you
```

```
reg.fit(features,labels)
```

```
---------------------------------------------------
------------------------
ValueError                                Traceback
(most recent call last)
<ipython-input-28-8d8076d8f986> in <cell line: 1>()
----> 1 reg.fit(features,labels)

                    3 frames
/usr/local/lib/python3.10/dist-
packages/sklearn/utils/validation.py in
check_array(array, accept_sparse,
accept_large_sparse, dtype, order, copy,
force_all_finite, ensure_2d, allow_nd,
ensure_min_samples, ensure_min_features, estimator,
input_name)
    900             # If input is 1D raise error
    901             if array.ndim == 1:
--> 902                 raise ValueError(
    903                     "Expected 2D array, got
1D array instead:\narray={}.\n"
    904                     "Reshape your data
```

```
reg.fit(student)
```

```
---------------------------------------------------
------------------------
TypeError                                 Traceback
(most recent call last)
<ipython-input-31-accc4baffd61> in <cell line: 1>()
----> 1 reg.fit(student)

TypeError: LinearRegression.fit() missing 1
```

```
features=features.reshape(25,1)
```

```
type(features)
```

```
numpy.ndarray
```

```
features.ndim
```

```
2
```

```
reg.fit(features,labels)
```

```
▾ LinearRegression
LinearRegression()
```

```
m=reg.coef_
```

```
m
```

```
array([9.77580339])
```

```
c=reg.intercept_
```

```
c
```

```
2.48367340537321
```

```
x=9
```

```
y=9.77580339*9+2.48367340537321
```

```
y
```

```
90.4659039153732
```

```
reg.predict([[9]])
```

```
array([90.46590392])
```

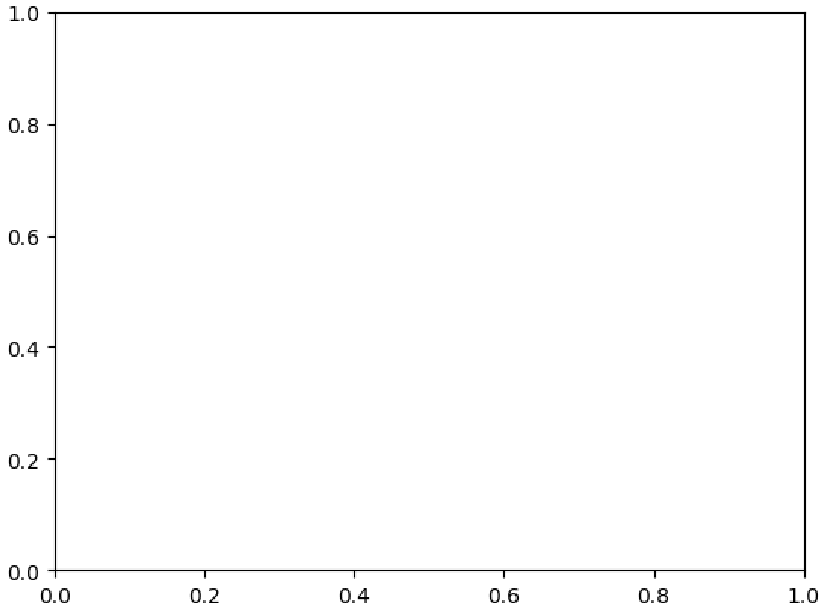
```
reg.predict([[5.1]])
```

```
array([52.3402707])
```

```
##########################          day 5      #########################
```

```
import matplotlib.pyplot as plt
plt.scatter(features,reg)
```

```
---------------------------------------------------
------------------------
ValueError                                Traceback
(most recent call last)
<ipython-input-45-5fc2ca8548af> in <cell line: 2>()
      1 import matplotlib.pyplot as plt
----> 2 plt.scatter(features,reg)

                    2 frames
/usr/local/lib/python3.10/dist-
packages/matplotlib/axes/_axes.py in scatter(self,
x, y, s, c, marker, cmap, norm, vmin, vmax, alpha,
linewidths, edgecolors, plotnonfinite, **kwargs)
   4582         y = np.ma.ravel(y)
   4583         if x.size != y.size:
-> 4584             raise ValueError("x and y must
be the same size")
   4585
   4586         if s is None:

ValueError: x and y must be the same size
```



```
reg.predict([[2.5]])
```

```
array([26.92318188])
```

```
student--->dataset--------> features and label
          train-test-split()
          train--->features_train,label_train
          test---->features_test,label_test
          fit(features_train,label_train)
          predict(features_test)
```

```
from sklearn.model_selection import train_test_split
features_train,features_test,label_train,labels_test=train_test_split(features,labels,test_s
```