*) Analysing Linear Regression using dummy data

```
import numpy as np
data = np.loadtxt ("data.csv", delimiter=",")
```

↓ default is space character

data.shape → (100, 2)

↓

splitting into x & y

↓

```
X = data [:,0] . reshape (-1, 1)
Y = data [:,1]
```

.fit function in sklearn requires 2D arrays. i.e. from (100,) to (100, 1).

↓ data split

```
from sklearn import model_selection
X_train, X_test, Y_train, Y_test = model_selection.
                                   train_test_split (X, Y)
```

↓ load LinearRegression

```
from sklearn.linear_model import LinearRegression
alg1 = LinearRegression ()
alg1.fit (X_train, Y_train)
```

↓

alg1 learns m and c value for the line fitted on the data.

↓

m → alg1.coef_[0] → returns the list of coefficient values for all features in the dataset.

↓

Here the length of the array is 1.

c → alg1.intercept_

↓

*) plotting the fit line wrt training data

```
import matplotlib.pyplot as plt
m = alg1.coef_[0]
c = alg1.intercept_

line_x = np.arange(30, 70, 0.1)
line_y = m*line_x + c
plt.plot(line_x, line_y)
```

↓

X_train will have to be reshaped to 1D array to be plotted.

↓

```
X_train1d = X_train.reshape(-1)
```

↓

```
plt.scatter(X_train1d, Y_train)
plt.grid()
plt.show()
```

*) The line can be plotted wrt to the test data to visualise how well the line fits.

alg. score (x_test , Y_tcet)

↓

coefficient of determination

→ coding linear regression (for single feature)

4 functions :

i) fit (X_train , Y_train) → m , c

ii) predict (x_test , m, c) → y_pred

iii) score (Y_test , Y_pred) → coefficient of determination

iv) cost (Y_test , Y_pred) → error value

↓

```
def fit ( X_train , Y_train ) :

    m_num = (x * y) . mean () - ( y. mean() * x.mean())
    m_den = (x * x) . mean() + (x . mean() * x.mean())

    m = m_num / m_den

    c = y.mean () - m * x.mean ()

    return m , c

def predict ( x_test , m, c ) :

    return x_test * m + c

def cost ( Y_test, Y_pred ):

    return (Y_test - Y_pred) ** 2
```

```
def score(y_test, y_pred):
    u = ((y_test - y_pred)** 2).sum()
    v = ((y_test - y_test.mean())** 2).sum()
    return 1 - (u / v)
```