In [1]:
```
import tensorflow as tf
import numpy as np
```

In [2]:
```
print("Tensorflow version : ",tf.__version__)
```

```
Tensorflow version :  2.3.0
```

In [3]:
```
# Input (temp, rainfall, humidity)
inputs = np.array([[73, 67, 43],
                   [91, 88, 64],
                   [87, 134, 58],
                   [102, 43, 37],
                   [69, 96, 70]], dtype='float32')
# Target (apples)
targets = np.array([[56],
                    [81],
                    [119],
                    [22],
                    [103]], dtype='float32')
m = np.shape(targets)
print("Data size is :",m[0])
```

```
Data size is : 5
```

In [4]:
```
x = tf.constant( inputs , dtype=tf.float32 )
y = tf.constant( targets , dtype=tf.float32)
print("Features :")
print(x)
print("Targets :")
print(y)
```

```
Features :
tf.Tensor(
[[ 73.  67.  43.]
 [ 91.  88.  64.]
 [ 87. 134.  58.]
 [102.  43.  37.]
 [ 69.  96.  70.]], shape=(5, 3), dtype=float32)
Targets :
tf.Tensor(
[[ 56.]
 [ 81.]
 [119.]
 [ 22.]
 [103.]], shape=(5, 1), dtype=float32)
```

In [5]:
```
#Add bias
bias = tf.ones([m[0],1],tf.float32)
new_input = tf.concat([x,bias],1)
print(new_input)
```

```
tf.Tensor(
[[ 73.  67.  43.   1.]
 [ 91.  88.  64.   1.]
 [ 87. 134.  58.   1.]
 [102.  43.  37.   1.]
 [ 69.  96.  70.   1.]], shape=(5, 4), dtype=float32)
```

In [6]:
```
#Intialize weight with random
random = tf.random.Generator.from_seed(74)
weight = random.normal(shape=[new_input.shape[1],1])
print(weight)
```

```
tf.Tensor(
[[-0.67008066]
 [-1.5614101 ]
 [ 0.6786617 ]
 [-1.0733451 ]], shape=(4, 1), dtype=float32)
```

In [7]:
```
#Define All Functions
def loss(y_pred,y):
  diff = y_pred-y
  diff_transpose = tf.transpose(diff)
  loss = tf.tensordot(diff_transpose,diff,axes=1)/(2*m[0])
  return loss

def predict(x,weight):
  y_pred = tf.tensordot(x,weight,axes=1)
  return y_pred

def gradientDescent(x,y,weight,alpha,num_of_epochs):
  for i in range(0,num_of_epochs):
     weight = weight - (alpha/m[0])*tf.tensordot(tf.transpose(x),(tf.tensordo
  return weight
```

In [8]:
```
#Intial pred
init_pred = predict(new_input,weight)
print("Init Predicate ans:")
print(init_pred)

#Intial loss
init_loss = loss(init_pred,y)
print("Init loss:")
print(float(init_loss))
```

```
Init Predicate ans:
tf.Tensor(
[[-125.42125]
 [-156.02043]
 [-229.23694]
 [-111.45172]
 [-149.69797]], shape=(5, 1), dtype=float32)
Init loss:
29202.693359375
```

In [9]:
```
num_of_epochs = 1000
alpha = 0.0001
#find out weight of each feature
final_weight = gradientDescent(new_input,y,weight,alpha,num_of_epochs)
```

In [10]:
```
print("Final weight:")
print(final_weight)
```

```
Final weight:
tf.Tensor(
[[-0.3919538]
 [ 0.8478244]
 [ 0.6945543]
 [-1.0719632]], shape=(4, 1), dtype=float32)
```

In [11]:
```
#Predict output
predicted_output = predict(new_input,final_weight)
print("predicted_output:")
print(predicted_output)
```

```
predicted_output:
tf.Tensor(
[[ 56.985477]
 [ 82.32027 ]
 [118.72068 ]
 [ 21.10371 ]
 [101.89317 ]], shape=(5, 1), dtype=float32)
```

In [12]:
```
final_cost = loss(predicted_output,y)
print("Final cost : ",float(final_cost))
```

```
Final cost :  0.48206907510757446
```