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

```
df=pd.read_excel("fake_reg.xlsx")
df.head()
```

	price	feature1	feature2
0	461.527929	999.787558	999.766096
1	548.130011	998.861615	1001.042403
2	410.297162	1000.070267	998.844015
3	540.382220	999.952251	1000.440940
4	546.024553	1000.446011	1000.338531

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   price     1000 non-null   float64
 1   feature1  1000 non-null   float64
 2   feature2  1000 non-null   float64
dtypes: float64(3)
memory usage: 23.6 KB
```

```
df.describe()
```

	price	feature1	feature2
count	1000.000000	1000.000000	1000.000000
mean	498.673029	1000.014171	999.979847
std	93.785431	0.974018	0.948330
min	223.346793	997.058347	996.995651
25%	433.025732	999.332068	999.316106
50%	502.382117	1000.009915	1000.002243
75%	564.921588	1000.637580	1000.645380
max	774.407854	1003.207934	1002.666308

```
sns.pairplot(data=df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f313644fa90>
```

```
df.describe()
```

	price	feature1	feature2
count	1000.000000	1000.000000	1000.000000
mean	498.673029	1000.014171	999.979847
std	93.785431	0.974018	0.948330
min	223.346793	997.058347	996.995651
25%	433.025732	999.332068	999.316106
50%	502.382117	1000.009915	1000.002243
75%	564.921588	1000.637580	1000.645380
max	774.407854	1003.207934	1002.666308

```
x=df.iloc[:,1:].values
x
```

```
array([[ 999.78755752,  999.7660962 ],
       [ 998.86161491, 1001.04240315],
       [1000.07026691,  998.84401463],
       ...,
       [1001.45164617,  998.84760554],
       [1000.77102275,  998.56285086],
       [ 999.2322436 , 1001.45140713]])
```

```
y=df['price'].values
y
```

```
array([461.52792939, 548.13001146, 410.29716167, 540.38221981,
       546.02455292, 542.9836716 , 555.48566416, 417.56972453,
       373.14653122, 633.35029248, 624.24709206, 475.37241721,
       600.36844486, 532.83295175, 472.8353628 , 506.31229096,
       508.414406  , 610.4553519 , 323.65776198, 446.21230389,
       362.12270299, 433.41064026, 562.00257647, 637.30962074,
       522.80800754, 469.8028243 , 543.10992778, 565.43416994,
       530.03285381, 610.58016503, 482.55641188, 327.56004052,
       579.73083872, 448.42981468, 628.97709187, 536.79737216,
       570.06729543, 357.82557519, 612.08492732, 444.67970846,
       600.1186364 , 523.53312776, 512.94994495, 614.8813169 ,
       404.35303251, 643.68851807, 488.95660398, 443.20468572,
       514.47906638, 514.88174058, 325.05852217, 554.6620585 ,
       451.39140001, 587.67887726, 477.73749721, 574.51560687,
       548.9107991 , 528.69088356, 443.21100482, 397.88209319,
       355.79535223, 460.69478138, 534.7673737 , 537.6067329 ,
       603.66990347, 547.27579153, 567.30862153, 454.32901321,
       492.60451447, 643.13593178, 477.4305163 , 497.41251837,
       559.04378534, 576.67843837, 292.1917811 , 542.99539996,
       547.73198239, 507.43439141, 408.12200074, 595.43348561,
       463.3552036 , 468.35698045, 462.00480189, 410.35912407,
       467.46059552, 497.23978613, 411.63794464, 583.06598409,
       441.95036691, 536.04090915, 530.98717439, 364.50842093,
       626.06865169, 434.62971003, 523.02344198, 484.10880902,
       411.57560703, 722.26394433, 519.46299177, 558.50321048,
       526.24297782, 531.27754687, 470.89067007, 459.92494538,
       577.58298501, 599.90573138, 584.92843756, 357.34223391,
       392.04726871, 515.77844597, 390.2092368 , 497.49148239,
       506.74032509, 395.51477512, 239.65626376, 507.6052943 ,
       628.84940437, 468.07324688, 444.9447339 , 527.31719596,
       415.44298463, 363.4303791 , 591.86909195, 391.12466656,
       526.1019196 , 402.00959595, 563.67781737, 362.24018346,
       514.31349869, 322.31435197, 513.519732  , 340.05814212,
       560.79471778, 489.79459784, 381.07367881, 558.34939279,
       451.8041724 , 451.79140134, 442.17202786, 488.78691774,
       433.09376956, 518.00709658, 511.69454654, 334.89459622,
       490.78489309, 510.46356567, 565.02540892, 542.30140508,
       655.42619806, 433.51753016, 373.06190748, 517.32199998,
       609.96955536, 401.28033698, 540.74381886, 410.53824963,
       441.7417974 , 448.70138654, 377.57042607, 450.85178618,
       695.08064351, 548.94851249, 479.70887661, 345.01181992,
       576.85255167, 337.95129476, 466.8394786 , 429.91541295,
       567.88189283, 422.77852605, 559.7847254 , 503.75687251,
       355.35228879, 622.37147383, 450.9760715 , 591.71413509,
       522.42993416, 474.08634639, 423.05243386, 466.70445806,
       481.92547045, 362.91867606, 566.05763935, 457.05958811,
       580.88060484, 497.33921897, 519.17237518, 640.40485258,
       477.25839763, 506.50370548, 634.2482933 , 311.87137765,
       556.61215428, 446.36031515, 543.83027912, 617.61509904,
       491.55128715, 441.90522674, 418.45717839, 524.47348399,
       483.13482709, 622.99764305, 424.60840775, 515.14753991,
       496.89047668, 534.23436696, 627.3802761 , 460.02329028,
       338.13384172, 397.62059618, 562.05793604, 504.69389644,
       490.99787402, 361.42640522, 608.12804021, 625.07938608,
       461.38078609, 533.07210805, 438.26540597, 404.14239475,
       458.52789767, 455.13953399, 521.87959462, 507.09798982,
       480.80694031, 539.68399729, 376.32625533, 449.458484  ,
       457.77794629. 531.16341758. 599.81185581. 526.67361757.
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
```

```
x
```

```
     array([[-0.23277507, -0.22551032],
            [-1.18389305,  1.12100994],
            [ 0.05762089, -1.19831824],
            ...,
            [ 1.47655837, -1.19452978],
            [ 0.77742953, -1.49494963],
            [-0.80318737,  1.55251439]])
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=6)
```

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

```
ann=Sequential()
```

```
ann.add(Dense(units=8,activation="relu"))
ann.add(Dense(units=8))
```

```
ann.compile(optimizer="adam",loss="mse")
```

```
ann.fit(x_train,y_train,epochs=200)
```

```
     Epoch 1/200
     22/22 [==============================] - 1s 2ms/step - loss: 255754.1250
     Epoch 2/200
     22/22 [==============================] - 0s 2ms/step - loss: 255676.4531
     Epoch 3/200
     22/22 [==============================] - 0s 2ms/step - loss: 255601.0781
     Epoch 4/200
     22/22 [==============================] - 0s 2ms/step - loss: 255524.2812
     Epoch 5/200
     22/22 [==============================] - 0s 2ms/step - loss: 255441.0000
     Epoch 6/200
     22/22 [==============================] - 0s 2ms/step - loss: 255348.0000
     Epoch 7/200
     22/22 [==============================] - 0s 2ms/step - loss: 255241.4844
     Epoch 8/200
     22/22 [==============================] - 0s 2ms/step - loss: 255121.6719
     Epoch 9/200
     22/22 [==============================] - 0s 2ms/step - loss: 254984.3438
     Epoch 10/200
     22/22 [==============================] - 0s 2ms/step - loss: 254829.1250
     Epoch 11/200
     22/22 [==============================] - 0s 2ms/step - loss: 254652.8750
     Epoch 12/200
     22/22 [==============================] - 0s 2ms/step - loss: 254454.9219
     Epoch 13/200
     22/22 [==============================] - 0s 2ms/step - loss: 254230.6562
     Epoch 14/200
     22/22 [==============================] - 0s 2ms/step - loss: 253982.4062
     Epoch 15/200
     22/22 [==============================] - 0s 2ms/step - loss: 253709.0000
     Epoch 16/200
     22/22 [==============================] - 0s 2ms/step - loss: 253407.7500
     Epoch 17/200
     22/22 [==============================] - 0s 2ms/step - loss: 253078.0156
     Epoch 18/200
     22/22 [==============================] - 0s 2ms/step - loss: 252723.7188
     Epoch 19/200
     22/22 [==============================] - 0s 2ms/step - loss: 252341.5312
     Epoch 20/200
     22/22 [==============================] - 0s 2ms/step - loss: 251932.9531
     Epoch 21/200
     22/22 [==============================] - 0s 2ms/step - loss: 251497.2188
     Epoch 22/200
```

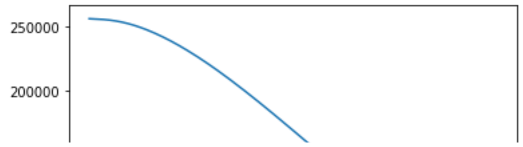
```
22/22 [==============================] - 0s 2ms/step - loss: 251040.6094
Epoch 23/200
22/22 [==============================] - 0s 2ms/step - loss: 250557.9688
Epoch 24/200
22/22 [==============================] - 0s 2ms/step - loss: 250049.1875
Epoch 25/200
22/22 [==============================] - 0s 2ms/step - loss: 249520.5469
Epoch 26/200
22/22 [==============================] - 0s 2ms/step - loss: 248962.4219
Epoch 27/200
22/22 [==============================] - 0s 2ms/step - loss: 248391.4531
Epoch 28/200
22/22 [==============================] - 0s 2ms/step - loss: 247795.4062
Epoch 29/200
```

```
ann.history.history
```

```
{'loss': [255754.125,
  255676.453125,
  255601.078125,
  255524.28125,
  255441.0,
  255348.0,
  255241.484375,
  255121.671875,
  254984.34375,
  254829.125,
  254652.875,
  254454.921875,
  254230.65625,
  253982.40625,
  253709.0,
  253407.75,
  253078.015625,
  252723.71875,
  252341.53125,
  251932.953125,
  251497.21875,
  251040.609375,
  250557.96875,
  250049.1875,
  249520.546875,
  248962.421875,
  248391.453125,
  247795.40625,
  247176.828125,
  246536.84375,
  245881.734375,
  245200.0,
  244501.625,
  243785.09375,
  243049.71875,
  242297.4375,
  241525.03125,
  240735.0625,
  239935.765625,
  239107.859375,
  238272.203125,
  237415.359375,
  236544.359375,
  235658.375,
  234758.125,
  233842.453125,
  232908.296875,
  231965.375,
  231006.421875,
  230031.640625,
  229051.703125,
  228044.953125,
  227038.609375,
  226014.0625,
  224975.5,
  223929.046875,
  222867.0,
  221805.03125,
```

```
loss=ann.history.history
loss_df=pd.DataFrame(loss)
plt.plot(loss_df['loss'])
```

```
[<matplotlib.lines.Line2D at 0x7f30d27c3a90>]
```



```
y_pred=ann.predict(x_test)
```

```
10/10 [==============================] - 0s 9ms/step
```

```
y_pred
```

```
array([[439.6848 , 446.07486, 440.11874, ..., 447.49136, 449.41287,
        458.0389 ],
       [342.35333, 336.4244 , 339.4998 , ..., 354.1713 , 345.4175 ,
        354.96097],
       [177.72949, 181.51526, 191.2848 , ..., 200.04102, 197.94427,
        185.71318],
       ...,
       [372.19482, 365.32672, 368.33957, ..., 384.57693, 375.40256,
        386.06216],
       [339.19556, 339.28888, 337.80942, ..., 347.62204, 345.01096,
        352.66788],
       [215.75162, 224.15443, 232.02335, ..., 238.47864, 239.3898 ,
        225.71269]], dtype=float32)
```

```
pd.DataFrame({"Actual value":y_test,"predicted value":y_pred.flatten()})
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-21-9a802400636d> in <module>
----> 1 pd.DataFrame({"Actual value":y_test,"predicted value":y_pred.flatten()})

                                    3 frames
/usr/local/lib/python3.9/dist-packages/pandas/core/internals/construction.py in
_extract_index(data)
    672             lengths = list(set(raw_lengths))
    673             if len(lengths) > 1:
--> 674                 raise ValueError("All arrays must be of the same length")
    675 
    676             if have_dicts:

ValueError: All arrays must be of the same length
```

SEARCH STACK OVERFLOW

```
from sklearn.metrics import r2_score
r2_score(y_pred,y_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-22-2e1e454e1c90> in <module>
      1 from sklearn.metrics import r2_score
----> 2 r2_score(y_pred,y_test)

                                    1 frames
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_regression.py in
_check_reg_targets(y_true, y_pred, multioutput, dtype)
    109 
    110     if y_true.shape[1] != y_pred.shape[1]:
--> 111         raise ValueError(
    112             "y_true and y_pred have different number of output ({0}!=
{1})".format(
    113                 y_true.shape[1], y_pred.shape[1]

ValueError: y_true and y_pred have different number of output (8!=1)
```

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred_train=lr.predict(x_train)
y_pred_test=lr.predict(x_test)
```

```
from sklearn.metrics import r2_score,mean_squared_error
def model_performance(y_actual,y_pred):
    r2=r2_score(y_actual,y_pred)
    RMSE=np.sqrt(mean_squared_error(y_actual,y_pred))
    print('R2 score:{}|RMSE:{}'.format(round(r2,2),round(RMSE,2)))
```

```
print('Train Performance')
r2_score(y_train,y_pred_train)
print('Test Performance')
r2_score(y_test,y_pred_test)
```

```
Train Performance
Test Performance
0.9970221031953577
```