Untitled25.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
import pandas as pd
import numpy as np
import keras
import sklearn
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
```

```
[7] concrete_data = pd.read_csv('https://cocl.us/concrete_data')
    concrete_data.head(10)
```

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30
5	266.0	114.0	0.0	228.0	0.0	932.0	670.0	90	47.03
6	380.0	95.0	0.0	228.0	0.0	932.0	594.0	365	43.70
7	380.0	95.0	0.0	228.0	0.0	932.0	594.0	28	36.45
8	266.0	114.0	0.0	228.0	0.0	932.0	670.0	28	45.85
9	475.0	0.0	0.0	228.0	0.0	932.0	594.0	28	39.29

```
[8] concrete_data.shape
```

```
(1030, 9)
```

```
[9] concrete_data.describe()
```

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Strength
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	73.895825	54.188350	181.567282	6.204660	972.918932	773.580485	45.662136	35.817961
std	104.506364	86.279342	63.997004	21.354219	5.973841	77.753954	80.175980	63.169912	16.705742
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.000000	594.000000	1.000000	2.330000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.000000	730.950000	7.000000	23.710000
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.000000	779.500000	28.000000	34.445000
75%	350.000000	142.950000	118.300000	192.000000	10.200000	1029.400000	824.000000	56.000000	46.135000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.000000	992.600000	365.000000	82.600000

```
[10] concrete_data.isnull().sum()
```

```
Cement              0
Blast Furnace Slag  0
Fly Ash             0
Water               0
Superplasticizer    0
Coarse Aggregate    0
Fine Aggregate      0
Age                 0
Strength            0
dtype: int64
```

```
[11] concrete_data_columns = concrete_data.columns
```

```
[12] target = concrete_data['Strength']
     target.head(10)
```

```
0    79.99
1    61.89
2    40.27
3    41.05
4    44.30
5    47.03
6    43.70
7    36.45
8    45.85
9    39.29
Name: Strength, dtype: float64
```

```
[13] predictors = concrete_data.iloc[:, :-1]
     predictors.head(10)
```

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360
5	266.0	114.0	0.0	228.0	0.0	932.0	670.0	90
6	380.0	95.0	0.0	228.0	0.0	932.0	594.0	365
7	380.0	95.0	0.0	228.0	0.0	932.0	594.0	28
8	266.0	114.0	0.0	228.0	0.0	932.0	670.0	28
9	475.0	0.0	0.0	228.0	0.0	932.0	594.0	28

```
[14] n_cols = predictors.shape[1]
     def regression_model():
         model = Sequential()
         model.add(Dense(10, activation='relu', input_shape=(n_cols,)))
         model.add(Dense(1))

         model.compile(optimizer='adam', loss='mean_squared_error')
         return model
```

```
[16] model = regression_model()
```

```
[19] list_of_mean_squared_error = []
     for cycle in range(50):
         X_train, X_test, y_train, y_test = train_test_split(predictors, target, test_size=0.3)
         res = model.fit(X_train, y_train, epochs=50, verbose=0, validation_data=(X_test, y_test))
         mean_squared_error = res.history['val_loss'][-1]
         list_of_mean_squared_error.append(mean_squared_error)
         print('Cycle #{}: mean_squared_error {}'.format(cycle+1, mean_squared_error))
```

```
Cycle #1: mean_squared_error 41.28075408935547
Cycle #2: mean_squared_error 54.21459197998047
Cycle #3: mean_squared_error 59.622718811035156
Cycle #4: mean_squared_error 53.92190170288086
Cycle #5: mean_squared_error 49.943138122558594
Cycle #6: mean_squared_error 52.33807754516016
Cycle #7: mean_squared_error 51.20048141479492
Cycle #8: mean_squared_error 48.66252136230469
Cycle #9: mean_squared_error 52.25123977661133
Cycle #10: mean_squared_error 50.22287368774414
Cycle #11: mean_squared_error 46.94742965698242
Cycle #12: mean_squared_error 48.91885757446289
Cycle #13: mean_squared_error 50.24974822998047
Cycle #14: mean_squared_error 48.705406188964844
Cycle #15: mean_squared_error 51.50592041015625
Cycle #16: mean_squared_error 55.116554260253906
Cycle #17: mean_squared_error 49.70653533935547
Cycle #18: mean_squared_error 49.84225845336914
Cycle #19: mean_squared_error 48.10299301147461
Cycle #20: mean_squared_error 46.61027908325195
Cycle #21: mean_squared_error 49.4514045715332
Cycle #22: mean_squared_error 48.370391845703125
Cycle #23: mean_squared_error 49.84026336669922
Cycle #24: mean_squared_error 53.76203536987305
Cycle #25: mean_squared_error 51.62146759033203
Cycle #26: mean_squared_error 42.66143035888672
Cycle #27: mean_squared_error 53.080326080322266
Cycle #28: mean_squared_error 49.580474853515625
Cycle #29: mean_squared_error 49.739219665527344
Cycle #30: mean_squared_error 55.8304443359375
Cycle #31: mean_squared_error 60.854408264160156
Cycle #32: mean_squared_error 44.44502639770508
Cycle #33: mean_squared_error 53.0426139831543
Cycle #34: mean_squared_error 51.09971618652344
Cycle #35: mean_squared_error 48.07730484008789
Cycle #36: mean_squared_error 46.719146728515625
Cycle #37: mean_squared_error 43.56586456298828
Cycle #38: mean_squared_error 45.751182556152344
Cycle #39: mean_squared_error 58.20309829711914
Cycle #40: mean_squared_error 47.78779983520508
Cycle #41: mean_squared_error 50.150028228759766
Cycle #42: mean_squared_error 55.6009635925293
Cycle #43: mean_squared_error 48.76174545288086
Cycle #44: mean_squared_error 57.0035400390625
Cycle #45: mean_squared_error 47.72990036010742
Cycle #46: mean_squared_error 49.98020935058594
Cycle #47: mean_squared_error 52.28877639770508
Cycle #48: mean_squared_error 46.959503173828125
Cycle #49: mean_squared_error 54.470703125
Cycle #50: mean_squared_error 51.947818756103516
```

```
[18] print('The mean of the mean squared errors: {}'.format(np.mean(list_of_mean_squared_error)))
     print('The standard deviation of the mean squared errors: {}'.format(np.std(list_of_mean_squared_error)))
```

```
The mean of the mean squared errors: 88.68272825387808
The standard deviation of the mean squared errors: 59.93241088768167
```

```
[20] predictors_norm = (predictors - predictors.mean())/predictors.std()
     predictors_norm.head(10)
```

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age
0	2.476712	-0.856472	-0.846733	-0.916319	-0.620147	0.862735	-1.217079	-0.279597
1	2.476712	-0.856472	-0.846733	-0.916319	-0.620147	1.055651	-1.217079	-0.279597
2	0.491187	0.795140	-0.846733	2.174405	-1.038638	-0.526262	-2.239829	3.551340
3	0.491187	0.795140	-0.846733	2.174405	-1.038638	-0.526262	-2.239829	5.055221
4	-0.790075	0.678079	-0.846733	0.488555	-1.038638	0.070492	0.647569	4.976069
5	-0.145138	0.464818	-0.846733	2.174405	-1.038638	-0.526262	-1.291914	0.701883
6	0.945704	0.244603	-0.846733	2.174405	-1.038638	-0.526262	-2.239829	5.055221
7	0.945704	0.244603	-0.846733	2.174405	-1.038638	-0.526262	-2.239829	-0.279597
8	-0.145138	0.464818	-0.846733	2.174405	-1.038638	-0.526262	-1.291914	-0.279597
9	1.854740	-0.856472	-0.846733	2.174405	-1.038638	-0.526262	-2.239829	-0.279597

```
[21] n_cols = predictors_norm.shape[1]
     def regression_model2():
         model2 = Sequential()
         model2.add(Dense(10, activation='relu', input_shape=(n_cols,)))
         model2.add(Dense(1))

         model2.compile(optimizer='adam', loss='mean_squared_error')
         return model2

     model2 = regression_model2()
```

```
[22] list_of_mean_squared_error = []
     for cycle in range(50):
         X_train, X_test, y_train, y_test = train_test_split(predictors_norm, target, test_size=0.3)
         res = model2.fit(X_train, y_train, epochs=50, verbose=0, validation_data=(X_test, y_test))
```

```
        mean_squared_error = res.history['val_loss'][-1]
        list_of_mean_squared_error.append(mean_squared_error)
        print('Cycle #{}: mean_squared_error {}'.format(cycle+1, mean_squared_error))
```

```
Cycle #1: mean_squared_error 248.80030822753906
Cycle #2: mean_squared_error 141.46994018554688
Cycle #3: mean_squared_error 129.63145446777344
Cycle #4: mean_squared_error 94.10957336425781
Cycle #5: mean_squared_error 82.68862915039062
Cycle #6: mean_squared_error 54.727657318115234
Cycle #7: mean_squared_error 56.879886627197266
Cycle #8: mean_squared_error 51.9527397157617
Cycle #9: mean_squared_error 54.18453598022461
Cycle #10: mean_squared_error 44.9055061340332
Cycle #11: mean_squared_error 49.151180267333984
Cycle #12: mean_squared_error 46.30262756347656
Cycle #13: mean_squared_error 42.41095733642578
Cycle #14: mean_squared_error 50.67615509033203
Cycle #15: mean_squared_error 49.512577056884766
Cycle #16: mean_squared_error 43.50089645385742
Cycle #17: mean_squared_error 41.03966522216797
Cycle #18: mean_squared_error 45.9974250793457
Cycle #19: mean_squared_error 38.036014556884766
Cycle #20: mean_squared_error 47.863807678222656
Cycle #21: mean_squared_error 45.801353454589844
Cycle #22: mean_squared_error 45.09261703491211
Cycle #23: mean_squared_error 39.28114318847656
Cycle #24: mean_squared_error 49.00874710083008
Cycle #25: mean_squared_error 44.22279357910156
Cycle #26: mean_squared_error 45.008399963378906
Cycle #27: mean_squared_error 47.40674591064453
Cycle #28: mean_squared_error 41.51381301879883
Cycle #29: mean_squared_error 41.762725830078125
Cycle #30: mean_squared_error 39.39617919921875
Cycle #31: mean_squared_error 46.577552795410156
Cycle #32: mean_squared_error 42.714786529541016
Cycle #33: mean_squared_error 43.330562591552734
Cycle #34: mean_squared_error 38.707881927490234
Cycle #35: mean_squared_error 34.853981018066406
Cycle #36: mean_squared_error 46.41719055175781
Cycle #37: mean_squared_error 40.367897033691406
Cycle #38: mean_squared_error 40.83223342895508
Cycle #39: mean_squared_error 47.64921951293945
Cycle #40: mean_squared_error 47.259098052978516
Cycle #41: mean_squared_error 44.61082077026367
Cycle #42: mean_squared_error 44.733367919921875
Cycle #43: mean_squared_error 38.70026397705078
Cycle #44: mean_squared_error 40.628543853759766
Cycle #45: mean_squared_error 44.6446647644043
Cycle #46: mean_squared_error 39.95962905883789
Cycle #47: mean_squared_error 40.66996765136719
Cycle #48: mean_squared_error 45.069679260253906
Cycle #49: mean_squared_error 40.36969757080078
Cycle #50: mean_squared_error 41.03737258911133
```

```
[23] print('The mean of the mean squared errors: {}'.format(np.mean(list_of_mean_squared_error)))
     print('The standard deviation of the mean squared errors: {}'.format(np.std(list_of_mean_squared_error)))
```

```
The mean of the mean squared errors: 54.02940933227539
The standard deviation of the mean squared errors: 34.274703722265215
```

```
[24] def regression_model3():
         model3 = Sequential()
         model3.add(Dense(10, activation='relu', input_shape=(n_cols,)))
         model3.add(Dense(1))

         model3.compile(optimizer='adam', loss='mean_squared_error')
         return model3

     model3 = regression_model3()
```

```
[ ] list_of_mean_squared_error = []
    for cycle in range(50):
        X_train, X_test, y_train, y_test = train_test_split(predictors_norm, target, test_size=0.3)
        res = model3.fit(X_train, y_train, epochs=100, verbose=0, validation_data=(X_test, y_test))
        mean_squared_error = res.history['val_loss'][-1]
        list_of_mean_squared_error.append(mean_squared_error)
        print('Cycle #{}: mean_squared_error {}'.format(cycle+1, mean_squared_error))
```

```
[25] print('The mean of the mean squared errors: {}'.format(np.mean(list_of_mean_squared_error)))
     print('The standard deviation of the mean squared errors: {}'.format(np.std(list_of_mean_squared_error)))
```

```
The mean of the mean squared errors: 54.02940933227539
The standard deviation of the mean squared errors: 34.274703722265215
```

```
[26] def regression_model4():
         model4 = Sequential()
         model4.add(Dense(10, activation='relu', input_shape=(n_cols,)))
         model4.add(Dense(10, activation='relu'))
         model4.add(Dense(10, activation='relu'))
         model4.add(Dense(1))

         model4.compile(optimizer='adam', loss='mean_squared_error')
         return model4
```

```
[27] model4 = regression_model4()
```

```
[28] list_of_mean_squared_error = []
     for cycle in range(50):
         X_train, X_test, y_train, y_test = train_test_split(predictors_norm, target, test_size=0.3)
         res = model4.fit(X_train, y_train, epochs=50, verbose=0, validation_data=(X_test, y_test))
         mean_squared_error = res.history['val_loss'][-1]
         list_of_mean_squared_error.append(mean_squared_error)
         print('Cycle #{}: mean_squared_error {}'.format(cycle+1, mean_squared_error))
```

```
Cycle #1: mean_squared_error 140.69586181640625
Cycle #2: mean_squared_error 99.71986389160156
Cycle #3: mean_squared_error 56.15390396118164
Cycle #4: mean_squared_error 39.3112907409668
Cycle #5: mean_squared_error 37.792240142822266
Cycle #6: mean_squared_error 38.635826110839844
Cycle #7: mean_squared_error 38.63139724731445
Cycle #8: mean_squared_error 28.976736068725586
Cycle #9: mean_squared_error 33.46844482421875
Cycle #10: mean_squared_error 34.9040641784668
```

```
Cycle #11: mean_squared_error 30.85513687133789
Cycle #12: mean_squared_error 28.800607681274414
Cycle #13: mean_squared_error 32.452003479003906
Cycle #14: mean_squared_error 30.408414840698242
Cycle #15: mean_squared_error 28.803272247314453
Cycle #16: mean_squared_error 32.60517501831055
Cycle #17: mean_squared_error 30.570201873779297
Cycle #18: mean_squared_error 25.309446334838867
Cycle #19: mean_squared_error 31.58930015563965
Cycle #20: mean_squared_error 29.151702880859375
Cycle #21: mean_squared_error 32.01768493652344
Cycle #22: mean_squared_error 28.38570785522461
Cycle #23: mean_squared_error 26.597782135009766
Cycle #24: mean_squared_error 28.345102310180664
Cycle #25: mean_squared_error 26.32594108581543
Cycle #26: mean_squared_error 27.666837692260742
Cycle #27: mean_squared_error 23.030284881591797
Cycle #28: mean_squared_error 26.094005584716797
Cycle #29: mean_squared_error 27.9251651763916
Cycle #30: mean_squared_error 25.565250396728516
Cycle #31: mean_squared_error 23.098196029663086
Cycle #32: mean_squared_error 20.767671585083008
Cycle #33: mean_squared_error 26.475229263305664
Cycle #34: mean_squared_error 26.13245964050293
Cycle #35: mean_squared_error 27.032794952392578
Cycle #36: mean_squared_error 25.122724533081055
Cycle #37: mean_squared_error 25.895462036132812
Cycle #38: mean_squared_error 24.954914093017578
Cycle #39: mean_squared_error 24.01709747314453
Cycle #40: mean_squared_error 22.40388679504395
Cycle #41: mean_squared_error 24.37986946105957
Cycle #42: mean_squared_error 23.823993682861328
Cycle #43: mean_squared_error 28.109956741333008
Cycle #44: mean_squared_error 26.67770004272461
Cycle #45: mean_squared_error 23.435436248779297
Cycle #46: mean_squared_error 21.935230255126953
Cycle #47: mean_squared_error 24.863786697387695
Cycle #48: mean_squared_error 20.726438522338867
Cycle #49: mean_squared_error 23.882659912109375
Cycle #50: mean_squared_error 18.733623504638672
```

```
[29] print('The mean of the mean squared errors: {}'.format(np.mean(list_of_mean_squared_error)))
     print('The standard deviation of the mean squared errors: {}'.format(np.std(list_of_mean_squared_error)))
```

```
The mean of the mean squared errors: 32.06515567779541
The standard deviation of the mean squared errors: 19.43531141042595
```