In [1]:
```
#importing the main libraries that we need them for our regression model
```

In [2]:
```
import pandas as pd
import numpy as np
import warnings
warnings.simplefilter('ignore', FutureWarning)
```

In [3]:
```
#we will download our dataset
concrete_data=pd.read_csv('https://s3-api.us-geo.objectstorage.softlayer.
concrete_data.head() #to see the first 5 data line  of the dataset
```

Out[3]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	St
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	

In [4]:
```
#to clean the dataset beofore using it , checking for any mising values
```

In [5]:
```
concrete_data.shape
```

Out[5]:
```
(1030, 9)
```

In [6]:
```
concrete_data.describe()
```

Out[6]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	A
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	103
mean	281.167864	73.895825	54.188350	181.567282	6.204660	97
std	104.506364	86.279342	63.997004	21.354219	5.973841	7
min	102.000000	0.000000	0.000000	121.800000	0.000000	80
25%	192.375000	0.000000	0.000000	164.900000	0.000000	93
50%	272.900000	22.000000	0.000000	185.000000	6.400000	96
75%	350.000000	142.950000	118.300000	192.000000	10.200000	102
max	540.000000	359.400000	200.100000	247.000000	32.200000	114

In [7]:
```
concrete_data.isnull().sum()
```

Out[7]:
```
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
```

In [8]:
```
concrete_data_columns=concrete_data.columns
predictors=concrete_data[concrete_data_columns[concrete_data_columns!='St
target=concrete_data['Strength'] #the Strenght cloumn is the Target
```

In [9]:
```
#we can store the predictors number in a variable called pred_no
concrete_no=concrete_data.shape[1]
predictors.head(9)
```

Out[9]:

	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

In [10]:
```
target.head(9)
```

Out[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
Name: Strength, dtype: float64
```

In [11]:
```
#first normalize the data then use it withthe model
```

In [12]:
```
predictors_normal=predictors-(predictors.mean())/predictors.std()
```

In [13]:
```
predictors_normal.head(5)
```

Out[13]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	
0	537.309562	-0.856472	-0.846733	153.497358	1.461362	1027.48721	6
1	537.309562	-0.856472	-0.846733	153.497358	1.461362	1042.48721	6
2	329.809562	141.643528	-0.846733	219.497358	-1.038638	919.48721	5
3	329.809562	141.643528	-0.846733	219.497358	-1.038638	919.48721	5
4	195.909562	131.543528	-0.846733	183.497358	-1.038638	965.88721	8

In [14]:
```
pred_no=predictors_normal.shape[1]
```

B.Build use the normalized data

In [15]:
```
import keras #first import keras and sklearn using this code
import sklearn
```

```
Using TensorFlow backend.
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/sklearn/uti
ls/validation.py:37: DeprecationWarning: distutils Version classes are dep
recated. Use packaging.version instead.
  LARGE_SPARSE_SUPPORTED = LooseVersion(scipy_version) >= '0.14.0'
```

In [16]:
```
#import the important pakages from keras liabrary to build a regression m
```

In [17]:
```
from keras.models import Sequential
```

In [18]:
```
from keras.layers import Dense  #to create our layers
```

In [19]:
```
#we split the data using sklearn
```

In [20]:
```
def regression_model():
    #define our neural network with one hidden layers
    model=Sequential()
    model.add(Dense(10, activation='relu', input_shape=(pred_no,)))
    model.add(Dense(10, activation='relu'))
    model.add(Dense(10, activation='relu'))
    model.add(Dense(1))
    #sompling our model using adam optimizer and the mean squared error a
    model.compile(optimizer='adam',loss='mean_squared_error')
```

In [21]:
```
from sklearn.model_selection import train_test_split
x=predictors_normal
y=concrete_data['Strength']
y.head()
```

Out[21]:
```
0    79.99
1    61.89
2    40.27
3    41.05
4    44.30
Name: Strength, dtype: float64
```

In [22]:
```
x.head()
```

Out[22]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	
0	537.309562	-0.856472	-0.846733	153.497358	1.461362	1027.48721	6
1	537.309562	-0.856472	-0.846733	153.497358	1.461362	1042.48721	6
2	329.809562	141.643528	-0.846733	219.497358	-1.038638	919.48721	5
3	329.809562	141.643528	-0.846733	219.497358	-1.038638	919.48721	5
4	195.909562	131.543528	-0.846733	183.497358	-1.038638	965.88721	8

In [23]:
```
x.shape
```

Out[23]:
```
(1030, 8)
```

In [24]:
```
#split the dataset to 30% for  test and the rest for training
```

In [25]:
```
x_train,x_test,y_train,y_test =train_test_split(x,y,random_state=10,test_
```

In [26]:
```
x_train.shape
```

Out[26]:
```
(721, 8)
```

In [27]:
```
y_test.shape
```

Out[27]:
```
(309,)
```

In [28]:
```
#training the model using 50 epochs
#first we call the model function that we have defiened it above  to buil
model=regression_model()
```

```
WARNING:tensorflow:From /home/jupyterlab/conda/envs/python/lib/python3.7/s
ite-packages/keras/backend/tensorflow_backend.py:68: The name tf.get_defau
lt_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /home/jupyterlab/conda/envs/python/lib/python3.7/s
ite-packages/keras/backend/tensorflow_backend.py:508: The name tf.placehol
der is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /home/jupyterlab/conda/envs/python/lib/python3.7/s
ite-packages/keras/backend/tensorflow_backend.py:3837: The name tf.random_
uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /home/jupyterlab/conda/envs/python/lib/python3.7/s
ite-packages/keras/optimizers.py:757: The name tf.train.Optimizer is depre
cated. Please use tf.compat.v1.train.Optimizer instead.
```

In [29]:
```
#we will train our model on the training data using fit function
```

In [30]:
```
model_fit=model.fit(x_train,y_train,epochs=50, verbose=0,validatio_data=(
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_1435/2163848.py in <module>
----> 1 model_fit=model.fit(x_train,y_train,epochs=50, verbose=0,validation_data=(x_test,y_test ))

AttributeError: 'NoneType' object has no attribute 'fit'
```

In [31]:
```
#evalute the model
scores=model.evaluate(x_test,y_test,verbose=0)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_1435/2593022917.py in <module>
      1 #evalute the model
----> 2 scores=model.evaluate(x_test,y_test,verbose=0)

AttributeError: 'NoneType' object has no attribute 'evaluate'
```

In [32]:
```
#computing the mean squarred error and print it
mean_squared_error=model_fit.history['val_loss'][-1]
print(mean_squared_error)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_1435/830681345.py in <module>
      1 #computing the mean squarred error and print it
----> 2 mean_squared_error=model_fit.history['val_loss'][-1]
      3 print(mean_squared_error)

NameError: name 'model_fit' is not defined
```

In [33]:
```
#we will repeat steps 1 to 3 50 times and calculate the error in each tim
```

In [34]:
```
msq_list= [] #create the mean squared error list in each iteration
for train_cycle in range(50):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=10,te
    model_fit=model.fit(x_train,y_train,epochs=50,verbose=0,validation_da
    mean_squared_error=model_fit.history['val_loss'][-1] #calculate the m
    msq_list.append(mean_squared_error) #fill up the mean squared error l
    print('train_cycle {}: mean_squared_error{}'.format(train_cycle+1,mea
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_1435/98300980.py in <module>
      2 for train_cycle in range(50):
      3     x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=10,test_size=0.30) #split the data to test_size 30%
----> 4     model_fit=model.fit(x_train,y_train,epochs=50,verbose=0,validation_data=(x_test,y_test)) #train and validate  the model
      5     mean_squared_error=model_fit.history['val_loss'][-1] #calculate the mean squared error
      6     msq_list.append(mean_squared_error) #fill up the mean squared error list during the 50 cycles

AttributeError: 'NoneType' object has no attribute 'fit'
```

In [35]:
```
#report the mean and the standard deviation of the mean squared error
```

In [36]:
```
print('the mean of the mean squared errors is : {}'.format(np.mean(msq_li
print('the standard deviation of the mean squared errors is : {}'.format(
```

```
the mean of the mean squared errors is : nan
the standard deviation of the mean squared errors is : nan
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/numpy/core/fromnumeric.py:3441: RuntimeWarning: Mean of empty slice.
  out=out, **kwargs)
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/numpy/core/_methods.py:189: RuntimeWarning: invalid value encountered in double_scalars
  ret = ret.dtype.type(ret / rcount)
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/numpy/core/_methods.py:263: RuntimeWarning: Degrees of freedom <= 0 for slice
  keepdims=keepdims, where=where)
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/numpy/core/_methods.py:223: RuntimeWarning: invalid value encountered in true_divide
  subok=False)
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/numpy/core/_methods.py:254: RuntimeWarning: invalid value encountered in double_scalars
  ret = ret.dtype.type(ret / rcount)
```

The mean of the mean squared error when w use 100 epochs is a bit higher than the case B using the 50 epochs,which means even increasing the epochs dose not help with the performance of the model

In [ ]:

In [ ]: