```
## ML  TYPES OF GD

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('./Datasets/Student_Performance.csv')

dataset.head()
```

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91.0
1	4	82	No	4	2	65.0
2	8	51	Yes	7	2	45.0
3	5	52	Yes	5	2	36.0
4	7	75	No	8	5	66.0

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Hours Studied                     10000 non-null  int64  
 1   Previous Scores                   10000 non-null  int64  
 2   Extracurricular Activities        10000 non-null  object 
 3   Sleep Hours                       10000 non-null  int64  
 4   Sample Question Papers Practiced  10000 non-null  int64  
 5   Performance Index                 10000 non-null  float64
dtypes: float64(1), int64(4), object(1)
memory usage: 468.9+ KB
```

```
dataset.isnull().sum()
```

```
Hours Studied                       0
Previous Scores                     0
Extracurricular Activities          0
Sleep Hours                         0
Sample Question Papers Practiced    0
Performance Index                   0
dtype: int64
```

```
Features = dataset.drop(columns = ['Performance Index'], axis = 1)

target = dataset['Performance Index']

## Lable encoding

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
Features['Extracurricular Activities'] =
le.fit_transform(Features['Extracurricular Activities'])

Features
```

```
      Hours Studied  Previous Scores  Extracurricular Activities
Sleep Hours  \
0                 7               99                           1
9
1                 4               82                           0
4
2                 8               51                           1
7
3                 5               52                           1
5
4                 7               75                           0
8
...             ...              ...                         ...
...
9995              1               49                           1
4
9996              7               64                           1
8
9997              6               83                           1
8
9998              9               97                           1
7
9999              7               74                           0
8

      Sample Question Papers Practiced
0                                    1
1                                    2
2                                    2
```

```
3                                     2
4                                     5
...                                 ...
9995                                  2
9996                                  5
9997                                  5
9998                                  0
9999                                  1

[10000 rows x 5 columns]
```

```
## One Hot Encoding

Features['Extracurricular Activities'] =
pd.get_dummies(Features['Extracurricular
Activities'],drop_first=True).astype('int')

Features
```

```
      Hours Studied  Previous Scores  Extracurricular Activities
Sleep Hours  \
0                 7               99                           1
9
1                 4               82                           0
4
2                 8               51                           1
7
3                 5               52                           1
5
4                 7               75                           0
8
...             ...              ...                         ...
...
9995              1               49                           1
4
9996              7               64                           1
8
9997              6               83                           1
8
9998              9               97                           1
7
9999              7               74                           0
8

      Sample Question Papers Practiced
0                                     1
1                                     2
2                                     2
3                                     2
4                                     5
```

```
...                                   ...
9995                                    2
9996                                    5
9997                                    5
9998                                    0
9999                                    1

[10000 rows x 5 columns]
```

```
from sklearn.model_selection import train_test_split
Training_Features , Testing_Features , Y_Feat_target , Y_target =
train_test_split(Features,target,random_state = 0,test_size = 0.25)

print(Training_Features.shape)
```

```
(7500, 5)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
Training_Features = sc.fit_transform(Training_Features)
Testing_Features = sc.transform(Testing_Features)

print(Training_Features.shape)
print(Y_Feat_target.shape)
print(type(Training_Features))
print(type(Y_Feat_target))
```

```
(7500, 5)
(7500,)
<class 'numpy.ndarray'>
<class 'pandas.core.series.Series'>
```

```
def Mini_batch(X_train,Y_train,epochs=100,batch_size = 75,lr = 0.01):
    Y_train = Y_train.values.reshape(-1,1)
    m , n = X_train.shape
    W = np.zeros((n,1))
    b = 0
    for epoch in range(epochs):
        indices = np.arange(m)
        np.random.shuffle(indices)
        for i in range(0,m,batch_size):
            batch_indx = indices[i:i+batch_size]
            X_batch = X_train[batch_indx]
            Y_batch = Y_train[batch_indx]
            Y_pred = np.dot(X_batch , W ) + b

            dW = np.dot(X_batch.T , (Y_pred - Y_batch))  /
len(X_batch)
            db = np.sum((Y_pred - Y_batch))  / len(X_batch)
```

```
            W -= lr * dW
            b -= lr * db
    return W,b


from sklearn.linear_model import LinearRegression , SGDRegressor
lr = LinearRegression()
sgd = SGDRegressor(loss = 'squared_error',eta0 = 0.01 , max_iter =
1000)
lr.fit(Training_Features,Y_Feat_target)
sgd.fit(Testing_Features,Y_target)
## MIni batch
W,b = Mini_batch(Training_Features,Y_Feat_target)

Y_pred_lr = lr.predict(Testing_Features)
Y_pred_sgd = sgd.predict(Testing_Features)
Y_pred_mbgd = np.dot(Testing_Features , W ) + b

from sklearn.metrics import mean_squared_error , r2_score
lr_mse = mean_squared_error(Y_target , Y_pred_lr)
lr_r2_score = r2_score(Y_target , Y_pred_lr)
sgd_mse = mean_squared_error(Y_target , Y_pred_sgd)
sgd_r2_score = r2_score(Y_target , Y_pred_sgd)
mbgd_mse = mean_squared_error(Y_target , Y_pred_mbgd)
mbgd_r2_score = r2_score(Y_target , Y_pred_mbgd)

print(f"SGD : MSE :{sgd_mse} , R : {sgd_r2_score}")
print(f"LR : MSE :{lr_mse} , R : {lr_r2_score}")
print(f"MBGD : MSE :{mbgd_mse} , R : {mbgd_r2_score}")

SGD : MSE :4.07100208302151 , R : 0.9885015000372696
LR : MSE :4.0827039600401065 , R : 0.9884684482161901
MBGD : MSE :4.085254205825721 , R : 0.988461245075422
```