```
import numpy as np
class GDRegression:

    def __init__(self,epoch=100,learning_rate= 0.01):

        self.epoch = epoch
        self.learning_rate = learning_rate
        self.coef_ = None
        self.intercept_ = None

    def fit(self,X_train,y_train):
        self.intercept_ = 0
        self.coef_ = np.ones(X_train.shape[1])

        for i in range(self.epoch):

            y_hat = np.dot(X_train , self.coef_) + self.intercept_

            intercept_der = -2 * np.mean(y_train - y_hat)
            self.intercept_ = self.intercept_ - (self.learning_rate *
intercept_der)

            coef_der = -2 * np.dot((y_train -
y_hat),X_train)/X_train.shape[0]
            self.coef_ = self.coef_ - (self.learning_rate * coef_der)

    def predict(self,X_test):
        return np.dot(X_test,self.coef_) + self.intercept_


class SGDRegression:

    def __init__(self,epoch=100,learning_rate= 0.01):

        self.epoch = epoch
        self.learning_rate = learning_rate
        self.coef_ = None
        self.intercept_ = None

    def fit(self,X_train,y_train):
        self.intercept_ = 0
        self.coef_ = np.ones(X_train.shape[1])

        for i in range(self.epoch):
            for j in range(X_train.shape[0]):
                id = np.random.randint(0,X_train.shape[0])

                y_hat = np.dot(X_train[id] ,self.coef_) +
self.intercept_
                intercept_der = -2 * (y_train[id] - y_hat)
```

```
                self.intercept_ = self.intercept_ -
(self.learning_rate * intercept_der)

                coef_der  = -2 * np.dot((y_train[id] -
y_hat),X_train[id])
                self.coef_ = self.coef_ - (self.learning_rate *
coef_der)


    def predict(self,X_test):
        return np.dot(X_test,self.coef_) + self.intercept_



class MBGDRegression:

    def __init__(self,batch_size=10,learning_rate=0.01,epoch=100):

        self.batch_size = batch_size
        self.learning_rate = learning_rate
        self.epoch = epoch
        self.coef_ = None
        self.intercept_ = None


    def fit(self,X_train,y_train):

        y_train = np.ravel(y_train)
        self.coef_ = np.ones(X_train.shape[1])
        self.intercept_ = 0


        for i  in range(self.epoch):
            for j in range(0,X_train.shape[0], self.batch_size):

                X_batch = X_train[j:j+self.batch_size]
                y_batch = y_train[j:j+self.batch_size]

                y_hat = np.dot(X_batch,self.coef_) + self.intercept_

                intercept_der = -2 * np.mean(y_batch - y_hat)
                self.intercept_ = self.intercept_ -
(self.learning_rate * intercept_der)

                coef_der = -2 * np.dot((y_batch - y_hat),X_batch)/
X_batch.shape[0]
                self.coef_ = self.coef_ - (self.learning_rate *
coef_der)

    def predict(self,X_test):
```

```
        return np.dot(X_test,self.coef_) + self.intercept_




import numpy as np
import pandas as pd
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Load diabetes dataset
diabetes = load_diabetes()
X = diabetes.data
y = diabetes.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

def adjusted_r2_score(y_true, y_pred, n_features):
    r2 = r2_score(y_true, y_pred)
    n = len(y_true)
    return 1 - (1 - r2) * (n - 1) / (n - n_features - 1)

# Initialize models
models = {
    "GDRegression": GDRegression(epoch=1000, learning_rate=0.01),
    "SGDRegression": SGDRegression(epoch=1000, learning_rate=0.01),
    "MBGDRegression": MBGDRegression(batch_size=20, epoch=1000,
learning_rate=0.01)
}

# Train and evaluate models
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Calculate metrics
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    adj_r2 = adjusted_r2_score(y_test, y_pred, X_train.shape[1])

    print(f"{model_name}:")
    print(f"  MSE: {mse:.4f}")
    print(f"  R2: {r2:.4f}")
```

```
    print(f"  Adjusted R2: {adj_r2:.4f}")
    print("-" * 40)
```

```
GDRegression:
  MSE: 4703.6028
  R2: 0.1122
  Adjusted R2: -0.0016
----------------------------------------
SGDRegression:
  MSE: 2947.3570
  R2: 0.4437
  Adjusted R2: 0.3724
----------------------------------------
MBGDRegression:
  MSE: 2923.3693
  R2: 0.4482
  Adjusted R2: 0.3775
----------------------------------------
```