BLENDED_LEARNING

Implementation-of-Stochastic-Gradient-Descent-SGD-Regressor

AIM:

To write a program to implement Stochastic Gradient Descent (SGD) Regressor for linear regression and evaluate its performance.

Equipments Required:

1. Hardware – PCs
2. Anaconda – Python 3.7 Installation / Jupyter notebook

Algorithm

1.Import Necessary Libraries: Import required libraries such as pandas, numpy, matplotlib, sklearn for the implementation.

2.Load the Dataset: Load the dataset (e.g., CarPrice_Assignment.csv) using pandas.

3.Data Preprocessing: Drop unnecessary columns (e.g., 'CarName', 'car_ID'). Handle categorical variables using pd.get_dummies().

4.Split the Data: Split the dataset into features (X) and target variable (Y). Split the data into training and testing sets using train_test_split().

5.Standardize the Data: Standardize the feature data (X) and target variable (Y) using StandardScaler() to ensure they have mean=0 and variance=1.

6.Create the SGD Regressor Model: Initialize the SGD Regressor model with max_iter=1000 and tol=1e-3.

7.Train the Model: Fit the model to the training data using the fit() method.

8.Make Predictions: Use the trained model to predict the target values for the test set.

9.Evaluate the Model: Calculate performance metrics like Mean Squared Error (MSE) and R-squared score using mean_squared_error() and r2_score().

10.Display Model Coefficients: Display the model's coefficients and intercept.

11.Visualize the Results: Create a scatter plot comparing actual vs predicted prices.

12.End: The program finishes by displaying the evaluation metrics, model coefficients, and a visual representation of the predictions.

Program:

```
/*
Program to implement SGD Regressor for linear regression.
Developed by: GANESH PRABHU J
RegisterNumber: 212223220023
*/
```

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# Load the dataset
data = pd.read_csv('CarPrice_Assignment.csv')
print(data.head())
print("\n\n")
print(data.info())

# Data preprocessing
# Dropping unnecessary columns and handling categorical variables
data = data.drop(['CarName', 'car_ID'], axis=1)
data = pd.get_dummies(data, drop_first=True)

# Splitting the data into features and target variable
x = data.drop('price', axis=1)
y = data['price']

# Standardizing the data
scaler = StandardScaler()
x = scaler.fit_transform(x)
y = scaler.fit_transform(np.array(y).reshape(-1, 1)).ravel()


# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=

# Creating the SGD Regressor model
sgd_model = SGDRegressor(max_iter=1000, tol=1e-3)

# Fitting the model on the training data
sgd_model.fit(x_train, y_train)
```

```
# Making predictions
y_pred = sgd_model.predict(x_test)

# Evaluating model performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print evaluation metrics
print()
print()
print("Mean Squared Error:", mse)
print("R-squared Score:", r2)

# Print model coefficients
print("\n\n")
print("Model Coefficients")
print("Coefficients:", sgd_model.coef_)
print("Intercept:", sgd_model.intercept_)

# Visualizing actual vs predicted prices
print("\n\n")
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices using SGD Regressor")
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red')  # Perfe
plt.show()
```

Output:

simple linear regression model for predicting the marks scored

LOAD THE DATASET:

```
   car_ID  symboling                   CarName fueltype aspiration doornumber  \
0       1          3        alfa-romero giulia      gas        std        two
1       2          3       alfa-romero stelvio      gas        std        two
2       3          1  alfa-romero Quadrifoglio      gas        std        two
3       4          2               audi 100 ls      gas        std       four
4       5          2                audi 100ls      gas        std       four

       carbody drivewheel enginelocation  wheelbase  ...  enginesize  \
0  convertible        rwd          front       88.6  ...         130
1  convertible        rwd          front       88.6  ...         130
2    hatchback        rwd          front       94.5  ...         152
3        sedan        fwd          front       99.8  ...         109
4        sedan        4wd          front       99.4  ...         136

   fuelsystem  boreratio  stroke compressionratio horsepower  peakrpm citympg  \
0        mpfi       3.47    2.68              9.0        111     5000      21
1        mpfi       3.47    2.68              9.0        111     5000      21
2        mpfi       2.68    3.47              9.0        154     5000      19
3        mpfi       3.19    3.40             10.0        102     5500      24
4        mpfi       3.19    3.40              8.0        115     5500      18

   highwaympg    price
0          27  13495.0
1          27  16500.0
2          26  16500.0
3          30  13950.0
4          22  17450.0

[5 rows x 26 columns]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
None
```

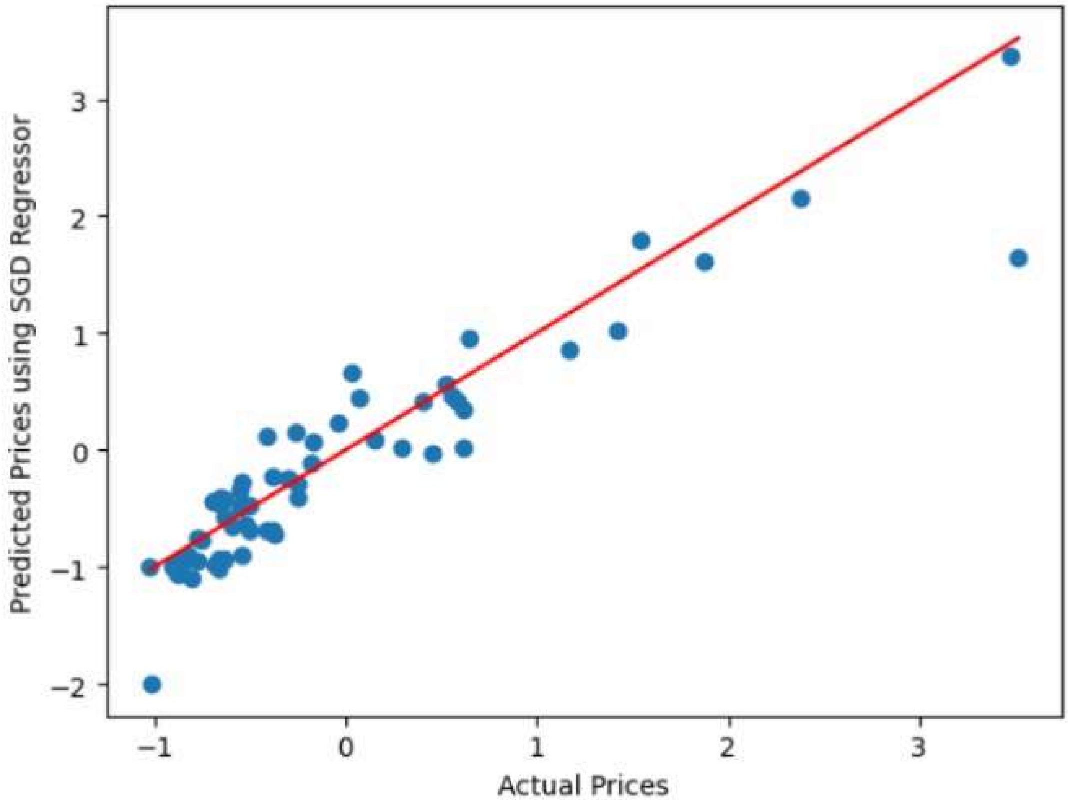
EVALUATION METRICS:

```
Mean Squared Error: 0.1342093761275513
R-squared Score: 0.8588105274203248
```

MODEL COEFFICIENTS:

```
Model Coefficients
Coefficients: [ 5.95324665e-02  1.04621666e-01 -1.50496499e-02  2.02200858e-01
  1.59477431e-02  1.70910766e-01  3.28878621e-01 -4.72514838e-02
 -8.08308399e-02 -1.31203362e-02  1.31879946e-01  2.72801451e-04
 -7.62997727e-03 -2.10543720e-02 -1.98904900e-02  7.10134411e-03
  7.59905720e-03 -4.16349472e-02 -9.69939527e-02 -1.84480888e-02
 -6.89098925e-02 -5.18959127e-02  7.79779092e-02  2.22160463e-01
  9.60416043e-03 -8.76711936e-02  8.80296430e-02  7.85793361e-05
  7.17936131e-03  3.48416168e-03 -4.29482616e-02 -1.59998058e-01
 -9.00523043e-02  1.65784211e-03 -3.40666404e-02  3.48416168e-03
 -1.81138301e-02  3.05131038e-03  1.98904900e-02  1.65784211e-03
 -3.87934514e-02 -3.60520608e-02 -1.76071141e-02]
Intercept: [-0.02368564]
```

VISUALIZATION OF ACTUAL VS PREDICTED VALUES:



Result:

Thus, the implementation of Stochastic Gradient Descent (SGD) Regressor for linear regression has been successfully demonstrated and verified using Python programming.