CS 6375
ASSIGNMENT 1

Names of students in your group:

Samyak Rokade (SJR220000)
Anthea Abreo (AXA210122)

Number of free late days used: 0

Note: You are allowed a **total** of 4 free late days for the **entire semester**. You can use at most 2 for each assignment. After that, there will be a penalty of 10% for each late day.

Please list clearly all the sources/references that you have used in this assignment.

1. Pandas: https://pandas.pydata.org/docs/
2. Scikit-Learn (sklearn): https://scikit-learn.org/stable/documentation.html
3. SGDRegressor from Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html
4. StandardScaler from Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
5. train_test_split from Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
6. Matplotlib (plt): https://matplotlib.org/stable/contents.html
7. Mean squared error (mean_squared_error) from Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html
8. Mean absolute error (mean_absolute_error) from Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html
9. R-squared (r2_score) from Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html
10. Explained variance score (explained_variance_score) from Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.explained_variance_score.html
11. NumPy (np): https://numpy.org/doc/stable/

Linear Regression Using Gradient Descent (Assignment 1)

By Anthea Abreo(AXA210122) and Samyak Rokade(SJR220000)

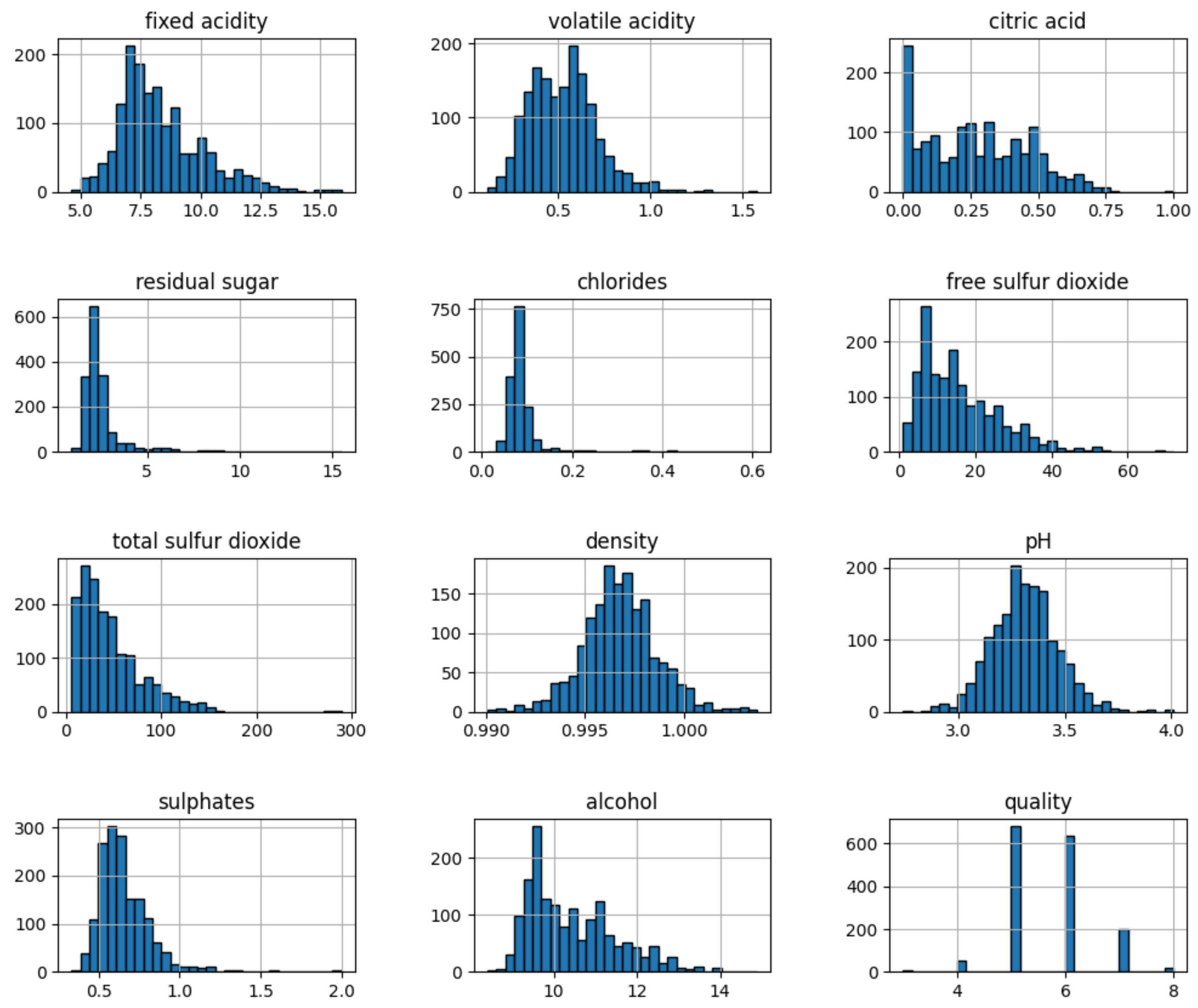
Dataset:

Wine Quality: https://archive.ics.uci.edu/dataset/186/wine+quality

Data Preprocessing

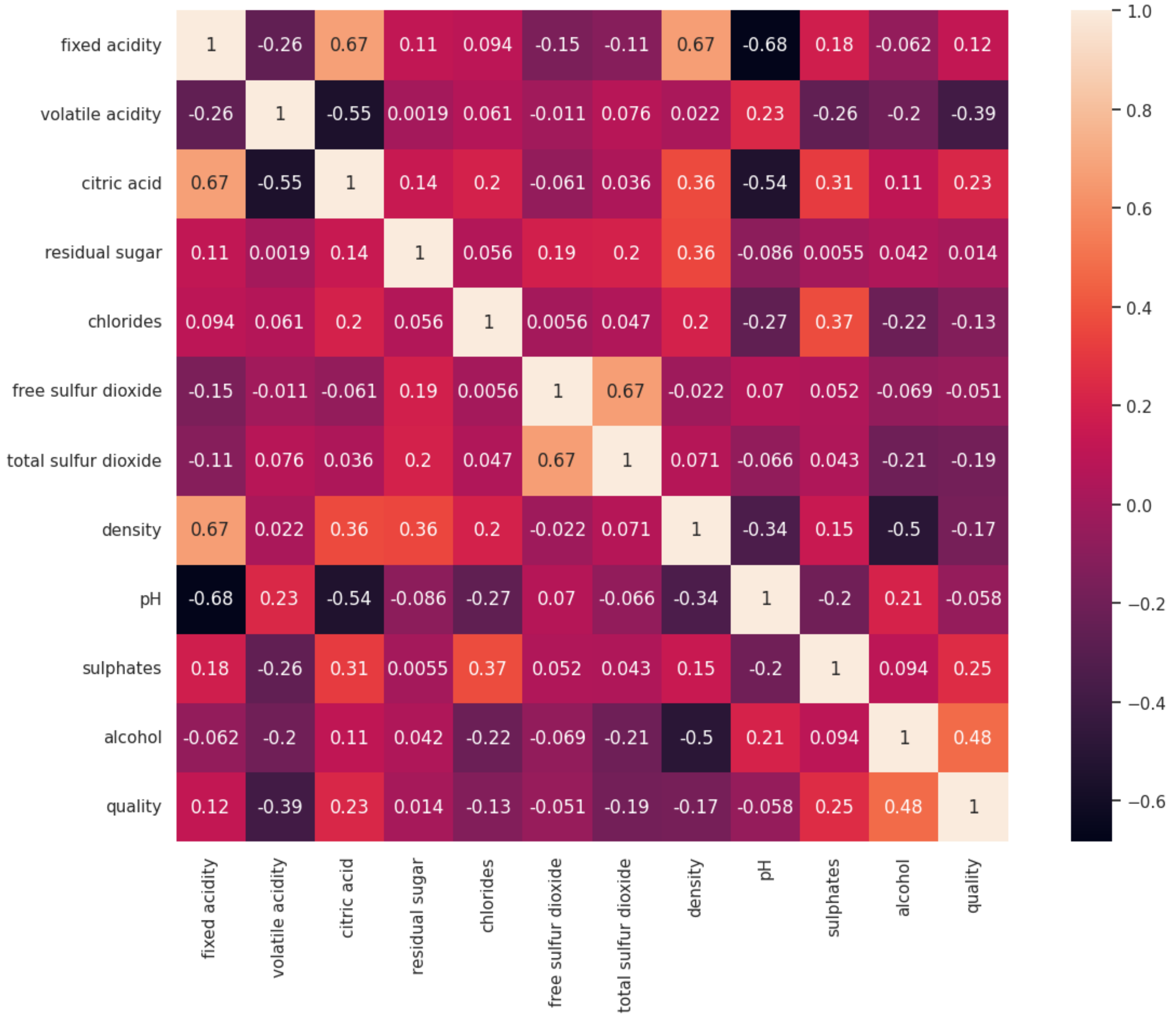
```
data =
pd.read_csv("https://github.com/anthea97/GradientDescent/raw/main/winequality-
red.csv",delimiter=";")
```

- Null and redundant values were removed.
- Histograms were plotted for all the variables



- Correlation matrix was analyzed and variables with weak correlation - residual sugar, free sulphur dioxide and pH - to the output variable (quality) were excluded.

```
correlation_matrix = data.corr()
sns.set(rc = {'figure.figsize':(15,10)})
sns.heatmap(correlation_matrix, annot=True,square=True)
plt.show()
```



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1	-0.26	0.67	0.11	0.094	-0.15	-0.11	0.67	-0.68	0.18	-0.062	0.12
volatile acidity	-0.26	1	-0.55	0.0019	0.061	-0.011	0.076	0.022	0.23	-0.26	-0.2	-0.39
citric acid	0.67	-0.55	1	0.14	0.2	-0.061	0.036	0.36	-0.54	0.31	0.11	0.23
residual sugar	0.11	0.0019	0.14	1	0.056	0.19	0.2	0.36	-0.086	0.0055	0.042	0.014
chlorides	0.094	0.061	0.2	0.056	1	0.0056	0.047	0.2	-0.27	0.37	-0.22	-0.13
free sulfur dioxide	-0.15	-0.011	-0.061	0.19	0.0056	1	0.67	-0.022	0.07	0.052	-0.069	-0.051
total sulfur dioxide	-0.11	0.076	0.036	0.2	0.047	0.67	1	0.071	-0.066	0.043	-0.21	-0.19
density	0.67	0.022	0.36	0.36	0.2	-0.022	0.071	1	-0.34	0.15	-0.5	-0.17
pH	-0.68	0.23	-0.54	-0.086	-0.27	0.07	-0.066	-0.34	1	-0.2	0.21	-0.058
sulphates	0.18	-0.26	0.31	0.0055	0.37	0.052	0.043	0.15	-0.2	1	0.094	0.25
alcohol	-0.062	-0.2	0.11	0.042	-0.22	-0.069	-0.21	-0.5	0.21	0.094	1	0.48
quality	0.12	-0.39	0.23	0.014	-0.13	-0.051	-0.19	-0.17	-0.058	0.25	0.48	1

The data was split into training and test data with an 80/20 split.

1. Linear Regression Using Gradient Descent (From Scratch)

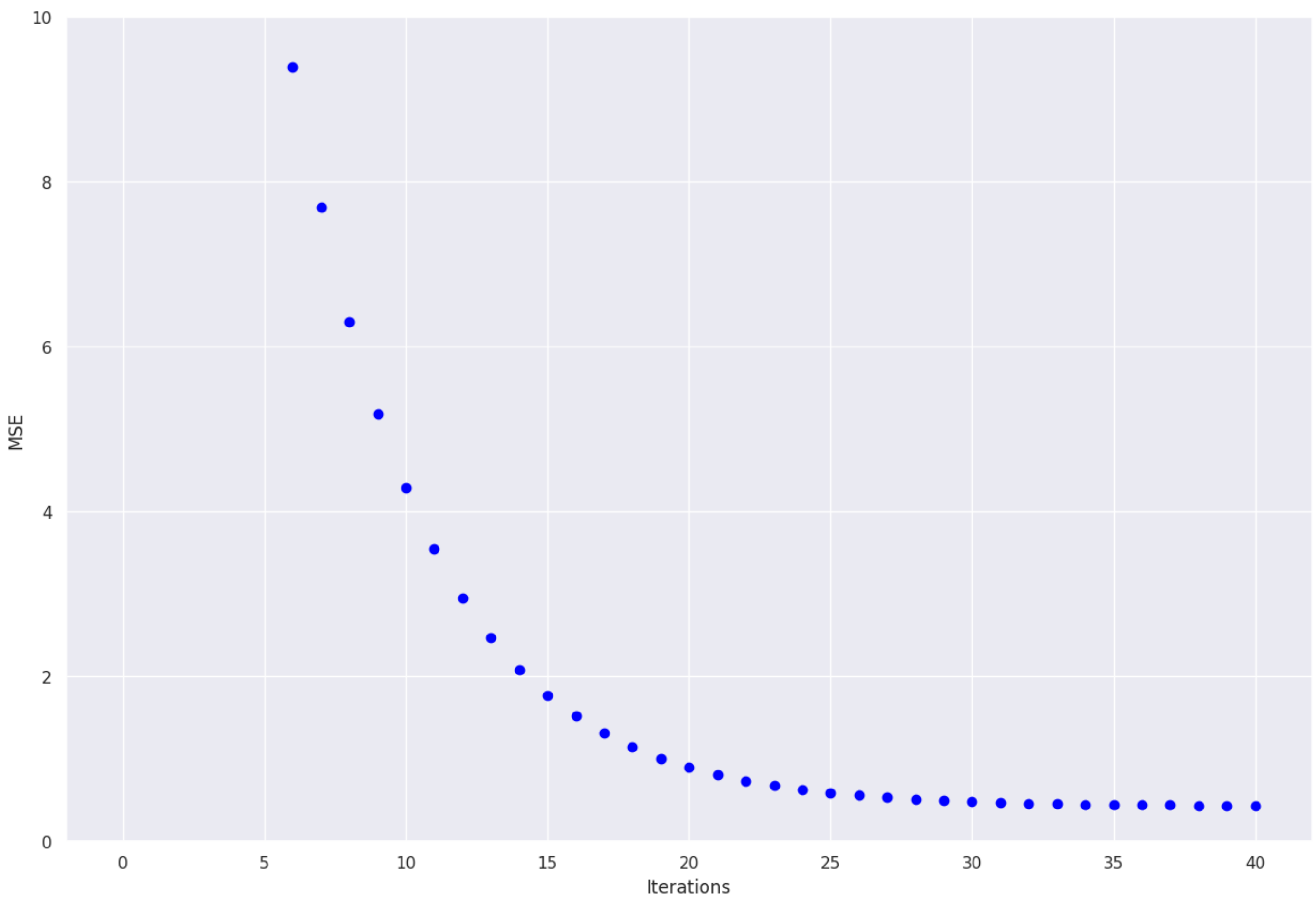
Error log for learning rate = 0.1 with increasing number of iterations:

S.No	# Iterations	MSE	MAE	EAV	R^2
1	10	4.418898011	2.001531109	0.35137986	-5.809046067
2	20	0.927470906	0.792843729	0.374701783	-0.429132808

S.No	# Iterations	MSE	MAE	EAV	R^2
3	50	0.414570221	0.485520261	0.379796538	0.361191924
4	100	0.414570221	0.485520261	0.379796538	0.361191924
5	1000	0.414570221	0.485520261	0.379796538	0.361191924
6	5000	0.414570221	0.485520261	0.379796538	0.361191924
7	10000	0.414570221	0.485520261	0.379796538	0.361191924
8	15000	0.414570221	0.485520261	0.379796538	0.361191924
9	20000	0.414570221	0.485520261	0.379796538	0.361191924
10	25000	0.414570221	0.485520261	0.379796538	0.361191924

Mean Square Error (MSE) vs Number of Iterations:

We observe that the MSE decreases as the number of iterations increases. The minimum number of iterations to convergence is around 30. The same can be observe from the log table, as the values of MSE stabilize after number of iterations cross 20.

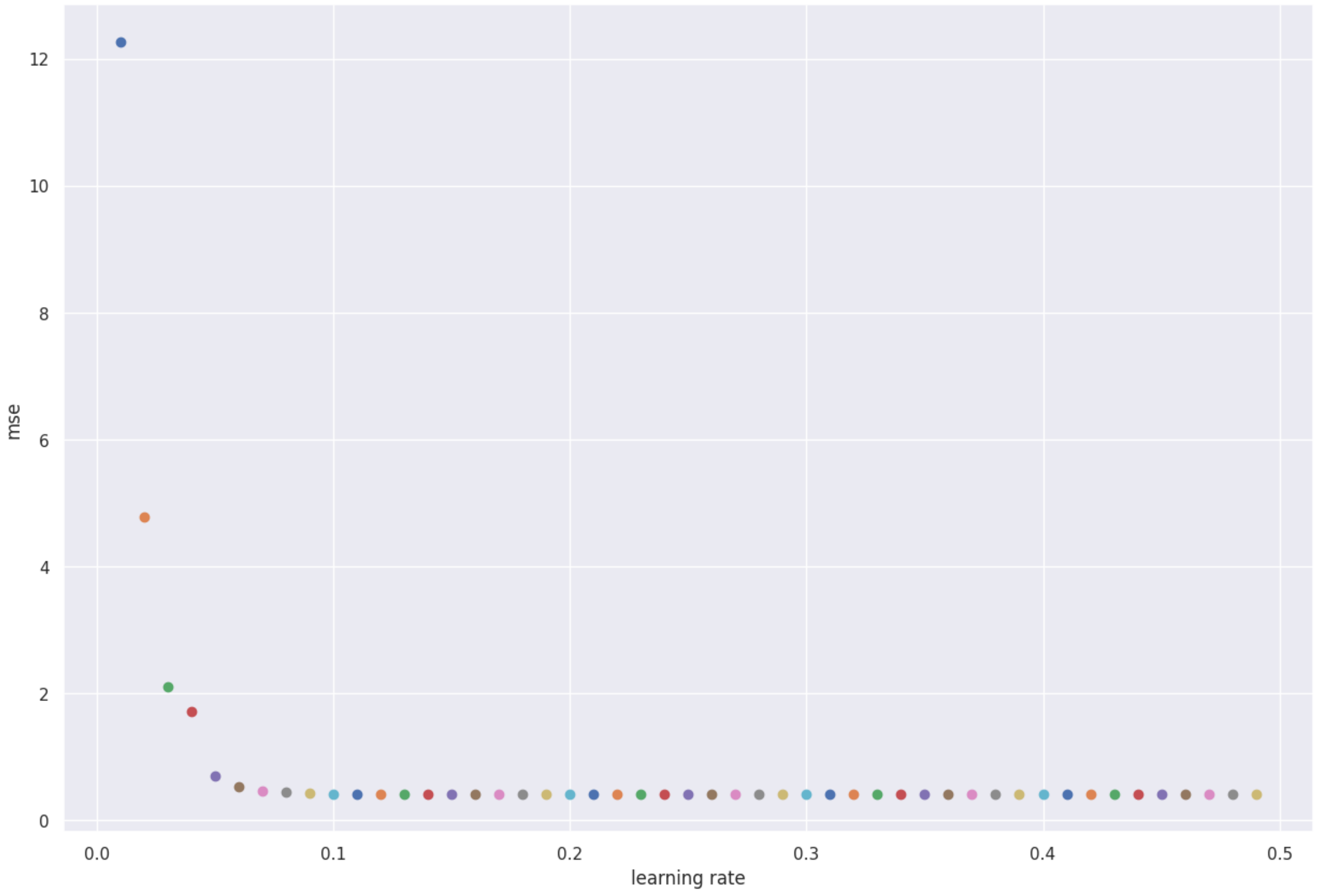


Mean Square Error (MSE) vs Learning Rate:

```
def mseVsLR(X_train, Y_train, X_test, Y_test, y_pred):
  gd = customGD(0.001)
  for lr in np.arange(0.01, 0.5, 0.01):
    w,b = gd.gradientDescent(X_train,Y_train,50,lr)
    y_pred = gd.predict(X_test,w,b)
    mse = mean_squared_error(Y_test,y_pred)
    plt.scatter(lr, mse)

  plt.xlabel("learning rate")
  plt.ylabel("mse")
  plt.show()
```

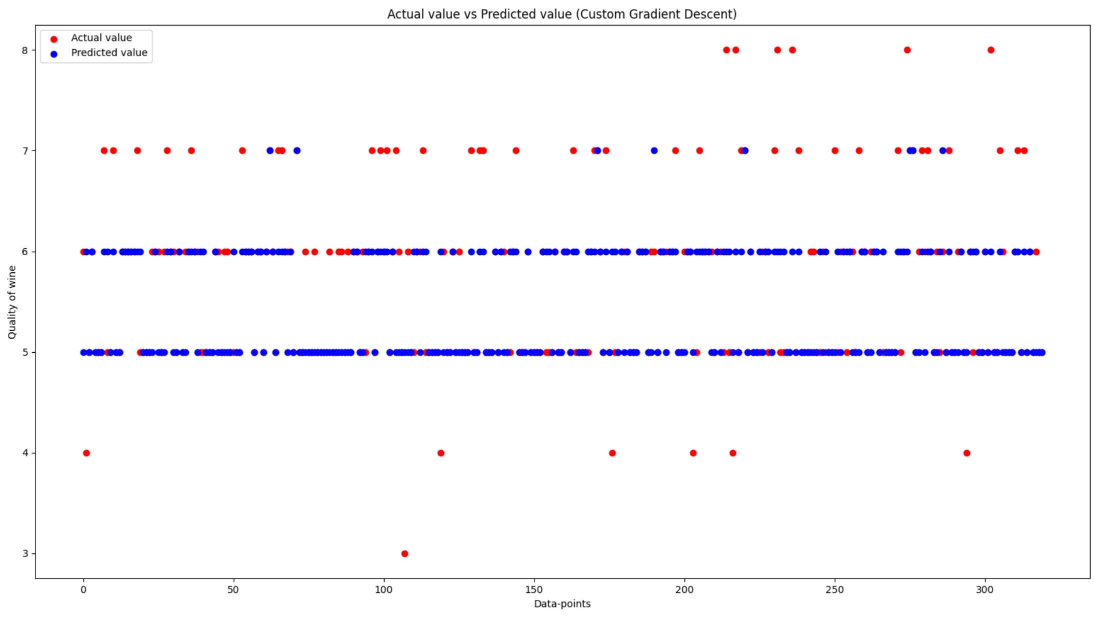
MSE was plotted for a range of learning rates between 0.01 and 0.5.



We observe that the MSE stabilizes at learning rate = 0.1. This is why we chose the final learning rate as 0.1.

Results:

- Weights: [0.09190231,-0.18631047, 0.00988079, -0.08262019, -0.06727607, -0.05484714, 0.14757514, 0.26096736]
- Bias: 5.553971075354176
- MSE: 0.41457022142942657
- MAE: 0.48552026122069414
- EAV: 0.3797965377406818
- R^2: 0.36119192424387503



Question:

Q. Are you satisfied that you have found the best solution? Explain.

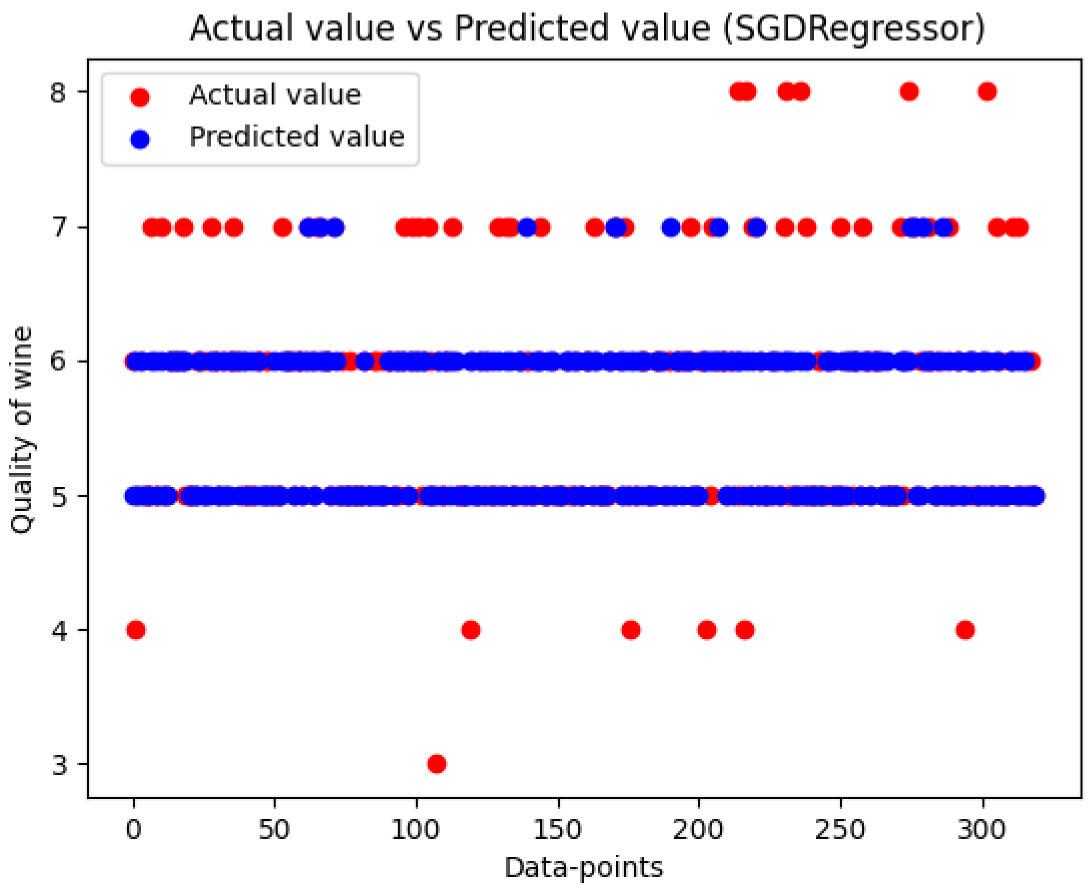
Ans: No, the model that we have built is not the best solution for the given dataset. There can be some parameters that can be modified like the learning rate, number of iterations etc., that will further decrease the error, but the current value of errors are permissible.

2. Linear Regression Using ML Library

SGDRegressor has been used as the linear model prediction of the wine quality dataset. By hit-and-trail, an observation was made that the SGDRegressor requires the data to be normalized in order to perform well. The coefficients of the SGDRegressor are as follows:

Results:

- Weights: [0.08640624, -0.20771388, -0.01689627, -0.07121951, -0.0534198, -0.03072766, 0.15382052, 0.26234732]
- Bias: 5.63082585
- MSE: 0.40621918708360366
- MAE: 0.49266536997221777
- EAV: 0.3756365448264052
- R^2: 0.3740599690412909



3. Conclusion

In this project, the team embarked on a journey to implement Linear Regression using the Gradient Descent optimization technique, leveraging the Wine Quality dataset from the UCI Machine Learning Repository. Their objective was to predict wine quality based on various input variables, and the results provided valuable insights into the model's performance.

Custom Gradient Descent vs. Library Implementation (SGDRegressor)

The team conducted two major experiments: one involving a custom implementation of Gradient Descent and another using a widely-used library, specifically the `SGDRegressor`. Both approaches aimed to achieve the same goal: predicting wine quality.

Custom Gradient Descent:

- After extensive iterations and fine-tuning, the custom Gradient Descent model demonstrated a commendable performance.
- The model's weight coefficients and bias were nearly identical to those obtained from the library implementation.
- Mean Square Error (MSE) converged to a satisfactory value of approximately 0.4145.
- Mean Absolute Error (MAE) and Explained Variance Score (EAV) also reached respectable levels.
- The R-squared (R^2) value, indicating the proportion of variance in the dependent variable (wine quality) that the model explained, was approximately 0.3612.

SGDRegressor:

- The library implementation using `SGDRegressor` provided comparable results in terms of weight coefficients and bias.
- It achieved a similar MSE of around 0.4062, implying a close approximation to the custom Gradient Descent model.
- MAE, EAV, and R^2 showed consistent performance with satisfactory values.

Summary

The project highlights the success of custom Gradient Descent in solving regression problems and emphasizes the importance of algorithm understanding and fine-tuning. The findings provide a strong foundation for future data science endeavors.