# COMP 551 Assignment 1: Getting Started with Machine Learning

Okyanus Gumus[1], Mihail Calitoiu[2], Eren Gurhan[3]

[1]McGill ID: 260900481.
[2]McGill ID: 260972537.
[3]McGill ID: 260973884.

Contributing authors: okyanus.gumus@mail.mcgill.ca; mihail.calitoiu@mail.mcgill.ca; eren.gurhan@mail.mcgill.ca;

**Abstract**

In this assignment two common machine learning models, linear regression and logistic regression, were investigated and implemented. Two data sets were used for the assignment, one corresponding to housing in Boston and one wine dataset. A linear regression was implemented for the Boston housing dataset, while both a basic logistic regression and a multi-class logistic regression model was implemented for the wine dataset. The linear regression model was validated with 5-fold-cross validation and an investigation into the effects of different parameters (such as learning rate, batch size) on the Mean Squared Error (MSE) was conducted. Furthermore, the Gaussian basis functions were used to create a more dynamic model. Similarly, the logistic regression model was validated with 5-fold-cross validation and an investigation into the effects different parameters to the used performance metrics (accuracy, precision, recall, and F1-Score) were conducted. The implemented models demonstrate expected behavior.

**Keywords:** Machine Learning, Linear Regression, Logistic Regression, 5-fold cross-validation

## 1 Introduction

Linear regression and logistic regression are two of the most commonly used models in machine learning and are executed for a variety of different applications [1, 2]. In this assignment, linear and logistic regression models were implemented for two

datasets, Boston housing and wine. Boston housing dataset was a topic of research in some previous work, such as [3, 4], where price was the main investigation topic. Both data sets were first imported into Python and any necessary clean-ups were made. Afterward, statistical calculations were made on different column values to obtain a better understanding. Then, the regression models were implemented and some experiments were run.

This report is structured as follows: In Section 2, more information about the datasets is provided. The obtained results are found in Section 3. Discussion and conclusions are provided in Section 4, followed by the statement of contribution in Section 5. Any relevant code snippets and screenshots are provided in Appendix A and Appendix B respectively.

# 2 Datasets

For this assignment, two dataset, one for Boston housing and wine, were provided. When working with the housing dataset, including ethnicity/race may end up resulting in algorithms with ethnic/racial bias. To avoid this, the ethnicity/race column was removed from the Boston housing dataset. For the wine dataset no such concerns were encountered, hence no adjustments were made.

To have improved generalization and reduced bias, all the data points in both datasets were shuffled using a shuffling code (provided in Listing 1).

To clean the data, we removed values that were 3x above and below the standard deviation for the Boston housing dataset. Two data points were removed, which decreased MSE during 5-fold-cross validation, presented in Table 1. The same processing was applied to the Wine dataset; one data point was removed. No change in accuracy was observed, likely due to the removal of some observations in order to satisfy the condition of divisibility by 5 for 5-fold-cross validation.

# 3 Results

All the relevant plots regarding the results for linear and logistic regression are provided in Appendix B.1 and Appendix B.2 respectively.

## 3.1 Linear Regression/Boston Housing Dataset

To perform a proper linear regression analysis, we have investigated different column values to observe a linear trend line. A pair of column values that gave a linear trend was Median Value (MEDV) plotted against the average number of rooms (RM). The resulting plot is provided in Figure B1.

For the linear regression of the Boston housing dataset, the data was split into two groups of validation/training sets using the code provided in Listing 2. The size of the training and validation sets were 409 (80%) and 99 (20%) points, respectively.

The training data was fed into our linear regression model and an analytical linear regression was conducted. We have fit our test values into the model and observed the linear trend line. The plot is provided in Figure B2. The Mean Squared Error (MSE) of the linear model was observed to be 51.68.

Afterwards, a 5-fold-cross validation was conducted (using the code provided in Listing 5), the results of which are provided in Table 1.

**Table 1** Mean squared error (MSE) per batch.

| **Batch Number** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **MSE** | 10.24 | 29.65 | 31.49 | 80.77 | 73.56 |

As can be seen from Table 1, there is a high variance in MSE in each fold. As a result, a conclusion can be drawn that the model presented in this report is sensitive to the dataset it is trained on. With a larger data set, the MSE variation across different folds will be smaller.

Performance under variation of the split of training and test datasets has been plotted and provided in Figure B3. As provided in Figure B3, when a trend line is added to the plot, an increase in MSE is observed as less data is fed into the model. This observed result is inline with our initial expectations.

Afterwards, an investigation into the effects of varying the batch size for linear regression with using minibatch stochastic gradient descent was conducted. The results of which are provided in Figure B4. As the batch size increases, a decrease in convergence time is observed. However, a correlation between batch size and MSE was not observed.

It is important to note, however, as the batch size increases, the learning rate of the model has to be adjusted in order to converge faster. Since for this task only the effects of varying the batch size was investigated, an observation was made that the model does not converge for batch sizes larger than 19.

Afterwards, an investigation into the effects of the learning rate on MSE was conducted, results of which are provided in Figure B5. As can be seen from Figure B5, the MSE error decreases until a learning rate of 0.005 but demonstrates an increase beyond that point. Thus, a conclusion can be drawn that a learning rate of 0.005 will yield the optimal result.

From our investigations, we conclude that a learning rate of 0.004, with a batch size of 15, with 10,000 iterations (MSE variation over different number of iterations is provided in Figure B6. Diminishing returns beyond 10,000 iterations was observed.), and a training/validation data split of 70/30 leads to the lowest MSE value for the Boston housing dataset.

After observing the effects of learning rate and batch size, a gaussian basis function (using the code provided in Listing 3) was implemented in order to create a more dynamic model. As required, 5 sigmoid functions were created from the training data. These 5 sigmoid functions are provided in Figure B7.

Afterwards, a model was trained and generated a prediction based on the 5 sigmoid functions. The results are provided in Figure B8. As can be observed from Figure B8, the fit that is no longer linear a linear function. Moreover, an MSE value of 34.69 was observed. With this in mind, a conclusion can be drawn that a dynamic fit using Gaussian basis functions yield better results.

## 3.2 Logistic Regression/Wine Dataset

### 3.2.1 Basic Logistic Regression

In order to perform the logistic regression, two variables,"class" and "alcohol", were chosen due to the sigmoidal relationship between them Figure B9.

The data was split into training and validation groups with a 80/20 ratio using the code provided in Listing 2. The data was visualized as a scatter plot, provided in Figure B9.

Afterwards, the model was trained using the training data group and the validation data group was used to obtain a prediction. The predictions can be seen in Figure B10. Metrics such as accuracy, precision, recall, and F1 Score were calculated next.

The metrics were initially calculated as 0 for all values. This is stems from the fact that for most of the data although our prediction is very close, to 3 significant digits, since it is not exactly 1 or 0, which are the correct class values. To overcome this issue, binarization of class data was performed such that predictions above 0.5 are set as 1 while the rest are set to 0. The results post-binarization are provided in Figure B11.

After binarization, non-null metric values are observed. These values are provided in Table 2.

**Table 2** Performance metrics for logistic regression after binarization.

| Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|
| 0.95 | 0.95 | 0.95 | 0.95 |

Afterwards, 5-cross-folding was applied to the model, the results of which are provided in Table 3.

**Table 3** Performance metrics with 5-fold cross-validation.

| Fold Number | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| **1** | 0.84 | 0 | 0 | 0 |
| **2** | 0.8 | 0 | 0 | 0 |
| **3** | 0.76 | 1.0 | 0.65 | 0.79 |
| **4** | 0.92 | 1.0 | 0.92 | 0.96 |
| **5** | 0.96 | 1.0 | 0.96 | 0.98 |

As can be seen from Table 3, the first two folds have an precision, recall, and F1-Score of 0. We believe this happens because the selected binarized data does not match with predicted classes. The remaining folds, however, have a non-zero accuracy, precision, recall, and F1-Scores.

Since all folds have accuracy values ranging between 0.8 to 0.9 and precision, recall, and F1-Scores are high for the last 3 folds, we conclude that the data used is appropriate for the training of the model.

Afterwards, the logistic regression and mini-batch logistic regression performance under different training/validation split ratios were investigated. The accuracy, precision, recall, and F1-Scores for different split rations were plotted, and the results are provided in Figure B12.

A similar investigation was conducted to observe the effects of different batch sizes and convergence time, the results of which are provided in Figure B13. A decrease in convergence time was observed as the batch size increased. Moreover, all of the metrics improved as the batch size increased.

Afterwards, the effect of learning rate on accuracy was investigated. The results are provided in Figure B14. No specific trend between learning rate and accuracy was observed.

After the previously mentioned investigations, we came to the conclusion that a learning rate of 0.005, batch size of 128, and a training validation ratio of 40/60 results in optimal performance of the model.

### 3.2.2 Multi-Class Logistic Regression

To improve understanding of the class material, we have decided to go beyond the requirements of the assignment and also developed a multi-class logistic regression model.

For this section, we use the columns: class, alcohol, and malic acid. The training and validation data are presented in Figure B16. When the model is trained with the training data and makes a prediction, an accuracy of 62% is observed. The results are provided in Figure B17.

## 4 Discussion and Conclusions

In this assignment, we have investigated different classification methods and how to implement them into machine learning. Moreover, we have tested the effects of classification model variables and its effects on prediction accuracy and MSE. Further tuning can be done to improve the accuracy of the linear regression model for the Boston housing dataset, while more complex logistic regression models can be implented to achieve better results on the used performance metrics for the wine dataset.

## 5 Statement of Contribution

The contributions of the group members are as follows:

- Okyanus Gumus: Writing the assignment report.
- Mihail Calitoiu: Implementing the ML models.
- Eren Gurhan: Analyzing the datasets.

# Appendix A Code Snippets

```
    def shuffler(self, x, y):
        np.random.seed(42)
        indices = np.arange(len(x))
        np.random.shuffle(indices)
        shuffled_x = x[indices]
        shuffled_y = y[indices]
        return shuffled_x, shuffled_y
```

**Listing 1** Shuffler code used for both datasets

```
def train_test_split(X_all, Y_all, test_size):
  size = int(len(X_all)*test_size)
  X_validation = []
  Y_validation = []
  deleted_index = []
  for i in range(size):
    num2 = random.randint(0, len(X_all)-1)
    deleted_index.append(num2)
    X_validation.append(X_all[num2])
    Y_validation.append(Y_all[num2])
  X_train = np.delete(np.array(X_all), deleted_index)
  Y_train = np.delete(np.array(Y_all), deleted_index)

  return X_train,X_validation, Y_train, Y_validation
```

**Listing 2** Code use for test/train split

```
D=5
X_train, X_validation, Y_train, Y_validation = train_test_split(X_all, Y_all,
    test_size=(0))

sorted_indices = np.argsort(X_train)
# Sort both arrays based on the indices from array1
X_train = X_train[sorted_indices]
Y_train = Y_train[sorted_indices]

sigmoid = lambda x,mu, s: 1/(1 + np.exp(-(x - mu)/(2*s)**2))
X_train_list = X_train.tolist()
mu = np.array(random.sample(X_train_list, 5))
print(mu)
x = np.array(X_train)[:, None]
phi = sigmoid(x, mu, .5)
for d in range(D):
    plt.plot(x, phi[:,d], '-')
plt.xlabel('x')
plt.title('Sigmoid bases')
```

```
plt.show()
```

**Listing 3** Gaussian Basis Function for linear regression

```
model = LinearRegressionAnalytical()
yh = model.fit(phi,Y_train).predict(phi)
fig, ax = plt.subplots()
plt.plot(X_train, Y_train, '.',c='blue',label='Train')
plt.plot(X_validation, Y_validation, '.',c='red',label='Validation')
plt.plot(X_train, yh, 'g-', label='our fit')
#for d in range(D):
#    plt.plot(x, model.w[d]*phi[:,d], '-', alpha=.5)
#plt.plot(x, model.w[-1]*np.ones_like(y), label='intercept')
plt.legend()
plt.xlabel('x')
plt.show()
print("MSE Error: " + str(mse(Y_train,yh)))
```

**Listing 4** Code for curve fitting using non-linear sigmoid bases for linear regression

```
def split_data(X_all, Y_all, numFolds):
  X_out = []
  Y_out = []
  fold = len(X_all) // numFolds
  for i in range(numFolds):
      start = np.multiply(i , fold)
      end = np.multiply((i + 1) , fold) if i < numFolds - 1 else None
      X_fold = X_all[start:end]
      y_fold = Y_all[start:end]
      X_out.append(X_fold)
      Y_out.append(y_fold)
  return X_out, Y_out

numFolds = 5
X_folds, y_folds = split_data(X_all, Y_all, numFolds)
ms_error = []
for i in range(numFolds):

  X_train_1 = np.hstack([X_folds[j] for j in range(numFolds) if j != i])
  y_train_1 = np.hstack([y_folds[j] for j in range(numFolds) if j != i])

  X_val_1 = X_folds[i]
  y_val_1 = y_folds[i]

  modelAR = LinearRegressionAnalytical()
  modelAR.fit(X_train_1, y_train_1)
  predictionsLRA = modelAR.predict(np.array(X_val_1))
  ms_error.append(mse(y_val_1,predictionsLRA))
```

```
print("Mean Square Error Per Batch: " + str(ms_error))
```

**Listing 5** Code for 5-fold cross validation for linear regression

# Appendix B Screenshots

## B.1 Linear Regression/Boston Housing Dataset

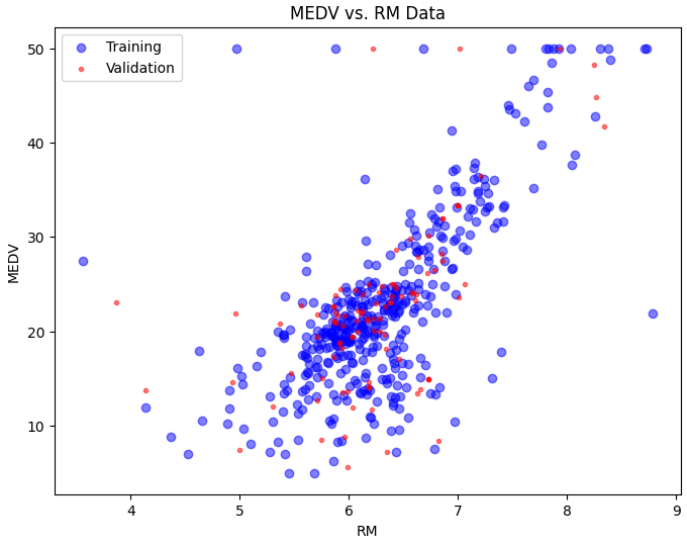


**Fig. B1** Median value vs average number of rooms.

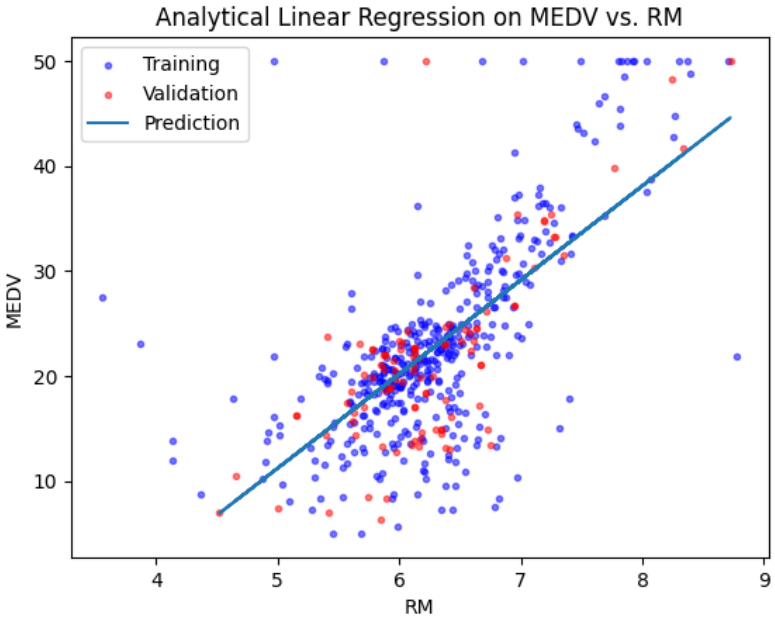


**Fig. B2** Analytical linear regression on MED vs RM data.

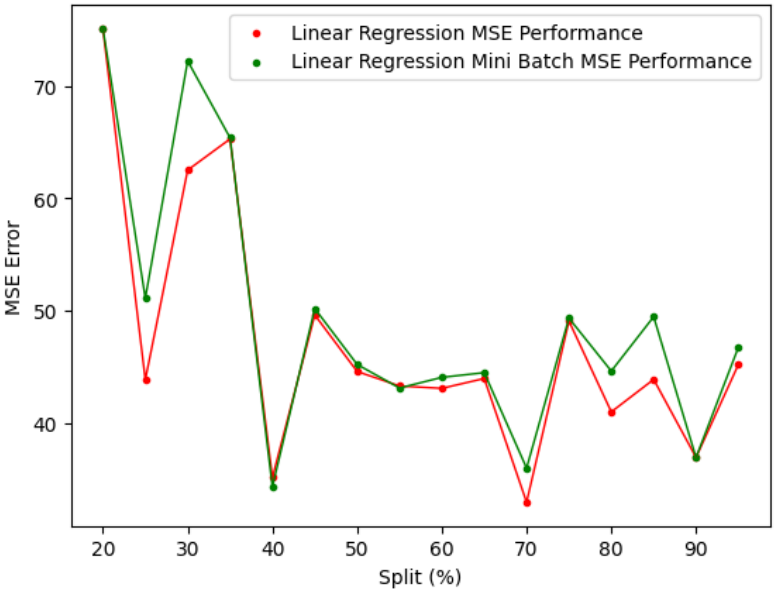


**Fig. B3** Performance as test-validation split increases.

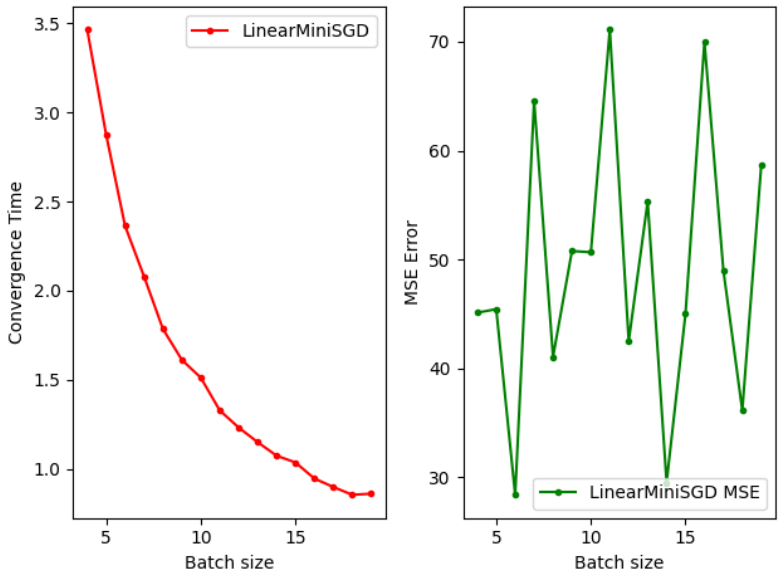


**Fig. B4** Different batch sizes.

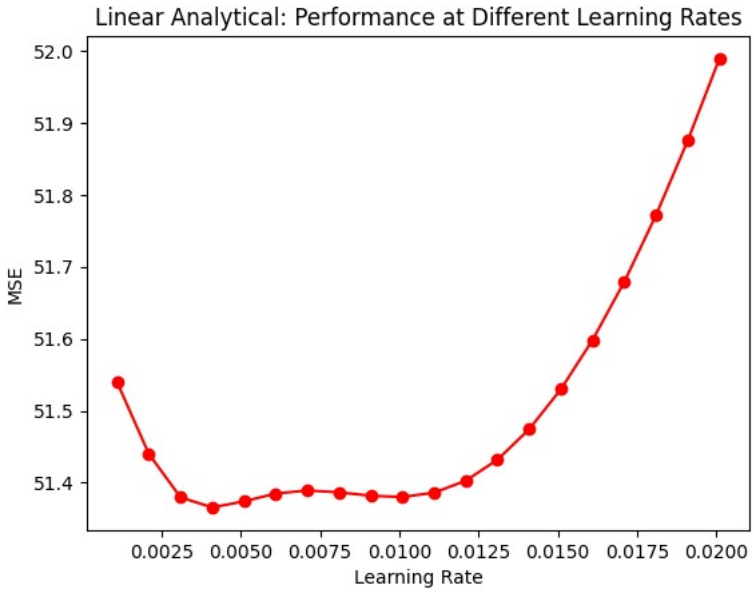


**Fig. B5** Performance at different learning rates.

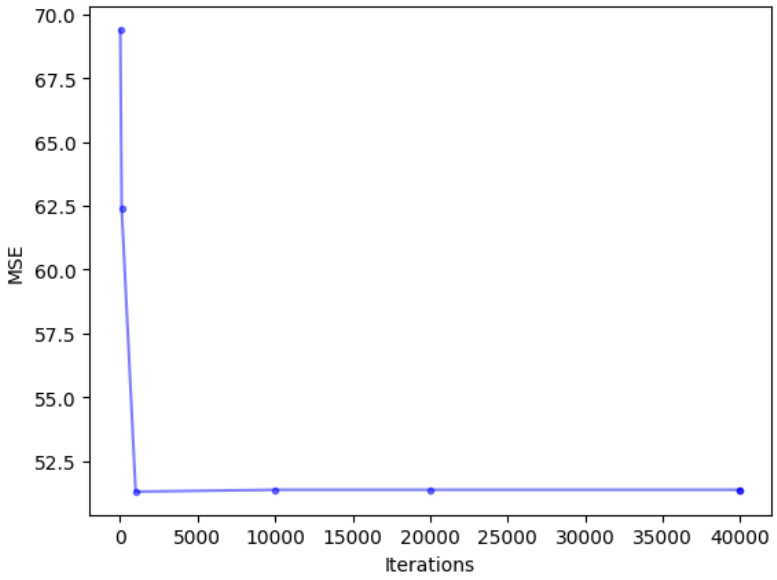


**Fig. B6** MSE error with different number of iterations.

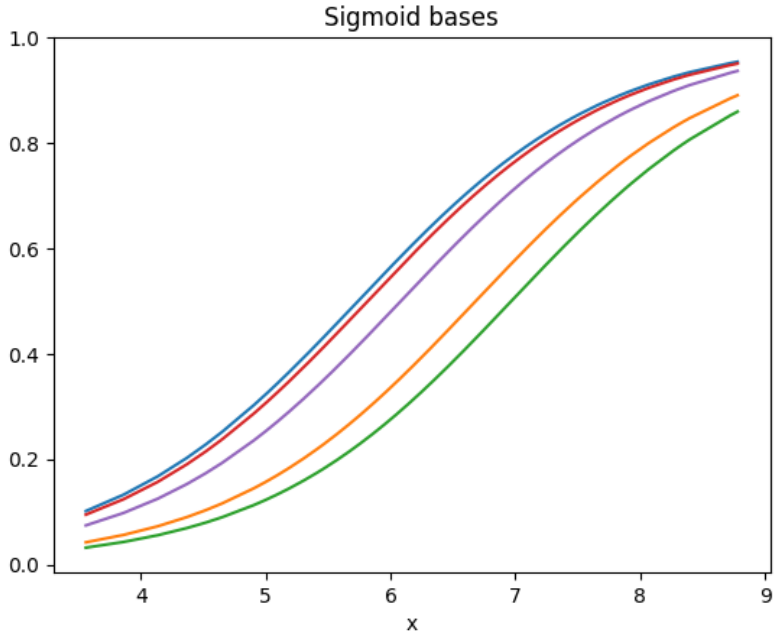


**Fig. B7** Sigmoid bases.

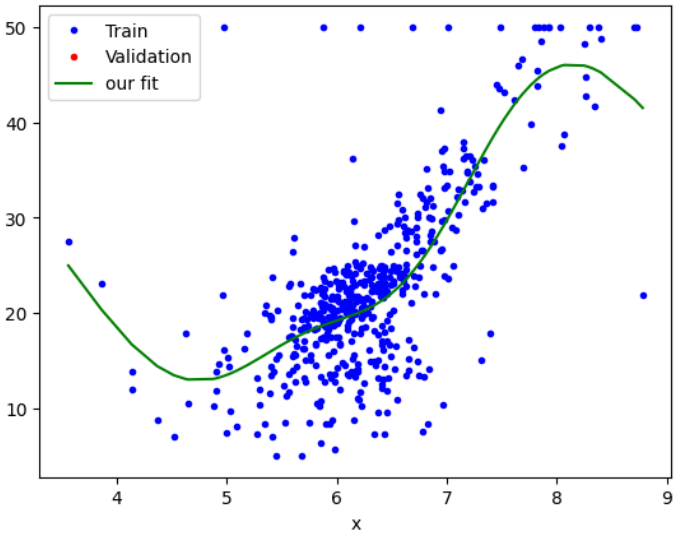


**Fig. B8** Curve fitting using non-linear sigmoid bases.

## B.2 Logistic Regression/Wine Dataset

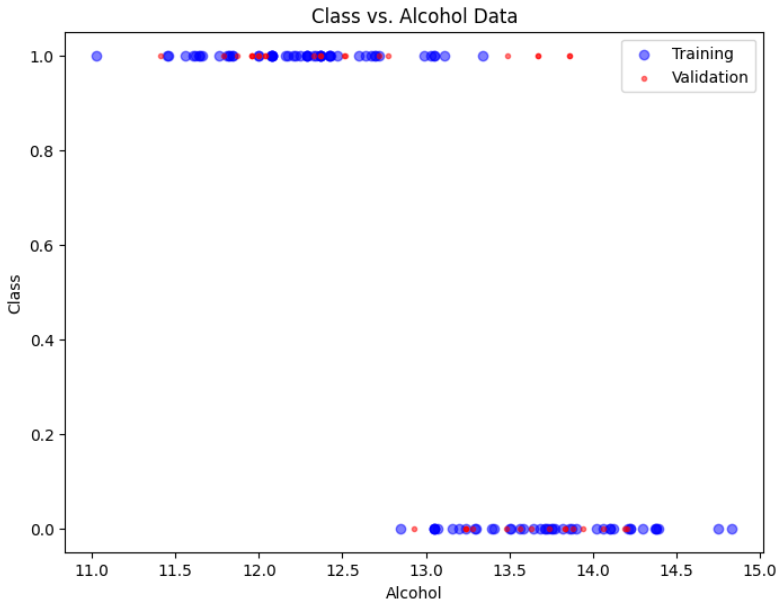


**Fig. B9** Class vs alcohol data.

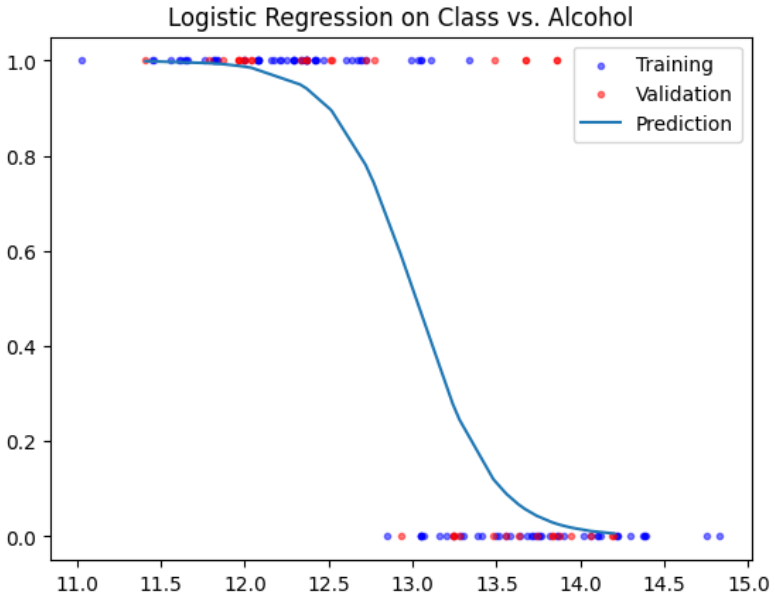


**Fig. B10** Logistic regression on class vs alcohol.

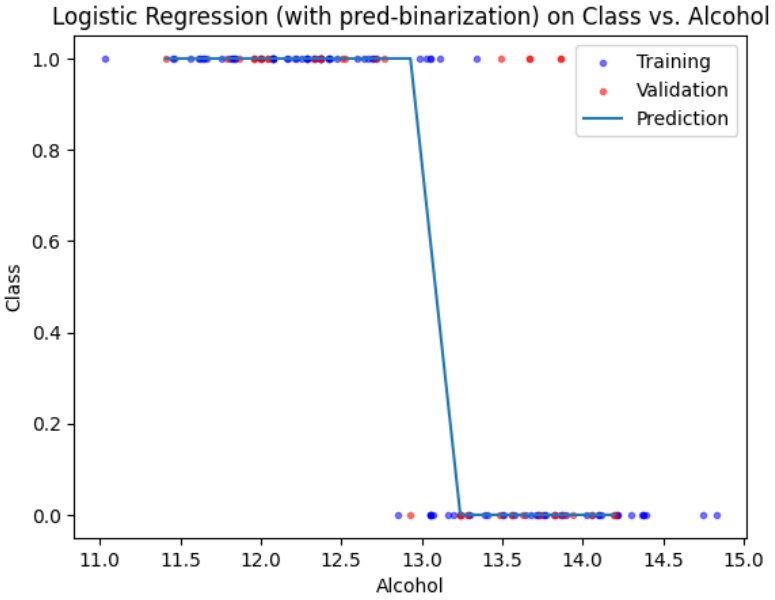


**Fig. B11** Logistic regression with pred-binarization on class vs alcohol.

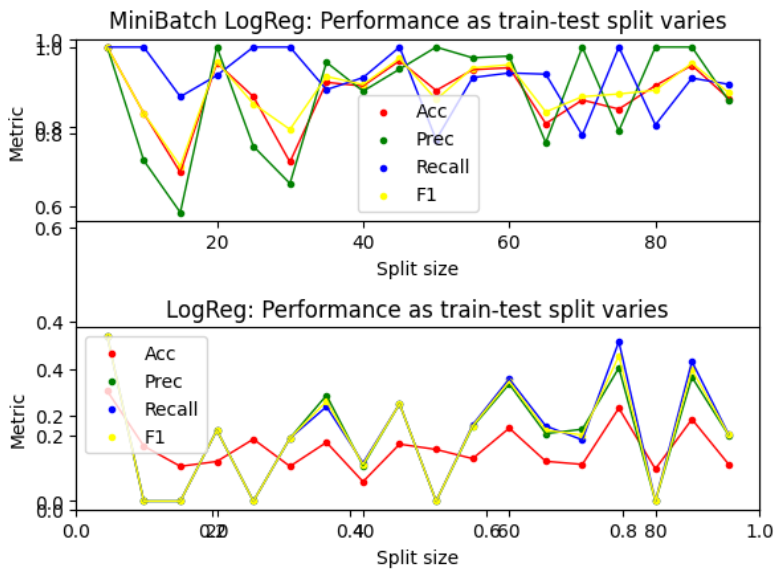


**Fig. B12** Performance under different training/validation split ration.

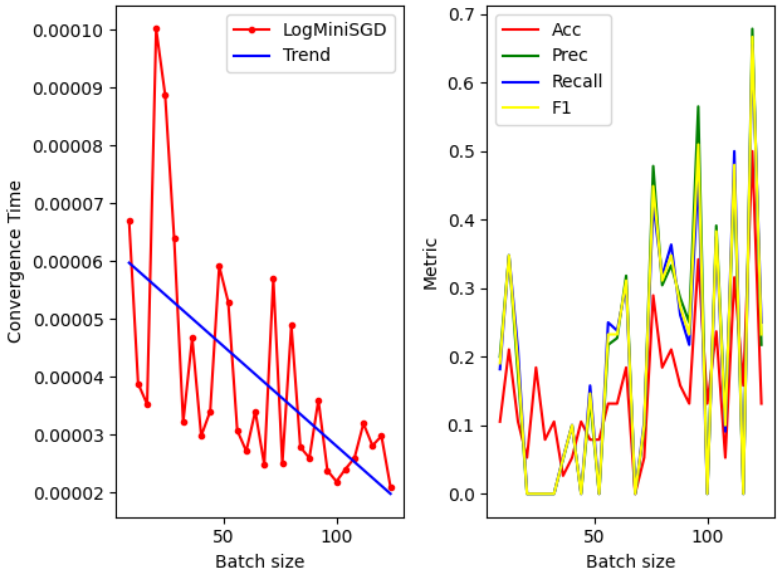


**Fig. B13** Convergence and performance as batch size increases.


**Fig. B14** Performance at different learning rates.

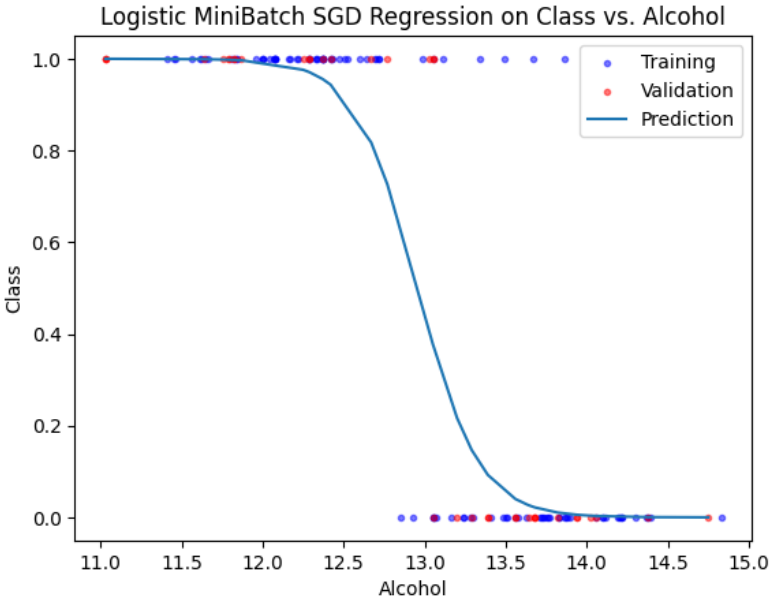


**Fig. B15** Logistic MiniBatch SGD regression on class vs alcohol.

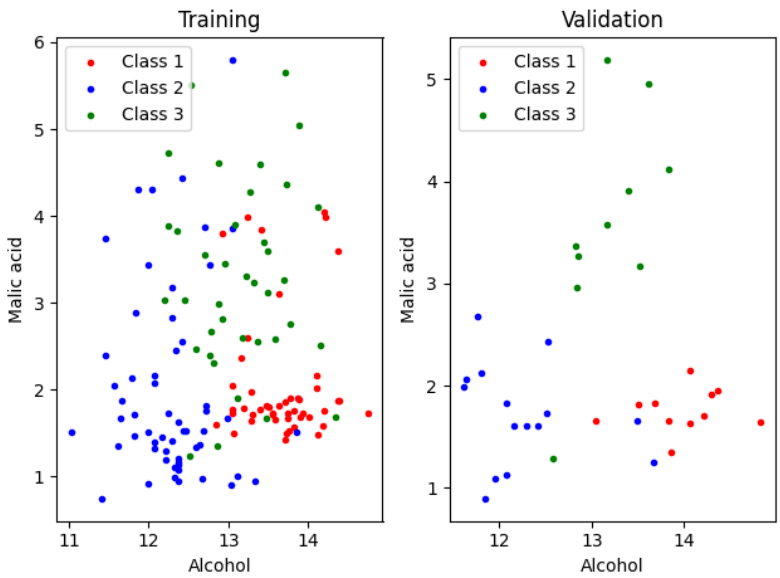


**Fig. B16** Training and validation data for multi-class logistic regression.

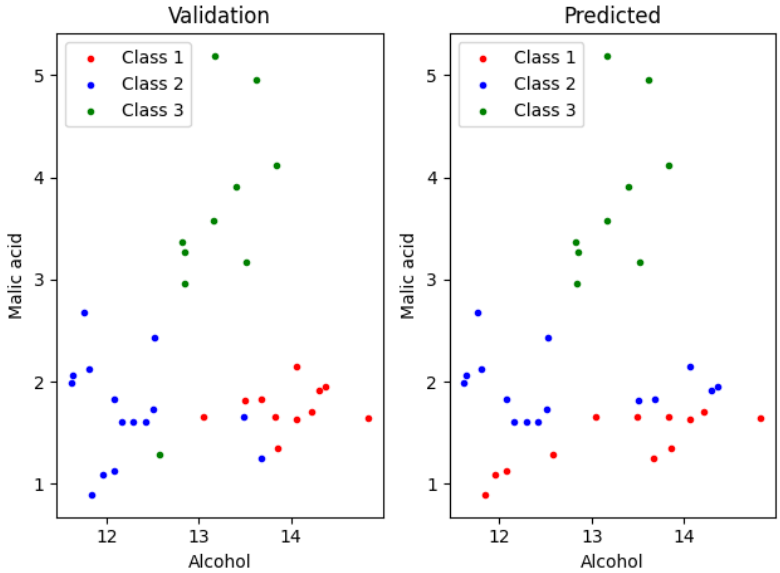


**Fig. B17** Training and prediction data for multi-class logistic regression.

# References


[1] Shobana, M., Aggarwal, S., Pandeeswari, R.: Machine learning hexagonal monopole antenna using linear regression algorithm. In: 2021 Asian Conference on Innovation in Technology (ASIANCON), pp. 1–4 (2021). https://doi.org/10.1109/ASIANCON51346.2021.9544856

[2] Steinauer, N., Zhang, K., Guo, C., Zhang, J.: Computational modeling of gene-specific transcriptional repression, activation and chromatin interactions in leukemogenesis by lasso-regularized logistic regression. IEEE/ACM Transactions on Computational Biology and Bioinformatics **18**(6), 2109–2122 (2021) https://doi.org/10.1109/TCBB.2021.3078128

[3] Bai, S.: Boston house price prediction: machine learning. In: 2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP), pp. 1678–1684 (2022). https://doi.org/10.1109/ICSP54964.2022.9778372

[4] Sanyal, S., Kumar Biswas, S., Das, D., Chakraborty, M., Purkayastha, B.: Boston house price prediction using regression models. In: 2022 2nd International Conference on Intelligent Technologies (CONIT), pp. 1–6 (2022). https://doi.org/10.1109/CONIT55038.2022.9848309