Margaret Atkins
Coding Assignment 2

Reflection

Part A.

In Part A, I implemented linear regression using the least-squares method to predict citric acid levels from wine physicochemical features. I began with a simple two-feature model using alcohol and density, then incrementally added volatile acidity, fixed acidity, and finally all available features to evaluate how performance changed. As shown in Figure 1, the predicted values increasingly aligned with the actual citric acid values as more features were included. The results table in Figure 2 shows that the error steadily decreased as features were added, from 0.1686 with two features to 0.1055 using all features. The largest improvement occurred when volatile acidity was added, suggesting it has a strong relationship with citric acid. Although the full model performed best, all models still struggled to capture extreme spikes, indicating some limitations of a purely linear approach. Overall, this part demonstrated how feature selection impacts predictive accuracy and how adding informative features can systematically improve performance.

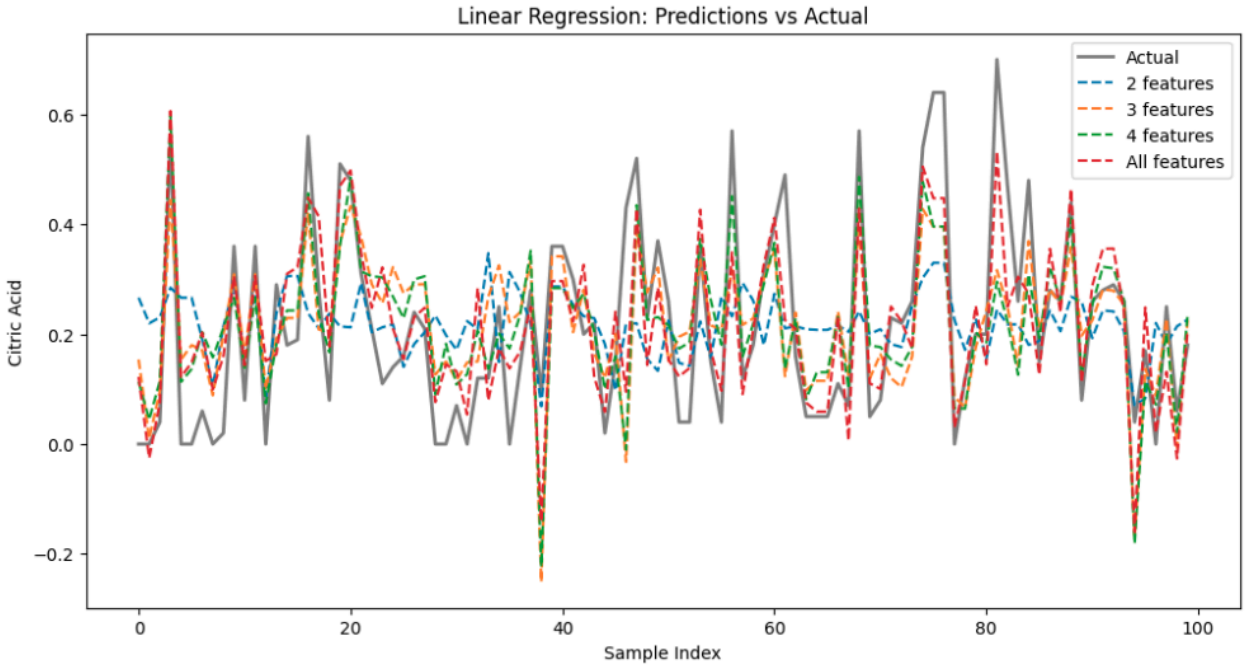

Figure 1. Linear Regression graph

Results Table

Model	Features	Error
Model 1	alcohol, density	0.1686
Model 2	alcohol, density, volatile acidity	0.1320
Model 3	alcohol, density, volatile acidity, fixed acidity	0.1242
Full Model	all features	0.1055

Figure 2. Results table from Part A

Part B.

In Part B, I implemented k-NN classification from scratch and evaluated it on both the Lenses and Credit Approval datasets. The approach involved computing Euclidean distance between samples, selecting the k nearest neighbors, and predicting class labels using majority voting. Proper preprocessing was essential for the Credit Approval dataset, which contained missing values and mixed data types. I replaced missing categorical values with the mode and continuous values with label conditioned means, then encoded categorical features numerically and normalized continuous features using z-score scaling. The results are summarized in Figure 3. For the Lenses dataset, all tested values of k (1, 3, 5, and 7) achieved perfect accuracy because the dataset is small, clean, and has clear separations between classes. In contrast, the Credit Approval dataset showed sensitivity to k. Accuracy improved from 0.7464 at k=1 to 0.8043 at k=5, then slightly decreased at k=7. The best performance at k=5 suggests a balance between reducing sensitivity to noise and maintaining important local patterns. This part reinforced how the choice of k reflects a bias variance trade off and how preprocessing dramatically affects performance for distance-based models.

Dataset	k=1	k=2	k=5	k=7
Lenses	1.0000	1.0000	1.0000	1.0000
Credit Approval	0.7464	0.7826	0.8043	0.7971

Figure 3. Results table from Part B

Part C.

In Part C, I implemented a Gaussian Naive Bayes classifier for spam detection. The model estimated class priors along with the mean and variance of each feature per class, then used the Gaussian probability density function combined with log probabilities for numerical stability. I also computed discriminative scores for each feature by measuring the standardized difference in means between spam and non spam emails. The top five discriminative features (“your,” “000,” “remove,” “$,” and “hp”) are shown in Figure 6, and their distributions are visualized in Figure 5. The classifier achieved an accuracy of 0.8217, precision of 0.7233, recall of 0.9385, and F1-score of 0.8170, as shown in Figure 7. The high recall indicates the model was very effective at identifying spam, although the lower precision suggests some false positives.

Despite the independence assumption and Gaussian distribution assumption, Naive Bayes performed well because many individual word frequency features strongly distinguish spam from non spam.

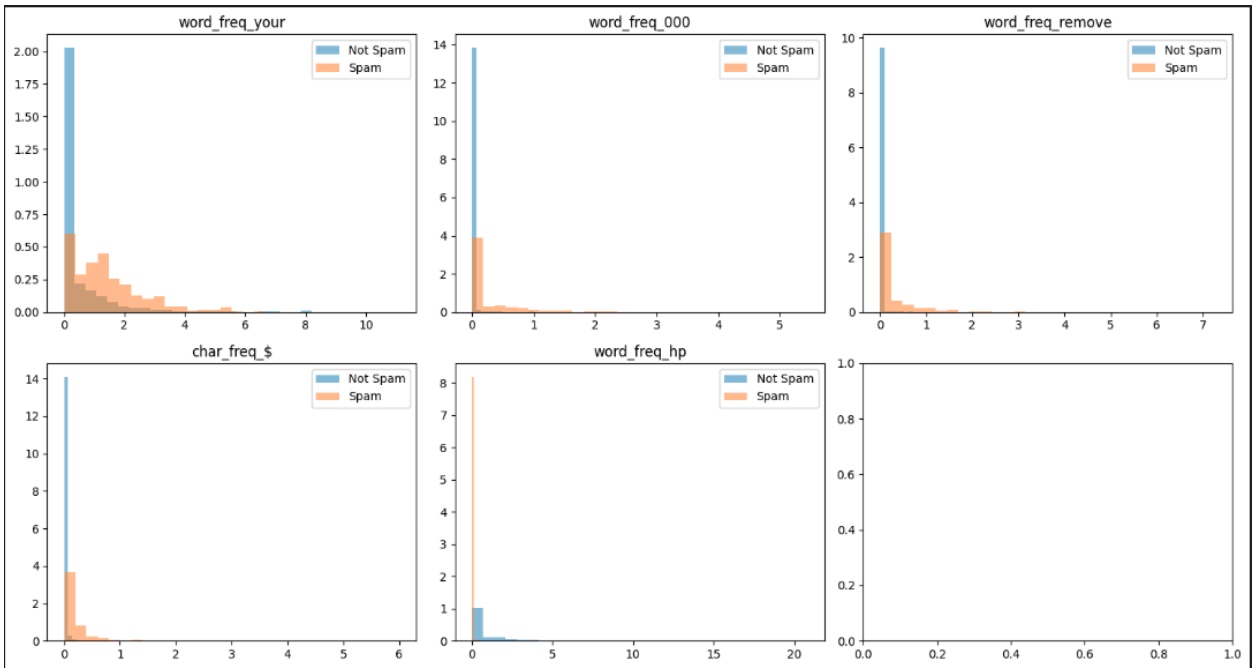

Figure 5. Visualized Distributions of Top Features for Spam vs. Non-Spam

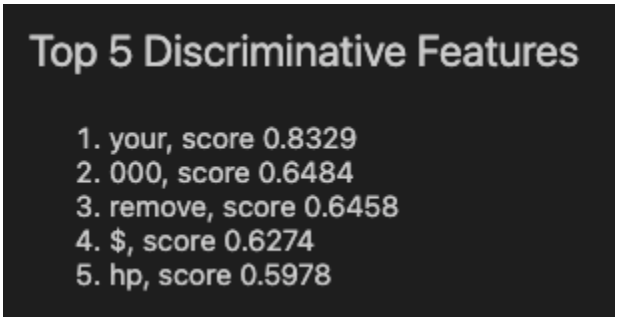

Figure 6. Top five Discriminitive Features

Metric	Value
Accuracy	0.8217
Precision	0.7233
Recall	0.9385
F1-Score	0.8170

Figure 7. Results table from part C

Conclusion

Overall, this lab strengthened my understanding of how different machine learning algorithms operate under different assumptions. Implementing each model from scratch helped me better connect the mathematical formulas to actual code. I learned how critical preprocessing is, especially for k-NN, where feature scaling directly affects distance calculations. I also saw how hyperparameter tuning, such as selecting k, can significantly change performance. The Naive Bayes implementation highlighted the importance of numerical stability and demonstrated how relatively simple probabilistic models can perform well on high-dimensional data. This assignment emphasized that model performance depends heavily on both the dataset characteristics and the implementation details, and that thoughtful design decisions are just as important as the algorithm itself.