COSC179 - Coding Assignment II Writeup

Eric Brown

Feburary 2026

Problem 2A - Linear Regression

Design/Approach

The goal of this problem was to predict the citric acid level of wine using a linear regression model. The linear regression model was implemented with "np.linalg.lstsq(X_b, y)" which computes the least squares. Then we found the 3rd and 4th best variables by RMSE.

Model Overview and Parameter Selection

Which features are most important for predicting citric acid?

Alcohol and density where selected at the start as per instructions. Then, for Task 3, we needed to implement a method to find the 3rd best candidate. This was done by searching through the other variables and testing the RMSE. Then the variable with the lowest RMSE is taken. Low RMSE = true to target values. Then we found fixed_acidity to be the 4th best variable.

Does adding more features always improve the model?

Typically, starting from one feature and then iteratively adding the best feature improves the model greatly. Eventually it will hit a limit. Many Python libraries streamline this testing. In datasets with many features, adding all features will increase computational costs with little improvement to the overall model.

Evaluation Results

Model	Features	Error
Model 1	alcohol, density	0.1686
Model 2	alcohol, density, volatile_acidity	0.132040
Model 3	alcohol, density, volatile_acidity, fixed_acidity	0.124159
Full	all features	0.105519

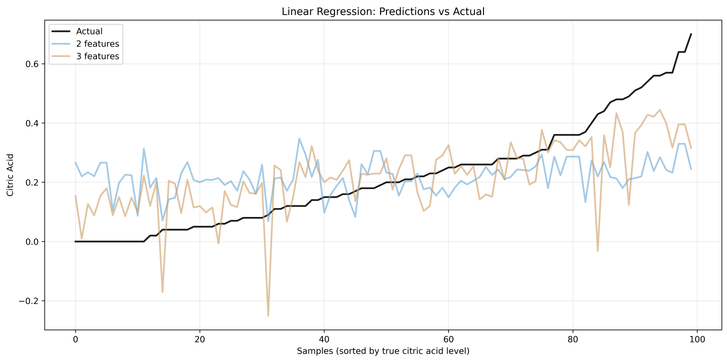

Figure 1: Plot comparing the 2nd and 3rd best feature model predictions for citric values to the true citric value. The X axis is sorted by the true citric acid level from lowest to highest.

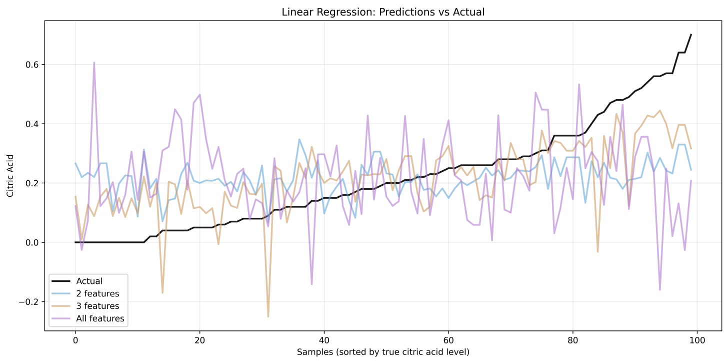

Figure 2: Plot comparing all model predictions for citric values to the true citric value. The X axis is sorted by the true citric acid level from lowest to highest

What did you learn from this exercise?

I have experience with applying data analytics libraries to datasets, especially ML methods. However, I have not dove into the mechanism and theory behind these libraries/functions. This course is very interesting. I feel I learn more because I can visualize how something we are working on (i.e RMSE for feature selection) fits into the overall process. In this case preprocessing.

Problem 2B - KNN for Lenses and Credit Approval Dataset

Design/Approach

The objective of this section is to implement a KNN model from scratch and evaluate how classification accuracy changes with different k values on two different datasets. For each input sample the distance to neighbors is computed using the L2 Euclidean distances, then the k closest neighbors are chosen. The class with the most NN is the predicted class. For the preprocessing stage, we needed to encode some features and clean the data values as specified in the readMe.

Model Overview and Parameter Selection

Which value of k works best for each dataset? Why do you think that is?

For the Lenses dataset a lower k value works best. This is because the dataset is small and limited, meaning edge cases will be misclassified.

For the credit score, higher k values yield better results. This is due to the dataset being much larger, and having more features. This allows the data to separate into classes and allow machine learning methods to understand the underlying patterns between features.

How did preprocessing affect your results on the Credit Approval dataset?

Preprocessing is essential in eliminating bias and noise. Most importantly, scaling the data allows for an "even playing field" between data points. Larger values are not overshadowing the smaller more subtle patterns.

What are the trade-offs of using different values of k?

When you have a limited dataset and a high number of K neighbors, the edge cases tend to get misclassified. It applies to larger datasets as well, however, the effects are mitigated because there are more datapoints to reflect the real world. The best course of action would be to preprocess the dataset, then begin training the model. A method such as Bayesian Optimization can be employed to automatically tune a model to the best k nearest neighbors.

Evaluation Results

Dataset	k=1	k=3	k=5	k=7
Lenses	100.00	100.00	50.00	83.33
Credit Approval	74.64	78.26	80.43	79.71

Reflection

What did you learn from this exercise?

This exercise was very interesting. Preprocessing is the most essential stage in machine learning. If not implemented correctly the ML model can crash. It was very cool implementing the statistical methods from scratch, such as KNN with L2 distance

Problem 2C - Naive Bayes

Design/Approach

For this exercise, we need to implement Naive Bayes from scratch to predict and evaluate spam with the given equations using log probabilities to avoid underflow:

- Class prior: $P(C) = \frac{N_C}{N}$
- Gaussian likelihood: $P(x_i|C) = \frac{1}{\sqrt{2\pi\sigma_{i,C}^2}} \exp\left(-\frac{(x_i-\mu_{i,C})^2}{2\sigma_{i,C}^2}\right)$

There was also a variance introduced. This is redundant for the assignment, but it is good practice to implement to avoid dividing by 0.

Model Overview and Parameter Selection

What are the limitations of your implementation?

We are using a fixed epsilon and no tuning on parameters. There were also 140 real emails identified as spam emails. I hypothesis this is due to a limited dataset and not parameter tuning. From the top 5 discriminative features we can see "your" and "you" play a key role. These are common words that can be throwing the model off.

How could you improve the classifier?

We could modify the features, such as combining "you" and "your" into one category, to see if we can reduce the weight of the feature. This way, we can understand the underlying pattern and hopefully improve the model score. We can also tune the decision threshold to find optimal parameters.

Evaluation Results

```
### Results Table
| Metric    | Value   |
|-----------|---------|
| Accuracy  | 82.1739 |
| Precision | 72.3320 |
| Recall    | 93.8462 |
| F1-Score  | 81.6964 |

### Top 5 Discriminative Features
1. Feature 20 - word_freq_your
2. Feature  6 - word_freq_remove
3. Feature 22 - word_freq_000
4. Feature 52 - char_freq_$
5. Feature 18 - word_freq_you
```

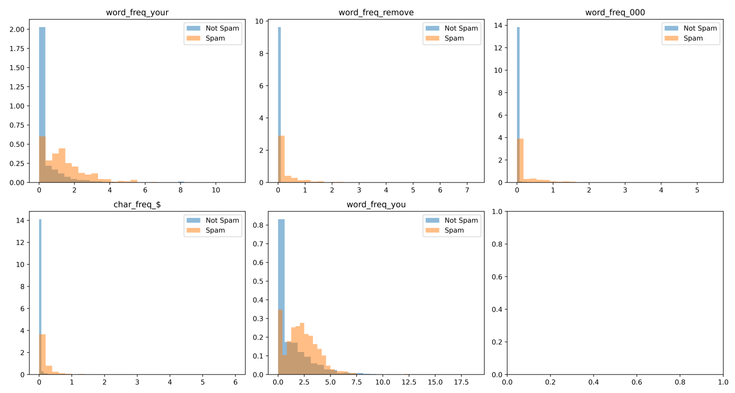

Figure 3: Distribution of top features for spam vs non spam.

Reflection

Why is Naive Bayes effective for spam detection despite the independence assumption?

Naive Bayes is a simple algorithm that is computationally efficient. Computational efficiency is a key aspect when implementing ML methods. In our case we are scrubbing emails for spam. We need to have real time capabilities so the email is being sent to the recipient as quick as possible. Models like SVM will not scale with the demand.

What did you learn from this exercise?

In this exercise we built a Gaussian Naive Bayes model from scratch, which helped link the theoretical side to implementation in code. The exercise also helped understand the model parameters and what tuning them does.