Final Project Report
Lead by Takira Boltman

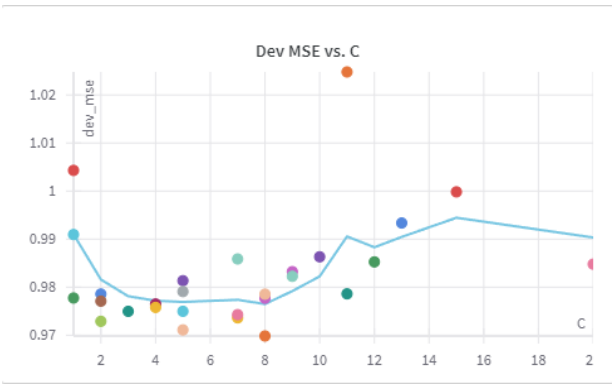
Task: I chose to work on Task 3, a regression task predicting the rating a user gives a product based on the term-frequency vector of their review of the product.

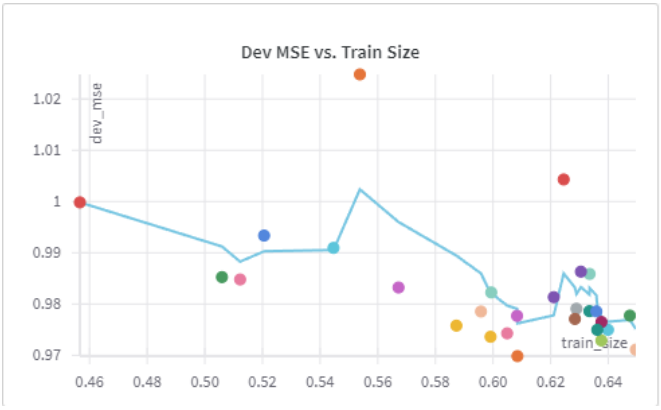
Distribution of Work: The work done for Task 3 was entirely done by Takira Boltman.

Methods: I started with Linear Regression as the baseline but changed to Logistic Regression to capture more complexity. I employed Lasso (L1 Regularization) for its ability to reduce features and mitigate overfitting. I explored non-linear relationships using Random Forests, which enhanced my model's robustness. I chose Support Vector Regression (SVR), with its variety of kernels, for its adaptability with sparse datasets. I utilized TruncatedSVD (Truncated Singular Value Decomposition), to reduce dimensions. To normalize my data distribution, I applied log and TF-IDF (Term Frequency-Inverse Document Frequency) transformations. These methods were carefully selected for their compatibility with the sparse matrices.

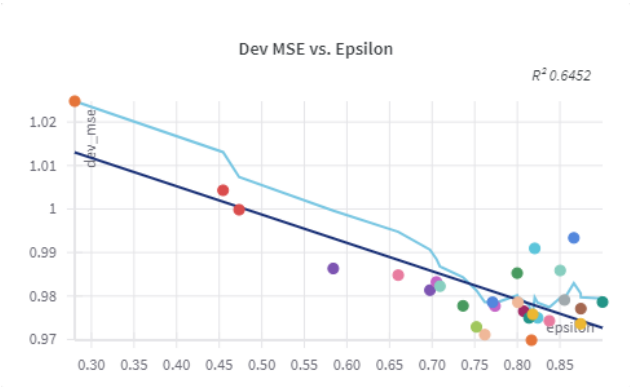
Submission Model Details: I applied TF-IDF transformation on my Train and Dev features since this gave better results as opposed to the log transformation. This helped reduce the bias towards high-frequency terms. I applied TruncatedSVD with a fixed value of dimensions (1006) which was needed to reduce computational resources to both Train and Dev sets. Hyperparameter tuning was done entirely in the Weights & Bias toolkit using the Bayes method. The values of those hyperparameters for the model are C = 8, ε = 0.82, train_size = 0.61 (61% of my training data), and a linear kernel. C is a way to tell your model how much you care about avoiding mistakes (also called the Regularization factor). Since our C is higher, we work harder to make each prediction correct. ε, or Epsilon, refers to the loss function that allows for a certain degree of “slack” in our predictions and acts as a margin around our hyperplanes. Our ε is smaller to focus on getting predictions close to the actual values but allows some room for predictions of a certain range. The kernel refers to the linear transformation used to capture the essence of our data.

Results: I took the “best” 20 runs with my best model (the 20 with the lowest Dev MSE scores) and took the average, which resulted in an average Dev MSE of 0.9758. The average Dev MSE for my baseline was 2.024 (over 20 runs), which is much better than the average of Linear Regression, which was 13.246. Lasso had an average Dev MSE of 1.614, and the average Dev MSE for Random Forests was 1.35. While Lasso and RandomForests were promising, SVR’s assumptions and characteristics aligned better with the underlying distribution and the overall structure of the data. Typical log transformations before the introduction of TF-IDF transforming yielded an average Dev MSE of 1.1, but I believe it was because the log transformations weren’t taking into account the frequency of words that TF-IDF does since it is specifically for term frequency vectors. While log transformations may help normalize the distribution better, it wasn’t as efficient in handling the sparse data format of the data as TF-IDF was.





These images show how the Dev MSE is affected by the final hyperparameters, and how I tuned them based on these relationships. We can see that a low Regularization value correlates to a lower Dev MSE, but then starts to increase as the C value increases. This means that if we increase C too much, it could start overfitting and not capturing the proper underlying patterns in unseen data. For our train size, at first, it was unrealistic to use the full amount of data due to the size and dimensions. As we increase our training size, the Dev MSE starts to decrease but tends to spike in some places. Gradually increasing the size of the training set to fully capture the variability and complexity of the underlying data distribution is important, but adding too much may have diminishing returns, such as computational resources.



Additionally, we see that an increase in ε correlates with a decrease in the Dev MSE. This could mean several things, but in my case, my model was previously overfitting quite a bit, and a higher ε balanced with our Regularization parameter allowed me to add a bit more bias to the training data, which could have resulted in better generalization in the Dev set. In our final model, our best Dev MSE was 0.96 and our best Train MSE was 0.91, meaning there is very slight overfitting still, but an improvement overall.