Predict Hawaii Google Local Business Review Rating

ABSTRACT

This project focuses on analyzing user reviews from the Google Local Reviews dataset for Hawaii, sourced from UCSD's McAuley Group.[1] [2] We aimed to build a recommendation system to predict user ratings based on review data. Various machine learning models, including Linear Regression, SGD Regressor, Latent Factor Models (Baseline and SVD++), Random Forest tree referencing Alomari's work[3], and Bi-GRU with Glove model referencing Ahmed's work [4] were implemented and evaluated. Mean Squared Error (MSE) and Confusion matrix were used to compare the models' performance. This study highlights how data-driven machine learning models can be applied to recommendation systems, providing a better understanding of user preferences and improving recommendations.

1 INTRODUCTION

Recommendation systems are everywhere these days, helping users find what they like on platforms like Amazon. This report explores the application of machine learning algorithms to predict user ratings in a recommendation system, focusing on accuracy and efficiency. The study implements several models, including Linear Regression, SGD Regressor, Latent Factor Models, Random Forest tree[3], and Bi-GRU with Glove model[4], to compare their performance. Using user-item interactions, the goal is to identify the optimal configuration of the model for an accurate rating prediction. This research underscores the impact of algorithmic choices and parameter optimization.

2 METHOD

2.1 Data Exploration

The dataset used for this project consists of reviews of local businesses in Hawaii. The dataset comprises 777,174 entries with 8 distinct features, providing a rich resource for exploring user feedback and ratings.

The main features of the dataset include:

- **user_id**: A unique identifier for each user, used to analyze individual review patterns and contributions.
- **name**: The name associated with the reviewer or business, providing contextual relevance to the review data.
- **time**: Timestamp of the review, capturing temporal patterns and trends in user feedback.
- **rating**: Numeric score (ranging from 1 to 5), representing user satisfaction.
- **pics**: URLs or metadata for attached images, which are not utilized in this analysis but represent potential for future work.
- **resp**: Business responses to reviews that highlight business engagement and interaction.
- **gmap_id**: Geolocation identifier for businesses that enables geographic and spatial analysis.

2.2 Relevant Data

For this project, we focused on four key features from the dataset: user_id, gmap_id, text, and rating. These features were selected for their significance in capturing user behavior, spatial information, and review sentiment, which collectively contribute to predicting business ratings.

The user_id feature represents a unique identifier for each user. This feature allows us to analyze patterns in individual user behavior, such as the frequency of reviews, consistency in ratings, and possible biases. By examining the historical activity of users, the model can identify trends that might influence ratings. For instance, users who habitually provide low ratings may influence the predictive power of their reviews. Moreover, incorporating user_id helps the model recognize repeat interactions between specific users and businesses, which may affect the rating.

The gmap_id feature acts as a geolocation identifier for businesses, linking each review to a specific place. This feature is crucial for understanding spatial trends in reviews. Including gmap_id enables the model to consider the geographic context of a review, such as proximity to popular landmarks or competition density.

The text is the review written by the user. It contains a detailed description of user experiences, also correlated to the numerical rating. By processing the feature with TF-IDF vectorization, we can extract key phrase and sentiment for reviews.

The rating feature is the dependent variable and represents the main metric to assess user satisfaction. It ranges from 1 to 5, with higher values indicating greater satisfaction. This feature serves as the target variable for the model, making it the most critical component of the analysis.

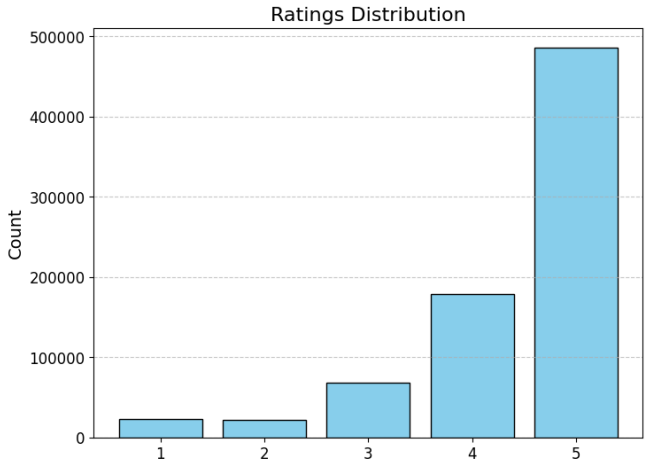

Figure 1: Rating Distribution

2.3 Data Preprocessing

To prepare the dataset for analysis and modeling, a preprocessing step was applied to clean and refine the data. Our goal was to handle missing values and filter out irrelevant entries.

The text column, which contains unstructured review content, was the only column with missing values. We removed all the None or null value and ensure that it only contains meaningful and Since textual data plays a crucial role in predicting the ratings, all rows with None or null values, and non-alphanumeric characters in the text column were removed.

2.3.1 ***Outliers and Null Values in Other Columns.*** After a thorough examination of the remaining columns (user_id, gmap_id, rating), no outliers or null values were found. As a result, no additional cleaning was required for these fields.

2.4 Model Training - Similarity

2.4.1 ***User-User Similarity with Jaccard Predict (baseline).*** We use Jaccard similarity to measure the overlap in items rated by different users. The main idea is to identify users who have rated similar items and use their ratings to predict the target user's rating.

$$J(u, u') = \frac{|I_u \cap I_{u'}|}{|I_u \cup I_{u'}|} \tag{1}$$

where:

- $J(u, u')$: Jaccard similarity between users u and u',
- I_u: Set of items reviewed by user u,
- $I_{u'}$: Set of items reviewed by user u',
- $|A|$: Cardinality (number of elements) in set A.

In the predictRating_Jaccard function, ratings from similar users are adjusted by subtracting itemAverages to account for item-specific biases. These adjusted ratings are weighted by their Jaccard similarity scores, summed, and normalized to predict the target user's rating. If no similar users exist, the model defaults to the global average ratingMean.

$$\hat{r}_{u,i} = \mu_i + \frac{\sum_{i' \neq i} (r_{u,i'} - \mu_{i'}) \cdot J(i, i')}{\sum_{i' \neq i} J(i, i')} \tag{2}$$

where:

- $\hat{r}_{u,i}$: Predicted rating for user u and item i,
- μ_i: Average rating for item i,
- $r_{u,i'}$: Rating given by user u to item i',
- $J(i, i')$: Jaccard similarity between items i and i',
- $i' \neq i$: Only items other than i are considered in the summation.

The Jaccard User-based Similarity Model achieved an MSE of 0.803, indicating a reasonable level of accuracy in predicting user ratings.

2.4.2 ***User-User similarity with Cosine predict.*** The user-user similarity model with Cosine similarity predicts a user's rating for an item by evaluating the cosine similarity between the items rated by the user.

Cosine similarity is defined as:

$$\text{Cosine}(i, i') = \frac{\sum_{u \in U_i \cap U_{i'}} r_{u,i} \cdot r_{u,i'}}{\sqrt{\sum_{u \in U_i} r_{u,i}^2} \cdot \sqrt{\sum_{u \in U_{i'}} r_{u,i'}^2}} \tag{3}$$

where:

- U_i: Set of users who have reviewed item i,
- $r_{u,i}$: Rating given by user u to item i,
- $\cap$: Denotes the intersection of users who reviewed both i and i'.

Similar to the Jaccard model, the predicted rating for a user u and an item i is calculated as a weighted sum of the deviations of similar items' ratings from their averages:

$$\hat{r}_{u,i} = \mu_i + \frac{\sum_{i' \neq i} (r_{u,i'} - \mu_{i'}) \cdot \text{Cosine}(i, i')}{\sum_{i' \neq i} \text{Cosine}(i, i')} \tag{4}$$

where:

- $\hat{r}_{u,i}$: Predicted rating for user u and item i,
- μ_i: Average rating for item i,
- $r_{u,i'}$: Rating given by user u to item i',
- $\text{Cosine}(i, i')$: Cosine similarity between items i and i',
- $i' \neq i$: Indicates that only items other than i are considered in the summation.

If no similar items exist ($\sum_{i' \neq i} \text{Cosine}(i, i') = 0$), the global mean rating is used as the prediction.

This model performed better than the Jaccard-based method, achieving a lower MSE of 0.798. However, it faces scalability issues, as calculating pairwise cosine similarities between all items can become computationally expensive in larger datasets.

2.5 Model Training - Text Mining

2.5.1 ***Bag-of-words models with Ridge Regression (Sentiment analysis).*** The bag-of-words model with Ridge Regression predicts ratings by converting text reviews into numerical feature vectors using a bag-of-words representation. Text preprocessing includes lowercasing, punctuation removal, and stopword elimination using NLTK's stopword list. The most frequent 1000 words are selected to create a fixed vocabulary, and each review is represented as a vector of word counts (feature function), with an additional bias term. The Ridge Regression model is well-suited for this task due to its ability to handle high-dimensional feature spaces like bag-of-words while controlling overfitting with L2 regularization.

Text Preprocessing: To prepare the review text for analysis, the following preprocessing steps were performed:

- **Lowercasing**: All text was converted to lowercase to ensure uniformity.
- **Punctuation** Removal: Punctuation was removed to retain only the core words.
- **Stopword Removal**: Common English stopwords, such as "the" and "is", were removed using the NLTK stopword library to reduce noise and focus on meaningful words.

The top 1,000 most frequent words from the dataset were identified. These words form the basis of the feature vectors, where each dimension represents the frequency of a specific word in a review. Each review was transformed into a vector of word counts for these top words. A bias term (intercept) was also added to the feature vector.

1. Feature Vector Construction
For each review d, the feature vector $\mathbf{x}_d$ is constructed as:

$$\mathbf{x}_d = [f_1, f_2, \ldots, f_n, 1] \tag{5}$$

where f_i represents the frequency of the i-th word in the review, and the final component 1 is added for the intercept term.

2. Prediction Function
The predicted rating $\hat{y}_d$ for a review is computed as:

$$\hat{y}_d = \mathbf{x}_d \cdot \boldsymbol{\theta} = \sum_{i=1}^{n} f_i \theta_i + \theta_{n+1} \tag{6}$$

where:

- $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_n, \theta_{n+1}]$ are the coefficients (weights) learned by Ridge Regression,
- θ_{n+1} is the bias (intercept) term.

3. Objective Function
Ridge Regression minimizes the regularized Mean Squared Error (MSE):

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{d=1}^{m} (y_d - \hat{y}_d)^2 + \lambda \sum_{i=1}^{n} \theta_i^2 \tag{7}$$

where:

- m: Number of training samples,
- y_d: True rating for review d,
- λ: Regularization parameter that penalizes large weights to prevent overfitting.

Most Influential Words: The weights assigned by the Ridge Regression model to individual words reveal their relative importance in predicting ratings. The top 10 words with the highest positive weights are: "awesome", "reasonably", "excellent", "heaven", and "onolicious" indicate positive experiences and strongly correlate with higher ratings. On the other hand, "Worst", "horrible", "terrible", "rude", and "poor" indicate negative experiences and strongly correlate with lower ratings

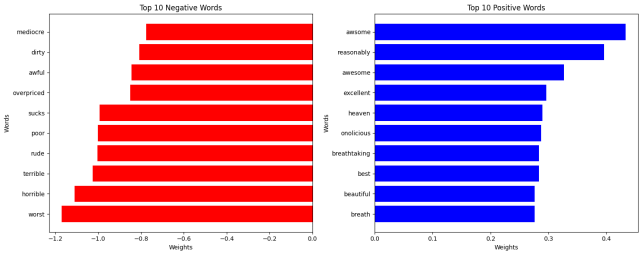

Figure 2: Top 10 words for positive and negative

Ridge regression, a linear model with L2 regularization, is then trained to predict ratings (y) using the bag-of-words feature matrix (X). The training and testing model both achieved around 0.65 for MSE.

2.5.2 ***Bag-of-words models with transformation techniques.*** The bag-of-words model, evaluated with transformation techniques, which is inspired by by Alomari et al.[3], which is based on logistic regression and is used to classify reviews into three sentiment categories: positive, negative, and neutral. A logistic regression model was trained on the labeled data set and achieved an accuracy of 87.6%.

True Positive (TP): The number of instances where the true label matches the predicted label for a specific class.

False Positive (FP): The number of instances where the model predicted the class, but the true label is something else.

False Negative (FN): The number of instances where the true label is the class, but the model predicted something else.

True Negative (TN): The number of instances where the true label is not the class, and the model predicted something else.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Accuracy} = \frac{\text{Total Correct Predictions (TP + TN)}}{\text{Total Number of Predictions}}$$

The confusion matrix results reveal varying performance across classes. For the positive class, the model achieves a high precision (0.89) and recall (0.99), indicating its strong ability to correctly identify positive reviews and minimize false negatives. For the negative class, the precision is moderate (0.67), but recall is low (0.37), suggesting difficulty in capturing all negative reviews. The neutral class shows the weakest performance, with low precision (0.48) and recall (0.14), reflecting the challenge of distinguishing neutral reviews, often due to the nuanced and ambiguous nature of such sentiments.

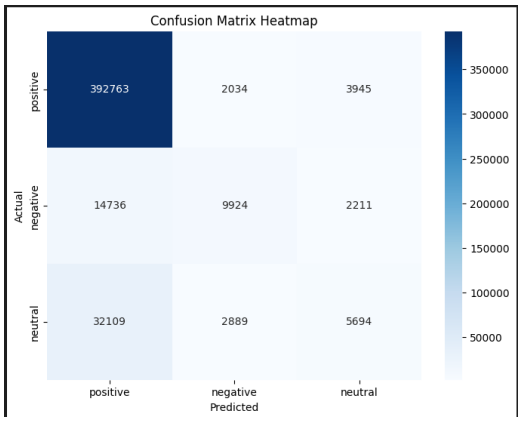

Figure 3: Confusion Matrix

2.5.3 ***Bag-of-words models with Linear Regression.*** Linear Regression predicts user ratings based on text reviews converted into numerical feature vectors. The CountVectorizer transforms the processed text into a bag-of-words representation, selecting the top 2000 most frequent words as features. Linear regression is trained on this feature matrix to predict numerical ratings, achieving a MSE of 0.628 on the test set.

However, the bag-of-words representation has inherent limitations, such as ignoring word order and semantic meaning. Consequently, the model may struggle with nuanced reviews where sentiment depends on context or word relationships.

Steps in the Bag-of-Words Model.

(1) **Bag-of-Words Representation**: Convert the preprocessed text T' into a numerical feature vector X using a word-count representation. Let the vocabulary $V = \{w_1, w_2, \ldots, w_n\}$ contain the top 2000 most frequent words across the corpus. The feature vector X for a review is:

$$X_i = \text{Count}(w_i, T'), \quad \forall w_i \in V$$

(2) **Linear Regression Prediction**: The model predicts user ratings based on the feature matrix X and learned weights β. The predicted rating $\hat{y}$ for a review is given by:

$$\hat{y} = \beta_0 + \sum_{i=1}^{n} \beta_i X_i$$

where:
- β_0: Bias term
- β_i: Weight corresponding to feature X_i

(3) **Mean Squared Error (MSE)**: To evaluate the model, compute the MSE between the predicted ratings $\hat{y}_j$ and the true ratings y_j over all test samples j:

$$\text{MSE} = \frac{1}{m} \sum_{j=1}^{m} (y_j - \hat{y}_j)^2$$

where m is the number of test samples.

2.5.4 ***TFIDF - Linear Regression.*** The TF-IDF transformation, which weighs word frequencies by their importance across the corpus, reduces the impact of commonly occurring but less meaningful terms while preserving informative features. Linear Regression then fits these TF-IDF vectors to the corresponding user ratings, predicting continuous values clipped to the range [1, 5]. This approach achieves a MSE of 0.559 for Linear Regression, demonstrating strong predictive performance.

Linear Regression is a natural choice for this task due to its simplicity and efficiency in handling high-dimensional feature spaces like TF-IDF. Unlike classification-based models, it directly predicts numerical ratings, making it well-suited for datasets with continuous target variables.

TF-IDF Formula in the Context of Text Reviews:

For a **word** w, a **text review** r, and the entire **corpus** C of reviews:

$$\text{TF-IDF}(w, r, C) = \text{TF}(w, r) \times \text{IDF}(w, C)$$

1. TF Definition: Measures the frequency of a **word** w in a given **text review** r:

$$\text{TF}(w, r) = \frac{\text{Number of times the word } w \text{ appears in the text review } r}{\text{Total number of words in the text review } r}$$

2. IDF Definition: Measures how unique or rare a **word** w is across all **text reviews** in the entire **corpus** C:

$$\text{IDF}(w, C) = \log\left(\frac{\text{Total number of text reviews in the corpus } C}{1 + \text{Number of text reviews containing the word } w}\right)$$

2.5.5 ***TFIDF - SGDRegressor.*** The TF-IDF model with SGDRegressor predicts user ratings by using a stochastic gradient descent algorithm to optimize a linear regression model. Text reviews are transformed into TF-IDF vectors, which highlight the relative importance of words in the dataset while down-weighting common terms.

SGD Update Rule: Parameters ($\mathbf{w}$) are updated iteratively:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \cdot \nabla \ell(\mathbf{w}; x, y)$$

Where:
- η: Learning rate
- $\nabla \ell$: Gradient of the loss function

Final Results:
- **SGD Regression MSE**: 0.5837

By processing data in mini-batches, the model reduces memory usage compared to batch gradient descent and is well-suited for datasets with millions of features. Additionally, its iterative nature allows for early stopping based on convergence criteria, improving computational efficiency.

2.6 Model Training - Latent Factor

2.6.1 ***Latent Factor Model.*** The Latent Factor Model predicts user ratings by incorporating user and item biases using a baseline prediction algorithm. This model estimates a user's rating for an item as the sum of the global average rating, user-specific bias, and item-specific bias. The Baseline Model provides an interpretable framework that focuses on biases inherent in user behavior (e.g., some users rate consistently higher) and item characteristics (e.g., some items receive universally higher ratings).

The predicted rating is computed as:

$$\hat{r}_{u,i} = \mu + b_u + b_i \tag{8}$$

where:
- $\hat{r}_{u,i}$: Predicted rating for user u and item i,
- μ: Global average rating across all users and items,
- b_u: User bias, representing the deviation of user u's ratings from the global average,
- b_i: Item bias, representing the deviation of item i's ratings from the global average.

By calculating these biases, the model achieves reasonable MSE of 0.738 on the test set.

2.6.2 ***SVD++.*** This Latent Factor Model using SVD++ predicts user ratings by utilizing both explicit user-item interactions and implicit rating feedback, such as user engagement patterns. This model extends the traditional SVD by incorporating implicit data (e.g., items a user interacted with but did not rate) to improve predictive accuracy. SVD++ learns latent factors for users and items, with parameters such as n_factors (number of latent factors), lr_all (learning rate), and reg_all (regularization strength) optimized using a grid search with cross-validation.

$$\hat{r}_{u,i} = \mu + b_u + b_i + q_i^T \cdot \left(p_u + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} y_j\right) \tag{9}$$

where:

- μ: Global average rating across all users and items,
- b_u: User bias, representing the deviation of user u's ratings from the global mean,
- b_i: Item bias, representing the deviation of item i's ratings from the global mean,
- q_i: Latent factor vector for item i,
- p_u: Latent factor vector for user u,
- $N(u)$: Set of items interacted with by user u,
- y_j: Implicit feedback factor for item j.

To optimize the performance of the SVD++ model, a grid search was conducted over the following hyperparameters:

- $n\_factors$: Number of latent factors (p_u, q_i), tested values: {20, 50, 100},
- lr_all: Learning rate for gradient descent, tested values: {0.01, 0.1, 0.5},
- reg_all: Regularization parameter, tested values: {0.05, 0.1, 0.2}.

The best hyperparameters identified were:

Best parameters: { n_factors: 100, lr_all: 0.01, reg_all: 0.1 }

The SVD++ model achieved an MSE of 0.720 on test set, significantly improving upon the BaselineOnly(latent) model (MSE = 0.738). The incorporation of implicit feedback and optimized hyperparameters allowed the SVD++ model to capture nuanced relationships between users and items.

While the SVD++ model demonstrates superior predictive performance on the test set compared to the BaselineOnly model, the significant difference between the train MSE (0.480) and the test MSE (0.720) suggests potential overfitting. This may be due to the fact that the model does not generalize well to unseen data because it captures and fits noise and specific patterns in the training data. To mitigate overfitting and further improve generalization, the following strategies could be considered: 1. Regularization Tuning: While the current regularization parameter (reg_all:0.1) was optimized, further refinement or testing a broader range of values could help in controlling model complexity. 2. Learning Rate Adjustments: Experimenting with adaptive learning rates or smaller increments around lr_all:0.01 may yield better convergence to an optimal solution, preventing overstep. 3. Early Stopping: Monitoring performance on a validation set and halting training once improvement plateaus could prevent overfitting.

2.7 Model Training - other

2.7.1 ***Random Forest Tree***. Random Forest is an ensemble model that combines predictions from multiple decision trees, improving accuracy and robustness by handling high-dimensional data effectively. This method was inspired by Alomari [3], they predict ESRB by feeding it 38 titles of Content Descriptors. This approach is a solid learning technique known for its ability to efficiently handle high dimensional complex data, but less forgiving noise and outlier. Since combining the predictions of multiple decision trees, Random Forest ensuring better generalization to unseen data and requires minimal parameter tuning and its ability to handle high-dimensional feature spaces effectively, we believe using this with TFIDF which contain large amount of text features from the review corpus is a practical choice for predict regression problem. Specifically, the Random Forest model was configured with 20 decision trees (n_estimators=20), balancing computational efficiency with prediction accuracy. The model's performance was assessed using Mean Squared Error (MSE), achieving a value of 0.592 on the test set, and 0.168 on the train set.

During implementation, we encountered scalability issues due to the large size of the TF-IDF feature matrix, which increased memory usage and computation time. To address this, dimensionality reduction techniques, such as Principal Component Analysis (PCA), were explored but required careful tuning to preserve predictive information. Also, we notice our MSE for train and test has considerable gap in performance suggesting that the model struggled to generalize to unseen data. This disparity is a classic symptom of overfitting, where the model captures noise and specific patterns in the training data that do not generalize well to new examples. We can address overfitting with several strategies 1. balance our validation and training set. 2. Adjusting the minimum number of samples required to split a node and the minimum number of samples per leaf helped create a more generalized model. 3. Constrain the maximum depth of individual decision trees, which we reduced their complexity and prevented them from modeling overly specific patterns.

2.7.2 ***Bi-GRU with Glove model***. Bi-GRU is an efficient sequence modeling tool that excels in capturing contextual relationships in text through its bidirectional structure, making it particularly suitable for tasks like sentiment analysis and rating prediction. Inspired by the article on leveraging Bi-GRU for rating prediction[4], we employed Bi-GRU to predict Google Local Review ratings. This model processes text sequences in both directions, significantly enhancing contextual understanding. Its gated mechanism mitigates the vanishing gradient problem effectively, and the integration of pre-trained GloVe embeddings (100 dimensions), sourced from Stanford NLP[5], further improved semantic representation.

Experimental results demonstrated that the Bi-GRU model with randomly initialized embeddings achieved a test MSE of 0.45, while incorporating GloVe embeddings reduced the test MSE to 0.43, highlighting the importance of pre-trained embeddings in improving model performance. GloVe embeddings further enhanced the model by reducing the burden of learning basic semantic relationships, allowing the network to focus on dataset-specific patterns.

The following is the GRU model structure figure from Kostadinov's article[6]:

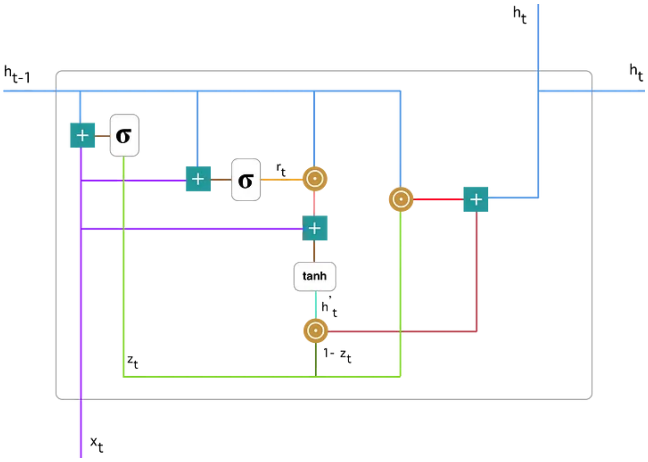

Figure 4: Single GRU structure

The formula for each update gate is defined as:

$$z_t = \sigma(W_z X_t + U_z h_{t-1}) \tag{10}$$

$$r_t = \sigma(W_r X_t + U_r h_{t-1}) \tag{11}$$

$$h'_t = tanh(W X_t + r_t \odot U h_{t-1}) \tag{12}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \tag{13}$$

3 LITERATURE DISCUSSION

The Google Local Reviews dataset for Hawaii, was sourced from UCSD's McAuley Group.[1] [2] When using unsupervised methods like UCTOPIC, show us how pre-trained embeddings and context-aware representations can be applied to topic mining. UCTOPIC's use of contrastive learning ensures high-quality phrase embeddings, which can enhance the textual understanding in recommendation systems. The goal for this technique it to be able to accurately capture semantic relationships may be adapted to better interpret user reviews and improve model robustness. UCTOPIC approach and Ahmed's work[4] inspire us to use Bi-GRU-based deep learning model for sentiment polarity prediction and rating estimation. Since both studies emphasize the importance of sentiment analysis for capturing the emotional tone of user feedback. In addition, the use of pre-trained word embeddings, such as GloVe in RRPF, is particularly noteworthy. These embeddings capture semantic and syntactic relationships between words, similar to how textual features are extracted in Assignment2 to improve model input quality. The adoption of embeddings like GloVe could be a promising enhancement for recommendation systems to better capture nuances in user reviews. So we assuming using Bi-GRU with Glove model doing rating prediction related jobs, could have outstanding results.

As we explore the importance of semantic and syntactic relationships between words, the work by Alomari et al.[3] gives us another perspective. They focus on binary descriptors (e.g., violence, strong language), which map directly to the prediction target using transformation techniques. This inspires us to transform our data into three categories, positive, negative, and neutral. Also, the article by Alomari et al.[3], indicates Random Forests versatility for high-dimensional, complex data. However, in order to balance the computational efficiency, we reduced the number of trees, which results in intermediate performance compared to other models. If we integrate with the other approaches such as sentiment analysis and binary encoding across domains, both projects could benefit from improved prediction accuracy and explainability. Exploring hybrid approaches and cross-domain feature engineering represents a promising direction for future research.

4 RESULTS

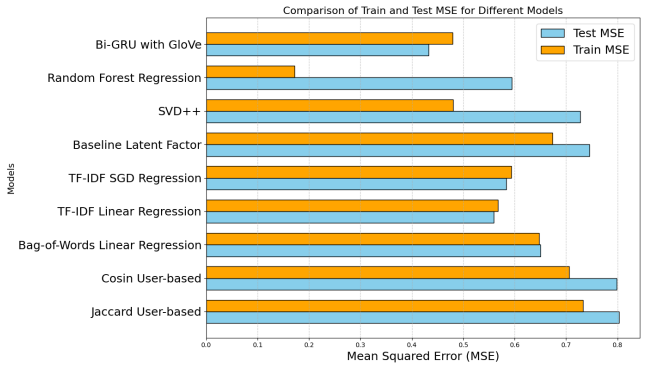

Figure 5: MSE results of different models

TOP 7 BEST MODEL ANALYSIS TABLE

Model	Strengths	Limitations
Bi-GRU with GloVe	Captures sequence and context; uses semantic embeddings.	Computationally intensive.
TF-IDF Linear Regression	Simple and interpretable; leverages word importance.	Ignores word order and context.
TF-IDF SGD Regression	Scalable and flexible.	Optimization variability; lacks sequence modeling.
Random Forest	Captures nonlinear relationships; robust ensemble.	Ignores word order and textual context.
Bag-of-Words Regression	Simple and computationally efficient.	Ignores context and sequence entirely.
SVD++	Captures latent factors in user-item interactions.	Does not utilize text.
Jaccard & Cosine Similarity	Simple similarity-based collaborative filtering.	Ignores textual data entirely.

Based on the experimental results, the Bi-GRU model with GloVe embeddings achieved the best performance in predicting Google Local Review ratings, with a test MSE of 0.43. Its ability to process text sequences bidirectionally allowed it to effectively capture nuanced sentiment and context, while GloVe embeddings provided

rich, pre-trained semantic representations that enhanced its generalization capability.

The performance ranking highlights the strengths and limitations of the tested approaches. While TF-IDF-based models, such as linear and SGD regression, performed moderately well due to their strong feature engineering, they lacked the ability to capture sequential dependencies. Traditional collaborative filtering models, like SVD++ and user-based similarity methods, performed poorly because they relied solely on user-item interactions and failed to utilize textual information. This shows the critical advantage of context-aware deep learning models like Bi-GRU, which particular outperform in integrating sequential patterns and semantic embeddings to achieve state-of-the-art performance in text-based rating prediction tasks.

5 CONCLUSIONS

This project showed how machine learning models can be used to predict user ratings for Google Local Reviews in Hawaii. Out of all the models tested, the Bi-GRU with GloVe embeddings performed the best. While traditional models like TF-IDF regression and collaborative filtering were useful, they fell short when it came to handling the deeper patterns and meanings in user reviews.

In the future, this experiment could be expanded by including different types of data, like images or business responses, to make the predictions even more accurate. Another exciting direction would be to explore modern transformer models like BERT or GPT, which could bring new ways to understand and analyze text. These improvements could lead to smarter and more personalized recommendation systems that better meet the needs of users.

More details and the code for our models can be founded here:

https://github.com/SamuelKao/HawaiiGoogleReview/blob/main/assignment2.ipynb

REFERENCES

[1] J. M. Jiacheng Li, Jingbo Shang, "Uctopic: Unsupervised contrastive learning for phrase representations and topic mining," *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

[2] J. L. T. Z. J. M. An Yan, Zhankui He, "Personalized showcases: Generating multimodal explanations for recommendations," *arXiv:2207.00422*, 2022.

[3] K. Alomari, A. Q. Alhamad, H. Mobaideen, and S. Salloum, "Prediction of the digital game rating systems based on the esrb," *Opcion*, vol. 35, pp. 1368–1393, 06 2019.

[4] B. H. Ahmed and A. S. Ghabayen, "Review rating prediction framework using deep learning," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 7, pp. 3423–3432, Jul 2022. [Online]. Available: https://doi.org/10.1007/s12652-020-01807-4

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[6] S. Kostadinov, "Understanding gru networks," *Towards Data Science*, Dec 2017. [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be