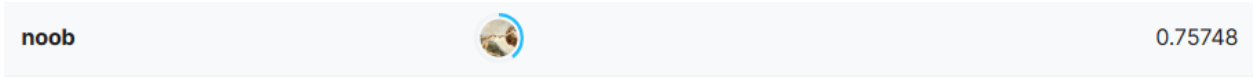
CS506 Midterm Report

Kaggle Username: qondojanfaf
Name on Leaderboard: noob



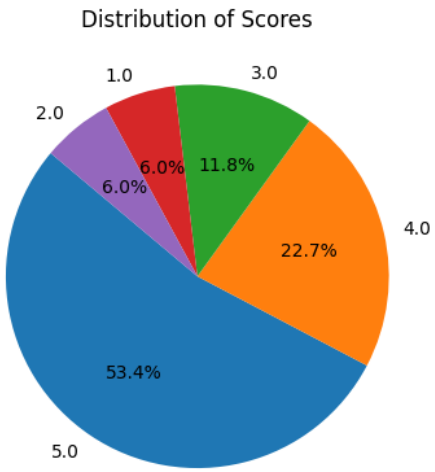
Name: Ketan Suhaas Saichandran
BU ID: U68176921

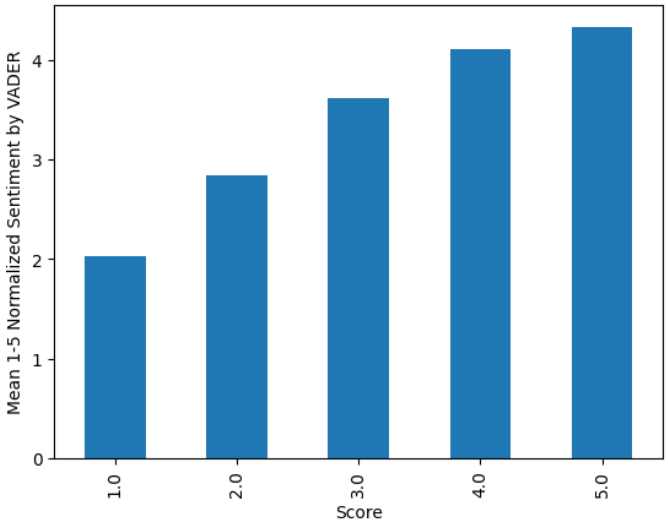
Finding Key Features

I employed a methodical approach to preprocess text data and discover essential features.

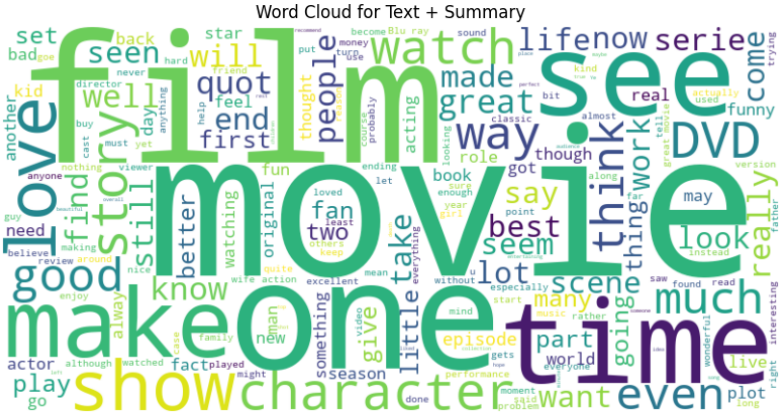
- **Text Cleaning:** I initiated the process by meticulously cleaning the text data. This entailed the removal of special characters and digits, converting all text to lowercase, eliminating stopwords, and applying lemmatization to the words. The choice of lemmatization over stemming consistently yielded better results.
- **Balancing the Dataset:** I observed that the dataset was imbalanced because a lot of customers had higher scores compared to lower scores. Initial efforts to balance the dataset, using techniques such as SMOTE, did not yield the expected improvements. As a result, I explored alternative approaches to enhance the model's performance.
- **Ensemble Methods:** To address the skewness, I harnessed the power of ensemble methods. This involved leveraging insights from multiple models and resampling, allowing for a more robust predictive framework.
- **Feature choice:** I dropped features like ProductId and UserId based on logical reasons and scaled the remaining numerical features using StandardScaler from sklearn.
- **Text Statistics:** Basic text statistics, including character count, word count, sentence count, average word length, average sentence length, etc provided valuable insights into the text's structure and I added some of them as feature columns.
- **TF-IDF:** I leveraged the "term frequency-inverse document frequency (TF-IDF)" representation. The optimal TF-IDF configuration was determined empirically, featuring a maximum of 50,000 features and n-grams ranging from unigrams to four-grams (1,4).
- **Dimension Reduction:** While dimension reduction techniques such as Latent Semantic Analysis (LSA) and Hashing Vectorizer were explored, their impact on performance was limited compared to using TF-IDF directly.
- **Sentiment Analysis:** The integration of VADER, a sentiment analysis tool, introduced a new column with sentiment scores ranging from -1 to 1. This addition significantly enhanced the model's performance by considering sentiment in the analysis.
- **GloVe Embeddings:** To enrich the feature space, 300-dimensional GloVe embeddings were incorporated. These embeddings were calculated by averaging embeddings per document, introducing valuable contextual information and leading to substantial performance improvements.

This systematic and detail-oriented approach allowed for the identification and effective utilization of key features, ultimately leading to an enhanced predictive accuracy for the model.





Selecting the Model



In the early stages, my primary focus was on classification models to predict class labels effectively. I explored several classifiers, including RandomForest, Logistic Regression, Decision Tree, and XGBoost. During this phase, I achieved promising results, especially with the RandomForestClassifier and XGBoost Classifier, which demonstrated their capabilities.

However, a significant turning point emerged when I made a pivotal discovery related to the Kaggle platform, which played a central role in my project. Kaggle's acceptance of float values for the "Score," a factor that significantly impacted my decision-making process, although I already knew that it was a ranking problem. Recognizing the potential of predicting fine-grained scores, I made a deliberate shift toward regression models. This transition was motivated by my desire to provide more nuanced and continuous predictions for the "Score" variable, aligning better with Kaggle's evaluation metrics. I realized that this approach would allow me to capture the subtleties of performance more accurately.

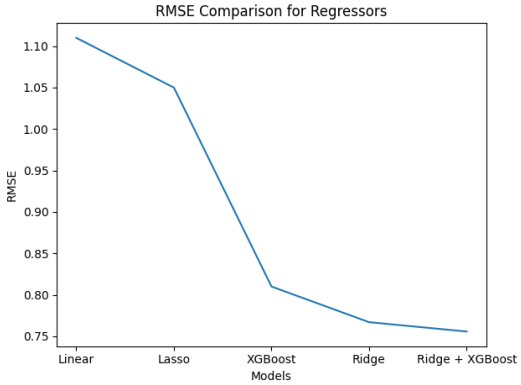
With this new direction in mind, I embarked on a comprehensive exploration of regression models, including Linear Regression, Lasso, Ridge, ElasticNet, XGBoost Regressor, and LightGBM Regressor. Each model underwent rigorous testing to assess its performance. Additionally, I experimented with ensemble techniques, using the BaggingClassifier from sklearn to combine multiple models with five estimators. The results were notable, especially for the XGBoost Regressor. However, when compared individually, Ridge Regression consistently outperformed the ensemble approach. Surprisingly, Ridge Regression consistently outperformed its counterparts in the regression task. This unexpected success can be attributed to its stability and adaptability. Ultimately, Ridge Regression emerged as my preferred choice, a decision driven by empirical evidence and the project's unique requirements. The transition from classification to regression was a significant and deliberate move that allowed me to harness the full potential of predictive modeling, delivering more accurate and nuanced results. This decision aligned my approach with Kaggle's distinctive demands and marked a key milestone in my journey to select the most suitable model.

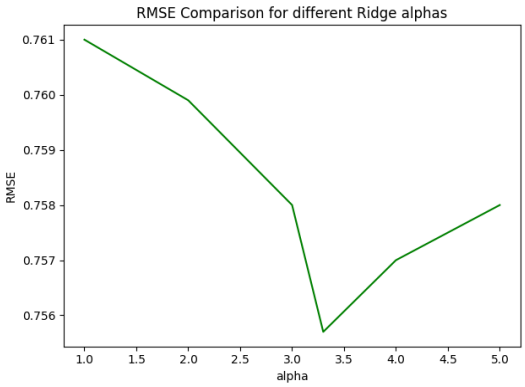
Tuning model parameters

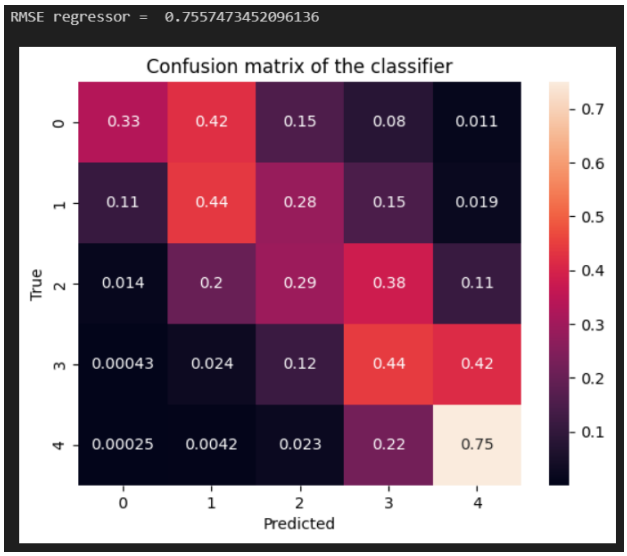
In the pursuit of optimal performance, I delved into fine-tuning the hyperparameters of the Ridge Regression model, employing the RidgeCV package from sklearn. The key objective was to determine the best alpha value, a crucial factor in Ridge Regression. This process involved a manual binary search, where I sought to identify the alpha that would yield the best results. After thorough experimentation, I found that an alpha value of 3.3 produced significantly improved results, achieving at least a 1% performance boost (RMSE). Not content with settling for a single model, I embarked on an exploration to leverage the strengths of both the Ridge Regression and XGBoost Regressor. Initially, I attempted to take an average of both their predictions but the results were bad. Then I tried stacking Ridge Regression on top of XGBoost, this approach failed to yield the desired outcomes but was still better than averaging the predictions. XGBoost, while powerful in its own right, couldn't surpass the performance of Ridge Regression when deployed solo. The turning point came when I reversed the order, using Ridge Regression as the initial model and feeding its results into the XGBoost Regressor. This innovative approach led to a remarkable improvement of 1% lower RMSE. According to me, I believe that the XGBoost Regressor adds a slight non-linearity to the outputs of the Ridge Regression - which is a linear regressor. To validate the consistency of this performance enhancement, I conducted multiple runs with varying settings and random states, and the results consistently reinforced the effectiveness of this strategy. This combination of hyperparameter tuning and model stacking proved to be a pivotal step in achieving the best possible results for this project.

Validating the Model

In the quest for robust model validation, I initially explored Stratified K-Fold Cross-Validation during the classification phase. However, as the project transitioned towards regression-based models, I leveraged Cross-Validation using the RidgeCV package. This approach allowed me to assess model performance effectively and identify the optimal alpha value. To ensure the reliability of the results, I explicitly tested various random states for train-test splits. Remarkably, all of these tests yielded similar outcomes, reinforcing the model's stability. Given the regression context, I introduced a critical step—clipping the model's outputs between 1 and 5. This adjustment was essential to align the predictions with the score range required for the Kaggle competition. Subsequently, I employed these clipped predictions to calculate the Root Mean Squared Error (RMSE), which served as a more appropriate evaluation metric for regression tasks. To facilitate a comparison with the ground truth, I rounded the model's continuous predictions to the nearest integer, effectively classifying them. This process allowed me to construct a confusion matrix, providing a valuable overview of the model's performance. The transition to regression and the meticulous validation approach played a pivotal role in ensuring the accuracy and reliability of the model's predictions as I achieved an RMSE of **0.75574**.







References

- scikit-learn: machine learning in Python — scikit-learn 1.3.2 documentation
- NLTK :: Natural Language Toolkit
- XGBoost Documentation — xgboost 2.0.1 documentation
- GloVe: Global Vectors for Word Representation (stanford.edu)