# Project Overview and Analysis Write-Up

## Data

The difference between the binary and multiclass classification tasks are related to the target variable and evaluation metrics. The target variable is changed in this task by subtracting 1 to make sure class labels start from 0 with XGBoost. The preprocessing steps stay the same, but label encoding and handling of imbalanced classes are changed for this part of the project. The XGBoost model is used, but hyperparameter tuning is used for multiclass scenarios.

## Methods

We began by one-hot encoding the category and extracting the unixReviewTime column. This is the same as what was done in the previous part of the project. We substrate 1 from the target variable (y), which aligns with the zero-based indexing expected by XGBoost for multiclass tasks. The VADER sentiment analysis was applied to both 'reviewText' and 'summary,' which finds the sentiment scores and incorporates them as additional features.

We used three models, Decision Tree, Random Forest, and XGBoost as our three models. Hyperparameter tuning was performed within the pipelines, and the models were evaluated using F1 score and accuracy metrics. XGBoost was found to be the best model and the hyperparameters were fine tuned. For the XGBoost model, we employed a ColumnTransformer to handle both text and numeric features, integrating TF-IDF vectorization for text data. The hyperparameters for XGBoost used in the previous part of the project were fine-tuned using grid search with cross-validation. We explored different values for each hyperparameter and selected the ones that gave us the highest cross-validated F1 score.

Unlike the binary classification task, we changed the evaluation metrics and incorporated class-wise F1 scores, precision, and recall to find model performance across different classes. The preprocessing steps for text data, including stopword removal and TF-IDF vectorization stayed the same as in the previous part.

## Results

| Model | ROC_AUC | Macro F1 | Accuracy |
|---|---|---|---|
| Decision Trees | 0.6616666345736525 | 0.46417897955260723 | 0.46385748544021926 |
| Random Forest | 0.8404928195636246 | 0.552640881663313 | 0.5553271668379582 |
| XGBoost | 0.8644469483506164 | 0.5715283021660719 | 0.5719424460431655 |

**Confusion Matrices:**
Decision Trees -
[[580 272 169 101  70]
 [275 483 211 138  85]
 [165 218 500 189 100]
 [ 94 154 190 504 212]
 [ 64  68 109 246 641]]

Random Forest -
[[869 203  63  33  24]
 [377 511 199  71  34]
 [153 242 553 174  50]
 [ 64 124 202 575 189]
 [ 54  56  68 216 734]]

XGBoost -
[[779 254  85  47  27]
 [286 555 229  92  30]
 [109 225 586 201  51]
 [ 39  88 183 632 212]
 [ 32  40  62 207 787]]

These metrics are appropriate for this task because they help understand the model's accuracy and effectiveness. The weighted F1 score is a suitable metric for this task. The model does provide the best performance across all metrics. The classes that are harder to predict are the middle ones (2 to 4). The distinction between these classes is very narrow, and if we continue to tailor it to the training set, we run into the risk of overfitting.

## Future Work

In the future, it would be interesting to use other datasets outside of the provided ones to explore the accuracy of our model.