SINGAPORE MANAGEMENT UNIVERSITY

CS421 Principles of Machine Learning
AY 2024/2025 Term 1

Group Project Final Report

Prepared by:
G2 - Kernel Lennel

Choo Yang Hwee	yhchoo.2023	01431875
Chua Jia Jie Lennel	lennel.chua.2023	
John Ho Weng Shing	john.ho.2022	01438098
Ng Zhen Cong Matthew	matthew.ng.2023	01448212
Shiv Iyer	shiv.iyer.2023	

1. Introduction

The aim of this project is to determine models that can best predict the anomaly level and type. Thus, there will be 2 parts to this project, part 1 to achieve the lowest possible MAE on the given unseen data, and part 2 to achieve the highest possible accuracy on the given unseen data. This was repeated over 4 weeks, where each week we dealt with an unseen data set with new data. For part 1, we focused on supervised learning, since we were given the anomaly level in our data. For part 2, we focused on semi-supervised learning, since only a small fraction of the data had the anomaly type while the rest were unlabelled. For the purposes of this report, we will mainly focus on the results of the final week, since the data set was just a compilation of the data from the first 3 weeks.

2. Exploratory Data Analysis

To better understand the different anomaly types, we extracted the reviews under the labelled user and grouped them by different anomaly types. Our key observations revealed several patterns across the three types (supporting graphs in Figure A1 & A2). Type 0 has no duplicates, Type 1 has some duplicates, and Type 2 has the most duplicates. The mean rating increases progressively from Type 0 to Type 1 to Type 2. Additionally, Type 0's skew is closest to 0, and its higher standard deviation indicates a more even rating distribution compared to the other types.

We further explored the duplicates and found several noteworthy patterns (supporting graphs in Figure A3). Type 2 only has 4's and 5's as duplicates, which explains why its mean rating is the highest among all types. Type 1's duplicate distribution follows its own natural distribution pattern. Even when duplicates are removed from consideration, the distribution between the different types varies significantly.

We also investigated the relationship between features and the anomaly level for each anomaly type. Our analysis revealed distinct correlations for each type (supporting graphs in Figure A4). Type 0's anomaly level shows a high negative correlation (-0.836) between anomaly level and average ratings, meaning that Type 0 users with lots of very low ratings usually have high anomaly scores. Type 1's anomaly level demonstrates a high positive correlation (0.876) between anomaly level and duplicate rate, indicating that Type 1 users with more duplicate ratings usually have higher anomaly scores. Similarly, Type 2's anomaly level has a high positive correlation (0.939) between anomaly level and duplicate rate, meaning that Type 2 users with more duplicate ratings usually have higher anomaly scores.

3. Methodology

3.1 Part 1 Methodology

We fixed the `seed = 42` across the entire code to ensure reproducibility. We did an initial 80:20 Train-Test split based on users to have a test set for unbiased evaluation. Next, we did feature engineering by transforming the original dataset into a dataset with its rows as users and its columns as the engineered features. Our engineered features could be categorised into 3 types of features:

1. User-based features, which rely solely on each user's own ratings
2. Global/item-based features, which rely on global users rating or item rating statistics
3. Duplicate-based features, which were created based on our findings in our exploratory data analysis. They are features that focus on duplicate ratings for each user.

All global and item statistics were calculated using the train set only, and then applied to the test set to prevent data leakage. (See Table A1 for a full list of features, See Set of Equations B1 for further mathematical explanation of selected features.)

Next, we did Cross Validation (CV) with k=5 to continue the 80:20 split. Within each CV run, we scaled on the train data and transformed the test data for models that required scaling. Ensemble Models did not require scaling. We realised that there was a small amount of data leakage within the CV, as we did not recalculate the global and item statistics within each CV run. However, since it was relatively insignificant, we left it as it is. We tried various models, doing both hyperparameter tuning and model selection based on the model with the highest mean CV MAE. We then compared the training MAE to the mean CV MAE to check for overfitting, and then looked at the test MAE for an unbiased evaluation. For hyperparameter tuning, we had to employ a mix of normal grid search with a greedy approach, meaning that we tuned one hyperparameter at a time with all others fixed. We also tuned hyperparameters that we believed were of most importance — this was due to time constraints of conducting an extensive grid search. Finally, we retrained over all the data, with scaling and calculating global and item statistics based on all the data and then applying this to the unseen data.

3.2 Part 2 Methodology

We had a similar methodology with some minor changes due to the semi-supervised nature of the task. First, we did not do a train-test split, as we wanted to have as much labelled data for training, thus we had no test set for unbiased evaluation. For CV, we used k=5 but split only on the labelled data, giving us a 80:20 train-test split on the labelled data. Within each CV run we scaled on the labelled train data and transformed both the labelled test data and unlabelled data. Therefore, hyperparameter tuning and model selection is based on mean CV Accuracy instead. For the final predictions, we retrained over all labelled and unlabelled data, scaling and calculating the global and item statistics on all labelled data and transforming both the unlabelled and unseen data.

3.3 Feature Selection

Referenced Work: [1] "A Unified Approach to Interpreting Model Predictions," S. Lundberg and S.-I. Lee

We had a total of 48 features, all of which were self-engineered. We wanted to see if we could reduce the number of features by removing relatively unimportant features to reduce dimensionality for better and faster performance. We conducted a feature correlation matrix (See Figure A5). By doing this we realised that we had some feature pairings (***zscore_product_mean, contrarian_mean, is_extreme_mean)*** that had a correlation of 1 due to accidental duplication of features during transformation. We also realised that we had 2 features (***item_max_mean, item_max_std***) that had 0 correlation with every other feature — this was because these 2 features had a constant value across all users. We then tried removing these 5 features thinking that it would lower the mean CV MAE/Accuracy or, at the very least, remain the same. However, we found that across all models, generally the mean CV MAE/Accuracy increased which seems to suggest that keeping these features would improve the performance. Since the goal was to obtain the lowest MAE/highest Accuracy, we decided to keep them in and let the regularisation functions in some of the models deal with it.

To further explore the importance of each feature, we employed a CV-leave-one-out method. This essentially takes one feature out at a time and shows the change in the mean CV MAE/Accuracy. If the MAE/Accuracy improves or remains the same, then it suggests that removing the feature is beneficial. However, for some models that are highly complex, this method would take too long. Hence, we turned to another method known as Shapley Additive Explanations (SHAP). For each sample prediction, it is decomposed into the base value and the sum of all SHAP values of each

feature, where the base value is simply the average of all sample predictions. The SHAP value for each feature shows the contribution from the base value to the predicted value for a single prediction. Thus, the greater the mean absolute SHAP value for a feature across all predictions, the greater the feature importance. For ensemble models like Random Forest and Extreme Gradient Boosting, we relied on their inbuilt feature importance functions. Finally, the decision whether to remove or keep the features was based on the mean CV MAE/Accuracy.

3.4 Weekly Predictions

Since each week we had new data made available to us for training, we had to redo the entire process including model selection and hyperparameter tuning. This is because our previous best model with its optimal hyperparameters may not be optimal with the inclusion of the new data.

4. Part 1: Predicting Anomaly Level (Regression)

4.1 Model Selection

We first started with a simple linear regression and then moved on to Ridge, Lasso, Elastic Net (combination of Ridge and Lasso) with varying polynomial degree (including 1) to implement regularisation. However, since there might be non linear patterns within the data, we explored ensemble methods which are able to capture such non linear patterns, as well as having regularisation. This includes Random Forest (RF), Adaptive Boost (Ada Boost), Extreme Gradient Boosting (XGB) and Light Gradient Boosting Machine (LGBM). Finally we also explored Dense Neural Network (DNN) which can potentially capture even more complex patterns in the data. DNNs also have greater customisability to tailor to the data such as choosing the architecture, the activation function and type of regularisation.

Based on the model results (See Table A2), we were surprised to see Ridge performing better than the ensemble models, however, we do recognise that we could have done a more rigorous grid search for hyperparameter tuning for the ensemble models. We found that DNN had the best performance of mean CV MAE 0.0510 and an unbiased test MAE of 0.0507. There was no overfitting for our DNN as our mean CV MAE was close to our training MAE of 0.0473. Underfitting was determined based on our unseen data MAE compared to other groups, where generally we performed well across all weeks (See Figure A6).

As mentioned earlier, we tried to implement feature selection using SHAP as CV leave-one-out would take too long for DNN. We removed the bottom 20 (See Figure A7) features with the lowest absolute SHAP values and reran the model and obtained a mean CV MAE of 0.0542, which suggests a drop in performance. Hence, we kept all features in for the DNN.

4.2 Dense Neural Network Model

Our DNN architecture (See Figure A8) comprised of an input layer with 48 nodes, 2 hidden layers (4 and 64 nodes respectively), each having L2 (Ridge) regularisation and ReLU activation function, and an output layer with 1 node with sigmoid activation function as the anomaly level ranges from 0 to 1. We chose 2 hidden layers as 1 hidden layer seemed to be underfitting and 3 hidden layers did not seem to improve the performance of the model.

The hyperparameters that we tuned were the number of nodes in each hidden layer, the L2 regularisation level, the learning rate and the batch size. Each CV run was done on 2000 epochs with early stopping of 300 epochs as we wanted sufficient iterations yet not taking too long. The optimal values of our hyperparameters are: Hidden Layer 1 = 4, Hidden Layer 2 = 64, Batch Size = 64, Learning Rate = 0.001, L2 Regularisation = 0.001. We did rerun the model with our optimal hyperparameters over 5000 epochs for each CV run to determine the average optimal epoch number to use for the final retraining since we can't use early stopping and we found that the optimal epoch number for each CV run varied from 800 to 5000 (See Figure A9). So we decided to stick to 2000 epochs for the final retraining.

5. Part 2: Predicting Anomaly Type (Classification)

5.1 Anomaly Level vs Anomaly Type

We wanted to examine if there was a correlation between anomaly level and anomaly type. If there was a significant correlation, we would want to use the predicted anomaly level as an additional feature for predicting anomaly type, since the predicted anomaly level would be a close enough

substitute for the actual anomaly level. From the visualisation we see that there is not really much of a significant pattern between the anomaly level and type. However, we ultimately decided to keep it in and let the regularisation and feature selection decide its importance. This meant that we had 49 features for part 2.

5.2 Model Selection

We tried several semi-supervised learning models which included Label Spreading (LS), tri-training with RF, tri-training with XGB and Expectation Maximization (EM) Naive Bayes. For LS and EM Naive Bayes, we also tried without SHAP unimportant features and without CV leave-one-out unimportant features. For the tri-training models we tried without the unimportant features based on the inbuilt function within the models.

From the results (See Table A3), we see that all models performed generally well, with EM Naive Bayes without CV leave-one-out unimportant features performing the best with mean CV Accuracy of 0.9500. The unimportant features are listed in the appendix (See Table A4). There was no overfitting as the mean CV accuracy was better than our training accuracy of 0.9333. For underfitting, we once again looked at the weekly group performance which we generally performed well apart from the second week (See Figure A6). This was because there was an error in our methodology for the first 2 weeks for part 2 where we did not have a proper way of choosing our best model and we just had a lucky choice for our first week.

5.3 Expectation Maximization Naive Bayes Model

Referenced work: [2] "Pattern Recognition and Machine Learning," C. Bishop

Referenced work: [3] "Text Classification from Labeled and Unlabeled Documents using EM Editor," K. Nigam, A. Kachites, M. Zy, S. Thrun, T. Mitchell, and W. Cohen

EM Naive Bayes consists of 2 parts, Naive Bayes which by itself is a supervised learning model and the EM algorithm which transforms the Naive Bayes into a semi-supervised learning model. Focusing on how Naive Bayes works, the model predicts the type based on the type with the highest predicted probability as shown in the equation below:

$$\hat{c} = \arg\max_{c} P(c|x)$$

And we obtain $P(c|x) = \frac{P(x|c)\cdot P(c)}{P(x)}$ using Bayes Theorem where $P(x|c)$ is the likelihood of observing features x given type c, $P(c)$ is the prior probability of type c, $P(x)$ is the evidence which is the probability of features x occurring.

The main issue is that $P(x|c)$ and by extension, $P(x)$ are generally really hard to compute. Hence, the model makes a "Naive" assumption that all features x_i are conditionally independent given the type c and thus, we can rewrite the equation for the 2 probabilities and ultimately, the initial equation as shown below:

$$P(x|c) = P(x_1, x_2, \dots, x_n|c) \approx \prod_{i=1}^{n} P(x_i|c)$$

$$P(x) = \sum_{c} P(c)P(x|c) = \sum_{c} P(c) \prod_{i=1}^{n} P(x_i|c)$$

$$\hat{c} = \arg\max_{c} \left(\frac{P(c)\prod_{i=1}^{n} P(x_i|c)}{P(x)} \right)$$

Combining the EM algorithm with the Naive Bayes Model, the model first calculates the 3 probabilities, $P(x|c)$, $P(c)$, $P(x)$ using the labelled data only. It then predicts the type for the unlabelled data using the 3 probabilities, then it recalculates and updates the value of the 3 probabilities using the labelled and confident unlabelled data (determined by the confidence threshold). It repeats this process until it reaches the convergence tolerance or the max iterations. Further mathematical explanations on the EM Naive Bayes model and the EM algorithm specifically, can be found in Appendix B (See Set of Equations B2).

The hyperparameters that we tuned were Variance Smoothing which is a form of regularisation by adding a very small constant to the variance in the Gaussian PDF for continuous features, Max Iterations and Convergence Tolerance which are both forms of early stopping, Confidence Threshold which is the level that determines whether the unlabelled data is used for the recalculation of the parameters, and Use Weights where “True” means the unlabelled data are weighted by their confidence level and “False” means the unlabelled data are weighted equally. The optimal values of our hyperparameters are: Variance Smoothing = 1e-07, Max Iterations = 50, Convergence Tolerance = 0.1, Confidence Threshold = 0.99, Use Weights = “False”.

5.4 t-distributed Stochastic Neighbour Embedding (t-SNE)

To visualise the clustering of the EM Naive Bayes Model, we used t-SNE which is a dimensionality reduction technique. Further explanation on the mathematical derivation of t-SNE can be found in Appendix B (See Set of Equations B3). From the visualisation (See Figure A10), we see that the 3 types are generally well separated, especially type 0 (represented by the purple square in the dotted red circle) where they are perfectly clustered as they are stacked perfectly on top of one another, with the exception of one type 2.

5.5 Prediction Analysis

We conducted the same ratings and duplicates analysis on our predicted anomaly types and compared it to our earlier findings in the exploratory data analysis to check if they are aligned. We found that the ratings and duplicates distribution are almost identical (See Figure A1, A2, A3, A11). Together with our high prediction accuracy, we can conclude that our duplicate-based features and our earlier findings are indeed crucial to predicting anomaly type.

6. Conclusion

6.1 Final Model Recommendation

Our final model recommendation for Part 1 is a Dense Neural Network Model with all 48 features, with hyperparameters: Hidden Layer 1 = 4, Hidden Layer 2 = 64, Batch Size = 64, Learning Rate = 0.001, L2 Regularisation = 0.001, Epochs = 2000.

Our final model recommendation for Part 2 is a EM Naive Bayes Model with the CV leave-one-out unimportant features removed (See Table A4), with hyperparameters: Max Iterations = 50, Convergence Tolerance = 0.1, Variance Smoothing = 1e-07, Confidence Threshold = 0.99, Use Weights = False.

6.2 Difference between the Anomaly Class

Anomaly Type 0 users are characterized by their relatively balanced rating distribution and lack of duplicates. This suggests natural rating behaviour from real movie watchers. **Anomaly Type 1** users are characterized by their duplicates that are most concentrated in the 3-4 range, mirroring the overall rating distribution. The duplication pattern suggests users who repeatedly rate items, possibly indicating engagement manipulation or data quality issues, while maintaining somewhat realistic rating patterns. **Anomaly Type 2** users exhibit a very high volume of duplicate ratings, almost exclusively 4's and 5's. This pattern is highly indicative of bot activity or review boosting services.

6.2 Limitations and Further Exploration

One limitation we identified was that since we had too extensive of a grid search, we relied on a greedy grid search which may not give the optimal set of hyperparameters. Due to a lack of time, we were unable to try or properly conduct a good grid search on other models that could have potentially performed better. Thus for further exploration, we would conduct a more extensive grid search on our other models.

7. References

We will be using IEEE formatting for references. Please look out for these references in 3.3 Feature Selection and 5.3 EM Naive Bayes in the report, as well as the mathematical derivations in Appendix B.

[1] S. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” 2017. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[2] C. Bishop, “Pattern Recognition and Machine Learning,” Springer, Information Science and Statistics, Feb. 2006. Available: https://tjzhifei.github.io/links/PRML.pdf

[3] K. Nigam, A. Kachites, M. Zy, S. Thrun, T. Mitchell, and W. Cohen, “Text Classification from Labeled and Unlabeled Documents using EM Editor,” Mar. 1998. Available: https://www.ri.cmu.edu/pub_files/pub1/nigam_k_1999_1/nigam_k_1999_1.pdf

[4] A. Dempster, Laird, and Rubin, “Maximum Likelihood from Incomplete Data via the EM Algorithm,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977, Available: https://www.ece.iastate.edu/~namrata/EE527_Spring08/Dempster77.pdf

[5] L. van der Maaten and G. Hinton, “Visualizing Data using t-SNE”, *Journal of Machine Learning Research,* vol. 9, pp. 2579-2605, 2008,
Available: https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

8. Appendix

APPENDIX A

Figure A1: Combined Statistics

anomtype	n_ratings	mean	std	skew	n_unique_items	dup_rate	label_y
0	308.50	2.8490	1.0874	-0.3157	308.50	0.0000	0.5504
1	273.05	3.3332	0.9606	-0.4490	256.40	0.0533	0.4842
2	303.60	3.9041	0.8788	-0.6983	260.65	0.1460	0.5045

Figure A2: Rating Distribution by Anomaly Type

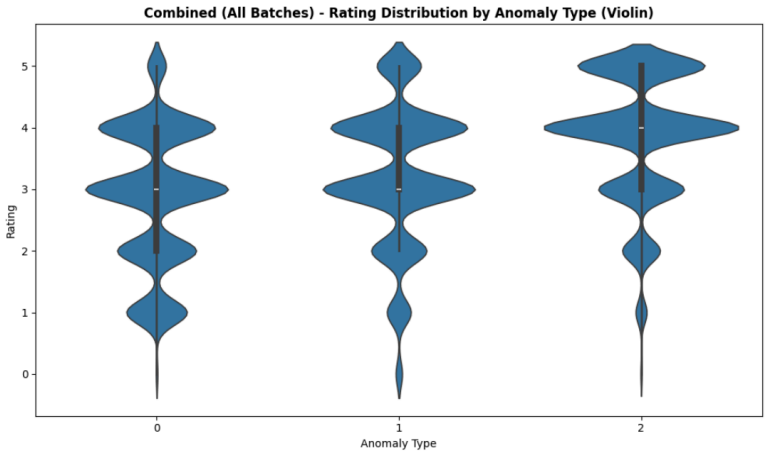


Figure A3: Duplicate Analysis

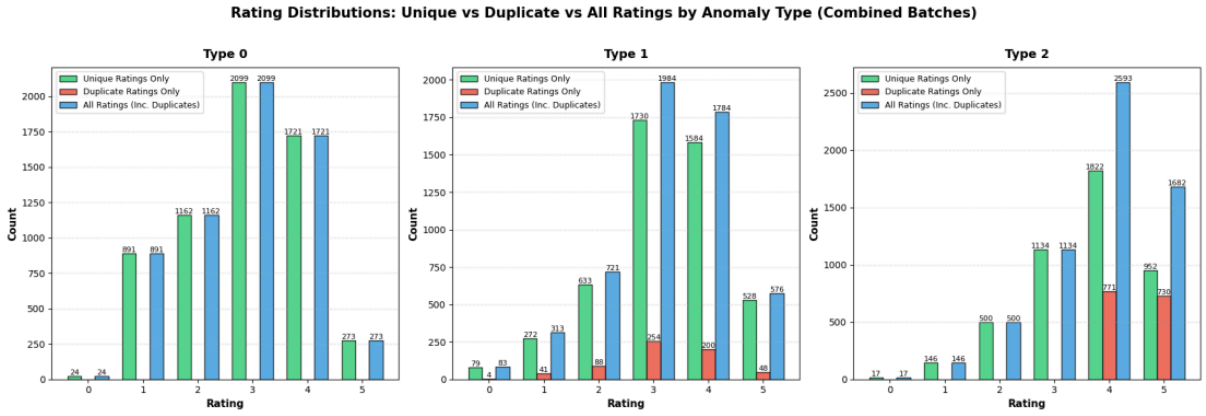


Figure A4: Anomaly Level Analysis

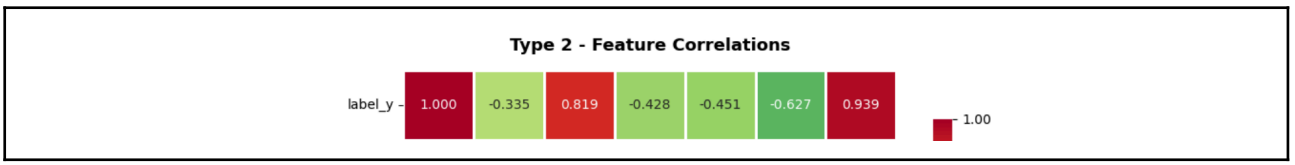


Table A1: Full Feature List

#	User-Based	#	Global/Item-Based	#	Duplicate-Based
1	user_mean_mean	16	item_mean_mean	38	duplicate_ratio
2	user_std_mean	17	item_mean_std	39	avg_duplicate_variance
3	user_skew_mean	18	item_std_mean	40	avg_duplicate_count
4	user_min_mean	19	item_std_std	41	dup_rating_mean
5	user_max_mean	20	item_min_mean	42	dup_rating_std
6	user_count_mean	21	item_min_std	43	dup_rating_mode
7	rating_mode	22	item_max_mean	44	dup_rating_range
8	rating_entropy	23	item_max_std	45	dup_unique_ratings
9	entropy_x_count	24	item_skew_mean	46	dup_pct_only_4_5
10	user_zscore_mean	25	item_skew_std	47	dup_pct_mid_rating
11	user_zscore_std	26	item_count_mean	48	dup_pct_low_rating
12	user_below_3_mean	27	item_count_std		
13	user_above_3_mean	28	item_below_3_mean		
14	extreme_rating_ratio	29	item_above_3_mean		
15	is_extreme_mean	30	item_zscore_mean		
		31	item_zscore_std		
		32	contrarian_mean		
		33	contrarian_score		
		34	global_zscore_mean		
		35	global_zscore_std		
		36	zscore_product_mean_x		
		37	zscore_product_mean_y		

Figure A5: Feature Correlation Matrix Heatmap

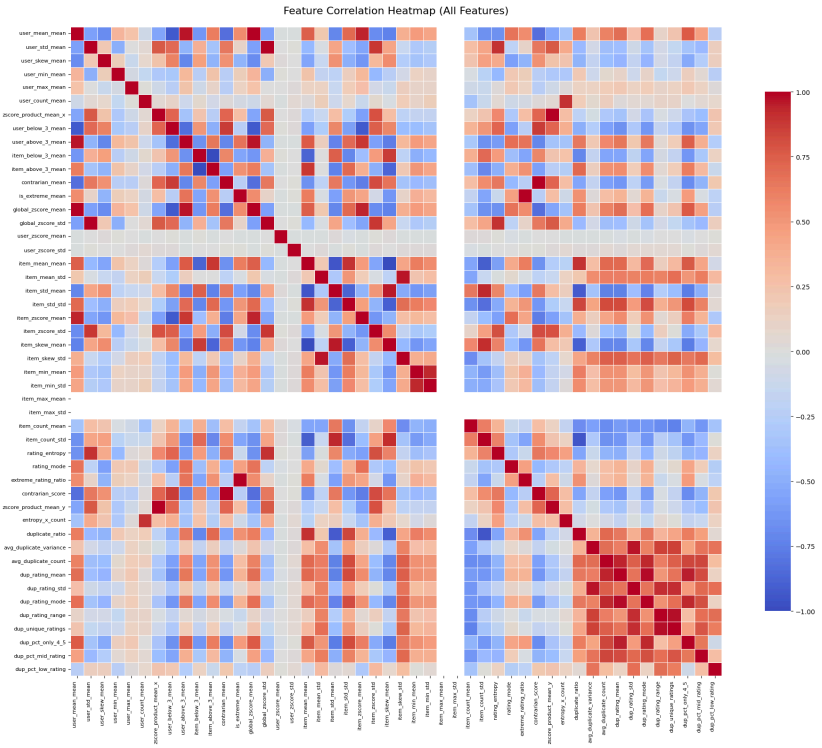


Table A2: Part 1 Results

*Row highlighted in green is our best performing model

Model	Hyperparameters	Results
Linear regression	Nil	Mean: 0.0858 Std Dev: 0.0025
Ridge	Poly Degree: 2 Alpha: 10	Mean: 0.0568 Std Dev: 0.0016
Lasso	Poly Degree: 3 Alpha: 0.001	Mean: 0.0596 Std Dev: 0.0021
Elastic Net	Poly Degree: 2 Alpha: 0.001 L1/L2 Ratio: 0.25	Mean: 0.0567 Std Dev: 0.0022
Random Forest	No. of Learners: 100 Max Depth: None Max Features: 23	Mean: 0.0596 Std Dev: 0.0014
Adaptive Boost	No. of Learners: 300 Learning Rate: 1 Loss Type: square	Mean: 0.0729 Std Dev: 0.0025
XGB	No. of Learners: 500 Learning Rate: 0.1 Max Depth: 5	Mean: 0.0598 Std Dev: 0.0012
LGBM	No. of Learners: 500 Learning Rate: 0.1 Max Depth: 5	Mean: 0.0591 Std Dev: 0.0012
Dense Neural Network	Learning Rate: 0.001 Layer 1 Units: 4 Layer 2 Units: 64 L2 regularization: 0.001	Mean: 0.0510 Std Dev: 0.0016

Figure A6: Groups' MAE and Accuracy Results

Teams	Members	Week 1				Week 2				Week 3				Week 4			
		MAE	Accuracy	Rank (MAE)	Rank (ACC)	MAE	Accuracy	Rank (MAE)	Rank (ACC)	MAE	Accuracy	Rank (MAE)	Rank (ACC)	MAE	Accuracy	Rank (MAE)	Rank (ACC)
clanker_learning	KANG TAEHO LEE SHAO DONG MICHAEL SUTEJA YONG ZHONG RUI MARCUS	0.089067	0.447778	3	8	0.073325	0.683333	3	4	0.15868	0.751111	8	3	0.056388	0.731111	2	3
dropout_rates	JADEN TOH YINGHENG KENNETH TAY VISHNU SO LETCHUMANAN YAP SENG HOW COBEN	0.178137	0.396667	9	9	0.290811	0.317778	9	9	0.120249	0.692222	7	6	0.063697	0.712222	4	3
eigenforce	DENISE LIE LOH SZE ERN CHERYL SIAH JIA LIN TEH JING WEI VARIDHI GOYAL	0.103725	0.593333	7	5	0.057203	0.708889	2	2	0.060636	0.738889	3	3	0.051078	0.74	1	3
empirical_risk_maximisers	BENEDICT LEE ZI LE DHRUV BENEGAL LAI WAN XUAN JOANNE RYAN MIGUEL MORALDE SIA	0.096351	0.686667	6	1	0.080157	0.674444	3	4	0.079451	0.683333	5	6	0.327056	0.584444	9	8.5
kernel_lennel	CHOO YANG HWEE CHUA JIA JIE LENNEL JOHN HO WENG SHING NG ZHEN CONG MATTHEW SHIV IYER	0.077636	0.675556	2	2	0.054664	0.604444	1	6	0.054139	0.793333	2	1	0.053949	0.918889	2	1
random_forest_gump	DORIS LIM LAY TENG LIM JIA EN SONIA PUA JIN CHONG IAN TAN YU XUAN XU DUO	0.090768	0.593333	4	5	0.247117	0.768889	8	1	0.051636	0.756667	1	3	0.051906	0.81	1	2
the_overfitters	CHANG JIA KAI ERIK HENRY SIRBORG SEAH KONG KHAI YAP ZU KAI	0.093899	0.656667	5	3	0.074616	0.585556	3	6	0.078986	0.704444	5	6	0.078228	0.657778	7	7.5
unclassical_logic	PEARLYN OW YANZHEN	0.074074	0.643333	1	4	0.078068	0.705556	3	2	0.064111	0.775556	3	2	0.060174	0.748889	3	3
unlearning_theory	CEMAL NISAN GAO MING MAN JUNLIANG	0.113823	0.578889	8	7	0.106236	0.398889	7	8	0.422666	0.444444	9	9	0.088055	0.526667	8	9.5
cemal_nisan														0.07621	0.731111	7	3

Figure A7: Feature importance (SHAP)

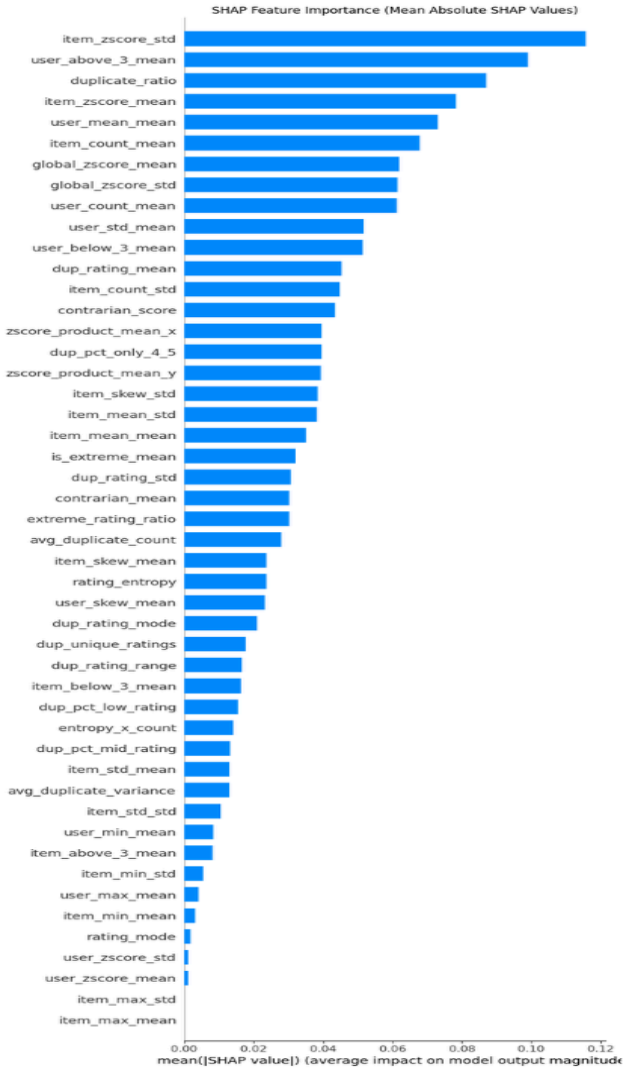


Figure A8: Neural Network Architecture

```
model = models.Sequential([
    layers.Input(shape=(48,)),
    layers.Dense(units=units1, activation='relu',
                 kernel_regularizer=regularizers.l2(l2_reg)),
    layers.Dense(units=units2, activation='relu',
                 kernel_regularizer=regularizers.l2(l2_reg)),
    layers.Dense(1, activation='sigmoid')
])
```

Figure A9: Grid Search CV

```
...  Starting Grid Search with 5 total fits (5 CV folds)...
     Restoring model weights from the end of the best epoch: 826.
     Restoring model weights from the end of the best epoch: 879.
     Restoring model weights from the end of the best epoch: 4947.
     Restoring model weights from the end of the best epoch: 4808.
     Restoring model weights from the end of the best epoch: 1727.
     Restoring model weights from the end of the best epoch: 4286.
```

Table A3: Part 2 Results

*Row highlighted in green is our best performing model

`Model	Hyperparameters	Results
Label Spreading	Kernel: rbf No. of Neighbours: NA Alpha: 0.8 Gamma: 0.1	Mean: 0.9167 Std Dev: 0.0527
Label Spreading (SHAP)	Kernel: rbf No. of Neighbours: NA Alpha: 0.1 Gamma: 0.1	Mean: 0.9333 Std Dev: 0.0624
Label Spreading (CV Leave-one-out)	Kernel: rbf No. of Neighbours: NA Alpha: 0.1 Gamma: 0.1	Mean: 0.9167 Std Dev: 0.0527
Self Training RF	No. of Learners: 300 Max Depth: 3 Max features: sqrt Max Iter: 20 Confidence Threshold: 0.95	Mean: 0.9333 Std Dev: 0.0333
Self Training RF (Inbuilt Feature Importance)	No. of Learners: 300 Max Depth: 3 Max features: sqrt Max Iter: 20 Confidence Threshold: 0.6	Mean: 0.9167 Std Dev: 0.0527
Self Training XGB	No. of Learners: 100 Max Depth: 6 Learning Rate: 0.5 Subsample: 1 Max Iter: 20 Confidence Threshold: 0.99	Mean: 0.9167 Std Dev: 0.0527
Self Training XGB (Inbuilt Feature Importance)	No. of Learners: 100 Max Depth: 6 Learning Rate: 0.1 Subsample: 1 Max Iter: 20 Confidence Threshold: 0.9	Mean: 0.9167 Std Dev: 0.0527
EM Naive Bayes	Var Smoothing: 1e-07 Max Iter: 50 Use Weights: True Convergence Tolerance: 0.00001	Mean: 0.9167 Std Dev: 0.0527
EM Naive Bayes (SHAP)	Var Smoothing: 2.64e-09 Max Iter: 50	Mean: 0.9167 Std Dev: 0.0527

`Model	Hyperparameters	Results
Label Spreading	Kernel: rbf No. of Neighbours: NA Alpha: 0.8 Gamma: 0.1	Mean: 0.9167 Std Dev: 0.0527
Label Spreading (SHAP)	Kernel: rbf No. of Neighbours: NA Alpha: 0.1 Gamma: 0.1	Mean: 0.9333 Std Dev: 0.0624
Label Spreading (CV Leave-one-out)	Kernel: rbf No. of Neighbours: NA Alpha: 0.1 Gamma: 0.1	Mean: 0.9167 Std Dev: 0.0527
	Use Weights: True Convergence Tolerance: 1e-05	
EM Naive Bayes (CV Leave-one-out)	Var Smoothing: 1e-0.7 Max Iter: 50 Use Weights: False Convergence Tolerance: 0.1 Confidence Threshold: 0.99	Mean: 0.9500 Std Dev: 0.0408

Table A4: CV leave-one-out Unimportant Feature List

#	Feature Name	#	Feature Name	#	Feature Name
1	item_max_mean	11	entropy_x_count	21	global_zscore_mean
2	user_min_mean	12	item_zscore_mean	22	dup_rating_mode
3	rating_mode	13	contrarian_score	23	item_min_mean
4	user_skew_mean	14	user_zscore_std	24	user_mean_mean
5	label	15	item_mean_std	25	user_zscore_mean
6	item_skew_std	16	user_count_mean	26	item_zscore_std
7	zscore_product_mean_y	17	avg_duplicate_count	27	user_std_mean
8	user_below_3_mean	18	item_max_std	28	global_zscore_std
9	rating_entropy	19	item_min_std		
10	user_max_mean	20	item_count_mean		

Figure A10: t-SNE Probability Distribution

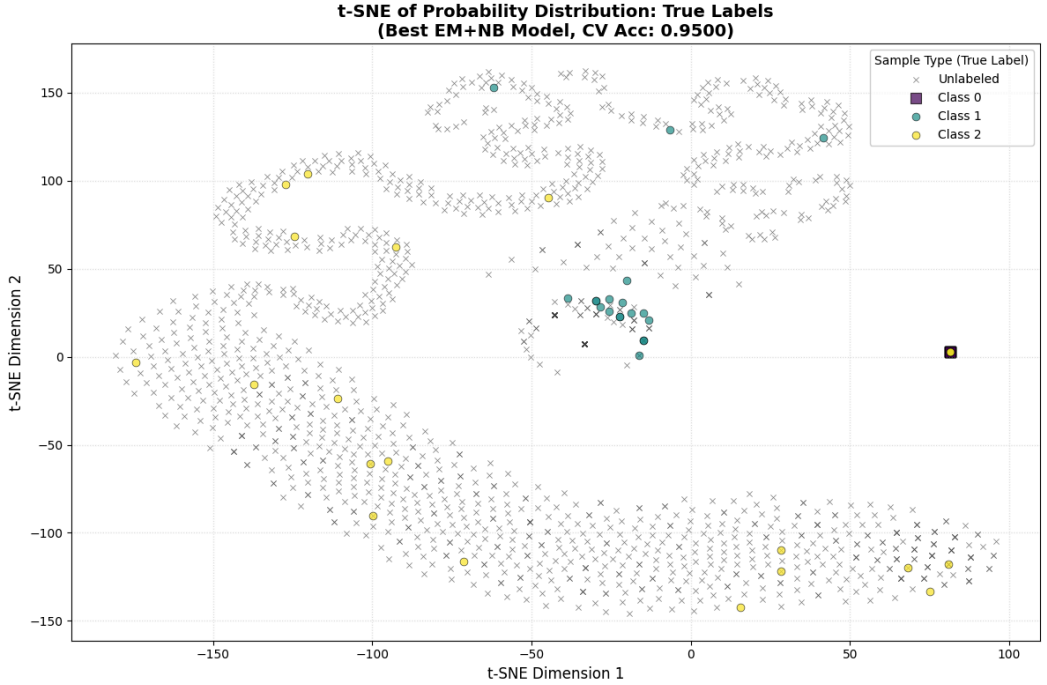
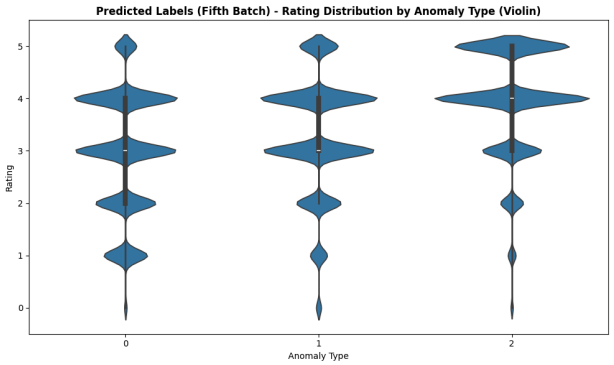
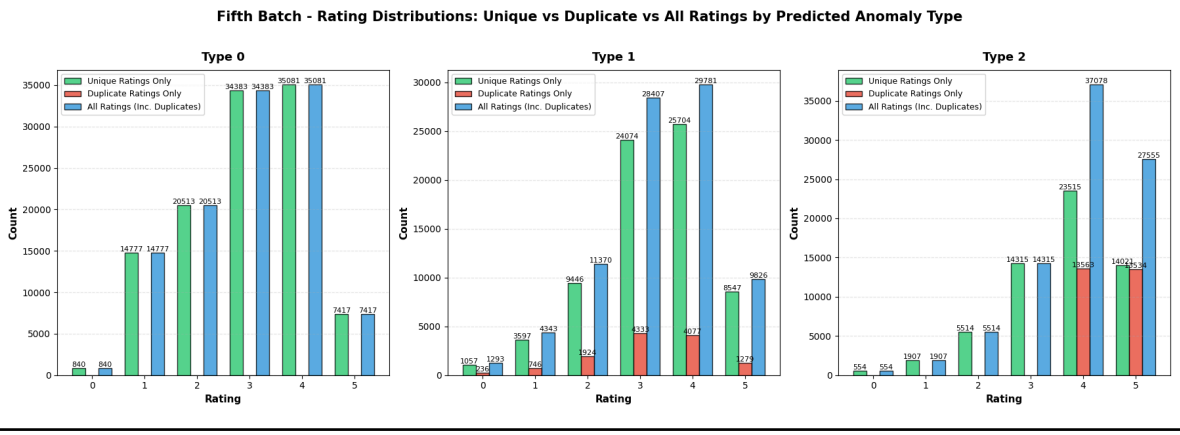


Figure A11: Fifth Batch Predicted Statistics and Graphs

anomtype	n_ratings	mean	std	skew	n_unique_items	dup_rate	label_y
0	313.0499	2.9839	1.0852	-0.3302	313.0499	0.0000	0.4481
1	313.7269	3.3344	0.9653	-0.4495	288.6716	0.0713	0.5356
2	324.3396	3.9509	0.9015	-0.7802	264.3769	0.1708	0.5497





APPENDIX B

Set of Equations B1: Mathematical Explanation of selected Features

1. rating_entropy: Shannon entropy of the user's rating distribution

   Shannon entropy = $-\sum_{i=0}^{5} p_i log_2(p_i)$ where p_i is the probability of the user rating = i

2. user_zscore_mean & user_zscore_std: mean and standard deviation of user z-score

   user z-score = $\frac{user\ rating - user\ rating\ mean}{user\ rating\ std\ dev}$

3. item_zscore_mean & item_zscore_std: mean and standard deviation of item z-score

   item z-score = $\frac{user\ rating - item\ rating\ mean}{item\ rating\ std\ dev}$

4. global_zscore_mean & global_zscore_std: mean and standard deviation of global z-score

   global z-score = $\frac{user\ rating - all\ ratings\ mean}{all\ ratings\ std\ dev}$

Set of Equations B2: Mathematical Explanation of EM Naive Bayes and EM algorithm

Referenced work: [3] "Text Classification from Labeled and Unlabeled Documents using EM Editor," K. Nigam, A. Kachites, M. Zy, S. Thrun, T. Mitchell, and W. Cohen

Referenced work: [4] "Maximum Likelihood from Incomplete Data via the EM Algorithm," A. Dempster, Laird, and Rubin

1. EM Naive Bayes (calculating $P(x_i|c)$)

   For categorical features (using MLE): $P(x_i|c) = \frac{Count(x_i, c)}{Count(c)}$

   For continuous features (using Gaussian PDF): $P(x_i|c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} exp(-\frac{(x_i-\mu_{ic})^2}{2\sigma_{ic}^2})$

   where x_i is the observed value of the feature for that data sample, μ_{ic} is the mean of feature x_i for all samples belonging to type c, σ_{ic}^2 is the variance of feature x_i for all samples belonging to type c

2. EM algorithm in general

   E-step computes the Q function, which is the expected value of complete data log-likelihood with respect to the conditional distribution of the latent variables Z given the observed data X and the current parameter estimate as shown below:

   $Q(\theta|\theta^{(t)}) = E_{Z|X,\theta^{(t)}}[logP(X, Z|\theta)]$

   If the latent variable Z is discrete then the formula changes to:

   $Q(\theta|\theta^{(t)}) = \sum_Z P(Z|X, \theta^t) logP(X, Z|\theta)$

   M-step finds the new parameter estimate that maximizes the Q function as shown below:

   $\theta^{(t+1)} = argmax_\theta\ Q(\theta|\theta^{(t)})$

Set of Equations B3: Mathematical Explanation of t-SNE

Referenced work: [5] "Visualizing Data using t-SNE", L. van der Maaten and G. Hinton

1. High Dimensionality Similarity (Gaussian Distribution):

   $p_{j|i} = \frac{\exp\left(-\frac{||x_i - x_j||^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{||x_i - x_k||^2}{2\sigma_i^2}\right)}$ where $p_{j|i}$ is the conditional probability that x_i selects x_j as its neighbour based on a Gaussian centred at x_i, $||x_i - x_j||^2$ is the squared euclidean distance and σ_i^2 is the variance of the Gaussian.

   $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$ where p_{ij} is the joint probability for the high dimensional distribution P and N is the number of data points.

2. Low Dimensionality Similarity (Cauchy Distribution):

   $q_{ij} = \frac{\left(1 + ||y_i - y_j||^2\right)^{-1}}{\sum_{k \neq l} \left(1 + ||y_k - y_l||^2\right)^{-1}}$ where q_{ij} is the joint probability for the low dimensional distribution Q, $||y_i - y_j||^2$ is the squared euclidean distance and $\left(1 + ||y_k - y_l||^2\right)^{-1}$ is the Cauchy kernel.

3. Objective Function (Kullback–Leibler Divergence):

   $Cost(Y) = KL(P||Q) = \sum_{i \neq j} p_{ij} log\left(\frac{p_{ij}}{q_{ij}}\right)$ is the cost function that t-SNE seeks to minimize using gradient descent, which would better match the low dimensional distribution Q with the high dimensional distribution P, by adjusting the low dimensional coordinates Y.

   $\frac{\delta C}{\delta y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(y_i - y_j)\left(1 + ||y_i - y_j||^2\right)^{-1}$ is the formula for the gradient.