Harnessing Helpfulness: Amazon Product Reviews to Elevate Marketplace Experience

DATA 270
Under guidance of Dr. Linsey Pang

Group 5

Presented by:

Eshita Gupta
Monica Lokare
Sneha Karri
Veena Ramesh Beknal

Model Proposals

Logistic Regression

- Logistic regression is popular in machine learning as it provides probabilities and classifies new samples using discrete measurements.
- It uses supervised learning techniques for classifying the dependent variable, which is binary. For our project, the dependent variable is helpfulness.
- As logistic regression outputs are discrete values, that is, if the variable has only two possible outcomes, in our project, it's helpful or not helpful.
- The sigmoid or logistic function gives the probability values between 0 and 1. It squashes the outcomes of a linear equation between 0 and 1.
- This function fits an S-shaped curve, which tells whether the probability of a review is helpful based on the weighted input, which is metadata and TF IDF scores of words. The y-axis on the sigmoid graph is the probability (review helpfulness).

Equations Used in Logistic Regression

Logistic function equations $\sigma(z) = \frac{1}{1 + e^{(-z)}}$

Using hypothesis of linear in logistic sigmoid functio $h_\theta(x) = \frac{1}{1 + e^{(-\theta^T x)}}$

Cost Function/ Loss FUnction of Logistic

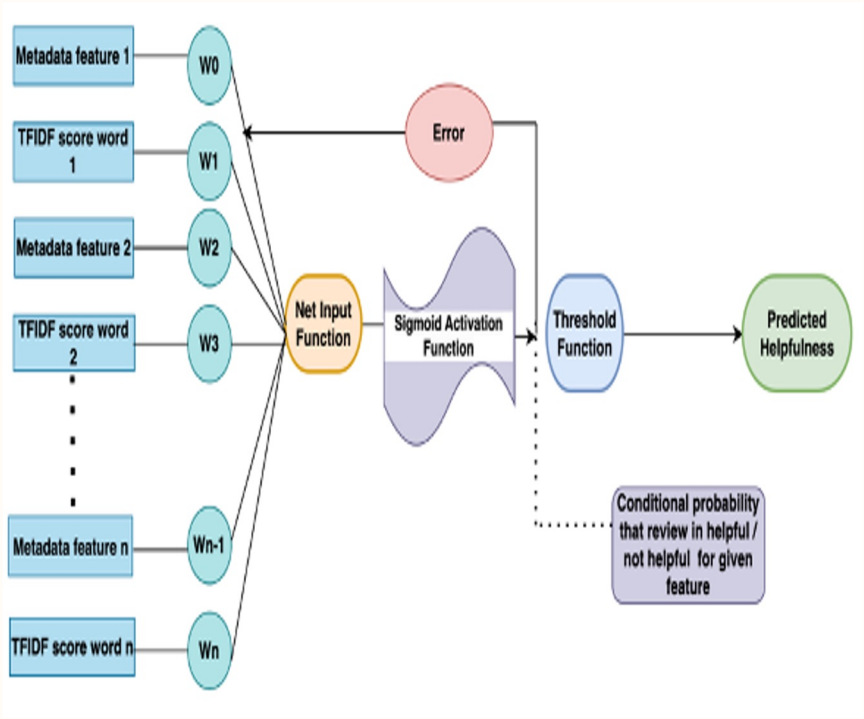
$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}[y^i \, log(h_\theta(x^{(i)})) + (1 - y^i) log(1 - h_\theta(x^{(i)}))]$$

To find optimal parameters, gradient descent is used and this will minimize the cost function of our model.

$$\theta_{new} = \theta_{old} - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

Logistic Regression Algorithm and Architecture

Input:
$x^{(i)}$: Feature vector of the i^{th} training sample
y^i : Corresponding label (0 or 1)
n: Number of records
Initialization:
Initialize parameter θ to 0 or some small value.
Define the hypothesis function $h_\theta(x)$ using the sigmoid function.
Define cost function J(θ) using the logistic loss/cost function.
Set iteration counter iter = 0
Set convergence criterion (e.g., threshold for change in cost function)
Set maximum number of iterations (optional)
Gradient Descent:
Repeat until convergence or maximum iterations reached:
For i = 1 to n:
Compute the hypothesis value for sample i: $h_\theta(x^{(i)})$
Compute the error (predicted - actual): error = $h_\theta(x^{(i)})$ - y^i
Update parameters θ using gradient descent: $\frac{\partial J(\theta)}{\partial \theta_j}$ for j = 0 to num_features
Compute the cost function J(θ) using the updated parameters.
Increment iteration counter: iter++
Output:
Return the optimized parameters θ.



XGBoost equations

- XGBoost (Extreme Gradient Boosting) is a tree-based ensemble machine learning algorithm for supervised learning (can be used for regression and classification).
- For our project, XGBoost was used for predicting helpfulness by clearly differentiating between helpful and non-helpful reviews
- XGboost relies on a technique called boosting it is a technique in ensemble modeling wherein the output of each decision tree is input into the next tree, which leads to weak learners getting stronger over multiple iterations
- The objective function of XGboost

$$Obj = \sum_{i=1}^{n} loss(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

- For binary classification, since helpfulness is binary, the logistic loss function can be used

$$Logistic\ Loss\ (Binary\ classification): loss(y_i, \hat{y}_i) = -[y_i\ log(\hat{y}_i) + (1 - y_i)\ log\ (1 - \hat{y}_i)]$$

XGboost equations (contd)

Regularization is represented as

$$\Omega(f_k) = \gamma T + \frac{1}{2} \lambda \|w\| \frac{2}{2}$$

Where gamma is the regularization parameter for controlling the number of leaves in the decision trees,

T is the number of leaves in tree k, lambda is the L2 regularization parameter and $\|w\| \frac{2}{2}$ is the L2 norm of the weights in the tree

For the actual decision tree ensemble construction, there are 3 key steps:

Gradient calculation, Tree construction, and Ensembling and updating learners

$$g_i = \frac{\partial L\,(y_i, \hat{y}_i)}{\partial \hat{y}_i{}^2}$$

$$h_i = \frac{\partial^2 L\,(y_i, \hat{y}_i)}{\partial \hat{y}_i{}^2}$$

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$$

XGboost algorithm

The algorithm of the XGBoost algorithm for predicting review helpfulness

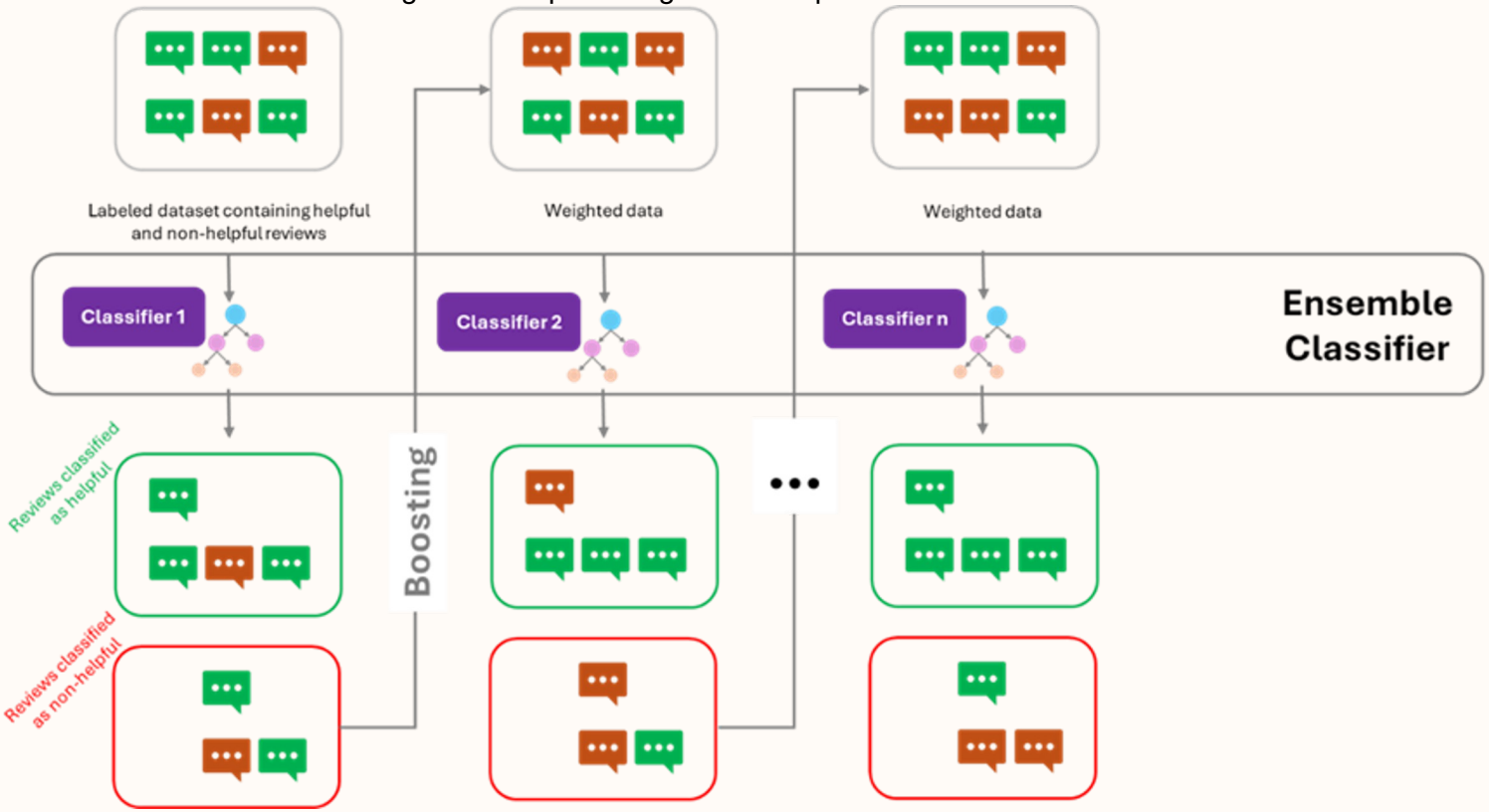
Input:
- Training data (X_train, y_train)
- Hyperparameters (e.g., learning_rate, max_depth, n_estimators)
- Loss function (e.g., logistic loss, squared loss)

Algorithm:
1. Initialize an ensemble model as an empty list of trees.
2. Initialize predictions for training data as an array of zeros
3. For each iteration (t = 1 to n_estimators):
   a. Compute the gradient of the loss function for the predictions (gradient):
      gradient = ∂L(y_true, y_pred) / ∂y_pred
   b. Compute the Hessian of the loss function for the predictions (hessian):
      hessian = ∂^2L(y_true, y_pred) / ∂y_pred^2
   c. Fit a decision tree to the negative gradient and Hessian:
      tree_t = fit_tree(-gradient, hessian)
   d. Update predictions by adding the predictions from the current tree multiplied by a learning rate:
      predictions += learning_rate × predict(tree_t, X_train)
   e. Add the fitted tree to the ensemble model.
      ensemble_model.append(tree_t)
4. Output the ensemble model.

Prediction:
- To make predictions for new data:
  a. Initialize prediction as an array of zeros
  b. For each tree in the ensemble model:
     i. Compute the prediction from the tree:
        prediction_t = predict(tree, X_new)
     ii. Add the prediction to the overall prediction
        prediction += prediction_t
  c. Output the final prediction

XGboost architecture

The architecture of the XGBoost algorithm for predicting review helpfulness



Random Forest

- Compared to a single decision tree, the primary goal of the Random Forest is to increase prediction accuracy while preserving robustness and lowering the risk of overfitting.
- In order to do this, it builds a large number of decision trees during training and outputs the class that represents the majority vote of the classes that each tree predicted.
- By successfully capturing a variety of patterns and correlations in the data, this ensemble method makes use of the strengths of several learners to achieve higher performance on complicated datasets.This method can be used for both regression and classification.
- The metrics like entropy and gini impurity are commonly employed in classification to assess the split quality. With the goal of reducing impurity across the network, these metrics calculate the impurity inside each group.
- For more precise classification, a more homogeneous grouping is indicated by lower impurity levels. In classification, recall, accuracy, and precision are additional metrics that are employed.

Equations used in Random Forest

Gini impurity - It is used to measure the impurity of the dataset. The formula to calculate the Gini impurity is given below.

$$Gini(p) = 1 - \sum_{i-1}^{n} {p_i}^2$$

where pi is the proportion of samples belonging to class i. The objective is to minimize the Gini impurity across all branches of the decision tree.

Entropy - It is a measure of uncertainty or randomness in a dataset. The formula used to calculate the entropy is shown below .

$$E(S) = -\sum_{i=1}^{n} p_i log_2(p_i)$$

where pi represents the proportion of data points in class i within the set S. The summation goes over all classes in the dataset.

Contd.

Accuracy - It is a metric used to measure the proportion of correct predictions made by a model out of the total predictions made.The formula used to calculate the accuracy is shown below.

$$Accuracy = \frac{Number\ of\ predictions}{Total\ number\ of\ predictions}$$

Precision - It measures the proportion of true positive predictions among all positive predictions made by the model.The formula to calculate precision is shown below.

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

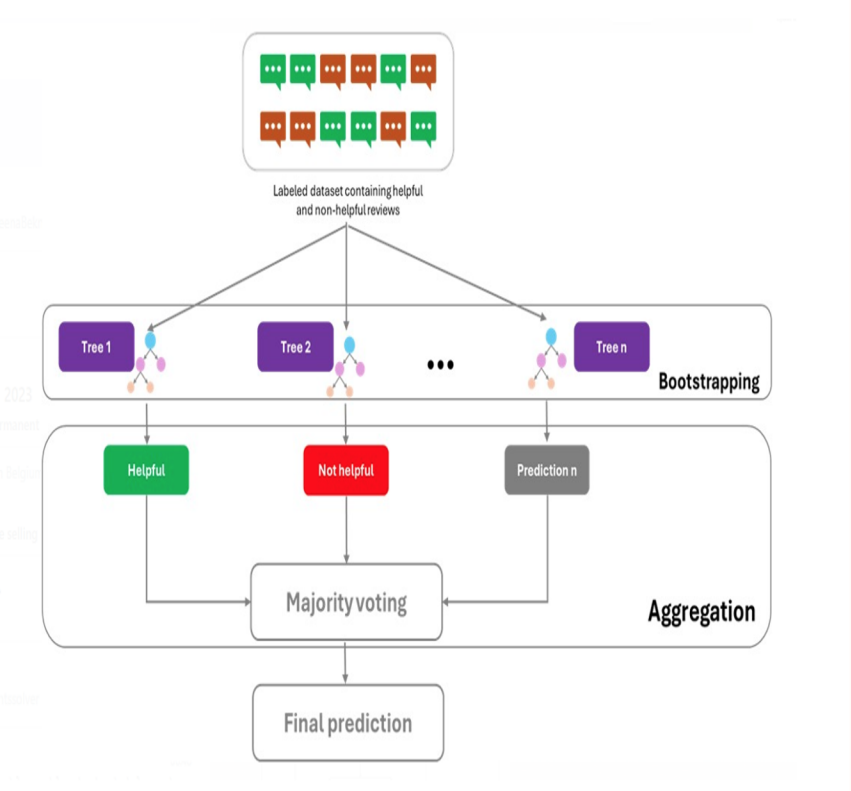
where, TP is the number of true positives and FP is the number of false positives.

Recall - It measures the proportion of actual positives that are correctly identified by the model, representing the model's ability to find all relevant cases within a dataset.The formula to calculate recall is shown below.

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

Random Forest Algorithm and Architecture

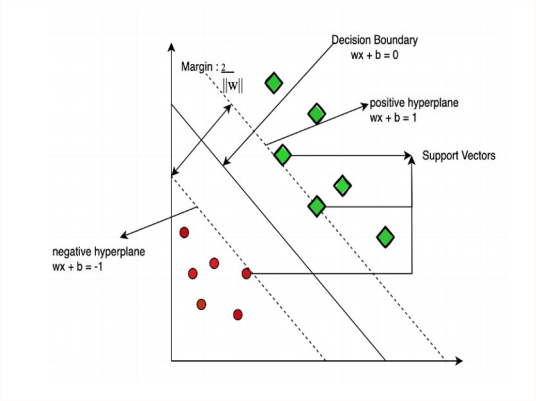
1. Create an empty forest.
In the case of i = 1 to n_estimators:
   2.1 Create a training data bootstrap sample.
   2. Start a decision tree from scratch.
   2.3 For every decision tree node:
      2.3.1 Choose max_features features at random from all of the features.
      2.3.2 Using metrics like entropy or Gini impurity, determine the optimal split based on the chosen features.
      2.3.3 Divide the parent node into its two offspring.
      2.3.4 Continue until max_depth is reached or min_samples_split or min_samples_leaf stops more splitting.
   2.4 Include the built tree in the woodland.
3. To forecast the outcome for a fresh sample:
   3.1 Assign a prediction to every tree in the forest.
   3.2 From all of the trees' predictions, choose the final prediction by majority voting (for classification).
4. Return to the Random Forest model of the forest.



Support Vector Machine

- The most sophisticated, widely used, and effective technique for resolving classification issues is support vector machines (SVM). Face recognition, spam filtering, image classification, and stock price prediction are a few applications of SVM.
- While they can also be used to solve regression problems, SVM is mostly used to solve classification tasks.
- Support Vector Machines (SVM) use a maximum margin technique to identify the hyperplane that best divides data into distinct classes.
- Supporting vectors determine the position of this hyperplane, which acts as the decision boundary and guarantees the best possible class separation.
- This approach maximizes the distance between the closest data points for each class, making it very successful for tasks like classifying tweets as true or false.

Support Vector Machine Architecture and Equations



- The goal is to find a hyperplane that maximizes the margin, which is the distance between the hyperplane and the closest point on either side of each class.
- This is done under the constraint that all data points lie on the correct side of the hyperplane. The distance between the hyperplane is called the margin and the middle hyperplane is the max-margin classifier.
- The kernel trick in SVM allows handling non-linear data by mapping it into a higher-dimensional space where it becomes linearly separable using kernel functions like RBF or polynomial kernels. This transformation lets SVM find an optimal hyperplane, enabling it to model non-linear decision boundaries effectively.

The equations for hyperplanes are shown below

Hyperplane	$w^T.x_i + b = 0$
Negative Hyperplane	$w^T.x_i + b = -1$
Positive Hyperplane.	$w^T.x_i + b = 1$

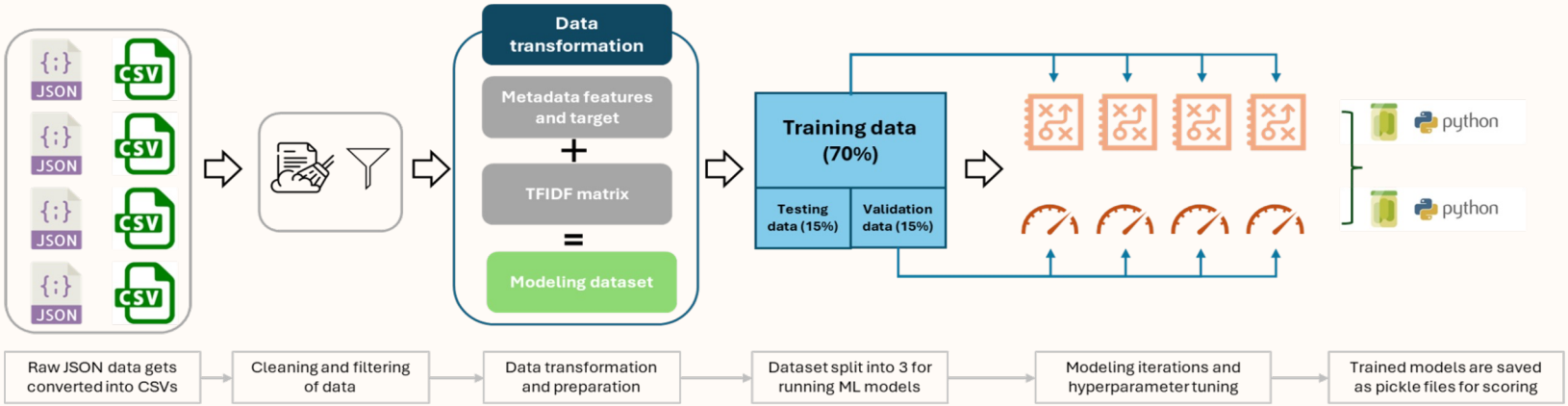
Model Supports

Environment, Platform and Tools

- We have used the Alienware AMD Ryzen 9 7950X 16-core processor, 4501 MHz high performance computing (HPC) machines at SJSU campus, Google Collab Pro, and local machines for developing and running machine learning algorithms.
- Due to the convenience, data cleaning and transformation were performed on local machines initially. When we finished the preprocessing, the entire code, including modeling was transferred to the HPC for faster execution, specifically for the model training and hyperparameter tuning process.
- For our entire project, we have used Python as our main programming language. All the coding was done on Jupyter Notebooks because of its interactive and collaborative development process. Some of the preprocessing was done in Collab Pro, and the model was executed in the HPC lab as we needed computational power.
- Some of the libraries used are pandas, numpy, Matplotlib.pyplot, random, plotly, Seaborn, Counter, Nltk, Contractions, sklearn.feature_extraction.text.TfidfVectorizer, TextBlob, Emoji, sklearn.model_selection.train_test_split, XGBClassifier, sklearn.metrics.classification_report, sklearn.metrics.accuracy_score. etc.

System Architecture

The first step is to convert the raw JSON to CSV, which simplifies the data handling and processing. In the next step, we perform data cleaning and filtering to remove duplicates and irrelevant data. During transformation, we create metadata features and target variables (review helpfulness) for stratified samples.Finally, metadata features and TFIDF matrices are used for modeling. Next, the dataset is split into three parts: train, test, and validate, which are 70%, 15%, and 15%, respectively. We have used Logistic Regression, SVM, Random Forest, and XG Boost machine learning algorithms. Multiple iterations of model training will occur on training data, and adjustments are made to the model's hyperparameters using the validation dataset, which aims to find an optimal result with the best model performance. We test these models on test data and Once models are trained and tuned, they are saved as pickle files.



Model Comparison and Justification

Model Comparison and Justification

Our project aimed to train machine learning models to classify and predict the helpfulness of online product reviews on Amazon across various product categories. The models considered text processing techniques and features extracted from review text, vocabulary richness, and metadata to drive the review helpfulness. As part of data transformation, we have implemented TF-IDF techniques to extract features from our cleaned and pre-processed review datasets.

These computed TF-IDF feature vectors were used as input to all the machine learning algorithms outlined. **For our use case, we explored these following Machine Learning models.** We have mentioned the strengths and weaknesses of each model based on factors like interpretability, scalability, training time, prediction time, handling of missing data, robustness to outliers etc, relevant to our use case

- **Logistic Regression**
- **Support Vector Machine (SVM)**
- **Random Forest**
- **Extreme Gradient Boosting (XGBoost)**

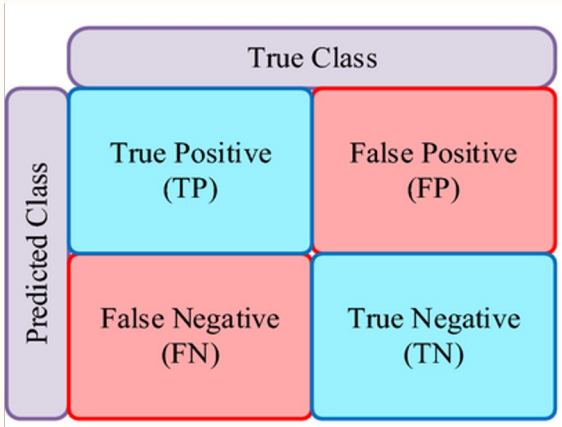
Model Comparison and Justification

Model	Strengths	Weakness
Logistic Regression	·Simple and relatively interpretable ·Beneficial for small datasets ·It provides the probabilistic output, can be helpful for scoring the review helpfulness ·Model coefficients can provide insights into feature importance	·Assumes the linear relationship which may not hold true for textual data ·Extensive feature engineering requirement ·Limited performance to complex problems
Support Vector Machine	·Can model non-linear decision boundaries using Kernel trick ·Effective for Binary classification ·Ability to handle imbalanced dataset ·Robustness to overfitting, even with high dimensionality data	·Increase the computational complexity and training time ·Less interpretable than logistic regression ·Constraints on memory requirements as data size grows
Random Forest	·Ensemble Learning can lead to better predictive performance ·Reduced overfitting ·Resistance to noise and outliers ·Parallelization and scalability ·Can handle class imbalance effectively	·Computational complexity ·Bias towards correlated features ·Sensitive to noisy features
XGBoost (Extreme Gradient Boosting)	·Includes built in regularization techniques, which help prevent overfitting ·Supports Parallel Processing ·Memory efficient ·Provides a measure of feature importance	·Sensitive to hyperparameters ·Computational complexity ·Potential overfitting if not properly tuned

Model Evaluation Methods

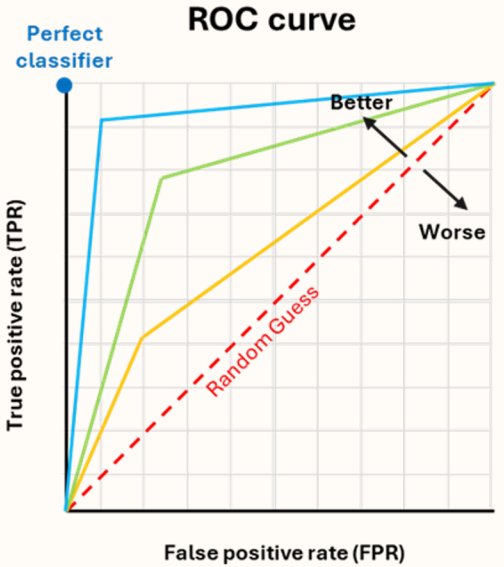
Model evaluation - overview

- We have 5 evaluation metrics for classification: Accuracy, Precision, Recall, F1-score and ROC-AUC
- The first 4 evaluation methods are derived directly from the confusion matrix
- ROC (Receiver Operating Characteristic) curve is a graph of TPR vs FPR
    - Additionally, from the ROC curve, we can calculate the Area Under the Curve (AUC), which measures the overall performance of predicting helpfulness of reviews



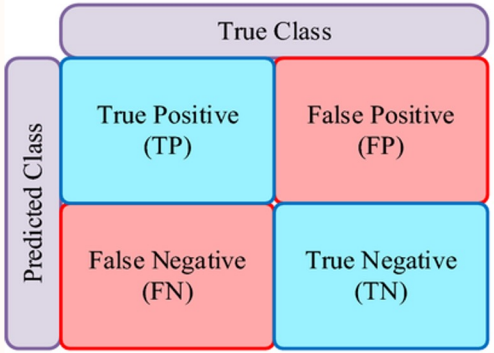
Function Name	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 score	$\frac{2.Precision.Recall}{Precision+Recall}$

Image source: https://www.researchgate.net/figure/Confusion-matrix-and-performance-metrics-formula_fig3_370070277



Model evaluation - Accuracy

- Accuracy is the overall percentage of reviews that are classified correctly
- In our project, accuracy of predicting the overall helpfulness gives us a sense of the overall correctness of a model's predictions
- While it doesn't give us the complete picture of predictions and may be misleading, especially since our data is imbalanced, it is a simple metric that can help us judge the overall performance of a model and can help in comparing against other models.

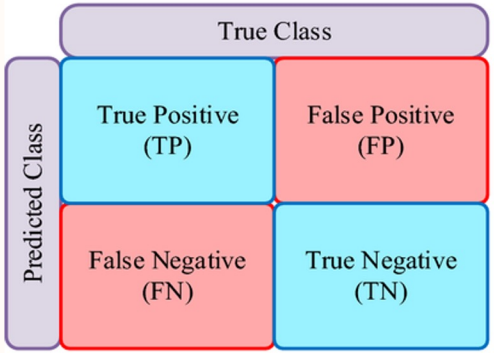


Function Name	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 score	$\frac{2.Precision.Recall}{Precision+Recall}$

Image source: https://www.researchgate.net/figure/Confusion-matrix-and-performance-metrics-formula_fig3_370070277

Model evaluation - Precision

- Precision gives us the proportion of correctly predicted helpful reviews (true positives or TP) among all reviews predicted as helpful
- In our project, precision is a measure of predicting helpfulness correctly
- It gives us an understanding of the proportion of predicted helpful reviews that are actually helpful
- A model that is highly precise for helpfulness prediction minimizes the number of false positive predictions
- To simplify, a highly precise model accurately identifies helpful reviews and avoids misclassifying not helpful as being helpful

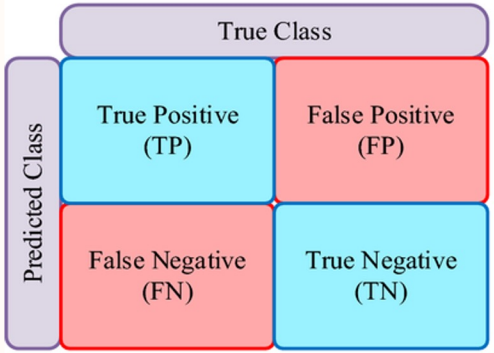


Function Name	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 score	$\frac{2.Precision.Recall}{Precision+Recall}$

Image source: https://www.researchgate.net/figure/Confusion-matrix-and-performance-metrics-formula_fig3_370070277

Model evaluation - Recall

- Recall measures the proportion of correctly predicted positive instances (true positives) among all actual positive instances
- Recall is also called Sensitivity and True Positive Rate; it gives us the proportion of correctly predicted helpful reviews (true positives or TP) among all reviews that are actually helpful
- In our project, recall is a measure of how many reviews that are actually helpful were identified correctly by the model. It gives us an understanding of whether the model is capturing a large proportion of reviews that are actually helpful

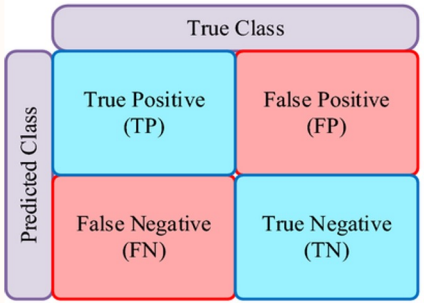


Function Name	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 score	$\frac{2.Precision.Recall}{Precision+Recall}$

Image source: https://www.researchgate.net/figure/Confusion-matrix-and-performance-metrics-formula_fig3_370070277

Model evaluation - F1 score

- F1-score is calculated as the harmonic mean of precision and recall and helps in striking a balance between precision and recall, effectively addressing the trade-off between the 2 metrics
- In our project, the F1-score provides a measure of the model's performance by giving us a holistic measure of a model's ability to classify helpful reviews correctly
- Since it is a harmonic mean, it gives equal weightage to precision and recall
- The F1 score is useful in our project since our data is imbalanced, where the share of helpful reviews is significantly smaller than the non-helpful reviews
- The F1 score is also an objective metric that helps in comparing the performance of different classification models built for predicting helpfulness

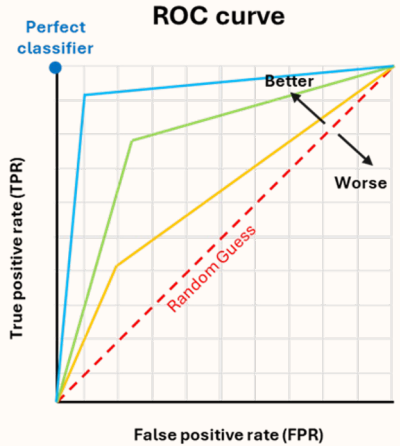


Function Name	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 score	$\frac{2.Precision.Recall}{Precision+Recall}$

Image source: https://www.researchgate.net/figure/Confusion-matrix-and-performance-metrics-formula_fig3_370070277

Model evaluation - ROC curve

- The Receiver Operating Characteristic (ROC) curve plots the True Positive Rate (TPR, also referred to as sensitivity) against the False Positive Rate (FPR, also calculated as 1 - specificity)
- There are two types of ROC curves in the context of predicting helpfulness: using binary outcomes and using probabilities
    - In our project, we are using the ROC curve with binary outcomes for its simplicity
- ROC curves can be plotted for different models for comparing the discriminative power of predicting helpfulness
- Additionally, from the ROC curve, we can calculate the Area Under the Curve (AUC), which measures the overall performance of predicting helpfulness of reviews
    - A higher AUC value indicates better differentiation between helpful and not helpful reviews, with an AUC being one is a perfect classifier that can fully differentiate between helpful and not helpful reviews
    - An AUC of 0.5 means that our model is as good as randomly guessing whether a review is helpful or not



Model Validation and Evaluation

Handling Imbalanced data: Class Weights

With Amazon Review dataset we have imbalanced data that is having the majority class as “Not Helpful” and “Helpful” reviews is the minority class. With such imbalance, all the models tend to be biased towards the majority class increasing the accuracy of the models and achieving a high precision. However, they are not able to identify any “True Positives” which is our primary use case here.

Handling Class weights is beneficial from multiple aspects.

- **Reduces Bias** - Without class weights, the models get biased towards majority classes as there are more instances to learn from.
- **Improves accuracy on Minority class** - By assigning higher weight to minority class, the models pay attention to minority class which helps to improve the model’s performance.
- **Reduces Overfitting** - Models are forced to generalize better and not overfit just on the majority class
- **Improves overall performance** - Models are forced to learn from a more balanced representation of each class.

Models are penalized for misclassifying minority class instances when using class weights which help to increase the sensitivity (true positive) of minority class. So we decided to apply class weights to one category “Tools & Home Improvement” to see if it adds any value to our overall modelling.

Model Validation and Evaluation - SVM

Baseline Model

We trained our models with default parameters to establish a baseline across all models. **The baseline models were trained using SVC class for SVM classification with parameters "kernel = rbf" across three review categories (Appliances, Automotive and Cellphone & Accessories)**. **Linear SVC class was used for SVM classification for the category 'Tools & Home Improvement" by considering class weights** to manage imbalance in data.

Models trained with SVC achieved a high accuracy and precision, however, with AUC of 0.5 and 0 as Recall, they were not able to identify any true positives. With Linear SVC and Class weights, the model was able to accurately identify ~70% of true positives with AUC and Recall both about 0.7, however, due to low precision, the number of false positives was also high. **We could not strike a balance, however, with class weights the performance was slightly better.**

Categories	Accuracy	AUC	Recall	Precision	F-1 score
Appliances	0.918656	0.5	0.00	1.0	0.0
Automotive	0.895600	0.5	0.00	1.0	0.0
Cell Phones and Accessories	0.937475	0.5	0.00	1.0	0.0
Tools and Home Improvement	0.738765	0.731792	0.722738	0.265898	0.388768

Interpretation of ROC-AUC Values

0.9-1.0: Excellent performance
0.8-0.9: Good performance
0.7-0.8: Fair performance
0.6-0.7: Poor performance
0.5-0.6: Very poor performance
<0.5: Failure

Model Validation and Evaluation - SVM

Hyperparameter Tuning

As the baseline models were not performing, we decided to use hyper parameter tuning to achieve the best parameters for modelling. A grid search was performed on the validation set for all categories to find the best parameters for the model training. The parameters considered for search with the SVC class were **“kernel = linear, rbf , poly”.** For Linear SVC the following parameters were considered for tuning “**C (Regularisation parameter) = 10, 100, 500, 1000, 10000”, “max_iter = 5000, 10000”, “loss = squared_hinge”, “penalty = l2 (Ridge), l1 (Lasso)”, and “dual = False”.**

Category	SVM Class Used	Parameters for Tuning	Best parameters
Appliances	SVC	kernel = linear, rbf, poly	kernel = rbf
Automotive	SVC	kernel = linear, rbf, poly	kernel = rbf
Cell Phones and Accessories	SVC	kernel = linear, rbf, poly	kernel = rbf
Tools and Home Improvement	Linear SVC	C = 10, 100, 500, 1000, 10000 max_iter = 5000, 10000 loss = squared_hinge penalty = l2 (Ridge), l1 (Lasso) dual = False	C= 10 penalty = l2 max_iter = 5000

Model Validation and Evaluation - SVM

Final Model & Testing

We noticed that with **SVC classifier for the categories "Appliances, Automotive and Cellphones & Accessories" the tuned models returned the exact same metrics as the baseline models**. This was primarily since the tuned parameter was same as the baseline model which was "kernel=rbf". For "Tools & Home Improvement" category **we saw a minor reduction in performance as compared to the baseline model in terms of AUC (0.730 – tuned vs 0.731 - baseline)** which indicates that the tuning of parameters did not make a significant impact the overall outcome. S**VM being highly sensitive to hyperparameters did not perform well with either of the approaches.** However, using class weights we were able to achieve slightly better results.

Categories	Accuracy	AUC	Recall	Precision	F-1 score
Appliances	0.918656	0.5	0.00	1.0	0.0
Automotive	0.895600	0.5	0.00	1.0	0.0
Cell Phones and Accessories	0.937475	0.5	0.00	1.0	0.0
Tools and Home Improvement	0.738899	0.730353	0.719258	0.265411	0.387742

Interpretation of ROC-AUC Values

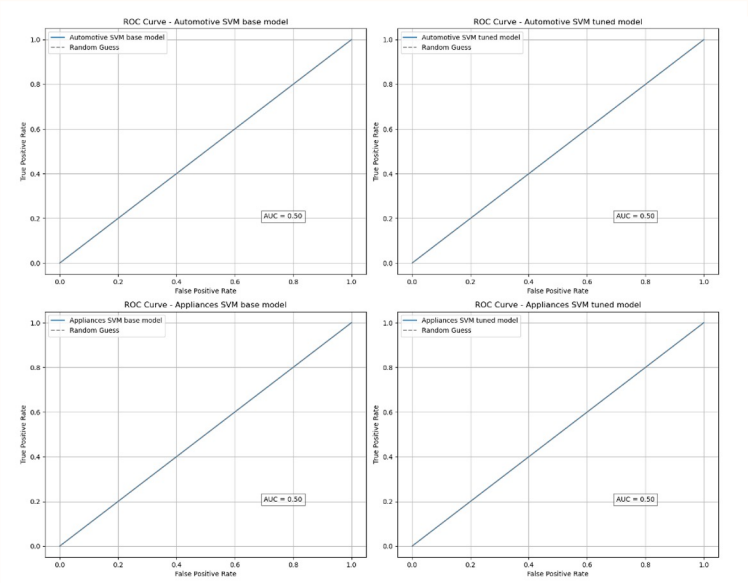
0.9-1.0: Excellent performance
0.8-0.9: Good performance
0.7-0.8: Fair performance
0.6-0.7: Poor performance
0.5-0.6: Very poor performance
<0.5: Failure

Model Validation and Evaluation - SVM

ROC-AUC Curves

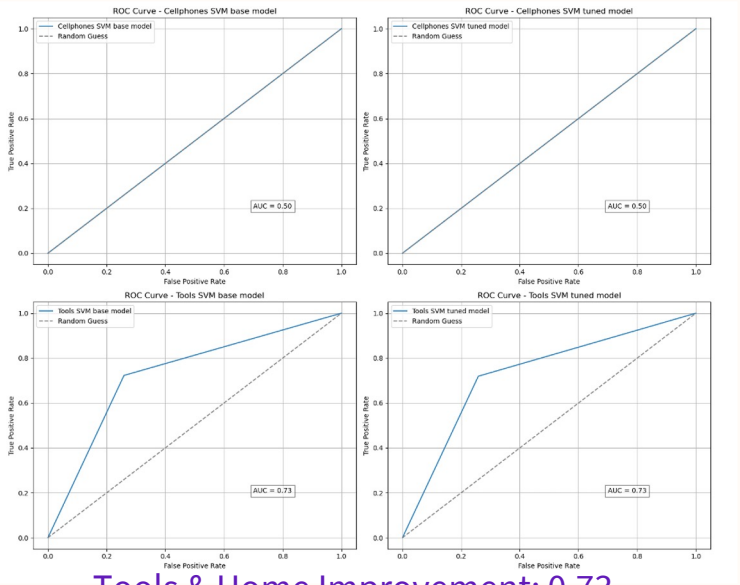
We can observe with the below curves that without class weights, SVM is not able to identify any true positive instances and instead is just random guessing. With Class weights, its able to identify ~70% of the positive instances, however, due to low precision noise is also added. To conclude, SVM perform fairly with class weights, however, it is not able to find a balance as the accuracy and precision are low.

Automotive: 0.5



Appliances: 0.5

Cellphones & Accessories: 0.5



Tools & Home Improvement: 0.73

Logistic Regression Baseline Model

Logistic regression baseline models for four different categories were trained and tested with 70,15,15 split for train, test and validation. The output includes accuracy, precision, recall, f1-score and classification outcomes labeled as not helpful (0) and helpful (1). For appliances the accuracy is 91.69%, automotive 89.95%, cellphones and accessories is 93.07%. For the category ‘Tools & Home Improvement” we considered class weights to manage imbalance in data. Hence, for tools and home improvement, accuracy is 74.5%, however, the recall and AUC improved significantly.

Logistic Regression Hyperparameter Tuning

Category	Inverse Regularization C	Penalty	Best Parameter
Appliances	[0.01, 0.1, 1, 10, 20]	[l1 and l2]	[C': 0.01, 'penalty': 'l1']
Automotive	[0.01, 0.1, 1, 10, 20]	[l1 and l2]	['C': 0.1, 'penalty': 'l2']
Cellphones and Accessories	[0.01, 0.1, 1, 10, 20]	[l1 and l2]	['C': 0.1, 'penalty': 'l2']
Tools and Home Improvement	[0.001, 0.01, 0.1, 1, 10]	[l1 and l2]	['C': 0.01, 'penalty': 'l1'}]

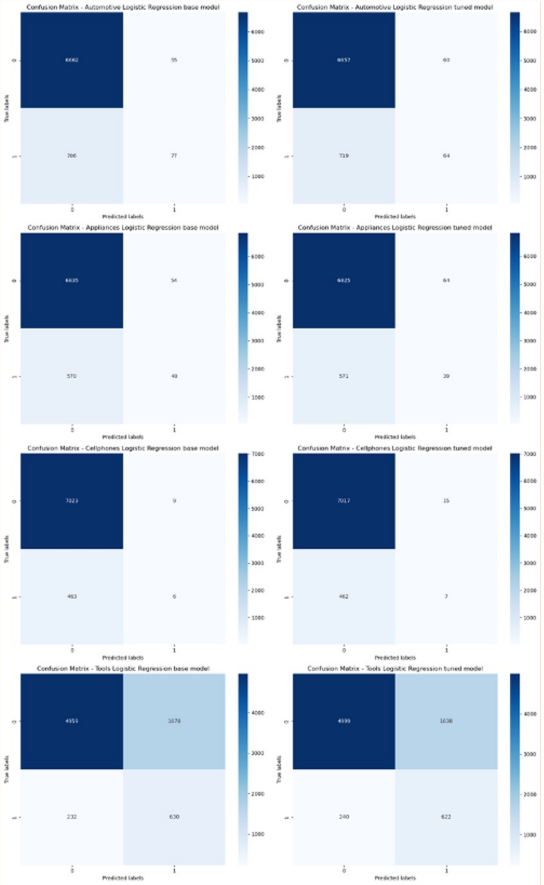
- Due to a low performance in baseline models, we have performed hyperparameter tuning using grid search which performs a search over all possible combinations of the hyperparameter ranges to find the optimal combination of hyperparameters
- For logistic regression models, we have tuned across two hyperparameters, C and penalty
- C is the inverse of regularization strength which means lower values indicate stronger regularization and Penalty is penalization.
- For review helpfulness prediction, we have used C values 0.001, 0.01, 0.1, 1, 10, 20, and for penalty, we have used L1(Lasso Regression) and L2 (Ridge Regression) regularization
- A lower C value indicates that stronger regularization helped a little in managing the overfitting but not enough to improve helpfulness prediction metrics

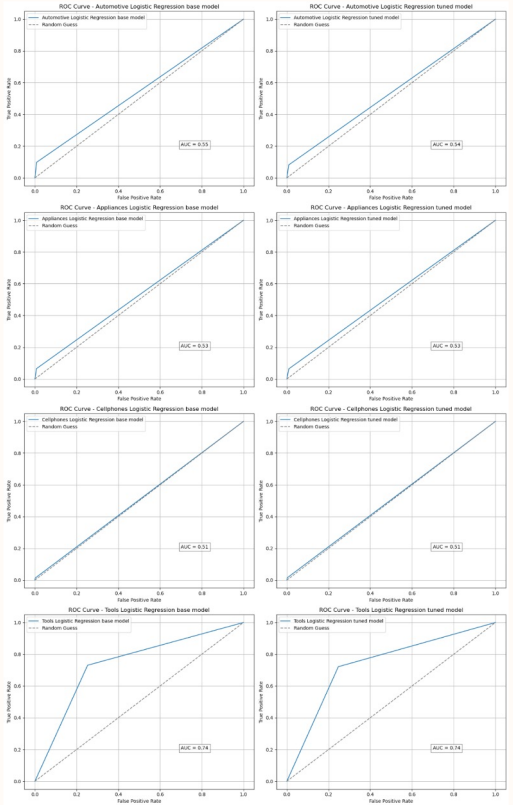
Logistic Regression predictions

The tuned models were tested on the testing dataset, which is 15% of stratified sample data. To understand the performance of classification model we are using confusion matrices. In all categories the class is imbalanced due to which there is high number of true negatives when compared to true positives. Appliances, cellphones and Automotive categories are unable to identify helpfulness which indicates the high bias towards majority class which is not helpfulness which is why there is underfitting for minority class. However, for Tools category, the AUC and Recall improved drastically similar to the base models as the model was trained using class weights. This improvement was with a trade-off with accuracy and precision.

Model	Accuracy	Precision	Recall	F1-Score	AUC	TPR	FPR
Automotive Logistic Regression base model	0.898533	0.583333	0.0983397	0.168306	0.545076	0.0983397	0.00818818
Automotive Logistic Regression tuned model	0.896133	0.516129	0.0817369	0.141125	0.536402	0.0817369	0.00893256
Appliances Logistic Regression base model	0.916789	0.425532	0.0655738	0.113636	0.528868	0.0655738	0.00783858
Appliances Logistic Regression tuned model	0.915322	0.378641	0.0639344	0.109397	0.527322	0.0639344	0.00929017
Cellphones Logistic Regression base model	0.937075	0.4	0.0127932	0.0247934	0.505757	0.0127932	0.00127986
Cellphones Logistic Regression tuned model	0.936408	0.318182	0.0149254	0.0285132	0.506396	0.0149254	0.00213311
Tools Logistic Regression base model	0.745299	0.272964	0.730858	0.397476	0.739017	0.730858	0.252825
Tools Logistic Regression tuned model	0.749567	0.275221	0.721578	0.398463	0.73739	0.721578	0.246798

Logistic Regression Comparison





Model Validation and Evaluation - Random Forest

Base Model. Each category baseline, or base, Random Forest model is trained using default parameters on the training dataset for various categories. There was a slight variation for the “Tools & Home Improvement” category as we used the concept of class weights to handle imbalance in the dataset. This helps the model focus on the minority class instead of being biased towards the majority class.

Hyperparameter-tuned. The validation dataset is used to fine-tune the models using the best possible parameters. The validation dataset was used for tuning since this method lessens overfitting and enhances the model's capacity for generalization. Additionally, since hyperparameter tuning occurs on a smaller dataset than training, which comprises 70% of the dataset, employing the validation dataset, which comprises 15% of the modeling dataset, speeds up the process.

The hyperparameters we have used for Random Forest include **n_estimators** and **max_depth**.

- The **"max depth"** option sets the maximum depth for each tree in the ensemble. It is a measure of a tree's potential complexity and affects the model's ability to identify patterns in the input data.
- The number of boosting rounds or trees to be constructed in the ensemble is indicated by the **"n_estimators"** parameter.

Random Forest Hyperparameters for each category

The following table lists the ideal Random Forest model **hyperparameters** for each category.

We have used **grid search** to tune the hyperparameters. It searches through all possible combinations of the hyperparameter ranges to identify the best possible combination. We have optimized the training time using 3-fold cross-validation because of our restricted computational resources. Finding the best possible parameter combination to get the highest accuracy is our aim when it comes to hyperparameter tweaking.

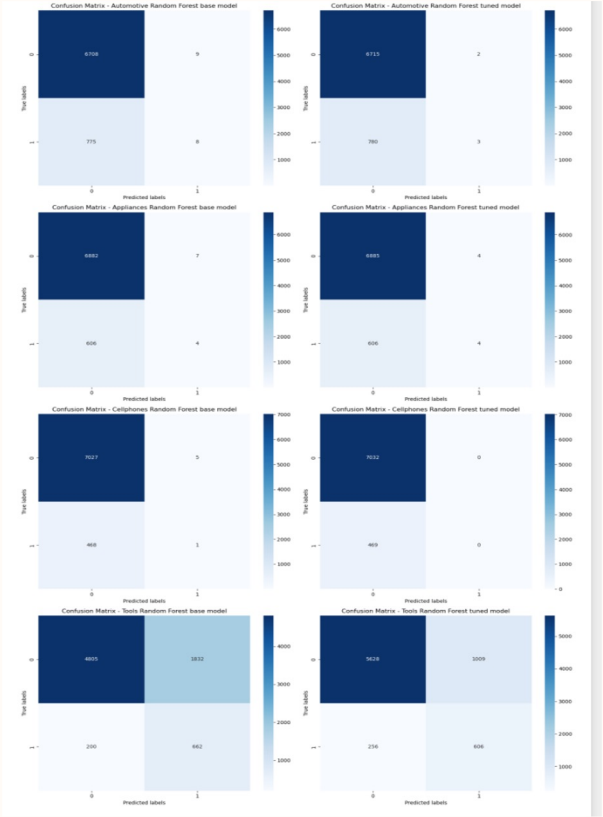
Category	max_depth	n_estimators
Appliances	None	50
Automotive	20	200
Cellphones and Accessories	10	50
Tools and Home Improvement	30	200

Classification metrics for Random Forest model across all categories

Category	Accuracy	Precision	Recall	F1-score	AUC-ROC
Appliances(Random Forest baseline)	91.82	0.3636	0.0065	0.0128	0.5027
Appliances(Random Forest tuned)	91.86	0.5000	0.0065	0.0129	0.5029
Automotive(Random Forest Baseline)	89.54	0.4705	0.0102	0.0200	0.5044
Automotive(Random Forest tuned)	89.57	0.6000	0.0038	0.0076	0.5017
Cellphones(Random Forest Baseline)	93.69	0.1666	0.0021	0.0042	0.5007
Cellphones(Random Forest tuned)	93.74	0.0000	0.0000	0.0000	0.5000
Tools(Random Forest Baseline)	71.88	0.2559	0.7394	0.3802	0.7277
Tools(Random Forest Baseline)	79.49	0.3053	0.5942	0.4034	0.7078

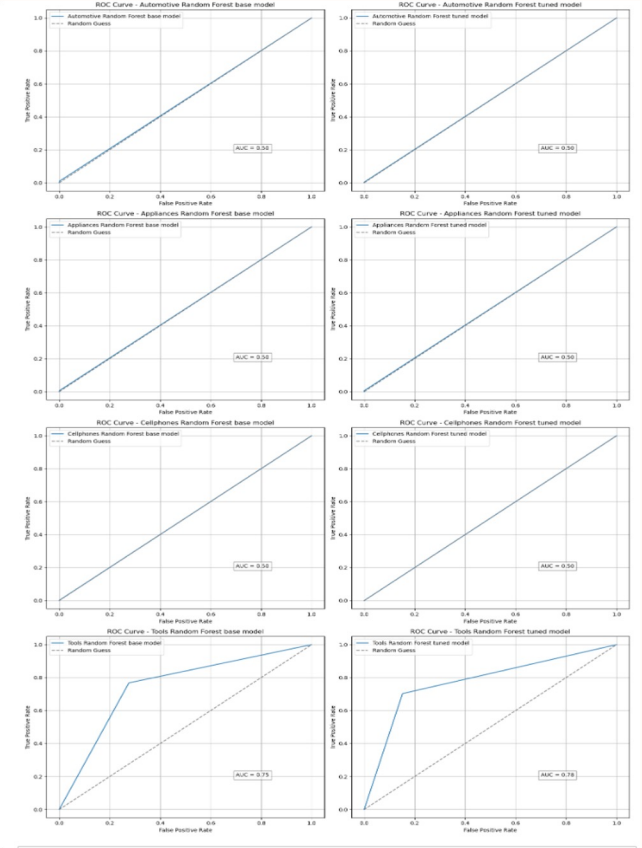
Random Forest Confusion matrix for all categories

- For every prediction produced by the models, the number of true positives, true negatives, false positives, and false negatives is shown in each matrix.
- Notably, tuning did not have any significant impact on models performance. Though precision and accuracy improved, however, the recall was down to 0 meaning the model is doing random guess and is not able to correctly distinguish between True and False.
- Tuning had no major impact on tools category as well, however, with a recall of 0.7, the model is performing better in distinguishing between true and false instances. However, this is with a trade-off on a accuracy and precision.



Random Forest ROC curves for all categories

- The performance of the model is depicted in each set of plots, the tuned models exhibit differing degrees of improvement, while the basic models frequently perform in close proximity to a random guess (as represented by proximity to the diagonal line) same is observed with tuned Models for Automotive, Appliances and Cellphone categories.
- The ROC curves after tuning rise towards the upper left corner of the tools category, showing model was able to capture true positives using class weights.



Model Validation and Evaluation - XGBoost

- The **baseline** or base XGBoost model for each category is trained with default parameters on the respective category's training dataset
- **Hyperparameter-tuned:** The models are tuned with optimal parameters using the validation dataset
    - Tuning was done on the validation dataset since this approach helps in reducing the overfitting of the model and improves the model's ability to generalize
- The hyperparameters that we experimented with for the XGBoost models across all categories were "**learning_rate**", "**max_depth**," and "**n_estimators**"
    - Since XGBoost is an ensemble of multiple trees, the "learning_rate" parameter controls the contribution of each tree in the ensemble to the final helpfulness prediction
    - The "max_depth" parameter controls the maximum depth of each tree in the ensemble to predict the helpfulness of reviews.Optimal value for the "max_depth" parameter is crucial, and we attempted three values, i.e., 3, 5, and 7
    - The "n_estimators" parameter pertains to the number of boosting rounds or trees to be built in the ensemble

Model Validation and Evaluation - XGBoost

Below table gives the optimal hyperparameters for XGBoost across all the categories:

Hyperparameter tuning was done based on Grid Search 3-fold cross validation

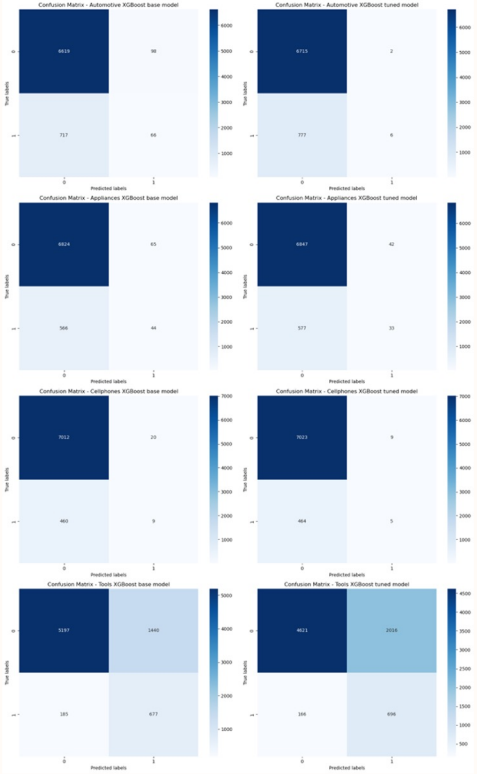
Category	learning_rate	max_depth	n_estimators
Appliances	0.1	5	50
Automotive	0.01	3	200
Cellphones and Accessories	0.2	3	50
Tools and Home Improvement	0.1	3	50

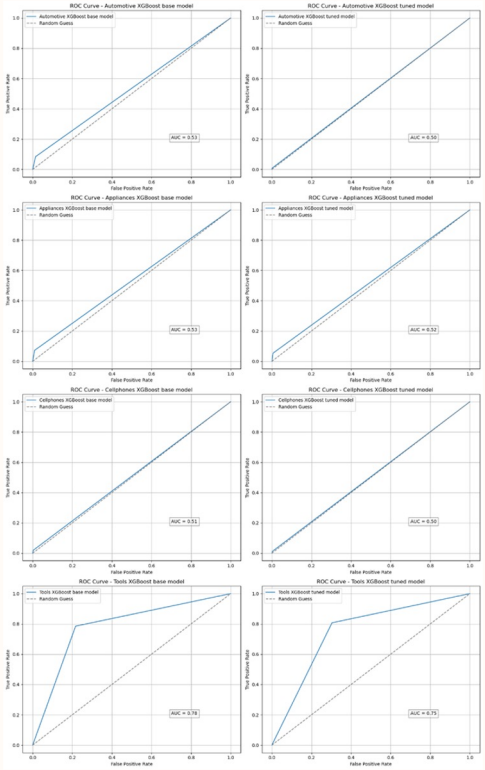
Model Validation and Evaluation - XGBoost

Classification metrics for XGBoost across all categories: a trade-off was observed between accuracy, precision and recall based on multiple approaches. This is subsequently visible in the F1-score as well.

Category	Accuracy	Precision	Recall	F1-Score	ROC AUC
Appliances (XGBoost Baseline)	91.33	0.3484	0.0754	0.1239	0.5314
Appliances (XGBoost Tuned)	91.47	0.3209	0.0426	0.0752	0.5173
Automotive (XGBoost Baseline)	89.13	0.4024	0.0842	0.1393	0.5348
Automotive (XGBoost Tuned)	89.61	0.7500	0.0076	0.0151	0.5036
Cellphones and Accessories (XGBoost Baseline)	93.60	0.3103	0.0191	0.0361	0.5081
Cellphones and Accessories (XGBoost Tuned)	93.69	0.3571	0.0106	0.0207	0.5046
Tools and Home Improvement (XGBoost Baseline)	76.62	0.2856	0.6685	0.4002	0.7238
Tools and Home Improvement (XGBoost Tuned)	69.66	0.2514	0.8091	0.3836	0.7454

Model Validation and Evaluation - XGBoost





- Classification metrics for XGBoost for all categories are shown in the adjoining charts
- While some minor improvements were observed in the AUC based on addressing imbalance, we couldn't observe any model with >0.8

Best performing model comparison

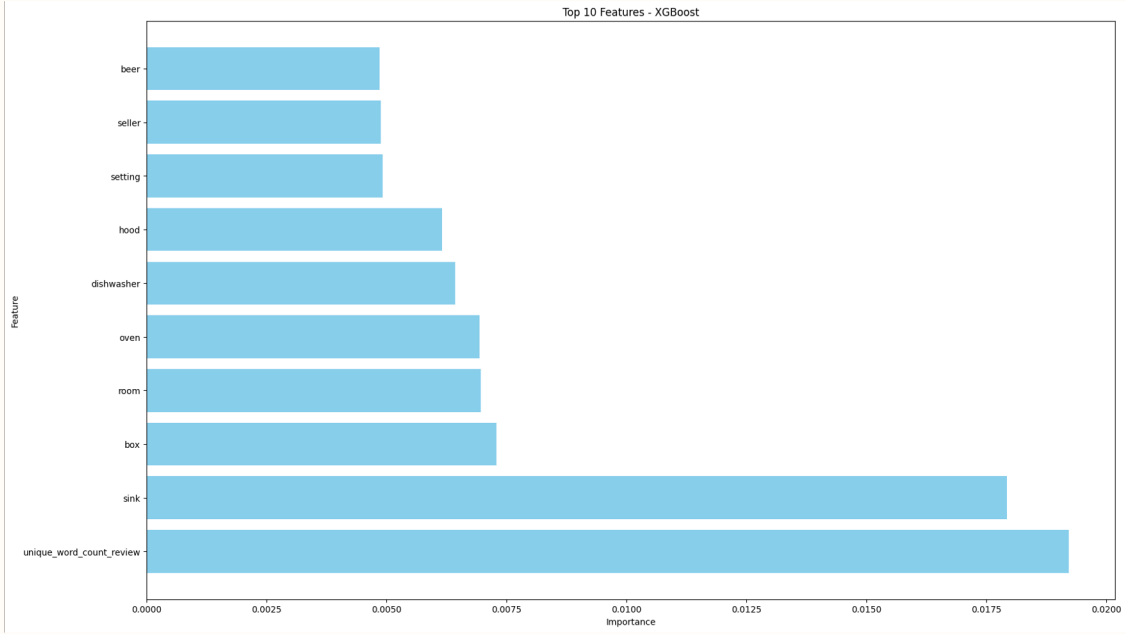
Algorithm	Appliances	Automotive	Cellphones	Tools
Logistic Regression	91.68% (base)	**89.85% (base)**	93.70% (base)	74.95% (tuned)
Random Forest	**91.86% (tuned)**	89.57% (tuned)	93.74% (tuned)	**83.13% (tuned)**
SVM	**91.86% (tuned)**	89.56% (base)	**93.75% (base)**	73.88% (tuned)
XGBoost	91.75% (tuned)	89.61% (tuned)	93.69% (tuned)	78.33% (base)

Model Explainability

Model explainability - Feature importance

- Model explainability is important to provide transparency about what features the model considers important in predicting helpfulness, what is the direction of influence of the feature (positive and negative) in predicting the outcome and do the features contribute to the model in a way that makes intuitive sense
- Feature importance in tree-based methods like Random Forest and XGBoost are directly available from the model object using the “feature_importances_” attribute which calculates the “gain” of each feature
- The importance score of a feature is calculated based on how much each feature contributes to reducing the loss function during the construction of decision trees in the ensemble to predict helpfulness
- This contribution is measured by the improvement in the model's performance (reduction in the loss) attributed to each split that involves the feature

Model explainability - Feature importance



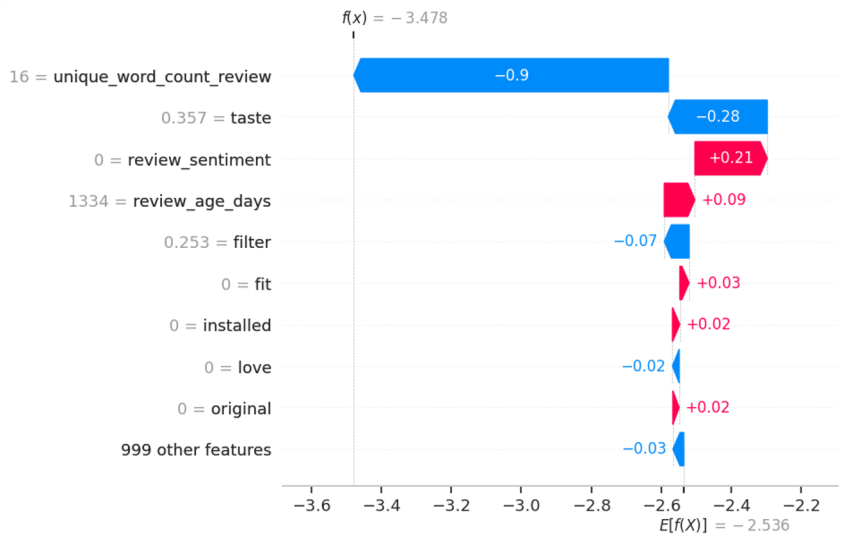
- The chart shows the relative feature importance from the XGBoost model for Appliances category
- We can observe that **“unique_word_count_review” is the most important feature in predicting helpfulness**
- However, the shortcoming of this method of feature importance is that we know the magnitude of the influence but not the direction i.e. **we can’t determine whether unique word count in the review is positively or negatively affecting helpfulness** and this is something that we need to analyze separately

Model explainability - SHAP

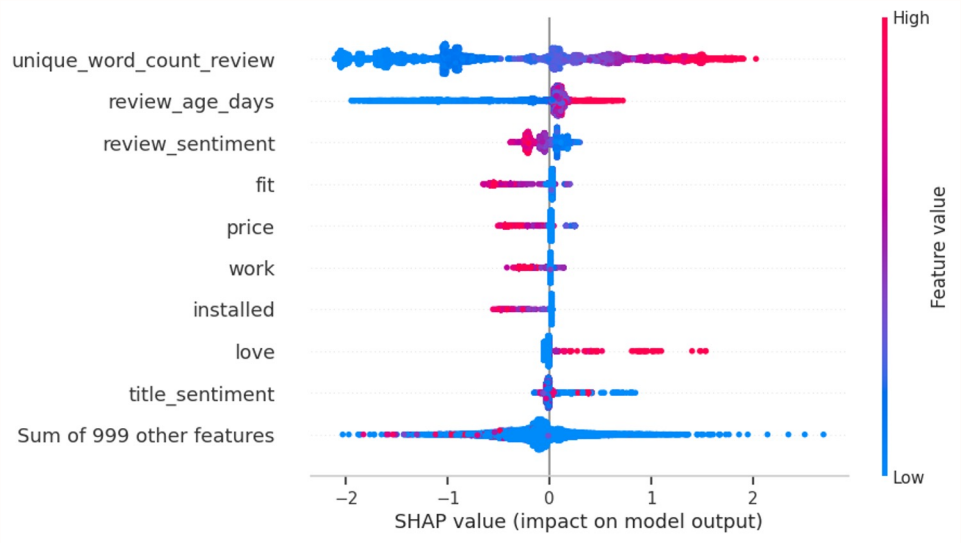
- SHAP (SHapley Additive exPlanations) values are a way for explaining the outcome of machine learning models. They provide a way to understand the contribution of each feature to the model's prediction for helpfulness
- SHAP values are based on cooperative game theory and the concept of Shapley values originate from economics. They aim to fairly distribute the "credit" or "contribution" of each feature across all possible combinations of features
- Positive SHAP values indicate that the feature contributes to increasing the prediction of helpfulness, while negative values indicate the opposite
- The magnitude of the values represents the impact of the feature on the prediction with larger magnitudes indicating greater influence

Model explainability - SHAP

SHAP waterfall chart



SHAP beeswarm chart



- Features pushing the helpfulness prediction higher are shown in **red** and those pushing the prediction lower are in **blue**
- We can now observe that **unique word count negatively impacts the prediction** i.e. more unique words in a review is associated with non helpful reviews
- Similarly, we can notice that review sentiment is positively influencing the model i.e. higher the sentiment score of the review, better is the helpfulness
- The beeswam chart, an alternative way of visualizing the SHAP values for the XGBoost tuned model for Appliances category with a distribution of the data points based on helpfulness

Conclusion

Conclusion

- Models for three categories, “Appliances, Automotive, and Cellphones & Accessories,” high accuracy but low AUC and recall, which can be attributed to the fact that our data is imbalanced and has high dimensionality.
- Our data has the majority class as “Not Helpful” with “Helpful” being the minority class. It's representative of real world data since helpful reviews are limited and exclusive.
- Another approach we explored was to use sample weights to offset the imbalance in the “Tools & Home Improvement” dataset. We were able to make the models focus more on the minority class that is “helpful” reviews.
- We were able to get decent recall values and an ROC score of more than 0.75, however, that was achieved at the cost of accuracy and precision.
- We conclude that even though we were able to arrive at the helpfulness with a certain degree of accuracy, these models do not perform optimally.
- Our hypothesis is that the helpfulness of reviews is subjective and a function of human behavior. Further exploration of different NLP techniques and may be required to achieve highly optimal models for our use case.

Appendix

Dataset and Files

- Dataset Link
- Drive Link
- GitHub Repository Link

Thank You