PROJECT REPORT

on

# Detecting Anomalies in Reviews and Ratings of Amazon Products

*Submitted by*

**Group 5**

**Mentor:** Mr. Koneti Naveen Kumar Yadav

*for the award of the degree*

*of*

**POST GRADUATE PROGRAM**

*in*

**DATA SCIENCE ENGINEERING**


Great Lakes Institute of Management, Chennai

March 2020

**Submitted by**

Charu Goyal

Navin Ramkumar

Neha Seth

Prachi Agarwal

Ramakrishnan Subramanian

# Table of Contents

ABSTRACT ..... 5
CHAPTER 1 ..... 6
INTRODUCTION ..... 6
1.1 Business Problem ..... 6
1.2 Organization of the Report ..... 8
CHAPTER 2 ..... 8
DATASET DESCRIPTION ..... 8
2.1 Amazon Dataset ..... 8
2.2 Pre-Processing ..... 10
Chapter 3 ..... 18
Text Vectorizer Model ..... 18
3.1 Need for a text vectorizer model ..... 18
3.2 Word Counts with CountVectorizer ..... 18
3.3 Word Frequencies with TfidfVectorizer ..... 18
3.4 Selection of Text to Vector Model ..... 19
3.5 Selection of N_Grams ..... 21
Chapter 4 ..... 22
Classification Models ..... 22
4.1 Logistic Regression ..... 22
4.2 Decision Tree Classifier ..... 26
4.3 Random Forest Classifier ..... 26
4.4 Multinomial Naive Bayes ..... 27
4.5 Ensemble Techniques ..... 27
Chapter 5 ..... 31
Important Model Evaluation Metrics ..... 31
5.1 Confusion Matrix ..... 31
5.2 Area under the ROC Curve (AUC- ROC): ..... 33
Chapter 6 ..... 34
Hyperparameter Tuning ..... 34
6.1 Need for hyperparameter tuning ..... 34
6.2 Selection of n_estimators ..... 35
6.3 GridSearchCV ..... 36
Chapter 7 ..... 38

Applying Tuned Model on the Training Data ..... 38
7.1 Passing the training data ..... 38
7.2 Preparing the Test Data ..... 39
7.3 Model Anomaly calculation ..... 39
Chapter 8 ..... 40
Amazon dataset (Test data) with the Trained Model ..... 40
8.1 Need for Sampling ..... 40
8.2 Final Model on the Test set ..... 42
8.3 Anomaly Detection Performance evaluation ..... 44
8.4 Threshold Tuning ..... 45
8.5 Results after threshold tuning: ..... 48
8.6 Analyzing False Positives and False Negatives ..... 49
REFERENCES ..... 52

## Table of Figures

Figure 1 Reviews per product source: blog.3dcart.com ..... 6
Figure 2 Flow chart of processing steps ..... 10
Figure 3 Word Cloud representation of Positive Labels ..... 12
Figure 4 Word Cloud representation of Negative Labels ..... 12
Figure 5 Percentage missing values ..... 13
Figure 6 Rating vs Count ..... 14
Figure 7 Rating vs Count ..... 14
Figure 8 Unique name and count of reviews ..... 15
Figure 9 Positive reviews word cloud ..... 16
Figure 10 Negative Reviews word cloud ..... 16
Figure 11 Count of reviews in the training dataset ..... 17
Figure 12 Word cloud of the training dataset ..... 17
Figure 13 Tf-IDF - CountVectorization performance ..... 19
Figure 14 CountVectorization and Tf-IDf comparison ..... 20
Figure 15 Confusion Matrix - Tf-IDF ..... 20
Figure 16 Confusion Matrix - CountVectorizer ..... 20
Figure 17 Confusion Matrix - Base Model ..... 23
Figure 18 K-FoldCV ..... 24
Figure 19 BE-VE Trade-Off ..... 25
Figure 20 Bagging vs Boosting ..... 28
Figure 21 Confusion Matrix ..... 31
Figure 22 ROC-AUC Curve ..... 33
Figure 23 n_estimators vs bias accuracy ..... 36
Figure 24 n_estimators vs coef. of variance ..... 36
Figure 25 GridSearchCV results ..... 37
Figure 26 Training Model Evaluation Matrices ..... 38
Figure 27 Normalized CF - Trained Model ..... 38

Figure 28 Confusion Matrix - Trained Model ..... 38
Figure 29 Training Sample value counts ..... 41
Figure 30 Training data (40.8k) value counts ..... 41
Figure 31 Confusion Matrix - Sentiment Prediction ..... 42
Figure 32 Count of True Anomalies by manual labelling ..... 43
Figure 33 Confusion Matrix - Anomaly prediction ..... 44
Figure 34 Confusion Matrix - Weightage ..... 46
Figure 35 Threshold tuning ..... 47
Figure 36 Positive Reviews - Model prediction ..... 49
Figure 37 WordCloud - Positive ..... 49
Figure 38 Negative reviews - Model prediction ..... 50
Figure 39 WordCloud - Negative ..... 50

# ABSTRACT

The e-commerce sales are surging which has increased the significance of online reviews and product ratings in influencing the decisions made by the users in day-to-day life. The users make use of these ratings and reviews to help them to make their respective purchases. However, there have been instances where these product ratings and reviews given by a customer do not synchronize with each other, which leads to the downfall of the reliability of the customer ratings. This gives birth to anomalies present between the ratings and reviews given by a customer. Unfortunately, automatically detecting such reviews and ratings is a challenging problem since incorrect ratings do not seem out-of-place next to genuine ratings and reviews.

In this project, addressing detecting anomalies in online reviews and ratings using an unsupervised approach. Most importantly, with the help of the models are explicitly labeling whether the anomaly or not an anomaly for the data.

Keywords: Anomaly Detection, Outlier Detection, Online Retail Sector, Online Reviews, Unsupervised learning, Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Multi-nominal Naives Bayes, Ensemble Techniques.

# CHAPTER 1

# INTRODUCTION

The introductory chapter lays out the business problem, its significance and the objective.

## 1.1 Business Problem

Over the years, there has been an exponential growth in the online retail business. In the world of e-commerce, customer trust is very vital and the predominant way to gain customer confidence is by exhibiting customer-generated reviews, ratings.

According to a study conducted by the Bazaarvoice network, one product review can result in a 10% increase in sales, and 200 reviews can result in as much as a 44% increase in sales. The network had found that not only did the product reviews increase the number of conversions, but also increased overall traffic, SEO rankings, and encouraged other reviews, post the analysis of 57 million reviews and more than 35 billion product page views. The following graphic displays the importance of how a product review increases the number of orders.


*Figure 1 Reviews per product source: blog.3dcart.com*

As online reviews play such an important role in the sale of the individual products as customers rely on the product ratings and reviews, therefore it's crucial to ensure that the ratings and reviews given by a customer conform. In the past, there have been cases where there has been incongruence between the reviews for the product and the ratings provided by a customer, leading to the deposition of the customer ratings.

Addressing this problem of finding patterns in the data that do not conform to expected behavior or in other words, detecting anomalies. These non-conforming patterns are often referred to as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants in different application domains. There are numerous applications of detecting such discrepancies within data. Anomaly detection finds extensive use in a wide variety of applications such as fraud detection for credit cards, insurance or health care, intrusion detection for cyber-security, fault detection in safety-critical systems, and military surveillance for enemy activities. The importance of anomaly detection is because anomalies in data translate to significant (and often critical) actionable information in a wide variety of application domains. For example, an anomalous traffic pattern in a computer

In this project, the focus is on the domain of reviews and ratings received on an e-commerce website, precisely for Amazon.com. The goal is to identify any inconsistencies between the ratings and the reviews given by a customer, thereby confirming if the rating is in line with the review. The importance of checking for such irregularities is significant and critical as could lead to actionable information and helps to gauge the customer response concerning product reviews and ratings.

The e-commerce retailers, in this case, Amazon.com on detecting the mismatch between the ratings and reviews can send an email notification to their end and ask them: "You have rated this product as 1 however, the review in text form is good. It would be great if you can help us in modifying the rating or provide a better review?"

Hence, this anomaly detection of rating and review can help the stakeholders to suggest to the end-user that the rating they had provided is different from the review they testified and therefore, can improve the same.

## 1.2 Organization of the Report

The report is organized as firstly, the description of the dataset is provided that gives the insights about the merging of the different CSV files and exploratory data analysis is performed which give the relationship of different features and also the Natural Language Processing (NLP) techniques that are applied to clean and pre-process the data. In addition to this, also explained is how the train and test dataset is formed.

Chapter 3 shows how the text data is converted into numerical data using the vectorizer models. Chapter 4 details the classification models and the ensemble techniques that are used and chapter 5 charts the model evaluation metrics. In chapter 6, the hyperparameters are explained followed by how the tuned model is applied to the training data in chapter 7. In Chapter 8, the evaluation of the methodology and the testing of the model on the Amazon dataset is detailed along with the detection of the anomalies.

# CHAPTER 2

# DATASET DESCRIPTION

This chapter gives an insight of the dataset such as a description of the dataset with all the features along with Exploratory Data Analysis explaining the dataset and performing the needed pre-processing steps.

## 2.1 Amazon Dataset

Amazon.com is an American multinational technology firm headquartered in Seattle, Washington. The company was founded in 1994 and has its stock on the National Association of Securities Dealers Automated Quotations (NASDAQ) exchange. The company had started as an online marketplace for books but expanded to sell electronics, software, video games, apparel, furniture, food, toys, and jewelry. In 2015, Amazon surpassed Walmart as the most valuable retailer in the United States by market capitalization. In 2018, the two-day delivery service, Amazon Prime was announced that had surpassed 100 million subscribers worldwide.

Amazon sells more than 12 million products, not including books, media, wine, and services. When Amazon Marketplace sellers are also counted in, the total product count surges to more than 353 million.

Our data is a list of over 67000 consumer reviews for Amazon products such as Kindle, Fire TV Stick, and more. The dataset includes basic product information, rating, reviews, text, and more for each product provided by Datafiniti's Product Database. Datafiniti gives access to web data without having to set up a web scraper. The dataset includes basic product information, rating, review text, and more for each product.

This dataset has been taken from Kaggle. The link hereby is Consumer Reviews of Amazon Products. The dataset described has been obtained because of combining three individual datasets found in the aforementioned link. There are 67992 records and ten attributes.

*The features are as follows:*

| Variable | Definition |
|---|---|
| **name** | name of the product |
| **categories** | name of the categories |
| **PrimaryCategories** | name of the Primary Categories |
| **reviews.dated** | date of the review |
| **reviews.rating** | ratings are given by the user for the product |
| **review.text** | text review is given by the user for the product |
| **reviews.title** | title of the review |
| **reviews.username** | name of the user or reviewer |
| **reviews.doRecommend** | whether the reviewer recommends the product |
| **reviews.numHelpful** | the number of people who found it helpful |

*Table 1 Variable Description*

For training our model, we have taken another merged dataset that combines the following three datasets:

- imdb.com
- amazon.com
- yelp.com

These datasets were created for a paper: From Group to Individual Labels using Deep Features',

Kotzias et. al, KDD 2015. It contains sentences labeled with positive or negative sentiment, extracted from reviews of movies, products, and restaurants, respectively.

Each of the records has a score that is either 1 (for the positive sentiment) or 0 (for the negative sentiment). For each website, there exist 500 positives and 500 negative sentences. These were selected randomly for larger datasets of reviews. Hence, we have 3000 records in total in this combined merged rating dataset.

## 2.2 Pre-Processing

Pre-processing refers to the transformations applied to data before feeding it to the algorithm. Data preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

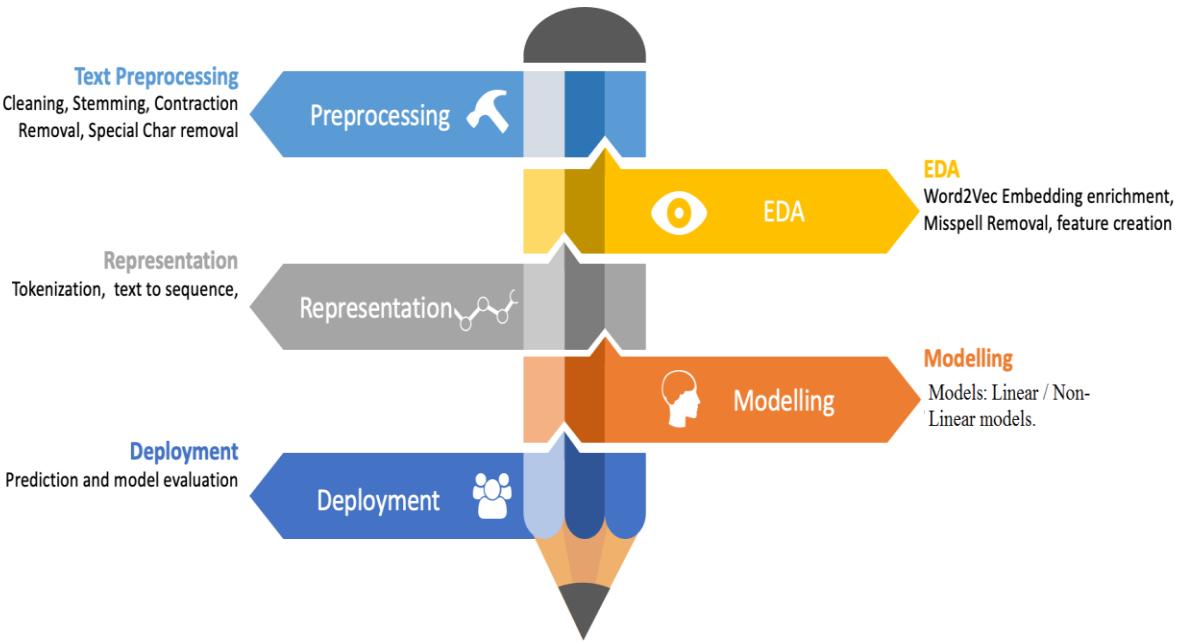



*Figure 2 Flow chart of processing steps*

Since the dataset is dealing with text columns which have raw text, that is pretty messy for these reviews so before we can do any analytics, we need to clean things up using the following steps:

- Regular expressions (regex or regexp) are extremely useful in extracting information from any text by searching for one or more matches of a specific search pattern (i.e. a specific sequence of ASCII or Unicode characters). Regex expressions are used to remove any unwanted symbols, punctuations & special characters. Lastly, the sentence is converted into lowercase.
- The second step involves spelling correction, with the help of **TextBlob.correct()** method spelling can be corrected.
- Lemmatization works by identifying the part-of-speech of a given word and then applying more complex rules to transform the word into its true root.
- "Stop words" are the most common words in a language like "the", "a", "on", "is", "all". These words do not carry important meaning and are usually removed from texts. It is possible to remove stop words using Natural Language Toolkit (NLTK).
- Since the project is dealing with anomaly detection so some stop words are formatted and removed from the stop words dictionary.

After applying all the above steps the dataset containing 3K records is visualized with labels negative (0) and positive (1) using a word cloud. The dataset is balanced with 1500 negative and 1500 positive reviews.

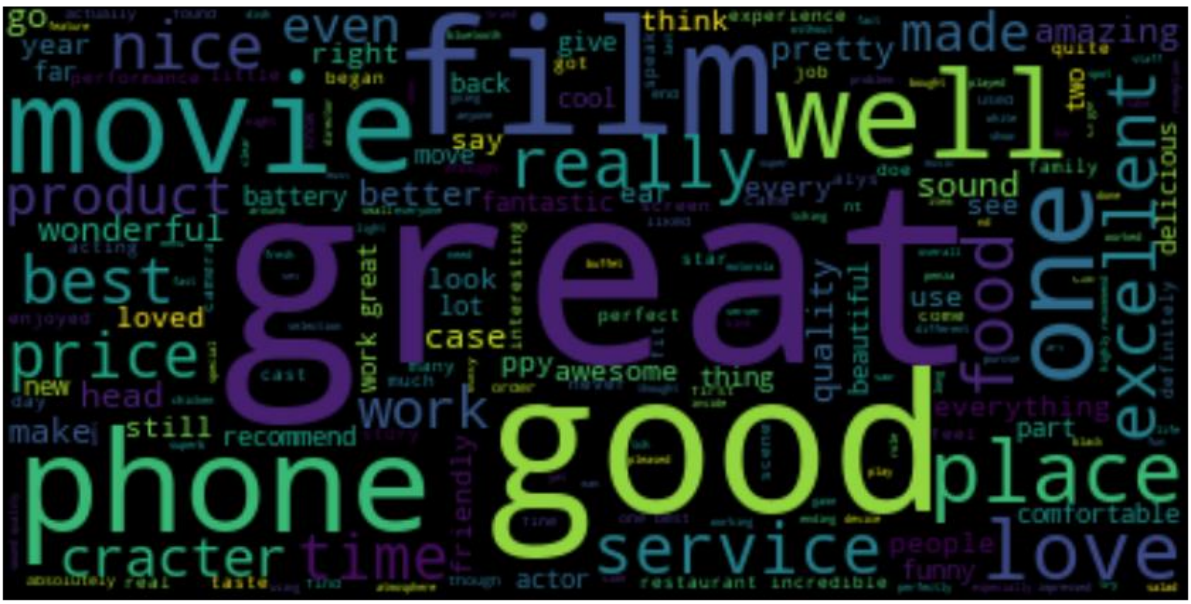



*Figure 3 Word Cloud representation of Positive Labels*


*Figure 4 Word Cloud representation of Negative Labels*

Training records were not sufficient for the testing dataset therefore some records from the testing dataset is concatenated for further analysis.

**Amazon electronics dataset analysis:**

- Missing values analysis: The dataset consists of 67,992 records/rows with 10 columns, the dataset consists of various electronic records like Amazon fire stick, Amazon Kindle, batteries, etc.

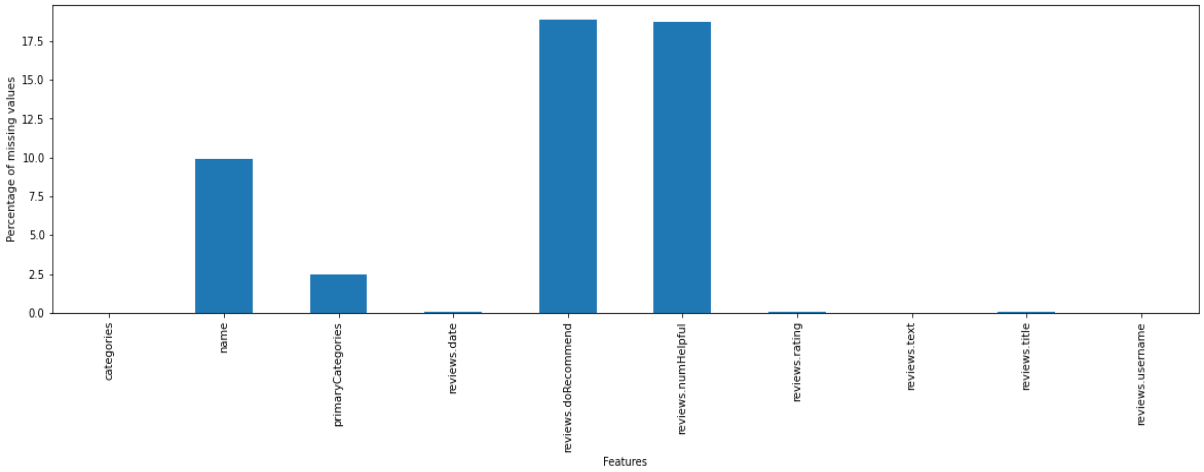



*Figure 5 Percentage missing values*

| Features | Total | Percentage |
|---|---|---|
| **categories** | 0 | 0.0 |
| **name** | 6760 | 9.9 |
| **primaryCategories** | 1692 | 2.48 |
| **reviews.date** | 39 | 0.057 |
| **reviews.doRecommend** | 12840 | 18.88 |
| **reviews.numHelpful** | 12746 | 18.74 |
| **reviews.rating** | 33 | 0.048 |
| **reviews.text** | 1 | 00.0147 |
| **reviews.title** | 18 | 0.026 |
| **reviews.username** | 2 | 0.0029 |

*Table 2 Percentage missing values*

Dropping all the columns except the name, reviews.title reviews.text and reviews.rating. Combining reviews.text and reviews.title into a new column named as text_title. All the further data pre-processing steps are applied the same as above mentioned to add more records to the 3K training records. TextBlob sentiment analysis is selected from all other available analysis techniques as it is giving more accurate results. Before diving into the selection of training records some of the basic analyses on reviews.ratings are plotted.

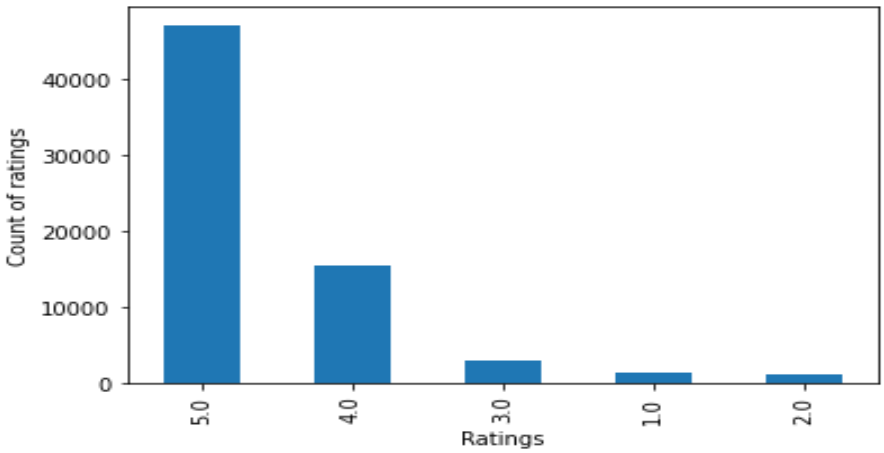



*Figure 6 Rating vs Count*

As it is impossible to detect anomalies with ratings of 2, 3, 4, therefore keeping only 1, 5 ratings for further analysis.

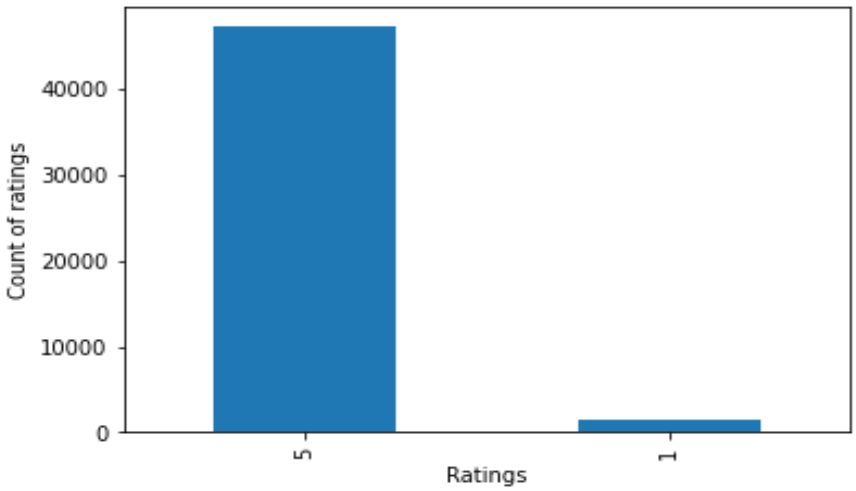



*Figure 7 Rating vs Count*

The Names of some of the products having reviewed by 50 customers are visualized. Total of 43 unique names are present in the dataset.

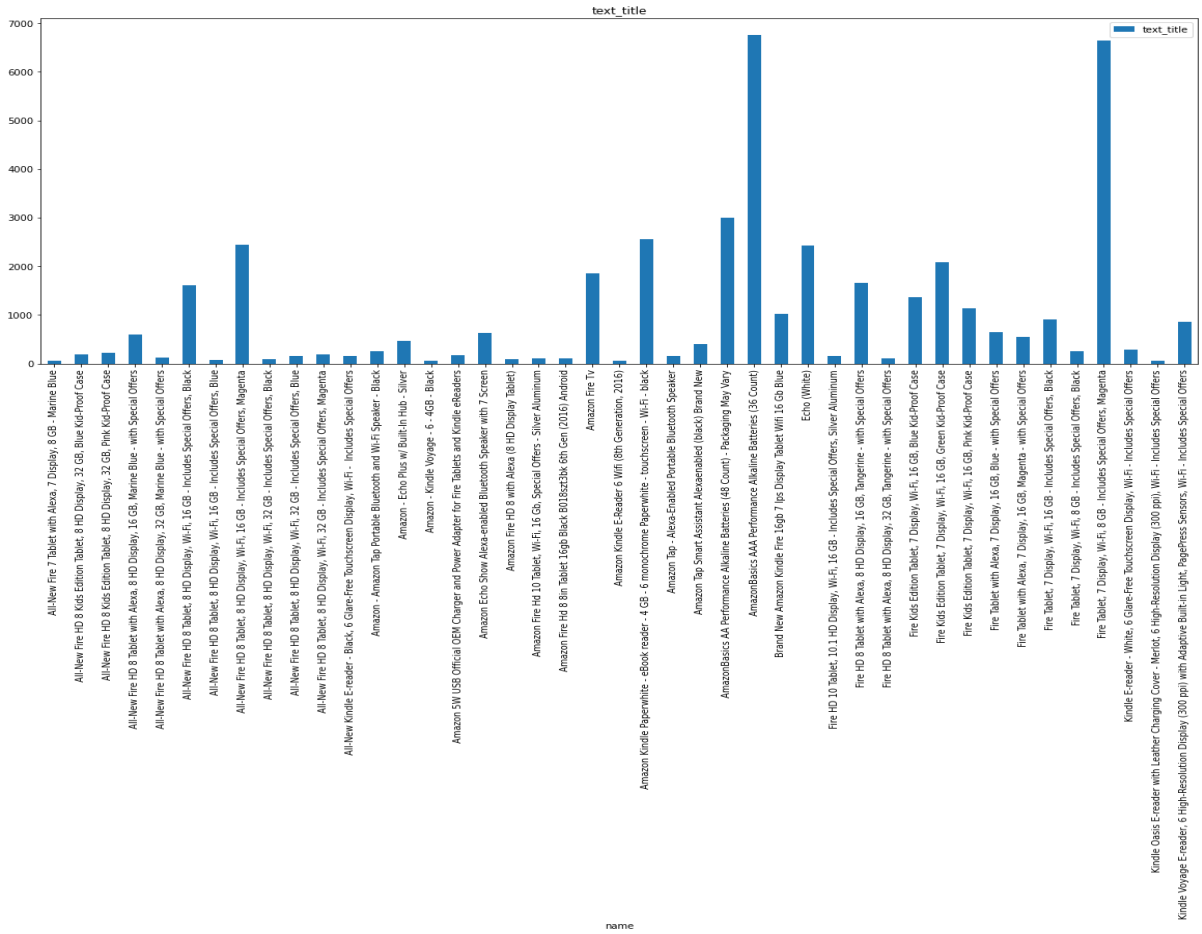



*Figure 8 Unique name and count of reviews*

Dropping column name, and after preprocessing text_title column grouping all the positive reviews (rating=5) and the negative reviews (rating=1) to visualize the words present in the dataset.

The word cloud is created, here dealing with the amazon dataset so word cloud is customized on the amazon logo.

It is visible that amazon products are loved by the customers, the positive word cloud clearly shows words like love, great, five stars, and easy use reviews.

Whereas with the negative word cloud worst, bad, return and one-star reviews are visible.


*Figure 9 Positive reviews word cloud*


*Figure 10 Negative Reviews word cloud*

TextBlob sentiment analysis is done on text_title column after data pre-processing step is done, where a threshold of 0.7 is set for positive sentiments (rating =5) and values with threshold 0.0 is set for selecting negative sentiments (rating =1), selected only the values which satisfy this criterion and added to the 3K dataset which leads to a total of 10,600 rows and training dataset.

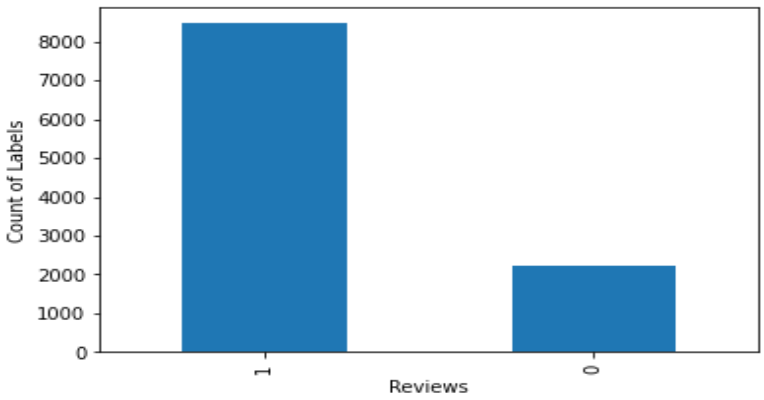



*Figure 11 Count of reviews in the training dataset*

Looking at the plot it is evident data is an imbalance that is required for the analysis of the test dataset. Looking at the word cloud of the training dataset.


*Figure 12 Word cloud of the training dataset*

# Chapter 3

# Text Vectorizer Model

## 3.1 Need for a text vectorizer model

Text data requires special preparation before you can start using it for predictive modelling.

The text must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating-point values for use as input to a machine learning algorithm, called feature extraction (or vectorization).

The scikit-learn library provides 2 different schemes that can be used, and so will briefly look at each.

## 3.2 Word Counts with CountVectorizer

The CountVectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

You can use it as follows:

1. Create an instance of the *CountVectorizer* class.
2. Call the *fit()* function in order to learn a vocabulary from one or more documents.
3. Call the *transform()* function on one or more documents as needed to encode each as a vector.

An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

Because these vectors will contain a lot of zeros, call them sparse. Python provides an efficient way of handling sparse vectors in the scipy.sparse package.

The vectors returned from a call to transform() will be sparse vectors, and you can transform them back to numpy arrays to look and better understand what is going on by calling the toarray() function.

## 3.3 Word Frequencies with TfidfVectorizer

Word counts are a good starting point but are very basic.

One issue with simple counts is that some words like "*the*" will appear many times and their large counts will not be very meaningful in the encoded vectors.

An alternative is to calculate word frequencies, and by far the most popular method is called TF-IDF. This is an acronym that stands for "*Term Frequency – Inverse Document*" Frequency which are the components of the resulting scores assigned to each word.

- **Term Frequency**: This summarizes how often a given word appears within a document.
- **Inverse Document Frequency**: This downscales words that appear a lot across documents.

Without going into the math, TF-IDF is word frequency scores that try to highlight more interesting words, e.g. frequent in a document but not across documents.

The TfidfVectorizer will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents. Alternately, if you already have a learned CountVectorizer, you can use it with a TfidfTransformer to just calculate the inverse document frequencies and start encoding documents.

The same create, fit, and transform process is used as with the CountVectorizer. [6]

## 3.4 Selection of Text to Vector Model

Since both CountVectorizer and Tf-IDF are popular models to convert a string of text to numerical values, both these models are tried and tested in this project, so the better ones can be chosen for further applying predictive models. It is also important to apply some of the most advanced NLP techniques to process or clean the text. These text cleaning or processing techniques that are considered for this project are Regex – Regular expressions, Removal of stopwords, Lemmatization, Stemming – Potter and Snowball, and spelling mistake correction. All these techniques improve the model by reducing the redundancy, data size, and thereby inefficient prediction by the model. Detailed explanations about these NLP text processing techniques are included in the above sections of this report.

Since each of these techniques works differently, some don't go well with another, therefore all different possible combinations of these text processing techniques are applied and converted into a sparse matrix using CountVectorizer and Tf-IDF, then Logistic Regression and SVM models are used for prediction of the sentiment. In the training data, they have positive and negative statements, the model's performance will depend on how efficiently the text is processed for the model to learn and differentiate between the two distinctive sentiments. The resulting comparison will help choose which of the two techniques will be considered for our project.

The bias accuracies and variance errors of the predictive models, using text processed by different NLP technique combinations can be observed in the graph below:

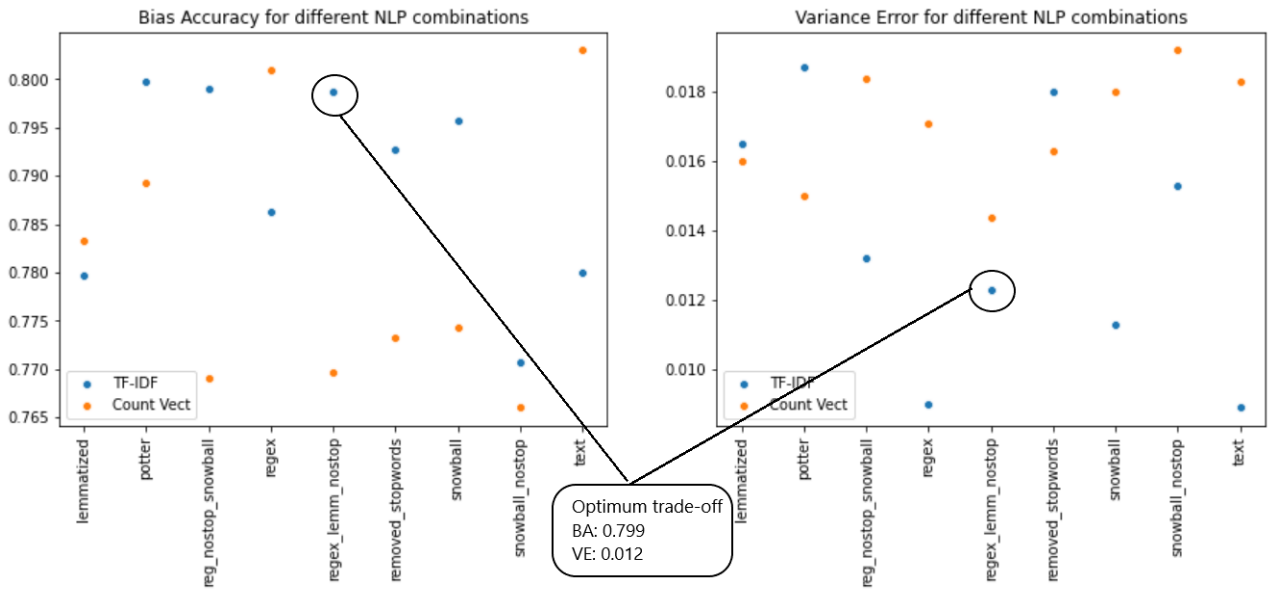



*Figure 13 Tf-IDF - CountVectorization performance*

The objective is to choose the best NLP combination and the best text to vector model based on high bias error and low variance error trade-off. High bias error signifies, better prediction, and low chances of overfitting, low variance yields to good model stability.

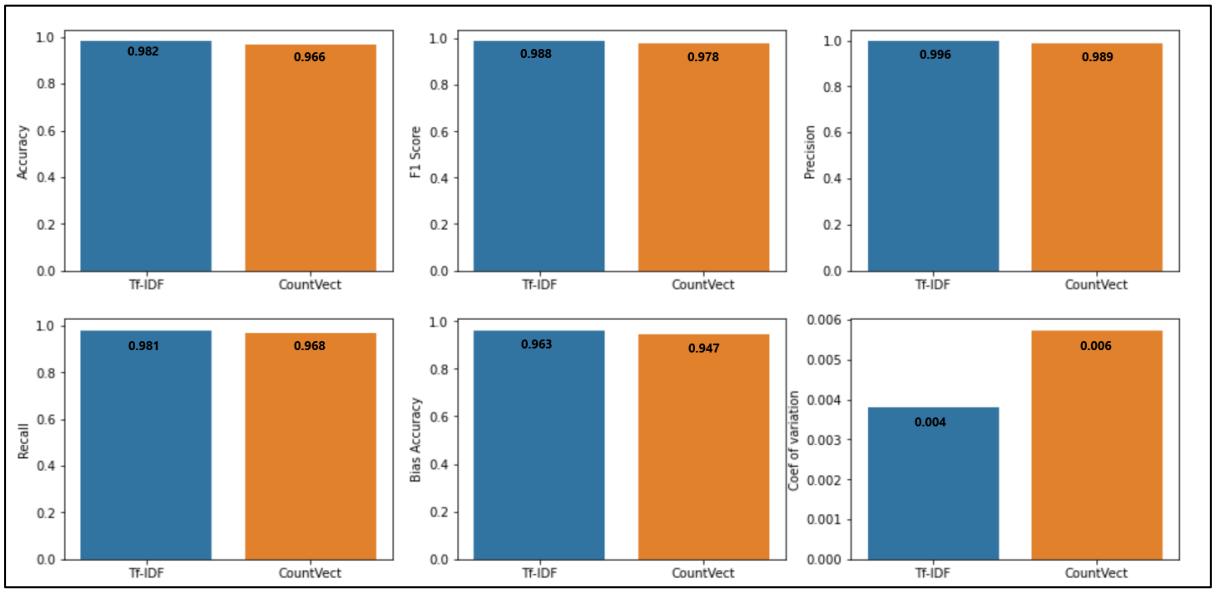



Figure 14 CountVectorization and Tf-IDf comparison

From the above graph, it is evident from all the following scoring metrics; Accuracy, Precision, Recall, Bias accuracy, and Coefficient of variation are very optimum for the Tf-IDF model over Count Vectorizer.

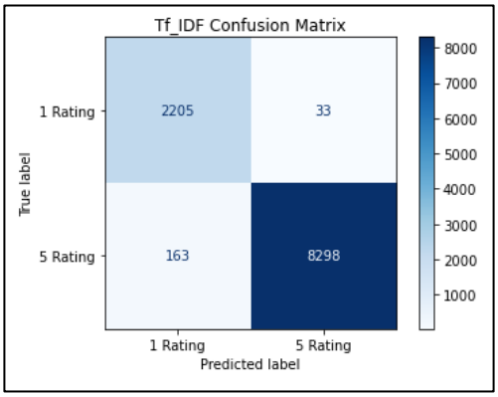



Figure 15 Confusion Matrix - Tf-IDF

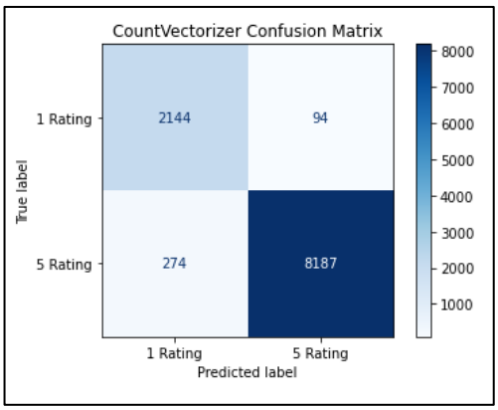



Figure 16 Confusion Matrix - CountVectorizer

From the two-confusion matrix above, one can see that Tf-IDF true positive and true negative for 1 and 5 stars respectively are better than CountVectorizer. 33 and 163 False Negative and False positives resp. are the wrong predictions of Tf-IDF whereas 94 and 274 are FN and FP, wrong predictions while using CountVectorizer. This clearly says that data to numerical preparation is more efficiently executed by Tf-IDF over CountVectorizer.

Considering these points, text that is of these processes: regex, lemmatization, and removal of stop words, using Tf-IDF and logistic regression model yields to 83.3 (%) bias accuracy and variance error of 0.1 (%). This trade-off is most optimal and therefore the training data and testing data will be processed similarly before predictive modeling is applied.

## 3.5 Selection of N_Grams

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus. When the items are words, n-grams may also be called shingles.

Turns out that is the simplest bit, an N-gram is simply a sequence of N words. For instance, let us take a look at the following examples.

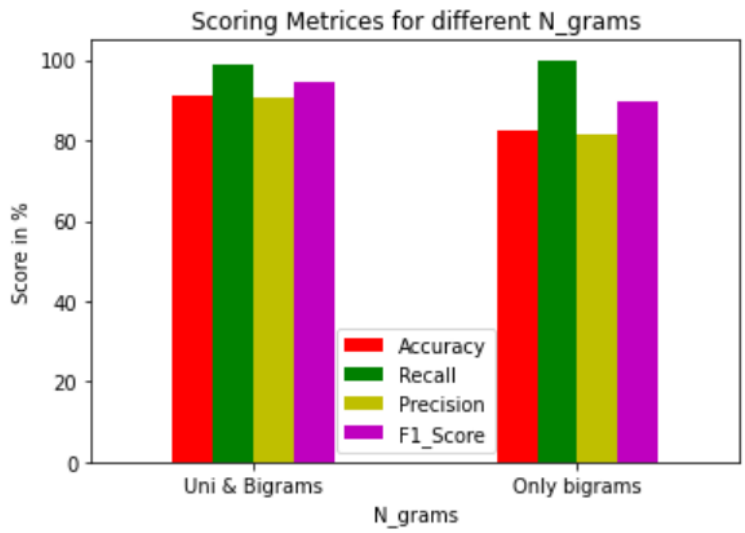
San Francisco (is a 2-gram)
The Three Musketeers (is a 3-gram)
She stood up slowly (is a 4-gram)

Now which of these three N-grams have you seen quite frequently? Probably, “San Francisco” and “The Three Musketeers”. On the other hand, you might not have seen “She stood up slowly” that frequently. Basically, “She stood up slowly” is an example of an N-gram that does not occur as often in sentences as Examples 1 and 2.

It can also help make next word predictions. Say you have the partial sentence “Please hand over your”. Then it is more likely that the next word is going to be “test” or “assignment” or “paper” than the next word being “school”.

It can also help to make spelling error corrections. For instance, the sentence “drink cofee” could be corrected to “drink coffee” if you knew that the word “coffee” had a high probability of occurrence after the word “drink” and also the overlap of letters between “cofee” and “coffee” is high.
As you can see, assigning these probabilities has a huge potential in the NLP domain.

For sentiment analysis purpose on reviews, it is simply sufficient to train the model based on bigrams (2 letters) that will include the words that give out the negative sentiment when paired for example: do not, did not, have not, were not, etc. These words give the entire negative meaning only then they are combined as bigrams. Now, in text vectorization technique, there’s option to selection N_gram.
For our project, analysis is done to verify if only bigrams are sufficient or combination of unigrams and bigrams are required for better prediction. The results can be seen below:




From this comparison plot, it is observed that the combination of uni and bigrams give better accuracy, precision and F1-score, whereas recall is just 0.009% higher for only bigrams over uni & bigrams combo. Therefore, it is beneficial for our project to adhere with the unigram and bigram combination.

# Chapter 4

# Classification Models

## 4.1 Logistic Regression

**Definition:** Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1/0, Yes / No, True / False) given a set of independent variables. To represent the binary/categorical outcome, we use dummy variables. Logistic regression is a special case of linear regression when the outcome variable is categorical, where we are using a log of odds as the dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a particularly important aspect of Logistic regression and is dependent on the classification problem itself.

The decision for the value of the threshold value is majorly affected by the values of precision and recall. Ideally, we want both precision and recall being 1, but this seldom is the case. In the case of a Precision-Recall trade-off, we use the following arguments to decide upon the threshold.

- **Low Precision/High Recall:** In applications where we want to reduce the number of false negatives without necessarily reducing the number of false positives, we choose a decision value which has a low value of Precision or a high value of Recall.
  **For example:** In the case of the medical field, employees churn or attrition rate, etc false negative would be costlier, thus the focus should be on reducing the number of false negatives in comparison to false positives.

- **High Precision/Low Recall:** In applications where we want to reduce the number of false positives without necessarily reducing the number of false negatives, we choose a decision value that has a high value of Precision or a low value of Recall.
  **For example:** In the case of email spam detection, cybersecurity space, etc, false-positive would be costlier, thus the focus should be on reducing the number of false-positive in comparison to false negatives.

**Advantages:**

- Effective and simple to implement.
- It is most useful for understanding the influence of several independent variables on a single outcome variable.
- Does not require input features to be scaled (can work with scaled features too but doesn't require scaling).

**Disadvantages:**

- Poor performance on non-linear data (image data e.g.)
- Poor performance with irrelevant and highly correlated features.
- Not a powerful algorithm and can be easily outperformed by other algorithms.

### 4.1.1 Base Model Results: Logistic Regression (using Sklearn)

| | Logistic Regression |
|---|---|
| ROC_AUC Score | 0.815020 |
| Accuracy Score | 0.913707 |
| F-1 Score | 0.947208 |

*Table 3 Base model results*

```
Confusion Matrix:
 [[ 448  252]
 [  25 2485]]

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.64      0.76       700
           1       0.91      0.99      0.95      2510

    accuracy                           0.91      3210
   macro avg       0.93      0.82      0.86      3210
weighted avg       0.92      0.91      0.91      3210
```

**Confusion Matrix Plot:**

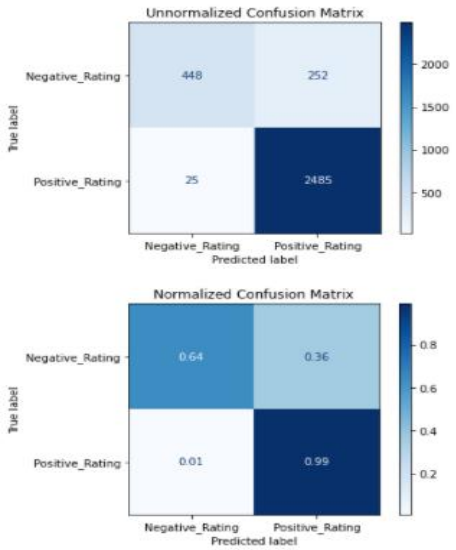



*Figure 17 Confusion Matrix - Base Model*

**Inference:** From the above base model it can be inferred that the model can detect 64% of the Negative Ratings and 99% of the Positive Ratings correctly and also it gives the different metrics from which the model can be explained.

### 4.1.1 K- Fold Cross-Validation

**Definition:** Splitting the data into train and test is necessary for supervised learning, however, the results of the model will be based on this only partition. A solution to this problem is a procedure called k-fold cross-validation, where data is divided into *k* smaller sets. The training data will be *k-1* folds, the remaining fold will be used for validation. This is a brilliant way of achieving the bias-variance tradeoff in the testing process and ensuring that the model itself has **high bias accuracy** (or low bias error) and **low variance error.**

**Steps for K- Fold Cross-Validation:**

- Partition the dataset into 'k' subsets
- Consider one subset as the test set and the remaining subsets as the train set
- Measure the model performance
- Repeat this until all k subsets are considered as the test set
- Here, each observation is used exactly k times for training and exactly once for testing
- The total error is obtained by summing up the errors for all the k runs

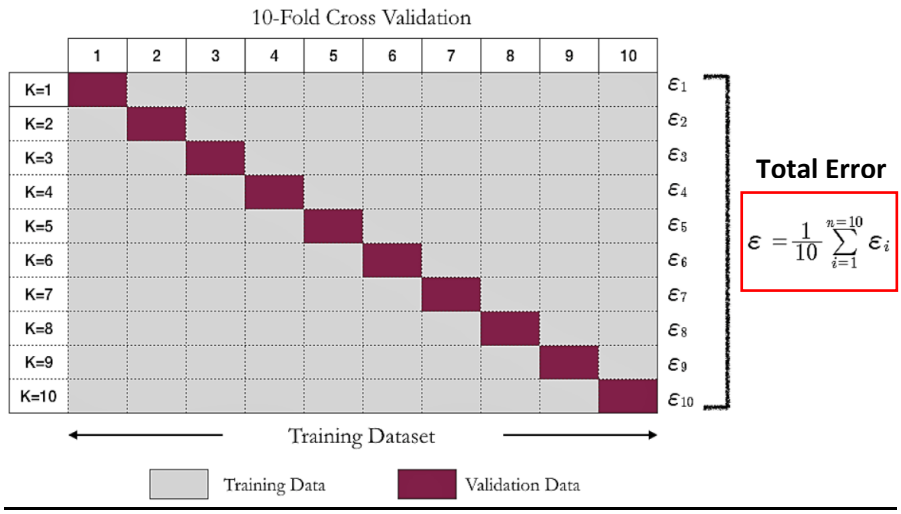



*Figure 18 K-FoldCV*

It's very important to assess the model's bias-variance characteristics using k-fold Cross-Validation. Ideally, we want both bias and variance error to below. While performing k-fold CV, we get *k* different estimates of model's error- say e1, e2, e3…. etc. Since each eye is an *error estimate,* it should ideally be zero. [3]

In the case of Bias- Variance trade-off we should understand the following terms to conclude the final results of a model.

- **Bias:** Bias is the difference between a model's predicted values and the observed values. To check out the model's bias, find out the mean of all the bias. If this value is low, it means that the model gives low error on an average– indirectly ensuring that the model's notions about the data are accurate enough.

- **Variance:** Variance is the variability of model prediction for a given data point or a value that tells the spread of the data**.** To check out the model's *variance*, compute the Coefficient of Variation (COV).

COV is defined as a measure of relative variability. It is the ratio of the standard deviation to the mean (average). If this value is high, it means that the model's performance (in terms of its error) varies a lot with the dataset used for training.

**Bias Variance Trade-Off**

If the model is too simple and has very few parameters, then it may have high bias and low variance. On the other hand, if the model has a large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data. This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time.

**To build a good model, we need to find a good balance between bias and variance such that it minimizes the total error.**

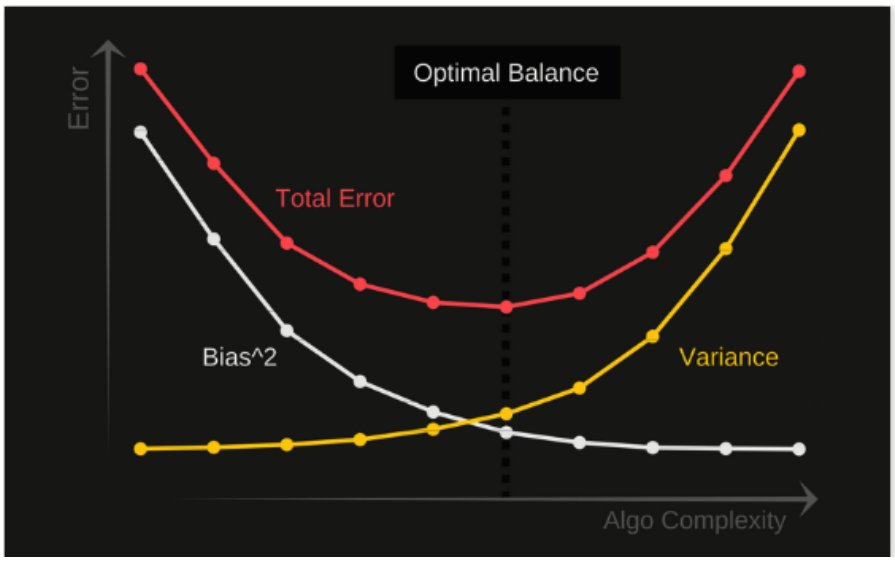



*Figure 19 BE-VE Trade-Off*

### 4.1.2 Model Validation using K-Fold

To achieve the best results different linear and non-linear models based on different scoring metrics namely: Accuracy, Roc_Auc, F1-Score were build using K-Fold Cross Validation, where the Number of Splits is considered as 7.

## 4.2 Decision Tree Classifier

Given data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data. The decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like a tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

**Advantages:**

- Simple, Fast in processing, and effective
- Does well with noisy data and missing data
- Handles numeric and categorical variables
- **Automatic Feature selection**: Irrelevant features won't affect decision trees

**Disadvantages:**

- Often biased towards splits or features have a large number of levels
- May not be optimum as modeling some relations on an axis parallel basis is not optimal
- Small changes in training data can result in large changes to the logic
- Large trees can be difficult to interpret

## 4.3 Random Forest Classifier

Random Forest is a trademarked term for an ensemble of decision trees. In Random Forest, we have a collection of decision trees (so-known as "Forest"). To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

**It works in four steps:**

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

**Advantages:**

- **Reduced error:** Random forest is an ensemble of decision trees. For predicting the outcome of a particular row, a random forest takes inputs from all the trees and then predicts the outcome. This ensures that the individual errors of trees are minimized, and overall variance and error are reduced.
- **Good Performance on Imbalanced datasets**: It can also handle errors in imbalanced data (one class is the majority and another class is the minority)
- **Good handling of missing data:** It can handle missing data very well. So if there is a large amount of missing data in your model, it will give good results.
- **No problem of overfitting:** Random forest considers only a subset of features, and the outcome depends on all the trees. So, there is more generalization and less overfitting.

- **Useful to extract feature importance** (we can use it for feature selection)

**Disadvantages:**

- Features need to have some predictive power else they won't work.
- Predictions of the trees need to be uncorrelated.
- Appears as Black Box: It is tough to know what is happening. We can at best try different parameters and random seeds to change the outcomes and performance.

## 4.4 Multinomial Naive Bayes

The general term **Naive Bayes** refers to the strong independence assumptions in the model, rather than the particular distribution of each feature. A Naive Bayes model assumes that each of the features it uses is conditionally independent of one another given some class. Naive Bayes is used in Text classification/ Spam Filtering/ Sentiment Analysis. It is used in text classification (it can predict multiple classes and doesn't mind dealing with irrelevant features).

Multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier that uses a multinomial distribution for each of the features. Multinomial Naïve Bayes consider a feature vector where a given term represents the number of times it appears or very often.

**Advantages:**

- Low computation cost.
- It can effectively work with large datasets.
- For small sample sizes, Naive Bayes can outperform the most powerful alternatives.
- Easy to implement, fast and accurate method of prediction.
- Can work with multiclass prediction problems.
- It performs well in text classification problems.

**Disadvantages:**

- Independence of features does not hold: The fundamental Naive Bayes assumption is that each feature makes an independent and equal contribution to the outcome. However, this condition is not met most of the time.
- 2. Bad estimator: Probability outputs from predict_proba are not to be taken too seriously.

## 4.5 Ensemble Techniques

Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model to **decrease variance** (bagging), **bias** (boosting) or **improve predictions** (stacking).

Ensemble methods can be divided into two groups:

- Bagging
- Boosting

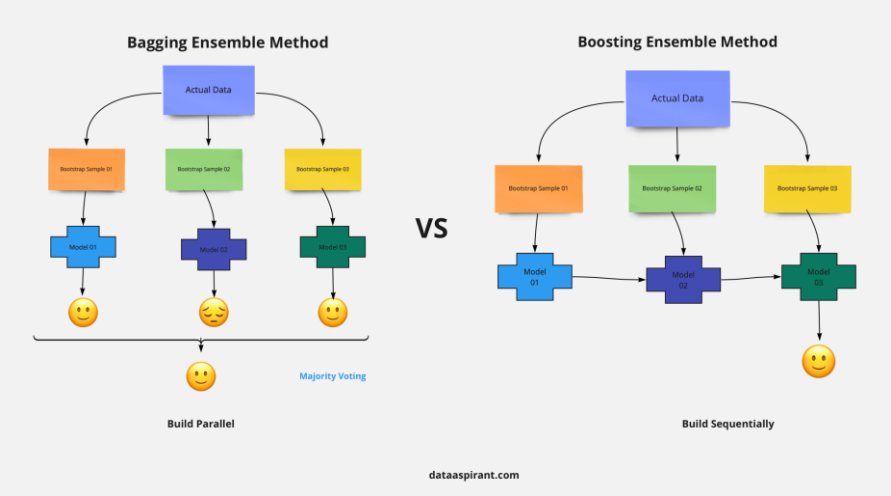



*Figure 20 Bagging vs Boosting*

### 4.5.1 Averaging method- Bagging (Bootstrap Aggregation):

The driving principle is to build several estimators independently and then to average / vote their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced. E.g. bagging methods, Forests of randomized trees**.**

- Designed to improve the stability (a small change in dataset change the model) and accuracy of classification and regression models
- It reduces variance errors and helps to avoid overfitting
- Can be used with any type of machine learning model, mostly used with Decision Tree
- .Uses sampling with replacement to generate multiple samples of a given size. The sample may contain repeat data points
- For a large sample size, sample data is expected to have roughly 63.2% ( 1 –1/e) unique data points and the rest being duplicates
- For classification, bagging is used with voting to decide the class of input while for regression average or median values are calculated.
- Multiple sample sets are created from the same data set using random function. Each sample data set is used to create a predictor

### 4.5.2 Boosting Method:

Base estimators are built sequentially, and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble. E.g. AdaBoost, Gradient Tree Boosting.

Below are different Boosting Algorithms, commonly used.

- **Adaptive Boosting (Ada Boost):**
  1. Similar to bagging, but the learners are grown sequentially; except for the first, each subsequent learner is grown from previously grown learners
  2. If the learner is a Decision Tree, each of the trees can be small, with just a few terminal nodes (determined by the parameter supplied).

3. During voting higher weight is given to the votes of learners who perform better in respective training data, unlike Bagging where all get equal weight.
4. Boosting slows down learning (because it is sequential) but the model generally performs well
5. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights to incorrectly classified instance. In AdaBoost, the successive learners are created with a focus on the ill-fitted data of the previous learner.

- **Gradient Boosting Algorithm:**

Gradient boosting, just like any other ensemble machine learning procedure, sequentially adds predictors to the ensemble and follows the sequence in correcting preceding predictors to arrive at an accurate predictor at the end of the procedure.

Gradient boosting aims at fitting a new predictor in the residual errors committed by the preceding predictor. It utilizes the gradient descent to pinpoint the challenges in the learners' predictions used previously. The previous error is highlighted, and, by combining one weak learner to the next learner, the error is reduced significantly over time.

- **XGBoost Algorithm:**

XGBoost stands for Extreme Gradient Boosting, another classic gradient boosting algorithm. The XGBoost has an immensely high predictive power which makes it the best choice for accuracy in events as it possesses both linear model and the tree learning algorithm, making the algorithm almost 10x faster than existing gradient booster techniques.

One of the most interesting things about the XGBoost is that it is also called a regularized boosting technique. This helps to reduce overfit modeling.

**Advantages of XGBoost Classifier:**

- Less feature engineering required (No need for scaling, normalizing data, can also handle missing values well)
- Feature importance can be found out (it output importance of each feature, can be used for feature selection)
- Fast to interpret
- Outliers have minimal impact.
- Handles large-sized datasets well and, has a good execution speed & model performance.

**Disadvantages of XGBoost Classifier:**

- Difficult interpretation, visualization tough.
- Overfitting possible if parameters are not tuned properly.
- Harder to tune as there are too many hyperparameters.

**Results obtained from K-Fold Cross-Validation**

| Metrix | Score | LR | DT | RF | Ada | GradB | XGB | MNB |
|---|---|---|---|---|---|---|---|---|
| Accuracy | BA | 0.9265 | 0.9311 | 0.9368 | 0.9180 | 0.9193 | 0.9407 | 0.8805 |
| | Coef_Var | 0.0098 | 0.0060 | 0.0100 | 0.0052 | 0.0092 | 0.0059 | 0.0074 |
| roc_auc | BA | 0.9864 | 0.9069 | 0.9853 | 0.9731 | 0.9763 | 0.9837 | 0.9823 |
| | Coef_Var | 0.0033 | 0.0091 | 0.0031 | 0.0027 | 0.0029 | 0.0028 | 0.0026 |
| f1 | BA | 0.9563 | 0.9556 | 0.9601 | 0.9471 | 0.9507 | 0.9620 | 0.9295 |
| | Coef_Var | 0.0043 | 0.0035 | 0.0026 | 0.0051 | 0.0054 | 0.0038 | 0.0043 |

*Table 4 K-FoldCV Results*

**Inference:** From the above, it can be inferred, for both Accuracy & F-1 Scoring metrics, XGB Classifier gives the best Bias Accuracy and Variance Error Trade-Off. Thus, choosing the XGB Classifier Model for evaluating our testing model.

**Note:** The above results were obtained using the default parameters for all seven models.

# Chapter 5

# Important Model Evaluation Metrics

Evaluating a model is a core part of building an effective machine learning model. There are several evaluation metrics for classification problems like F-1, Precision, Recall, Accuracy, and ROC-AUC Scoring. Different evaluation metrics are used for different kinds of problems. Evaluation metrics explain the performance of a model. An important aspect of evaluation metrics is their capability to discriminate among model results.

To get a proper understanding of F-1, Precision, Recall, and Accuracy, it's very important to understand the concept of the **Confusion Matrix.**

## 5.1 Confusion Matrix

**Definition:** A confusion matrix is an N X N matrix, where N is the number of classes being predicted. For the problem in hand, we have N=2, and hence we get a 2 X 2 matrix. Here are a few terms, we need to remember for a confusion matrix: [4]

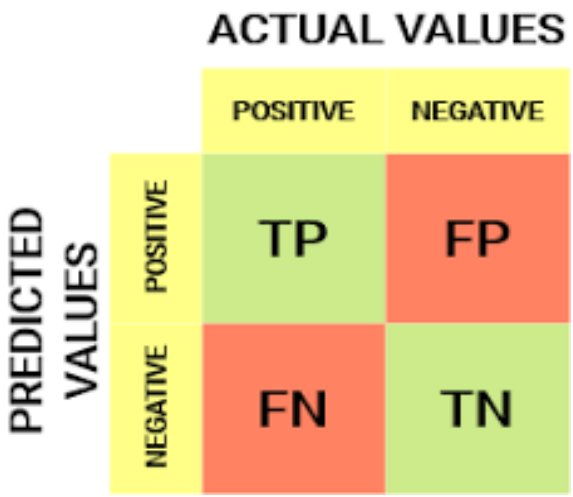



*Figure 21 Confusion Matrix*

Let's decipher the matrix:

- The target variable has two values: Positive or Negative
- The columns represent the actual values of the target variable
- The rows represent the predicted values of the target variable

**Understanding True Positive, True Negative, False Positive, and False Negative in a Confusion Matrix**

**True Positive (TP)**

- The predicted value matches the actual value.

- The actual value was positive, and the model predicted a positive value

**True Negative (TN)**

- The predicted value matches the actual value.
- The actual value was negative and the model predicted a negative value

**False Positive (FP) – Type 1 error**

- The predicted value was falsely predicted.
- The actual value was negative, but the model predicted a positive value

**False Negative (FN) – Type 2 error**

- The predicted value was falsely predicted.
- The actual value was positive, but the model predicted a negative value.

**Need for Confusion Matrix**

Most of the real-life datasets are imbalanced, and thus blindly relying on the most commonly used metrics which is Accuracy is not sufficient, as it might be misleading. Below are the few important terms we should know before deciding the evaluation metrics.

1. **Accuracy:** The accuracy of a machine learning classification algorithm is one way to measure how often the algorithm classifies a data point correctly. Accuracy is the number of correctly predicted data points out of all the data points. More formally, it is defined as the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives.

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

2. **Recall:** Recall tells us how many of the actual positive cases we were able to predict correctly with our model. To simplify this, when the actual value is positive how often his prediction correct. Also known as Sensitivity.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

3. **Precision:** Precision tells us how many of the correctly predicted cases turned out to be positive. To understand this in simpler terms, when a positive value is predicted, how often is prediction correct.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

4. **F-1 Score:** In practice, when we try to increase the precision of our model, the recall goes down, and vice-versa. The F1-score captures both the trends in a single value. F1-score is a harmonic mean of Precision and Recall, and so it gives a combined idea about these two metrics. It is maximum when Precision is equal to Recall.
   The interpretability of the F1-score is poor. This means that we don't know what our classifier is maximizing – precision or recall. So, we use it in combination with other evaluation metrics which gives us a complete picture of the result.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

## 5.2 Area under the ROC Curve (AUC- ROC):

**Definition:** This is again one of the popular metrics used in the industry. The biggest advantage of using ROC curve is that it is independent of the change in the proportion of responders. The ROC curve is the plot between sensitivity and (1- specificity). (1- specificity) is also known as false positive rate and sensitivity is also known as True Positive rate. [5]

**For instance:** The sensitivity at 0.5 thresholds is 99.6% and the (1-specificity) is ~60%. This coordinate becomes on point in our ROC curve. To bring this curve down to a single number, we find the area under this curve (AUC). Hence AUC itself is the ratio under the curve and the total area.

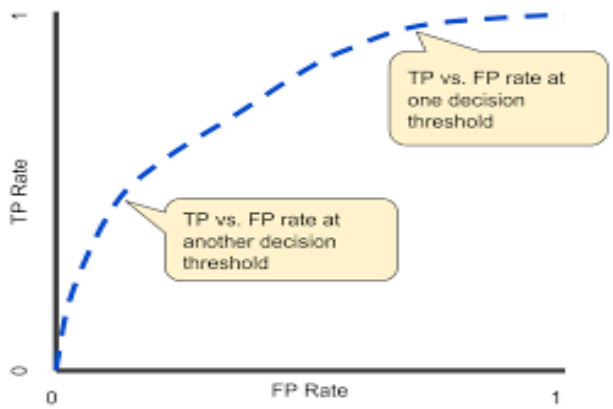



*Figure 22 ROC-AUC Curve*

# Chapter 6

# Hyperparameter Tuning

## 6.1 Need for hyperparameter tuning

Hyperparameters control the behavior of the algorithm that is used for modeling. [11]
Hyperparameters are passed in the arguments during the initialization of the algorithm.

The hyperparameters that are mostly used for XGBoost are n estimators, max depth, and learning rate. [14] Hyperparameters help us limit overfitting or underfitting of the model. They also aid in reducing the time taken to execute.

There are many hyperparameters for boosting.

### N – estimators:

- The number of sequential trees to be modeled.
- Though XGB is fairly robust at a higher number of trees it can still be overfitted at a point. Hence, this should be tuned using CV for a particular learning rate. [12]

### Max – depth:

- It gives the maximum depth of a tree.
- Used to control over-fitting as higher depth will allow the model to learn relations very specific to a particular sample. [12]

### Learning rate:

- This determines the impact of each tree on the final outcome. XGB works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates. [Ref 10]
- Lower values are generally preferred as they make the model robust to the specific characteristics of the tree and thus allowing it to generalize well. [Ref 10]
- Lower values would require a higher number of trees to model all the relations and will be computationally expensive. [12]

### Max Leaf Nodes:

- The maximum number of terminal nodes or leaves in a tree.
- Can be defined in place of max_depth. Since binary trees are created, a depth of ‘n’ would produce a maximum of 2^n leaves.
- If this is defined, XGB will ignore max_depth. [13]

### Booster [default=gbtree] [13]:

- Select the type of model to run at each iteration. It has 2 options:
    - gbtree: tree-based models
    - gblinear: linear models

### Max Delta Step [default=0]:

- In the maximum delta step, we allow each tree's weight estimation to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help to make the update step more conservative.
- Usually, this parameter is not needed, but it might help in logistic regression when the class is extremely imbalanced.
- This is generally not used.

### Reg alpha:

- L1 regularization term on weight (analogous to Lasso regression)
- Can be used in case of very high dimensionality so that the algorithm runs faster when implemented. [13]

### Reg lambda:

- L2 regularization term on weights (analogous to Ridge regression)
- This is used to handle the regularization part of XGBoost. Though many data scientists don't use it often, it could be explored to reduce overfitting. [13]

### Subsample [default=1]:

- Denotes the fraction of observations to be randomly selected samples for each tree.
- Lower values make the algorithm more conservative and prevent overfitting, but too small values might lead to under-fitting. [13]
- Typical values: 0.5-1

The Final model used was executed using max_depth, n_estimators, and learning_rate as it was widely used in XGBClassifier. These three parameters alone help reduce the time complexity and improve performance to an extent.

## 6.2 Selection of n_estimators

n_estimators is one of the most important hyperparameters for the extreme gradient boosting model. n_estimators is the number of trees to be used in the forest. Since XGBoost is an ensemble method comprising of creating multiple decision trees, this parameter is used to control the number of trees to be used in the process. [9]

It is important to choose the right number of trees/estimators that can learn about the data very well and thereby reducing the entropy and maximizing the information gain. As the number of trees increase, XGBoost algorithm works in a way that minimizes the error of the preceding tree.

Our approach is because GridSearchCV does not take into account Variance error while computing the best parameters for model tuning, therefore a manual looping is performed with 5 to 400 n_estimators to manually check the best bias accuracy and variance error trade-off.

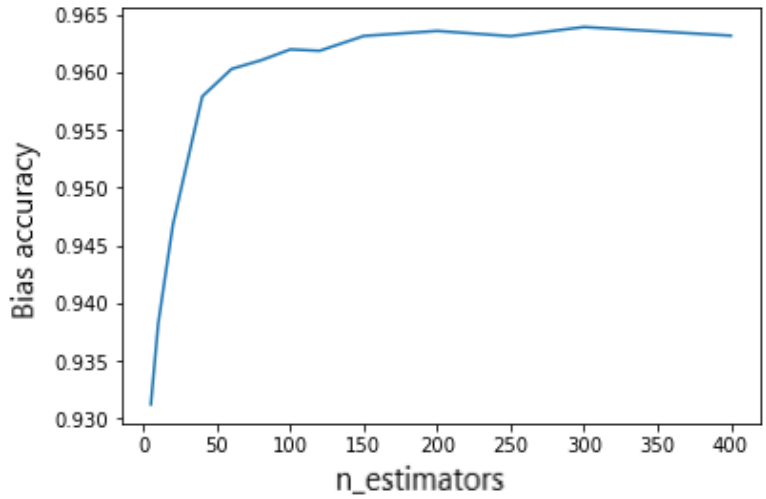



*Figure 23 n_estimators vs bias accuracy*

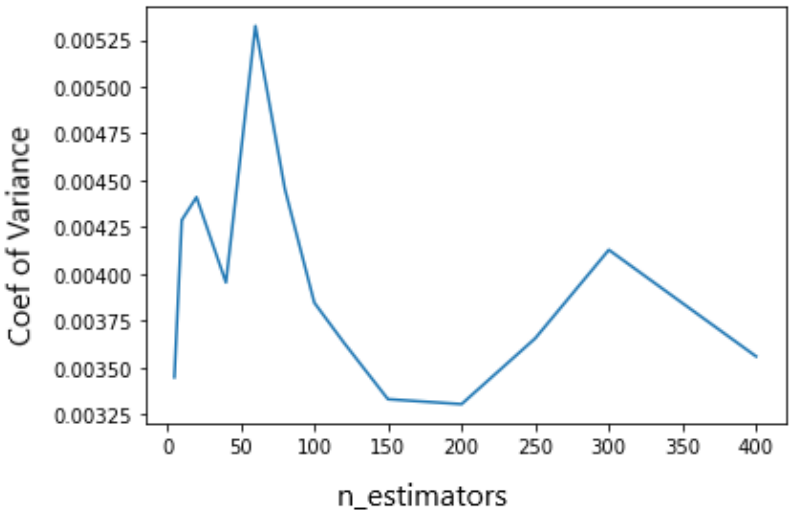



*Figure 24 n_estimators vs coef. of variance*

Comparing both the graphs above, the trade-off between bias accuracy and coefficient of variance can be performed. It's evident that n_estimators = 150 and 200, the coef. of variance reduces very well at 0.00333 and 0.0033. The bias accuracy at these points are 0.96313 and 0.96357 respectively. On choosing n_estimators as 200, the best trade-off between these two factors is observed.

## 6.3 GridSearchCV

Grid search is the process of performing hyperparameter tuning to determine the optimal values for a given model. This is significant as the performance of the entire model is based on the hyperparameter values specified.

The estimator parameter of GridSearchCV requires the models that are used for the hyperparameter tuning process. For this example, let's use the kernel of the XGBoost algorithm. The param_grid parameter requires a list of parameters and the range of values for each parameter of the specified

estimator. The most significant parameters required when working with the kernel of the XGB model are n_estimators, max_depth, and learning_rate. A list of values to choose from should be given to each hyperparameter of the model. You can change these values and experiment more to see which value ranges give better performance. A cross-validation process is performed in order to determine the hyperparameter value set which provides the best accuracy levels. [10]

For determining max_depth and learning rate for our model using GridSearchCV, let's use the n_estimators as 200 that is determined in the previous step. A range of values are chosen for max_depth [10,15,20,25,30,35] and learning_rate [0.01, 0.1, 1, 10, 100].

Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. 0 is only accepted in loss guided growing policy when tree_method is set as hist and it indicates no limit on depth. Eta, also known as the learning rate: Step size shrinkage used in the update to prevents overfitting. After each boosting step, one can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.

The following plot is the outcome of all the possible combination's variance error and bias accuracy.

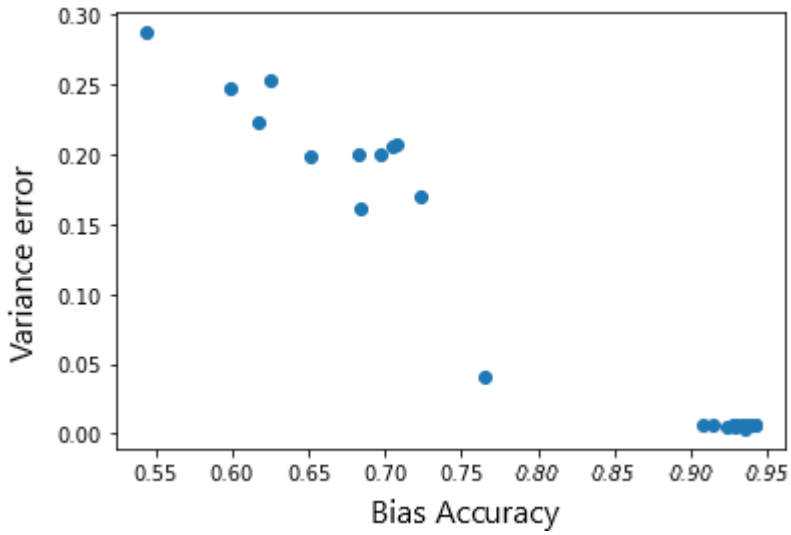



*Figure 25 GridSearchCV results*

From the above plot, it is evident that the zone at the extreme right bottom is the area to focus, i.e. having maximum bias accuracy and least variance error. Using filters, that particular zone is analyzed and the best combination is chosen based on the bias accuracy and variance error trade-off.

The chosen hyperparameters values are:

*max_depth : 15*

*learning_rate = 0.1*

# Chapter 7

# Applying Tuned Model on the Training Data

## 7.1 Passing the training data

After selecting the optimum hyperparameters using GridSearchCV, the tuned model is ready to be subjected to the 10,000 records of training data. The training data comprises data with dynamic sentiments, this helps in training the model in different verticals to predict the sentiments.

Let us look at the model's performance with the training data.

```
Accuracy:  0.9816805308907375
F1:  0.9883277751310149
Precision: 0.9960388908894491
Recall: 0.9807351376905803
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.99 | 0.96 | 2238 |
| 1 | 1.00 | 0.98 | 0.99 | 8461 |
| accuracy | | | 0.98 | 10699 |
| macro avg | 0.96 | 0.98 | 0.97 | 10699 |
| weighted avg | 0.98 | 0.98 | 0.98 | 10699 |

*Figure 26 Training Model Evaluation Matrices*

Looking into the above scoring matrices, F1, Precision, Accuracy, and Recall are explained in the above sections of the report. These matrices are all above 98% thanks to the hyperparameters and the Extreme Gradient Boosting algorithm, our model is well trained on the data.

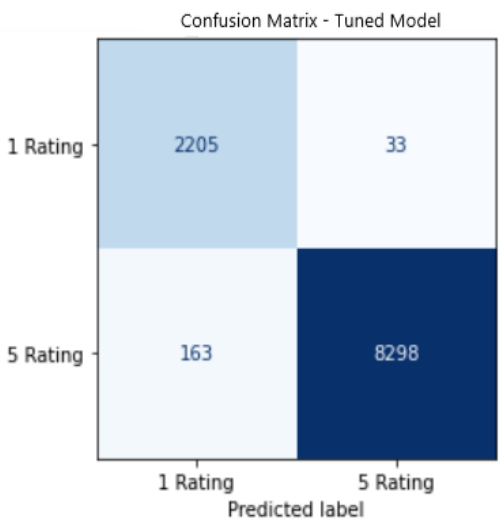



*Figure 28 Confusion Matrix - Trained Model*

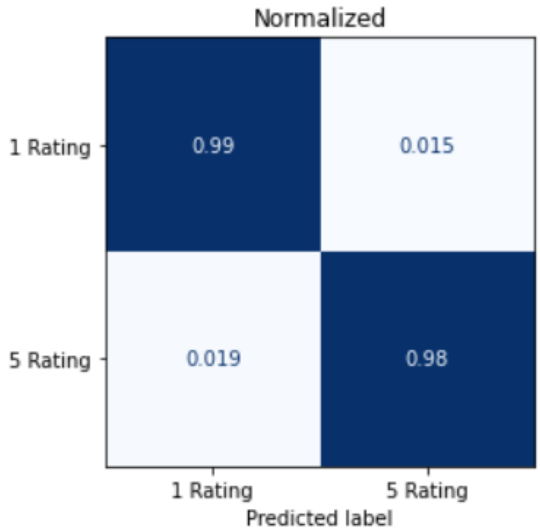



*Figure 27 Normalized CF - Trained Model*

The above two confusion matrix talk about our model's prediction and detection. The matrix at the left says that out of 2238 1-Rating Reviews, 2205 are correctly predicted (99%) and 33 (0.015%) are wrongly predicted. Out of 8461 5-Star Reviews, 8298 (98%) are correctly predicted whereas 163 (0.019%) are wrongly predicted. These wrong predictions are because reviews invoke sarcasm, praising, or degrading other items or products and are meaningless comments. These sentiments are hard for a machine to comprehend and therefore fail to classify them correctly based on a positive or negative review.

## 7.2 Preparing the Test Data

In the preceding topics, the reduction of test data size from 67k records to 40k records is seen, and therefore now the testing data has 40885 records. As the scope of the project is to detect an anomaly, an anomaly column is required that says if the rating given by a user does not match with the review written by him/her, only then our model can be evaluated and one could see if the model can detect anomalies. For this reason, a new column with manual labeling is required. Manual labeling comprises reading the review text and corresponding rating and determining whether the records are non-anomaly or anomaly. Manual work can be relied upon here as the complex sentiments are easily comprehendible by humans. This manually labeled will be taken as y_true and y_pred (anomaly) will be based on the model's output y_pred (1 or 5 star) and reviews.rating feature.

## 7.3 Model Anomaly calculation

As the model predicts whether a review comment is positive or negative (5 or 1-star equivalent), and does not directly predict if a record has an anomaly, a simple feature engineering is done to get this anomaly prediction on each record.

*df_result['Mod_Anomaly'] = np.where((df_result['reviews.rating'] == df_result['Model_Pred']),0,1)*

The above line of code will impute whether the record is an anomaly or not. In case the reviews.rating feature does not match with the model's y_pred (sentiment), it is an anomaly and '1' will be imputed in these columns for this particular record. In the other case, where reviews.rating matches with the model's y_pred (sentiment), '0' will be imputed – as it is not an anomaly.

# Chapter 8

# Amazon dataset (Test data) with the Trained Model

## 8.1 Need for Sampling

It is established that manual labeling is required to evaluate the model, to see how good the model is in predicting and not predicting the anomaly. For this reason a y_true (anomaly) – in actuality it is an anomaly is required to compare with the model's prediction. It'd be ideal to have this y_true (anomaly) for all the 40,885 records, however, it's a tedious process to manually label these entire records. Therefore, a sample is taken from 40.8k records to evaluate the model and manual labeling is also performed for this sample. Now, there is y_true (anomaly) for comparison.

**Selection of Sampling size and Sampling data**

To select the sample size, let's understand the following parameters:

**Margin of error**

The margin of error is the level of precision you require. This is the plus or minus number that is often reported with an estimated proportion and is also called the confidence interval. It is the range in which the true population proportion is estimated to be and is often expressed in percentage points (e.g., ±2%). Note that the actual precision achieved after you collect your data will be more or less than this target amount because it will be based on the proportion estimated from the data and not your expected sample proportion.

**Confidence level**

The confidence level is the probability that the margin of error contains the true proportion. If the study were repeated and the range calculated each time, you would expect the true value to lie within these ranges on 95% of occasions. The higher the confidence level the more certain you can be that the interval contains the true proportion.

**Population size**

This is the total number of distinct individuals in your population. In this formula, one can use a finite population correction to account for sampling from small populations.

**Sample proportion**

The sample proportion is what you expect the results to be. This can often be determined by using the results from a previous survey, or by running a small pilot study. If you are unsure, use 50%, which is conservative, and gives the largest sample size. Note that this sample size calculation uses the Normal approximation to the Binomial distribution. If the sample proportion is close to 0 or 1 then this approximation is not valid and you need to consider an alternative sample size calculation method.

**Sample size**

This is the minimum sample size you need to estimate the true population proportion with the required margin of error and confidence level. Note that if some people choose not to respond they cannot be included in your sample and so if non-response is a possibility your sample size will have to be increased accordingly. In general, the higher the response rate the better the estimate, as non-response will often lead to biases in your estimate. [7]

**Formula**

This calculator uses the following formula for the sample size n:

$n = N*X / (X + N - 1)$,

where,

$X = Z_{\alpha/2}^2 * p*(1-p) / MOE^2$,

and $Z_{\alpha/2}$ is the critical value of the Normal distribution at $\alpha/2$ (e.g. for a confidence level of 95%, $\alpha$ is 0.05 and the critical value is 1.96), MOE is the margin of error, p is the sample proportion, and N is the population size. Note that a Finite Population Correction has been applied to the sample size formula. [ref]

Data for estimating the sample size for our problem:

The accepted margin of error: 5%

Confidence level: 95%

Population size: 40,885

Sample proportion: 10%

Estimated sample size = 138

138 is the minimum sample size that must be taken, however, a sample size of 200 records is taken for positive and 200 for negative records to be on the safer side.

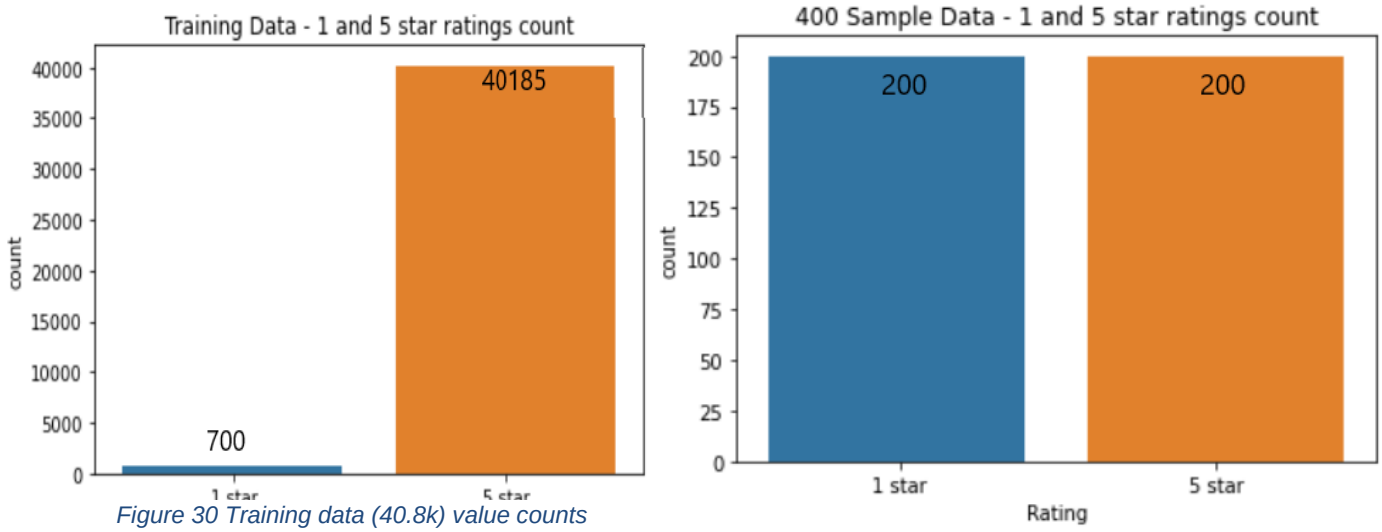



*Figure 30 Training data (40.8k) value counts*

From the above two plots, one can see the reduction in the number of positive and negative records. For 5-star records, the reduction is done with a factor of 1:200 and for 1-star, 1:3.5. The reason for such a huge difference in the proportion factors is the imbalance in the data. Only about 1.712% of the test data have negative (1-star) records and the rest – 98.28% are positive (5-star) records.

## 8.2 Final Model on the Test set

By applying the best model on the sample of 400 records from the test set of approx., 40,000 records, we get a predicted sentiment of the review.

This predicted sentiment is calculated with a threshold of 0.5. This predicted value is compared with the actual rating provided by the user. The results of the comparison are as below:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.94 | 0.80 | 0.86 | 200 |
| 1 | 0.82 | 0.95 | 0.88 | 200 |
| accuracy | | | 0.87 | 400 |
| macro avg | 0.88 | 0.87 | 0.87 | 400 |
| weighted avg | 0.88 | 0.87 | 0.87 | 400 |

The model gives an accuracy of 87. We can improve this performance by varying the threshold for sentiment analysis. The average F1 Score and accuracy are the same since the data is balanced.

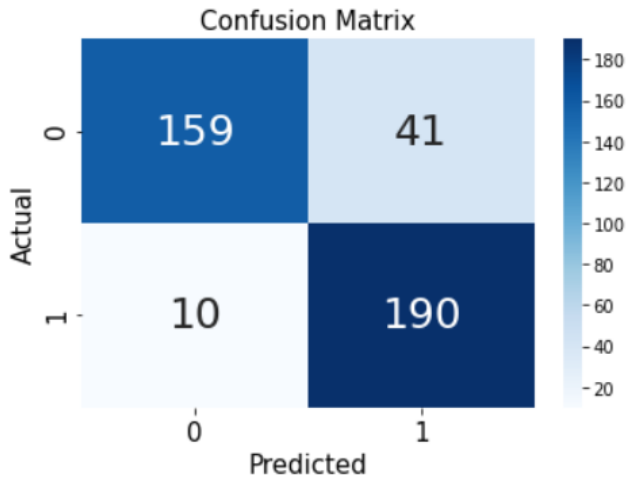



*Figure 31 Confusion Matrix - Sentiment Prediction*

- The model predicts 51 values with a different value from the user rating. These are anomalies according to the model. But we can not evaluate how well the model performs in detecting the anomalies as there are no actual values for this.
- Let's look at the evaluation method for the prediction of anomalies in the upcoming section.

**Evaluation:**

- A new column created to label if the model predictions match with the user rating or not. Those which do not match are labeled 1 and that which match is labeled 0.
- To compare the labeled values with the actual we need to manually create the anomaly or not column.
- If the sentiment of the statement is 100% accurate then we need not label them manually. But there is no metric to check the accuracy of the sentiment analysis except the human mind.
- This manual verification is done on a smaller sample from the 40k records since it is not possible to verify the whole dataset.
- By taking a sample size as explained earlier a new column is recorded whether the record is an anomaly or not.
- This manually labeled column and the model prediction as anomalies are compared and this performs the model in identifying the anomalies.

**Example of an Anomaly (Label created):**

| Review | Rating by user (0 or 1) | Anomaly |
|---|---|---|
| She uses it every day. She cannot get enough of it. | 0 | 1 |
| I got this for my kid because it is tailored for kids. They liked it and it worked. | 1 | 0 |

*Table. Anomaly Labelling: Filling the anomaly column by checking whether the sentiment of the statement is matching with the rating provided was done in the above table.*

The count of actual anomalies is really low compared to non-anomalies. This is expected since anomalies are usually very low in number. That is one of the reasons it is hard to identify.

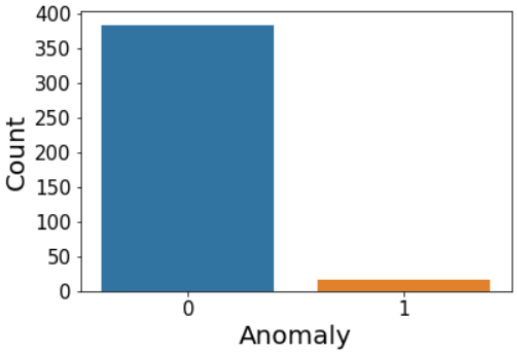



*Figure 32 Count of True Anomalies by manual labelling*

## 8.3 Anomaly Detection Performance evaluation

Performance of the model in identifying anomalies:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.91 | 0.95 | 384 |
| 1 | 0.31 | 1.00 | 0.48 | 16 |
| accuracy | | | 0.91 | 400 |
| macro avg | 0.66 | 0.95 | 0.71 | 400 |
| weighted avg | 0.97 | 0.91 | 0.93 | 400 |

The model produces a weighted average F1 score of 93. Here we consider the weighted average as there is an imbalance in data.

If the data was balanced the macro average and weighted average would be the same.

We can see from the confusion matrix that all 16 anomalies were predicted correctly by the model.

But the model identifies a few non-anomalies as anomalies this may be due to the sentiment of the statement being ambiguous to model due to sarcasm, short reviews, neutral reviews, or too elaborate reviews or mixed sentiment reviews.

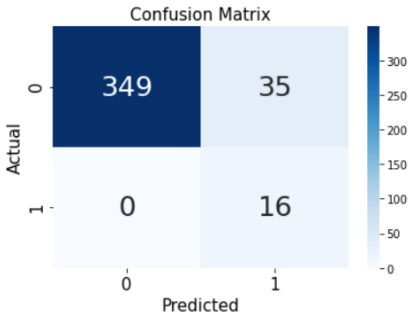



*Figure 33 Confusion Matrix - Anomaly prediction*

False Positive (35) is high but False Negative (0) is nil.

FP only costs a message that is sent to the user whereas FN affects the users' trust on the website as it directly affects the ratings.

Hence in this particular scenario, FN is costlier.

Hence, we can send a text message or an email to the user to inform them about the difference between their rating and review thus causing them to change anyone to match both. Thus, the rating will be true to users' experience.

## 8.4 Threshold Tuning

Classification predictive modelling typically involves predicting a class label [15].

Nevertheless, many machine learning algorithms are capable of predicting a probability or scoring of class membership, and this must be interpreted before it can be mapped to a crisp class label. This is achieved by using a threshold, such as 0.5, where all values equal or greater than the threshold are mapped to one class and all other values are mapped to another class.

For those classification problems that have a severe class imbalance, the default threshold can result in poor performance. As such, a simple and straightforward approach to improving the performance of a classifier that predicts probabilities on an imbalanced classification problem is to tune the threshold used to map probabilities to class labels.

In some cases, such as when using ROC Curves and Precision-Recall Curves, the best or optimal threshold for the classifier can be calculated directly. In other cases, it is possible to use a grid search to tune the threshold and locate the optimal value.

Here we have used the grid search method to tune the model with respect to the F1 score. In this case, we can define a set of thresholds and then evaluate predicted probabilities under each in order to find and select the optimal threshold.

The scoring metric considered is weighted recall and a hypothesis is considered where the expected score is 0.95 (since the maximum weighted recall score achieved was close to 0.95 by varying threshold).

Weighted recall is considered as False Negative are costlier than False positives. Since recall is ratio between TP and sum of TP and FN, it will tell what percentage of the class does the model predict correctly.

Recall:

TP/(FN+TP) for 1 class

TN/(FP+TN) for 0 class

Precision on the other hand will give how well the model is able to avoid errors. How much of the predictions are correct? It will how minimal the False Positives are with respect to True Positives.

Precision:

TP/(TP+FP) for 1 class

TN/(TN+FN) for 0 class

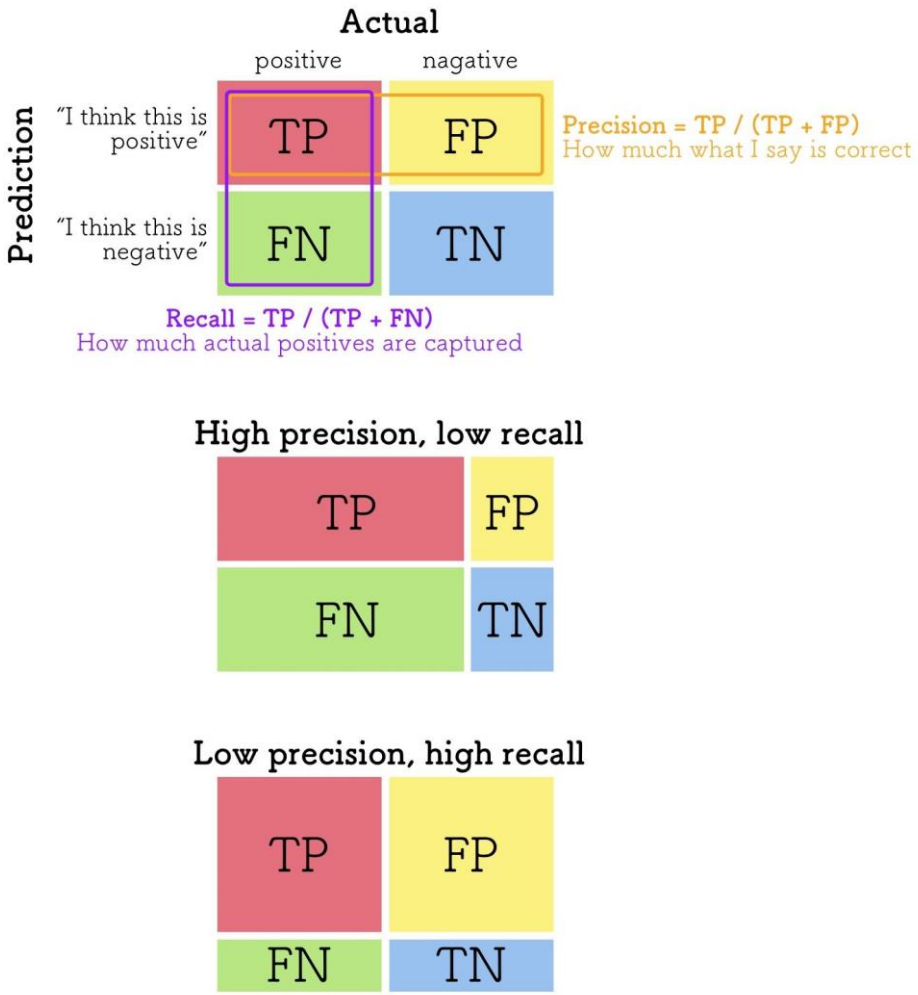



*Figure 34 Confusion Matrix - Weightage*

Since we are concerned about False Negative we go for recall. Weighted recall is chosen because there is imbalance in the classes (Anomalies and Non-Anomalies).

**Hypotheses:**

*Null hypothesis*: Weighted Recall = 0.95

*Alternate Hypothesis*: Weighted Recall ≠ 0.95

For this analysis, the significance level is 0.05. The test method, shown in the next section, is a one-sample z-test.

Specifically, the approach is appropriate because the sampling method was simple random sampling, the sample included at least 10 successes and 10 failures, and the population size was at least 10 times the sample size.

The threshold values tested were from 0.2 to 0.85 with step size of 0.05.
The best weighted recall was achieved at 0.25 where the False positive dropped from 35 to 25.
The P-value was accepting null hypotheses at thresholds 0.2, 0.25 and 0.4.

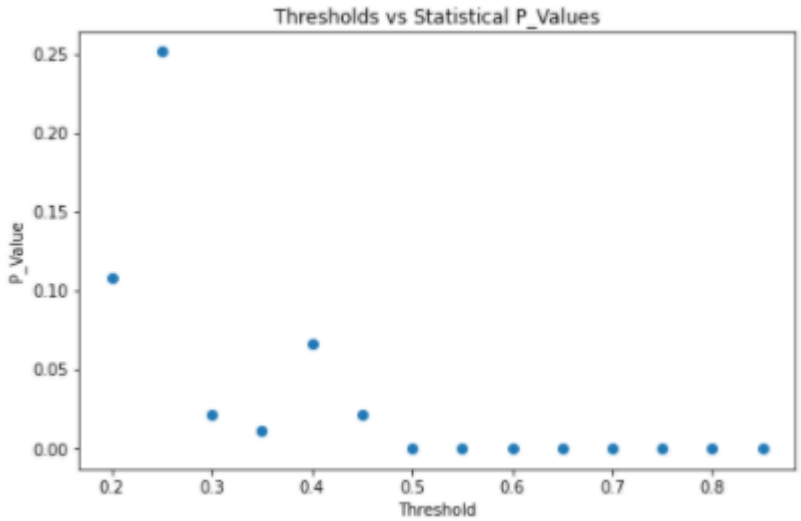



*Figure 35 Threshold tuning*

In order for the threshold to be unbiased on all kinds of test data it needs to be checked with the test data.
For thresholds 0.2 and 0.25 the predictions were biased towards the negative sentiment hence got high false negative.

```
The report for Th =  0.2
[[2230    8]
 [ 573 7888]]
```

```
The report for Th =  0.25
[[2230    8]
 [ 489 7972]]
```

For threshold 0.4, the sentiment analysis is more unbiased, and a good trade-off is achieved between false positive and false negative.

```
The report for Th =  0.39999999999999997
[[2220   18]
 [ 264 8197]]
```

Hence, 0.4 was chosen as the ideal threshold value for sentiment analysis.

**Hypothesis testing with threshold 0.4:**

Hypotheses:

*Null hypothesis*: Weighted Recall = 0.95

*Alternate Hypothesis*: Weighted Recall ≠ 0.95

Using sample data, we calculate the standard deviation (σ) and compute the z-score test statistic (z).
P = 0.95 (the expected weighted recall)
n = 400 (sample size)
p = 0.93 (actual weighted recall)

$$\sigma = \text{sqrt}[\ P * (\ 1 - P\ ) / n\ ]$$

$$\sigma = \text{sqrt}\ [(0.95 * 0.05) / 400]$$

$$\sigma = \text{sqrt}(0.0016) = 0.01$$

$$z = (p - P) / \sigma = (.93 - .95)/0.01 = -2$$

where P is the hypothesized value of population proportion in the null hypothesis, p is the sample proportion, and n is the sample size [16].

The P-value was then calculated using stats.norm.cdf(z)*2 since it is a two tailed test.

The resulting P-value is 0.06 which is greater than 0.05 hence it is concluded that we can accept the null hypothesis that weighted recall is equal to 0.95.

The 95% confidence interval for the weighted recall is 0.9286 – 0.9713.

## 8.5 Results after threshold tuning:

Following are the results of Anomaly detection on the 400-sample data with threshold as 0.4.

```
Positive class scores:

Threshold: 0.4                          Threshold: 0.49999999999999994
F1:  0.5333333333333333                 F1:  0.47761194029850745
Precision: 0.36363636363636365         Precision: 0.3137254901960784
Recall: 1.0                             Recall: 1.0

Confusion Matrix:                       Confusion Matrix:
 [[356  28]                              [[349  35]
 [  0  16]]                              [  0  16]]
```

Hence, we can see that the False positive has decreased from 35 to 28 by tuning the threshold from 0.5 to 0.4.

```
Classification Report (0.4):
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.93 | 0.96 | 384 |
| 1 | 0.36 | 1.00 | 0.53 | 16 |
| accuracy | | | 0.93 | 400 |
| macro avg | 0.68 | 0.96 | 0.75 | 400 |
| weighted avg | 0.97 | 0.93 | 0.95 | 400 |

After applying the hypothesis testing, following was the P-value and weighted recall:

```
P_value:  0.06645742001693253
weighted_recall:  0.93
```

We have been able to reduce the False positive without affecting the false negative by varying the threshold for classification.

This reduces cost wasted on falsely alerting consumers of their review.

## 8.6 Analyzing False Positives and False Negatives

### Examples of False Positives Anomalies

After the threshold tuning it is observed that False Positives has been significantly reduced from 35 to 28 in number without compromising on False Negatives, as its still zero.

From the 28 cases, below are the few examples of False Positives for both Positive & Negative Sentiments.

### Examples of Positive Sentiments

| | reviews.rating | reviews.text | reviews.title | lem_stpwrd | Anomaly | Model_Pred | Probab_0 | Mod_Anomaly |
|---|---|---|---|---|---|---|---|---|
| 47 | 1 | Glad I ordered this many. I had no idea I'd use nearly half the box as soon as I opened it. Now I need to do the same with AA size as many applications say specifically not to use rechargeables. Believe them! I tried it once before and pretty much melted down the item and destroyed the batteries as well. Apparently they are not made to supply as much current and they will let out the (magic) smoke. These came in quickly and are just what I needed | Quick, quality, quantity. Happy customer here. | quick quality quantity happy customer glad ordered many no idea i'd use nearly half box soon opened need do size many application say specifically not use rechargeables believe tried before pretty much melted item destroyed battery well apparently not made supply much current let magic smoke came quickly needed | 0 | 0 | 0.925889 | 1 |
| 84 | 1 | Great batteries here! These last just as long as Duracell or Energizer, but you get way more for your money! I mostly use these for video game controllers and remote controls. They last as long as I expect and I have never had a bad (dead) one. I don't understand why anyone would pay more for the expensive brands. These are NOT dollar store batteries. High quality and durability with a price that lets me always have spares ready to go! Since I found these I have been ordering them over and over in multiple sizes. | High Quality, Lasts As Long As Energizer/Duracell!! | high quality last long energy duracell great battery last long duracell energy but get way money mostly use video game controller remote control last long expect never bad dead one don't understand anyone would pay expensive band not dollar store battery high quality disability price let always spare ready go since found ordering multiple size | 0 | 0 | 0.996977 | 1 |
| 148 | 1 | I love my Amazon Fire HD 8. It's not too small and not too big. I was looking at other tablets, and decided on the Amazon Fire HD 8 because of the price, and because I have Amazon Prime, I purchased it online at Best Buy, and then picked it up at the store on Black Friday. It is my first tablet, and so I wanted to start with one that I can have fun with, and I am. I do have a laptop for important things. | First Tablet | first tablet love amazon fire 8 not small not big wa looking tablet decided amazon fire 8 price amazon prime purchased online best buy picked store black friday first tablet wanted start one fun do lawton important thing | 0 | 0 | 0.416825 | 1 |

*Figure 36 Positive Reviews - Model prediction*

In reality these sentiments are positive (5 rating one's), however the model has predicted them as negative (0 rating) because of the extreme negative Probab_0 Scores, highlighted above.

### WordCloud- Positive Sentiments having Negative Words

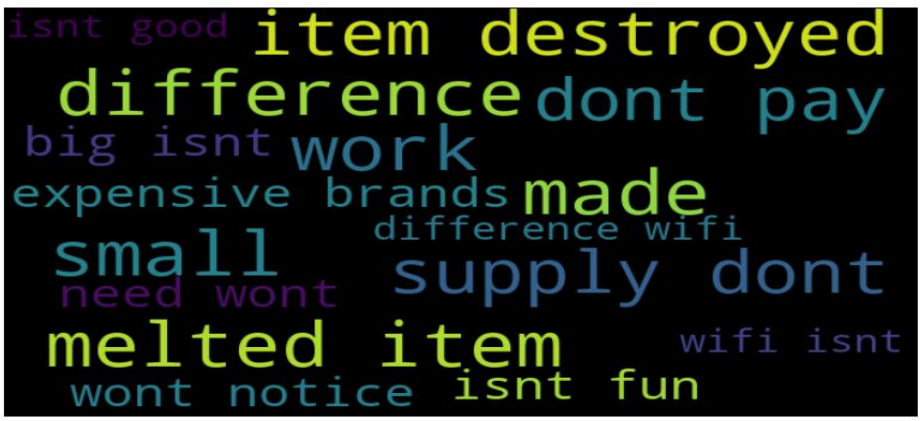



*Figure 37 WordCloud - Positive*

**Observation:**

Above are the few words used in the Positive Sentiments, due to which the Probab_0 Score is extremely negative. Thus the actual positive review rating (i.e. Rating 5) does not match with the Model_Pred rating (i.e. Rating 0), hence resulting in Anomalies.

**Examples of Negative Sentiments**

| | reviews.rating | reviews.text | reviews.title | lem_stpwrd | Anomaly | Model_Pred | Probab_0 | Mod_Anomaly |
|---|---|---|---|---|---|---|---|---|
| 210 | 0 | Got this during Christmas rush and didn't think much of it since it was slightly open. (They had none in stock and needed 4 of them for all my children ) upon opening item , one had a small crack (bearly noticeable ). Increased useage of item cause the screen to spider web... tried to return but no help with management .... store at fountains in El Paso Texas . So I have 3 perfect kindles and one cracked .... very disappointed 1 star | Open item | open item got christmas rush didn't think much since wa slightly open none stock needed 4 child upon opening item one small crack early noticeable increased usage item cause screen spider web tried return but no help management store mountain el past texas 3 perfect kindled one cracked disappointed 1 star | 0 | 1 | 0.260108 | 1 |
| 212 | 0 | I went through all 48 in less than 2 weeks. I can't lie and say they were awesome, when they were not. | Terrible | terrible went 48 le 2 week can't lie say awesome not | 0 | 1 | 0.034157 | 1 |
| 385 | 0 | We started out great. We had a great thing going. I loved her, she was easy on the eyes, and looked great in any light. Then 7 months into it, I could no longer turn her on. I tried the reboot with no luck. She is still under warranty but that doesn't matter because Best Buy's return policy stinks. If I do try to replace her, I will not buy it here. | Started out great | started great started great great thing going loved wa easy eye looked great light 7 month could no longer turn tried report no luck still warranty but doesn't matter best buy's return policy stick do try replace not buy | 0 | 1 | 0.034362 | 1 |

*Figure 38 Negative reviews - Model prediction*

In reality these sentiments are negative (0 rating one's), however the model has predicted them as positive (5 rating) because of the extreme positive Probab_0 Scores, highlighted above.

**WordCloud- Negative Sentiments having Positive Words**

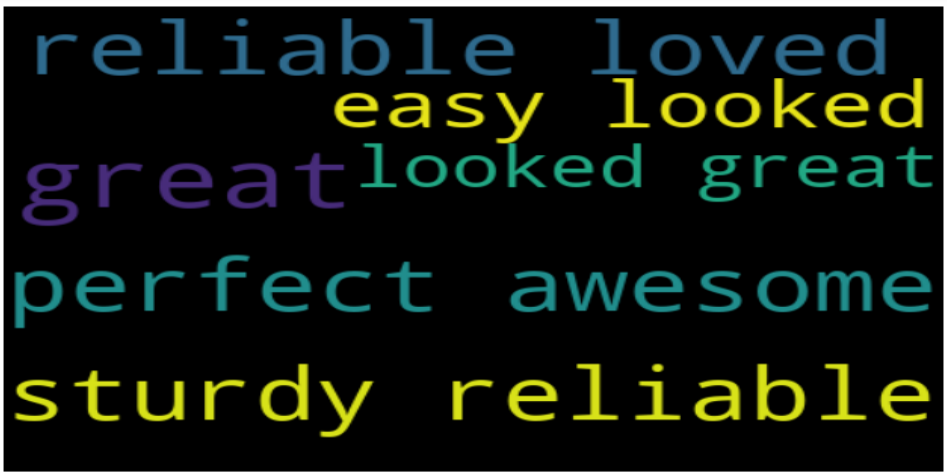



*Figure 39 WordCloud - Negative*

**Observation:**

Above are the few words used in the Negative Sentiments, due to which the Probab_0 Score is extremely positive. Thus the actual negative review rating (i.e. Rating 0) does not match with the Model_Pred rating (i.e. Rating 5), hence resulting in Anomalies.

Thus from above we can clearly say that these 28 records probab_0 scores are extreme towards positive or negative sentiments, therefore they do not alter on varying different thresholds (sentiment).

**Conclusion & Business Recommendation**

Predicting the anomalies in review and ratings for the Amazon Products which formed different classification models such as Logistical Regression, Decision Tree, Random Forest, Ada Boost, XGB Classifier, Gradient Boosting, Multinomial Naïve Bayes, the accuracy, f-1 scores for all the models are taken and compared. In order to get the best bias and variance error trade-off K-Fold Cross Validation was done. After comparing the BE and VE, XGB Classifier gave the best result in terms of Bias variance trade off. The scoring for XGB Classifier is then improved with the help of hyperparamter tuning.

By applying the best model on the sample of 400 records from the test set of approx., 40,000 records, we got a predicted sentiment of the review. This predicted sentiment is calculated with a threshold of 0.5, and is compared with the actual rating provided by the user.

After manually checking the anomalies by comparing the review with rating, a new column was created which acted as actual Y in terms of anomalies detection. Finally the anomalies predicted by model was calculated by comparing rating with the predicted sentiment.

A significant improvement was observed after threshold tuning for the XGB Classifier Model in order to detect and predict anomalies. Different threshold values were checked, calculating the p-values for respective threshold. The optimal threshold value is 0.4 with the p-value as 0.0664 which gave the weighted recall and F-1 score as 93% and 95% respectively.

The aim is to maximize the recall score, by reducing the number of FN, following are the key observations noted after threshold tuning:

- False Negatives became zero, resulting in 100% recall score.
- With 0.4 threshold False Positives reduced to 28 and the p-value got increased from 0.0005 to 0.0664.
- Having the p-value greater than 0.05, it can be said that the Null hypothesis stating that model is 95% accurate is accepted as the weighted recall of 93% lies in the Confidence interval of 95%.

**Business Recommendation:**

The objective is: "**Detection of Anomalies in Review & Ratings for Amazon Products**". Thus it came to know that False Negatives (FN) are much more costlier than the False Positives (FP). FP only costs a message that is sent to the user whereas FN affects the users' trust on the website as it directly affects the ratings.

So to the client, recommendation can be to send a text message/email to the user to inform them about the difference between their rating and review thus allowing them to change their rating or review. Hence, the rating will be true to users' experience.

# REFERENCES


[1] https://en.wikipedia.org/wiki/Amazon_(company)

[2] https://blog.3dcart.com/blog/product-reviews-increase-sales

[3] https://codesachin.wordpress.com/2015/08/30/cross-validation-and-the-bias-variance-tradeoff-for-dummies/

[4] https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/

[5] https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/

[6] https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/

[7] https://select-statistics.co.uk/calculators/sample-size-calculator-population-proportion

[8] Daniel WW (1999). Biostatistics: A Foundation for Analysis in the Health Sciences. 7th edition. New York: John Wiley & Sons.

[9] https://stackoverflow.com/questions/46234806/what-n-estimators-and-max-features-means-in-randomforestregressor

[10] https://medium.com/datadriveninvestor/an-introduction-to-grid-search-ff57adcc0998

[11] https://www.analyticssteps.com/blogs/introduction-model-hyperparameter-and-tuning-machine-learning

[12] https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/

[13] https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[14] https://www.datasciencelearner.com/gradient-boosting-hyperparameters-tuning/

[15] https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/

[16] https://stattrek.com/hypothesis-test/proportion.aspx