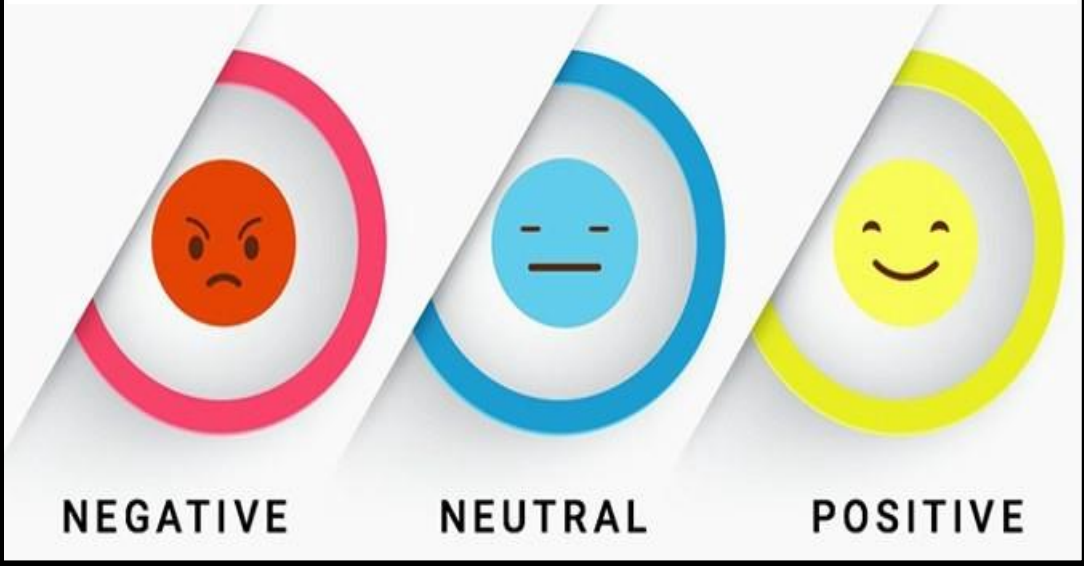
Sentiment Analysis of Google Playstore App reviews



By: **Saumya Jaiswal**

Independent Study (MKTG 899)

Under the supervision of

Prof. Ian Clark S Sinapuelas

TABLE OF CONTENTS

INTRODUCTION

The Internet has transformed how we use Android applications. In the fast-paced world of online applications, we have a wealth of alternatives to choose from. As a result, users must rely heavily on application reviews to help them make better decisions about using any new application. However, consumers may find it difficult to look for and compare text reviews.

Sentiment analysis is the categorizing of a user's evaluations or remarks as good, negative, or neutral. Most organizations examine their consumers' sentiments about their products or services to find out what their customers want from them. Google play store includes millions of applications with their reviews; thus, it will be an excellent use case of sentiment analysis to assess the perceived sentiment of apps accessible on the google play store. So, this report will focus on the process of leveraging automation using software (Python) to analyze Google Play Store applications' sentiment.

Goal

Application developers desire to locate useful reviews as fast as possible using rating system during their decision-making process. Therefore, algorithms that can estimate the user rating from the text review are extremely significant. Getting a holistic perspective of a textual evaluation might in turn improve user experience. It may also assist developers grow sales and enhance products by understanding customer needs and wants. The user's reviews and ratings for various applications, as well as feedback on the user's experience with the app will be considered.

The major purpose for this project is to construct a model to predict whether a user liked an App or not based on their reviews?

The intention is to take the raw customer reviews and perform sentiment analysis on them to produce the sentiment score, which involved changing the string data type to a numerical data type and then feeding those sentiment scores into the classification model for prediction.

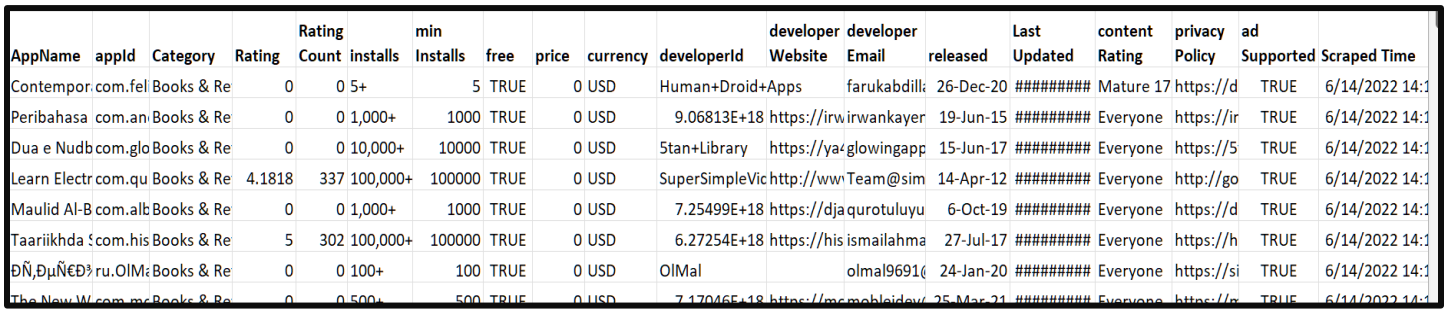
DATA COLLECTION

The data for this project was collected through methodologies employing web scrapping. For this study, a meta dataset comprising reviews and application details from the Google Play store was gathered. This dataset contains ratings, text, helpfulness votes, and application descriptions, as well as category information, price, developer details, and image attributes.

To scrape the real-time data, the Python library "Beautiful Soup" was utilized. Details on 797101 applications were recovered, with a total data size of 1.17 Gb. This metadata had the following columns:

Column	Details	Column	Details
App Name	Name of the App. E.g.: "TikTok"	Developer ID	Shows unique developer Id. E.g.: "56"
App Id	Unique Id of the App. E.g.: "ru.webvo.book.aaafb"	Developer Website	Gives the link to the developer website. E.g: ""
Category	Category to which the App belongs. E.g.: "Books & Reference"	Developer Email	Shows the email id of the developer

Rating	Rating for the Apps ranging from 0 to 5	Released	Gives the timestamp of the app when it was released to public
Rating Count	Total Count of the ratings given for an app	Last Updated	Shows the last updated timestamp of the app Eg: “2/8/2022 8:25:55 AM”
Installs	Number of total installs for the app	Content Rating	Shows whether the app is for adults, kids, teen, or everyone
Minimum Installs	Minimum number of installs	Privacy policy	Gives the link of page for the privacy policy. Eg: “https://sites.google.com/privacy”
Free	Whether the App is Free or not E.g.: “True” or “False”	Ad Supported	Whether the App supports advertisement or not E.g.: “True” or “False”
Price	Tells the price of the app if it is not free	Scraped Time	Time when the data was scrapped. Eg : “6/14/2022 2:16:00 PM”
Currency	Shows the currency in which the price is mentioned		



AppName	appId	Category	Rating	Rating Count	installs	min Installs	free	price	currency	developerId	developer Website	developer Email	released	Last Updated	content Rating	privacy Policy	ad Supported	Scraped Time
Contempor	com.fel	Books & Re	0	0	5+	5	TRUE	0	USD	Human+Droid+Apps		farukabdilla	26-Dec-20	#########	Mature 17	https://d	TRUE	6/14/2022 14:1
Peribahasa	com.an	Books & Re	0	0	1,000+	1000	TRUE	0	USD	9.06813E+18	https://irw	irwankayer	19-Jun-15	#########	Everyone	https://ir	TRUE	6/14/2022 14:1
Dua e Nudb	com.glo	Books & Re	0	0	10,000+	10000	TRUE	0	USD	5tan+Library	https://ya	glowingapp	15-Jun-17	#########	Everyone	https://5	TRUE	6/14/2022 14:1
Learn Electr	com.qu	Books & Re	4.1818	337	100,000+	100000	TRUE	0	USD	SuperSimpleVic	http://ww	Team@sim	14-Apr-12	#########	Everyone	http://go	TRUE	6/14/2022 14:1
Maulid Al-B	com.alb	Books & Re	0	0	1,000+	1000	TRUE	0	USD	7.25499E+18	https://dja	qurotuluyu	6-Oct-19	#########	Everyone	https://d	TRUE	6/14/2022 14:1
Taariikhda	com.his	Books & Re	5	302	100,000+	100000	TRUE	0	USD	6.27254E+18	https://his	ismailahma	27-Jul-17	#########	Everyone	https://h	TRUE	6/14/2022 14:1
ÐÑ‚ÐµÑ€Ð¾	ru.OlMa	Books & Re	0	0	100+	100	TRUE	0	USD	OlMal		olmal9691(	24-Jan-20	#########	Everyone	https://si	TRUE	6/14/2022 14:1
The New W	com.mc	Books & Re	0	0	500+	500	TRUE	0	USD	7.17046E+18	https://mc	mobleidev(	25-Mar-21	#########	Everyone	https://n	TRUE	6/14/2022 14:1

Using this metadata, the top 35 applications based on total installs were chosen, with a focus on the gaming category. The names of the chosen apps are as follows:

• Minecraft	• Subway Surfer	• Criminal Case	• Tank stars	• Cooking Madness
• Temple Run	• PUBG	• Jetpack	• Crowd City	• Clash of Clans
• Geometry Dash	• Hungry Shark Evolution	• Tiktok	• Google Chrome	• Paper.io 2
• Instagram	• Youtube	• Whatsapp	• Roblox	• Growtopia
• BlockCraft	• Angry Bird Transformer	• Zombie	• Tap Tap Dash	• Dragon Mania Legends
• Genshin Impact	• Brother in Arms	• Ant Smasher	• Stickman	• Rider
• Glow Hockey	• Ice Age Adventure	• Doodle jump	• Leps World	• Garena Free fire

Now, the reviews from the last 10 years (2012-2022) were scrapped for these mentioned apps. A total of 35 CSV files were created which had the following columns:

Column	Details	Column	Details
Review Id	Has alpha-numeric unique Id of the review	Thumbs up Count	Number of likes received on the review
Username	Name of the user	Review created version	The version of app when the review was posted
User Image	Thumb nail of the user	At	Timestamp of the review
Content	The text of the review which the user has posted	Reply content	The reply given to the review if any.
Score	The rating that the user has given to the app	Replied at	The timestamp of the reply given to the review

	reviewId	userName	userImage	content	score	thumbsUpCount	reviewCreatedVersion	at	replyContent	repliedAt
0	be6a53e4-9b9e-4ae6-86ea-50b2d8096608	Charles Ojegbulu	https://play-lh.googleusercontent.com/a/AItbvm...	Awesome	5	0	2.22.13.76	8/2/2022 17:00	NaN	NaN
1	eb61677d-f0ab-4f65-84ea-898e745930e7	Linet Kendi	https://play-lh.googleusercontent.com/a/AItbvm...	It was the best	5	0	2.22.3.77	8/2/2022 17:00	NaN	NaN
2	7a73726f-e86f-4026-8afc-f6afb73874bc	Veronica James	https://play-lh.googleusercontent.com/a/AItbvm...	Is so beautiful whatsapp	5	0	2.22.9.78	8/2/2022 17:00	NaN	NaN
3	33ccef53-d635-4dcf-8613-1c06971e95d0	Ggbeko Shigmah	https://play-lh.googleusercontent.com/a/AItbvm...	It really helpful and fast	5	0	NaN	8/2/2022 17:00	NaN	NaN
4	93566307-5fa6-4a39-9fd5-b0f3aa87e9ac	Rudra Awasthi	https://play-lh.googleusercontent.com/a/AItbvm...	Very useful for us no another words	5	0	2.22.15.74	8/2/2022 17:00	NaN	NaN

The average size of files was 170 mb with the minimum size being 6 mb and maximum size being 300 mb. The total data scrapped in this process was of 9 GB and the crawler crawled 1000s of google play store webpages.

Data Wrangling

Concatenating DataFrames:

The metadata obtained was as per categories, so to extract the top 35 apps based on installs it was necessary to merge the data frames. Concat() function simply adds DataFrames on top of each other. Since the columns names for all the files was same it was easy to perform this function.

Handling Duplicates, Missing Values:

1. Dropped rows which had missing values in “content” column.
2. Dropped irrelevant columns which would not contribute to the Sentiment Analysis like: “reviewId","userImage","replyContent","repliedAt”.
3. Checked Duplicate rows based on “username” “content” and “at” column
4. Ratings greater than or equal to 3 was categorized as “good” and less than 3 was classified as “bad”. This would later help us in prediction of reviews through Machine learning models.
5. “At” was converted to datetime ‘%m %d %Y’ format.

```
whatsappData.info()
```

```
RangeIndex: 804756 entries, 0 to 804755
Data columns (total 10 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   reviewId              804752 non-null  object
 1   userName              804756 non-null  object
 2   userImage             804756 non-null  object
 3   content               804714 non-null  object
 4   score                 804756 non-null  int64
 5   thumbsUpCount         804756 non-null  int64
 6   reviewCreatedVersion  661675 non-null  object
 7   at                    804756 non-null  object
 8   replyContent          0 non-null       float64
 9   repliedAt             0 non-null       float64
dtypes: float64(2), int64(2), object(6)
memory usage: 61.4+ MB
```

Fig: WhatsApp Dataset information

6. Data types for columns were correctly defined.
7. Columns were renamed for clarity purpose.

Descriptive Statistics:

The summary statistics was obtained for all the 35 applications. The sample of whatsapp is as follows:

```
================================================
           Number of reviews:  804714

      Number of unique reviewers:  697501
        Prop of unique reviewers:  0.867

          Average rating score:  4.107
================================================
```

```
##########################################
## Classify rating scores as good or bad
##########################################

good_rate = len(whatsappData[whatsappData['score'] >= 3])
bad_rate = len(whatsappData[whatsappData['score'] < 3])

# Printing rates and their total numbers
print ('Good ratings : {} reviews for Whatsapp App'.format(good_rate))
print ('Bad ratings : {} reviews for Whatsapp App'.format(bad_rate))

whatsappData['rating_class'] = whatsappData['score'].apply(lambda x: 'bad' if x < 3 else'good')
whatsappData.head()
```

```
Good ratings : 663983 reviews for Whatsapp App
Bad ratings : 140731 reviews for Whatsapp App
```

	userName	content	score	thumbsUpCount	reviewCreatedVersion	at	rating_class
0	Charles Ojegbulu	Awesome	5	0	2.22.13.76	8/2/2022 17:00	good
1	Linet Kendi	It was the best	5	0	2.22.3.77	8/2/2022 17:00	good
2	Veronica James	Is so beautiful whatsapp	5	0	2.22.9.78	8/2/2022 17:00	good
3	Ggbeko Shigmah	It really helpful and fast	5	0	NaN	8/2/2022 17:00	good
4	Rudra Awasthi	Very useful for us no another words	5	0	2.22.15.74	8/2/2022 17:00	good

Fig: Code for converting rating as good or bad for WhatsApp.

Preprocessing Text:

Since, text is the most unstructured form of all the available data, various types of noise are present in it and the data is not readily analyzable without any pre-processing. The entire process of cleaning and standardization of text, making it noise-free and ready for analysis is known as text preprocessing. In this section, the following text preprocessing methods were applied.

1. Removing HTML tags

HTML tags typically do not add much value towards understanding and analyzing text. Hence, HTML words were removed from the text.

2. Removing accented characters

Accented characters/letters were converted and standardized into ASCII characters.

3. Expanding Contractions
Contractions are shortened version of words or syllables. They exist in either written or spoken forms. Shortened versions of existing words are created by removing specific letters and sounds. In case of English contractions, they are often created by removing one of the vowels from the word.

By nature, contractions do pose a problem for NLP and text analytics because, to start with, we have a special apostrophe character in the word. Ideally, we can have a proper mapping for contractions and their corresponding expansions and then use it to expand all the contractions in our text.

4. Removing Special Characters
One important task in text normalization involves removing unnecessary and special characters. These may be special symbols or even punctuation that occurs in sentences. This step is often performed before or after tokenization. The main reason for doing so is because punctuation or special characters often do not have much significance when we analyze the text and utilize it for extracting features or information based on NLP and ML.

5. Lemmatization
The process of lemmatization is to remove word affixes to get to a base form of the word. The base form is also known as the root word, or the lemma, will always be present in the dictionary.

6. Removing stop words
Stop words are words that have little or no significance. They are usually removed from text during processing to retain words having maximum significance and context. Stop words are usually words that end up occurring the most if you aggregate any corpus of text based on singular tokens and checked their frequencies. Words like a, the, me etc. are stop words.

7. Building a Text Normalizer
Based on the functions which we have written above and with additional text correction techniques (such as lowercase the text, and remove the extra newlines, white spaces, apostrophes), we built a text normalizer to help us to preprocess the new text document.

A clean dataset will allow a model to learn meaningful features and not overfit on irrelevant noise. After following these steps and checking for additional errors, we can start using the clean and labelled data to train models in the modeling section.

Review Text	Clean Text
This is my favorite game â□¤ï¸□â□¤ï¸□â□¤ï¸□	favorite game
I really like this game and it does not even have ads	really like game not even ads

Clash Of Clans Game Taking Too Much Storage!! My Mobile Or Whose Players Mobile Is Smaller GB - They Can't Affordable To Play No Space No Game , Please Minimize The Clash Of Clans Game Application inner 400Mb only it's Too Big 600+ Mb So Please Fix It Thank..	clash clans game take much storage mobile whose players mobile smaller gb not affordable play no space no game please minimize clash clans game application inner mb big mb please fix thank
It's a very good game ðŸŽ®	good game
Very bad experience i completed to many clan games and i took reward but now I am board and Sad . what kind of this reward always in reward we get worth spells and worth gemes ðŸ’Ž in reward you reward only 100 gems.100 gemes i can get by tree ðŸŽ„ remove.we all need 1000 gemes and 4 hero book ok it's called reward..but now my experience going very bad .we all want real reward .do something and change reward list ok.we all are bo ard with this worth reward	bad experience complete many clan game take reward board sad kind reward always reward get worth spell worth game reward gems game get tree remove need game hero book ok call reward but experience go bad want real reward something change reward list ok we board worth reward
I PLAYED THIS GAME ALMOST 8YEARS AND THIS IS A GOOD GAME	play game almost years good game

Exploratory Data Analysis

After collecting data and subsequently wrangling the same, exploratory analyses were carried out on the dataset. The following insights were explored through exploratory analyses.

Distribution of Rating:

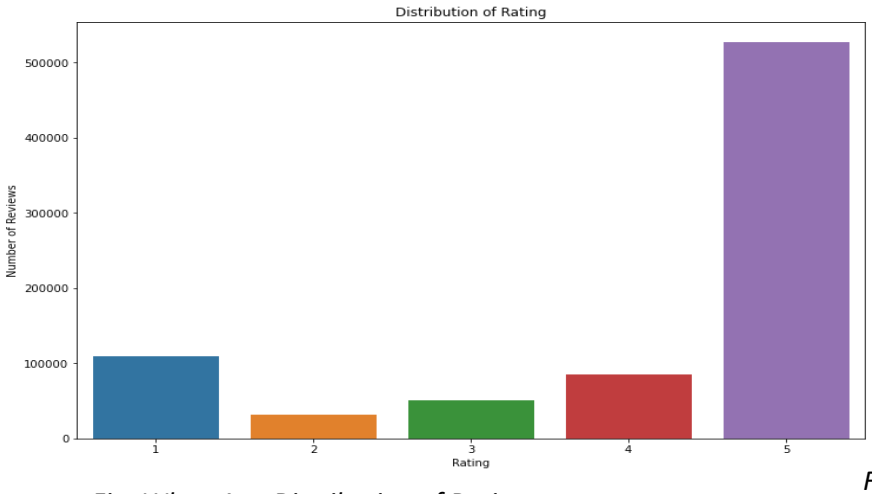


Fig: WhatsApp Distribution of Rating

```
fd.most_common(50)

[('good', 119420),
 ('nice', 45868),
 ('', 42934),
 ('good app', 14168),
 ('nice app', 13168),
 ('excellent', 12421),
 ('super', 12047),
 ('ok', 11018),
 ('best', 8035),
 ('great', 7674),
 ('love', 4946),
 ('awesome', 4942),
```

Fig: Most common words in the clean text

Word Cloud:

Fig: Word Cloud of Most common words

Letter-Value Plots (or Boxenplots):

These plots have been developed to overcome the problem of an inaccurate representation of outliers in boxplots. The wider a box in a boxen plot, the larger ratio of total population it contributes to.

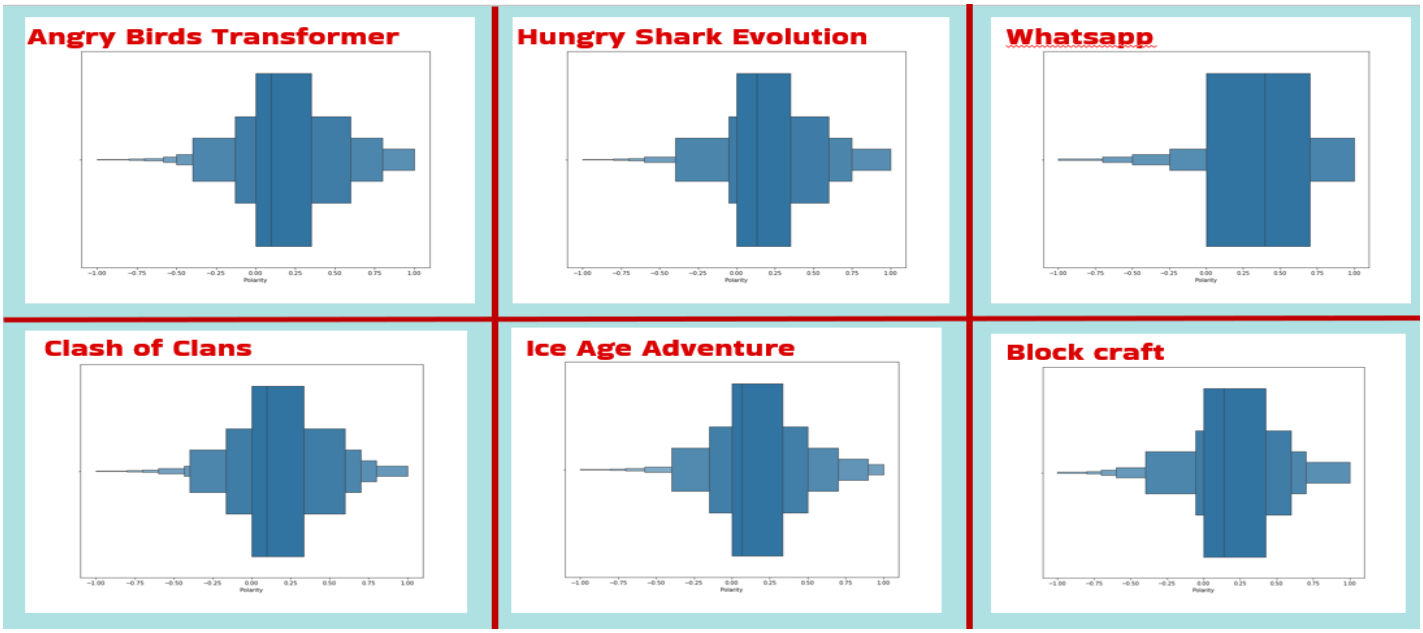


Fig: Boxen plot based on sentiment score

Most Similar Words:

‘Feel’	'name', 'turn', 'home', ‘experience', 'right’
'Good'	'nice', 'useful', 'helpful', 'wonderful', 'love'
'Product'	'application', 'things', 'better', 'quick', 'touch'
'Cheap'	'unblock', 'password', 'despite', ‘low’, ‘affordable’
'Bad'	'new', 'could', 'stop', 'something', 'notification'
‘Great’	'amaze', 'like', 'awesome', 'use', 'better’

Data Modelling

SENTIMENT ANALYSIS:
Machine Learning models take numerical values as input. The reviews are made of sentences, so to extract patterns from the data; we need to find a way to represent it in a way that machine learning algorithm can understand, i.e., as a list of numbers.

FEATURE EXTRACTION:
Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Features are usually numeric in nature and can be absolute numeric values or categorical features that can be encoded as binary features for each category in the list using a process called one-hot encoding. The process of extracting and selecting features is both art and science, and this process is called feature extraction or feature engineering.

DATA PREPROCESSING:
Due to computational considerations, top 10000 observations were only used for Modelling purposes. From the dataset, “clean text” and “rating class” were treated as “X” (feature) and “Y”(variable) respectively. Dataset were divided into 75% as training and 25% as testing.

MACHINE LEARNING:
In this project, the model needs to predict sentiment based on the reviews written by user for any application. This is a supervised binary classification problem. Python’s Scikit libraries was used to solve this problem.

Following machine learning algorithms were implemented:

1. **Logistic Regression:** Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-

entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

2. **Random Forest Classifier:** A random forest is a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

3. **Naive Bayes:** Naive Bayes implements the naive Bayes algorithm for multinomial distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts). This algorithm is a special case of the popular naïve Bayes algorithm, which is used specifically for prediction and classification tasks where we have more than two classes.

4. **XGBoost Classifier:** XGBoost means eXtreme Gradient Boosting. XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.

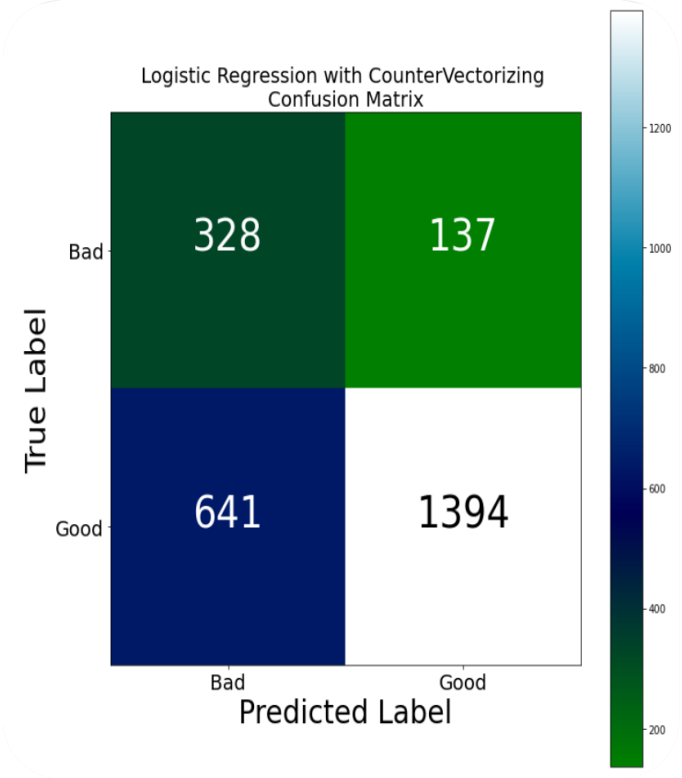
5. **CatBoost Classifier:** CatBoost is an algorithm for gradient boosting on decision trees. “CatBoost” name comes from two words “Category” and “Boosting”. the library works well with multiple Categories of data, such as audio, text, image including historical data.

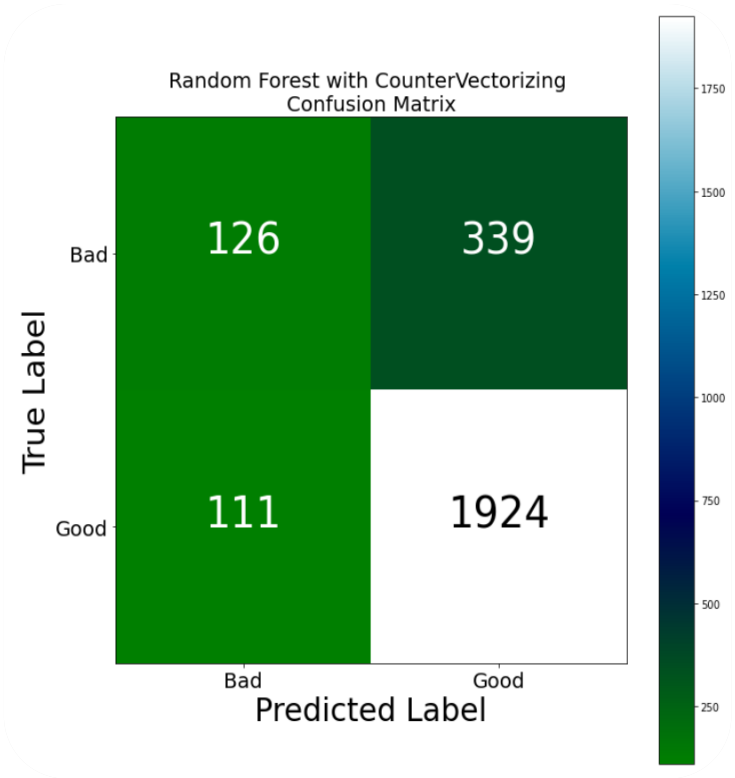
EVALUATION METRICS:

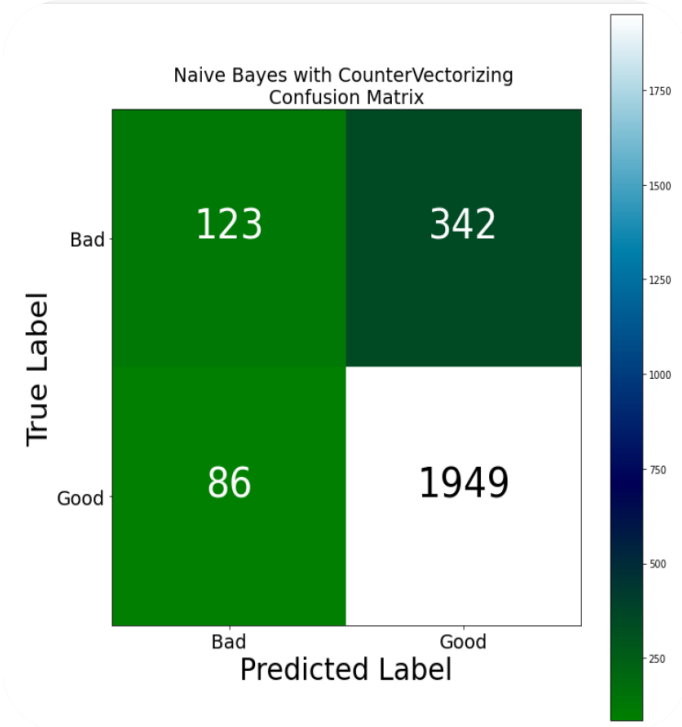
Since we have a data imbalance in our case, the evaluation of the classifier performance must be carried out using adequate metrics to consider the class distribution and to pay more attention to the minority class. Based on this thought, f1 score was used, which is harmonic average of precision and recall as my evaluation metric. Understanding the types of errors our model makes are important. A good way to visualize that information is using a Confusion Matrix, which compares the predictions our model makes with the true label. With that in mind, confusion matrix was used besides our evaluation metric (f1 score).

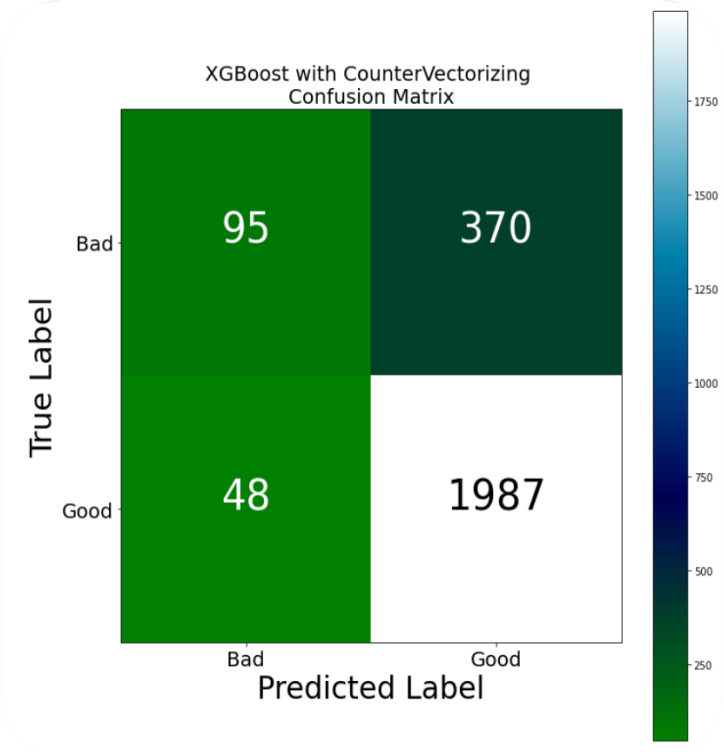
MODELLING:

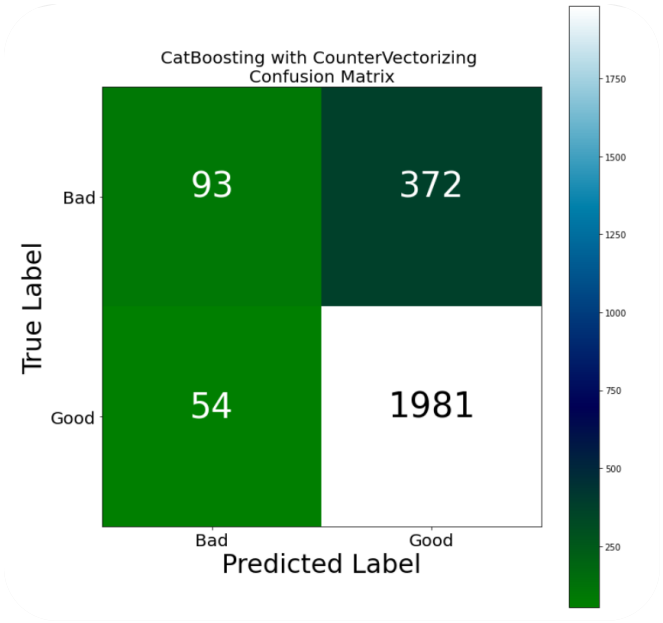
Since the ratings of the reviews were not distributed normally, rating classes from 1-2 were classified as ‘Bad’ and Rating 3-4-5 were classified as 'Good'. For feature selection, threshold for word occurrence with using min_df/max_df were applied. For feature engineering, CountVectorizer was applied to the text data to turn a collection of text documents into numerical feature vectors.


Logistic Regression with CounterVectorizing
Confusion Matrix
328
137
641
1394
Bad
Good
True Label
Bad
Good
Predicted Label
1200
1000
800
600
400
200


Random Forest with CounterVectorizing
Confusion Matrix
126
339
111
1924
Bad
Good
True Label
Bad
Good
Predicted Label
1750
1500
1250
1000
750
500
250


Naive Bayes with CounterVectorizing
Confusion Matrix
123
342
86
1949
Bad
Good
True Label
Bad
Good
Predicted Label
1750
1500
1250
1000
750
500
250


XGBoost with CounterVectorizing
Confusion Matrix
95
370
48
1987
Bad
Good
True Label
Bad
Good
Predicted Label
1750
1500
1250
1000
750
500
250



1. Bag of Words Model

The Bag of Words model is perhaps one of the simplest yet most powerful techniques to extract features from text documents. This specific strategy (tokenization, counting and normalization) is called the Bag of Words or "Bag of n-grams" representation. The essence of this model is to convert text documents into vectors such that each document is converted into a vector that represents the frequency of all the distinct words that are present in the document vector space for that specific document. The figure below shows that **XGBoost is the winner with 83% accuracy.**

vectorizer	model	accuracy	class	precision	recall	f1-score
CountVect	LogReg	0.6888	bad	0.338493	0.705376	0.457462
			good	0.910516	0.685012	0.781828
			average	0.804120	0.688800	0.721496
	Random Forest	0.8200	bad	0.531646	0.270968	0.358974
			good	0.850199	0.945455	0.895300
			average	0.790948	0.820000	0.795544
	Naive Bayes	0.8288	bad	0.588517	0.264516	0.364985
			good	0.850720	0.957740	0.901063
			average	0.801950	0.828800	0.801353
	XGBoost	0.8328	bad	0.664336	0.204301	0.312500
			good	0.843021	0.976413	0.904827
			average	0.809785	0.832800	0.794654
	CatBoost	0.8296	bad	0.632653	0.200000	0.303922
			good	0.841904	0.973464	0.902917
			average	0.802983	0.829600	0.791504

Fig: Results of various Machine Learning models employed.

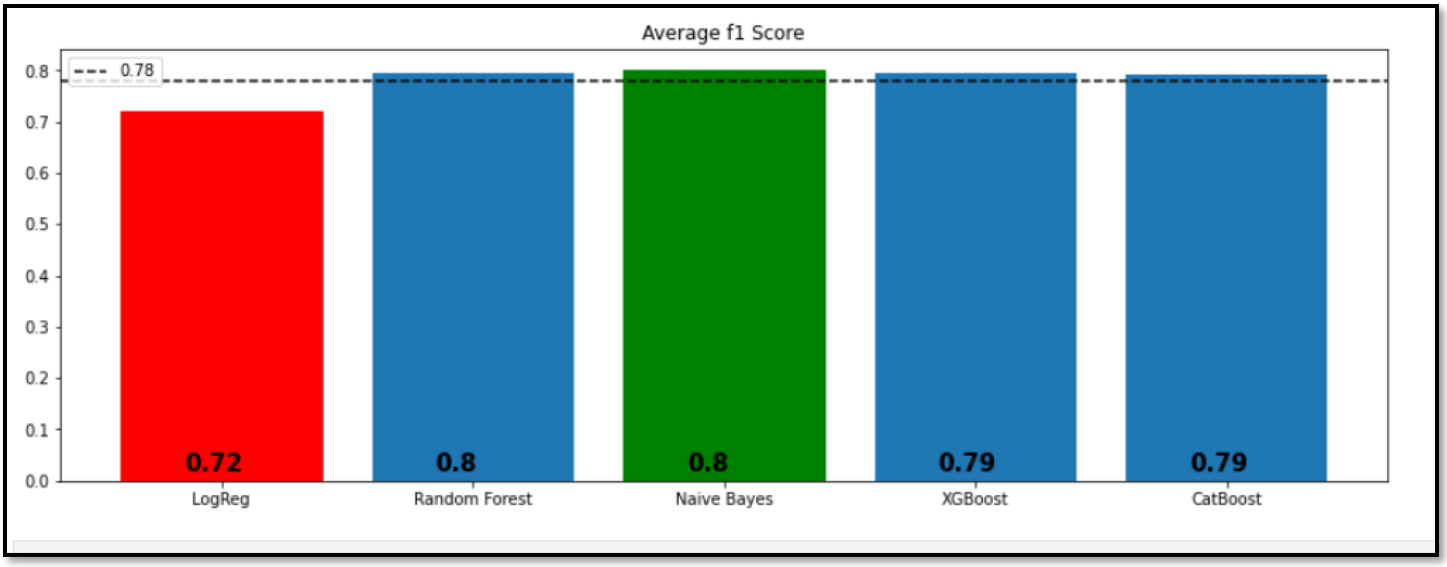


Fig: Average F1 score of Machine Learning models employed

CONCLUSION:

In this project, the rating scores based on the reviews left by the users were predicted using Count Vector, and Classification Models like: Logistic Regression, Random Forest Classifier, Naive Bayes, XGBoost Classifier, CatBoost Classifier. From the analyses, it was found that XGBoost with Count Vectorizing with accuracy of 83% & f1 score of 0.79 was the top models.

Future Scope

- The model can be used by Google play store for decreasing the negative comments by users by giving follow up questions to them to enhance their user experience.
- Recommendation system can be proposed using the sentiment analysis data.
- Implementation of Dask library for parallel processing to decrease run time.

Challenges

- Dealing with the size of the dataset.
- Learning about deploying code on Google cloud VM for running data mining at scale rather than personal PC.
- Tuning the parameters of ML model to increase the accuracy of predictions.

Code

Web scrapping: https://github.com/jsaumya20/Web_scraping
Sentiment Analysis: https://github.com/jsaumya20/Sentiment-Analysis-