RETWEET PREDICTION ON EARTHQUAKE TWEETS

DEPREM TWEETLERİ ÜZERİNDE RETWEET TAHMİNİ

SEVGİNUR İNCE

PROF. DR. EBRU AKÇAPINAR SEZER

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Science

in Computer Engineering

2024

To my family

ABSTRACT

RETWEET PREDICTION ON EARTHQUAKE TWEETS

Sevginur İNCE

Master of Science, Department of Computer Engineering

Supervisor: Prof. Dr. Ebru AKÇAPINAR SEZER

September 2024, 120 pages

On February 6, 2023, an earthquake centered in Kahramanmaraş killed or damaged many people. In the aftermath of these devastating earthquakes, the efficiency of communication channels in the crisis zone is of vital importance. While a few decades ago there was no indication of the existence of social media, today social media platforms have become people's main communication channels. Twitter, one of these platforms, is widely used in Turkey. Social media provides the opportunity to reach millions of people with a shared post. The amount of interaction a post receives increases the possibility of being noticed by other users on social media. In this thesis, using the tweets posted during and after the earthquake centered in Kahramanmaraş on February 6, 2023, the retweet interaction amounts were divided into two classes. These classes are *'non-low'* and '*moderate-high'* classes. The data was captured with Python's Snscrape Library as 38 days of data covering February 6, 2023 - March 15, 2023. The following operations were then performed respectively: Tweet text was cleaned. Spelling mistakes were corrected with the python Zemberek Module. Words were parse to their roots with Zeyrek Module. Stop words were deleted. Stop words were deleted. The dataset was simplified and IDF values of unique words in the first week tweets were calculated. Unique words were grouped according to their IDF value ranges. By adding 400 unique words from different IDF ranges to the dataset, 7 dataset versions consisting of different unique word groups were obtained. Among these sets, the word set that best represents the tweet text was investigated. The XGBoost model was used in the analysis. We also investigated the interaction type and class threshold limit that would be the best class label. The best class label was 'Retweet' and the best class distinction limit was observed as 2. The words that best represents the dataset were found to be the 400 words with the lowest IDF value. These words were added to the dataset as Binary Bag of Words. Then, classification was performed with various Deep Learning and Machine Learning models. These models are Random Forest, XGBoost, LSTM and DistilBERTurk. The XGBoost model gave the best performance. The results of the XGBoost model are as follows: Non-low class precision 0.75, recall 0.70, F1 score 0.73, Moderate-high class precision 0.72, recall 0.77, F1 score 0.74. Average accuracy 0.7340 and ROC-AUC score 0.81.

Keywords: Text Classification, Machine Learning, Natural Language Processing. XGBoost, LLM, Random Forest, LSTM

ÖZET

DEPREM TWEETLERİ ÜZERİNDE RETWEET TAHMİNİ

Sevginur İNCE

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Prof. Dr. Ebru AKÇAPINAR SEZER

Eylül 2024, 120 sayfa

6 Şubat 2023'te Kahramanmaraş merkezli depremde çok sayıda kişi hayatını kaybetti veya zarar gördü. Bu yıkıcı depremlerin ardından kriz bölgesindeki iletişim kanallarının etkinliği hayati önem taşımaktadır. Birkaç on yıl önce sosyal medya varlığına dair hiçbir belirti yokken, bugün sosyal medya platformları insanların temel iletişim kanalları haline geldi. Bu platformlardan biri olan Twitter Türkiye'de de yaygın olarak kullanılmaktadır. Sosyal medya paylaşılan bir gönderi ile milyonlarca insana ulaşma imkânı sağlamaktadır. Bir gönderinin aldığı etkileşim miktarı sosyal medyada diğer kullanıcılar tarafından fark edilme ihtimalini arttırır. Bu tezde, 6 Şubat 2023'te Kahramanmaraş merkezli deprem sırasında ve sonrasında atılan tweetler kullanılarak retweet etkileşim miktarları iki sınıfa ayrılmıştır. Bu sınıfar '*düşük olmayan'* ve *'orta yüksek'* sınıflarıdır. Veriler Python'un Snscrape Kütüphanesi ile 6 Şubat 2023 - 15 Mart 2023 tarihlerini kapsayan 38 günlük veriler olarak ele geçirildi. Daha sonra sırasıyla şu işlemler gerçekleştirildi: Tweet metni temizlendi. Yazım yanlışları Python Zemberek Modülü ile düzeltildi. Zeyrek Modülü ile kelimeler köklerine ayrıldı. Duraksama kelimeleri silindi. Veri seti basitleştirildi ve ilk

hafta verilerinin IDF değerleri hesaplandı. Eşsiz kelimelerin IDF değerleri hesaplandı. IDF değer aralıklarına göre eşsiz kelimeler gruplandı. Farklı IDF aralıklarından 400'er eşsiz kelime veri setine eklenerek farklı eşsiz kelime gruplarından oluşan 7 dataset versiyonu elde edildi. Bu setlerin içinden tweet metnini en iyi temsil eden kelime seti araştırıldı. Analizlerde XGBoost modeli kullanıldı. Ayrıca en iyi sınıf etiketi olacak etkileşim tipi ve sınıf eşik sınırı da araştırıldı. En iyi sınıf etiketi 'Retweet', en iyi sınıf ayrım sınırı ise 2 olarak gözlendi. Data seti en iyi temsil eden kelimelerin IDF değeri en düşük olan 400 kelime olduğu belirlendi. Bu kelimeler veri setine Binary Bag of Words olarak eklenmiştir. Ardından çeşitli Deep Learning ve Machine Learning modelleri ile sınıfama gerçekleştirildi. Bu modeller Random Forest, XGBoost, LSTM ve DistilBERTurk'tür. XGBoost modeli en iyi performansı verdi. XGBoost modeli sonuçları aşağıdaki gibidir: Düşük olmayan sınıf hassasiyeti 0.75, geri çağırma 0.70, F1 puanı 0.73, orta-yüksek sınıf hassasiyeti 0.72, geri çağırma 0.77, F1 puanı 0.74. Ortalama doğruluk 0.7340 ve ROC-AUC puanı 0.81.

Anahtar Kelimeler: Metin Sınıflandırma, Makine Öğrenmesi, Doğal Dil İşleme, XGBoost, Rastgele Orman, LSTM

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude my master thesis supervisor Prof. Dr. Ebru AKÇAPINAR SEZER for her precious advices and guidance. Since I started working with her, I have learned priceless information.

I would also like to thank all faculty members who give me contribution throughout my educational life.

Finally, I would like to express my sincere gratitude to my parents and sister for their endless support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviations	
AUC	Area Under Curve
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag of Words
DistilBERT	Distilled version of BERT
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GPU	Graphics Processing Unit
IDF	Inverse Document Frequency
LLM	Large Language Models
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
ROC	Receiver Operating Characteristics Curve
RNN	Recurrent Neural Network
SVM	Support Vector Machine
TF	Term Frequency
TF-IDF	Term Frequency- Inverse Document Frequency
TN	True Negative
TP	True Positive
TPR	True Positive Rate
XGBoost	Extreme Gradient Boost

1. INTRODUCTION

On February 6, 2023, an earthquake centered in Kahramanmaras, Turkey caused devastation. The disaster, which occurred at 04:17 in the morning on a winter day, caused people in the region to be caught in their sleep. This caused the damage to be much higher. Some people died on the spot, and others were trapped under the debris. The 7,7 and 7,6 magnitude earthquakes affected 11 provinces in Turkey. In some provinces, such as Hatay, the destruction was much greater than in other provinces. There are almost no buildings left standing in the city. People desperately tried to reach their relatives and get help. For this, they tried to make their voices heard on social media. They posted continuously to ensure that bulldozers arrived in the region to meet basic needs such as water, food, tents, and prefabricated houses, to receive medical aid, and to reach their loved ones trapped under the rubble. They also used the Twitter platform extensively for this purpose. In addition, Twitter is one of the platforms used not only by earthquake victims but also by people trying to reach them for help.

Twitter is among the top social media platforms in use nowadays. Here, a user uses various types of interaction if they want to spread information to other users. One of the fastest ways to spread information on a social network is through Retweet interaction. Retweet interaction is when another user re-shares a user's tweet on their own page. If a user adds a comment while sharing a post, this is an example of a quote interaction. If a user comments on a posted tweet, this is an example of reply to interaction. If a user refers to another user's name while writing a tweet, this is a mention type of interaction. Likewise, the amount of engagement a tweet receives makes it more likely to be noticed by other users on social media. Engagement is therefore vital for the spread of information. Especially if the information is the result of massive destruction.

Classical channels of communication, such as the telephone, allow a limited number of people to be reached. However, posts shared on social media can reach millions or even billions of people at the same time. It is simply enough to click on the 'share' button. For this reason, social media is a very powerful communication channel. These platforms, which have only become popular in the last 20 years, have become indispensable in our daily lives. In terms of its widespread use, speed, accessibility and scope, social media platforms provide benefits to survivors in the event of a disaster that no other communication platform can provide.

In this study, tweets sent during and after the earthquakes centered in Kahramanmaras, Turkey on February 6, 2023, were captured with the Python’s snscrape module. All tweets containing the word 'deprem' (means 'earthquake' in Turkish) between February 6, 2023 and March 15, 2023 were captured. The dataset also includes tweets in many different languages, as this disaster has reverberated around the world. Tweet texts were pre-processed, and the text data was cleaned. When cleaning the tweet text, all non-textual content was deleted. These are punctuation marks, whitespace characters, emojis, hashtags, numbers, mentions, URLs, etc. Then, the Turkish spelling correction library Zemberek was used to remove as many spelling mistakes as possible. Although the Zemberek library was developed in Java, it also has python support. After this step, words were parsed into their roots to simplify modeling and reduce data complexity. For this purpose, the Zeyrek framework of ython, which is widely used as a lemmatizer in the Turkish language, was used. Zeyrek library was developed as a tool of Zemberek library.

After lemmatization implemented, the stop words in the tweet text were deleted. Stop words can be defined as words that are frequently used in daily speech but do not allow any information to be extracted. Examples in Turkish include pronouns such as ‘ben', 'sen', 'o', nouns such as 'bugün', 'yarın', 'dün', verbs such as 'yapmak', 'etmek'. The deletion of stop words was not limited to Turkish. Words belonging to languages that do not utilize the Latin alphabet, e.g., Japanese, Chinese, Sanskrit, and Arabic, were eliminated with regex by the Unicode values of their alphabets. This is because the Twitter API is no longer able to retrieve the location where the tweet was shared. Stop words for languages that use the Latin alphabet have also been removed. Examples of these are European languages, Indonesian, etc. Some of the words in languages using the Latin alphabet have equivalents in Turkish. These words were excluded from the planned deletion of stop words. Even after removing stop words, it was still not possible to completely clean the tweet text. In order to make the data easier to model and to reduce its complexity, only the first week's data was considered instead of all 38 days. Although Zemberek and

Zeyrek contribute to normalizing the data to a large extent, unfortunately they do not provide perfect performance. To remove words from the dataset that still have typos or are written in a foreign language, words with a frequency of 2 or less were also removed from the dataset. If there is too much noise in data, there are two possible outcomes. Either the model that will use will underperform and will not be able to learn the data. Or it will contain so many unique words that the model will force itself to learn everything and eventually memorize the data. When memorization occurs, the model's ability to generalize is reduced. For this reason, frequency-based cleaning was performed.

In the next step, the IDF values of the unique words in the first week's data were calculated. It was divided into 1 interval each. The number of unique words in each interval was tabulated. Then, it was investigated how many unique words can be added to the dataset as binary bag of words style. As a result of the studies, it was discovered that up to 400 words could be encoded into the dataset in binary bag of words style.

There are approximately 69.000 unique words in the cleaned data of the first week. The IDF values of each of these words were calculated. Words with IDF values above 10 are words consisting of a high percentage of misspellings. The frequency of these words is also low. Because of this situation, we took the IDF value range of 1-10 as the working boundary. The 400 words with the lowest IDF value are in the idf value range of 1-7. For the second 400-word feature set, we took 400 randomly selected words with IDF values in the range of 7-8. This experiment was then repeated for the 2nd version. Similarly, the words from each of the 2 versions were selected from the IDF value ranges 8-9 and 9-10. In total, we have 7 dataset versions with 400 words each. Before finding which of these word sets best represents the text of the tweet in the dataset, we first investigated which label feature is the best label feature. For this purpose, the data consisting of 400 words with minimum IDF value was taken randomly as the base analysis data. Each dependent feature was applied to this data as a class label feature one by one. For each dependent variable, binary classification was performed with the XGBoost model. We also investigated at which threshold value gives the best results. For this purpose, the class separation threshold value was decreased by 1 in each experiment starting from 5 and the results were tabulated. For example, it was decided to investigate the class discrimination power of the Reply feature. For this, the class discrimination threshold value was initially set to 5. This means that tweets with a Reply engagement value greater than 5 are included in the high interaction class, while tweets with a Reply engagement value of 5 and less than 5 are included in the low interaction class. In this way, threshold values from 1 to 5 were tested one by one on each dependent variable. The best result was achieved at a threshold value of 2 for the Retweet dependent variable. After determining the best class label, we searched for which word set would best represent this dataset. Previously, 7 different versions of the dataset were prepared. It was seen that the dataset containing 400 words with minimum IDF value would best represent the text of this tweet. After finding the class label, class boundary and the set of words that best represent the dataset, we moved on to the model selection stage.

During the model selection process, multiple machine learning and deep learning techniques were benchmarked against one another. Random Forest, XGBoost, LSTM and DistilBERTurk models were utilized. We used a dataset of 400 word features for analysis with XGBoost, Random Forest and LSTM. In these experiments, no text data was given to the models. Only the DistilBERTurk model was trained with tweet text. The DistilBERTurk model represents a Turkish adaptation of the Distil BERT model, which serves as a more compact and efficient alternative to the BERT model. It uses the Attention mask structure to infer the position of each word in the sentences in which it appears. As a consequence of the experiments, It was ascertained that the XGBoost model achieved the pinnacle of performance amidst the tetra of models.

The subsequent are the contributions of this study. We endeavored to forecast the number of retweets a tweet in the earthquake ecosystem would receive. To this purpose, we identified the important words from the text data. We compared the performance of classical and popular models using a dataset that had not been studied before. In order to analyze text data, it must be vectorized. By adding the 400 words that best represent the tweet text as features, analyses were conducted using Random Forest, LSTM, and XGBoost models. On the other hand, LLM (Large Language Model) automatically vectorizes words thanks to the attention mechanism in its structure. Thus, the performance of manual vectorization and LLM's automatic vectorization were also compared.

2. RELATED WORKS

Daga et al. (2020) used the number of likes and retweets as metrics to predict a tweet's popularity. A set of classifiers was employed to predict tweet popularity, using the number of likes and retweets as the metric. The researchers concentrated on tweet content to determine the factors that generate an adequate number of likes and retweets. Using the 'tweepy' library in Python, two million tweets from the years 2015 to 2018 were gathered from Twitter in the field of Data Science. The research focused on "data science," selecting tweets tagged with hashtags like #datascience and #machinelearning. The dataset includes the following features: tweet date, likes, retweets, and tweet text. They created two diagrams to illustrate their feature set: one for the most used words in the top 25% of retweeted tweets and another for the most used words in the top 25% of liked tweets. Examples of words from these diagrams include 'datascience', 'bigdata', 'data', 'analytics', and 'machinelearning'. They employed two distinct word embedding techniques, Doc2Vec and TF-IDF, along with five dissimilar machine learning models: Random Forest (RF), Support Vector Machine (SVM), Logistic Regression (LR), Multinomial Naive Bayes and Neural Networks (NN). In their experiments, the Random Forest model using TF-IDF achieved the highest accuracy and the least overfitting, making it the best model compared to the others. The test accuracies for predicting retweets (in %) are as follows: LR (with TF-IDF): 50.7, LR (with Doc2Vec): 35.06, SVM (with TF-IDF): 51.1, SVM (with Doc2Vec): 34.8, RF (with TF-IDF): 40.8, RF (with Doc2Vec): 60.08, NN (with TF-IDF): 49.18, NN (with Doc2Vec): 40.65, Bayes (with TF-IDF): 50.1, Bayes (with Doc2Vec): 35.08 [1].

Chen and Deng (2020) delineated the problem statement as follows: when a user, designated as the target user, encounters a piece of information, they must ascertain whether to disseminate it by retweeting. The researchers delved into heterogeneous relation networks, scrutinizing various social interactions that influence how the act of retweeting is swayed by the sentiments of both the sender and the receiver. They extracted an array of features from identified elements across three dimensions—diffusion behavior, content semantics, and network and user structure. Their prediction challenge was approached through ensemble learning techniques, proposing a model dubbed Retweeting Behavior on Multiple Heterogeneous Diffusion Relation Networks

(RBMHDRN). This model amalgamates several models within an ensemble learning framework. To validate their hypothesis, they juxtaposed its performance against Logistic Regression (LR), Support Vector Machine (SVM), and Naive Bayes (NB) models. Their model aspires to anticipate user re-posting behavior in information dissemination scenarios within social networks. They amassed data from Weibo, encompassing 762,936 microblogs posted by 68,817 users. The statistical characteristics of the dataset include: Users: 68,817, Original Posts: 425,213, Retweets: 337,723, Direct Retweet Relations: 71,974, and Indirect Retweet Relations: 400,469. Their feature sets are categorized as follows: Dimension of Content Semantics, Dimension of User Diffusion Behavior, Dimension of Network Structure. To assess the efficacy of their model, RBMHDRN, the researchers compared it against two rudimentary baselines. The first baseline, termed HRB, is predicated on the target user's historical retweeting behavior, specifically whether they have previously retweeted another microblog from the same original user. The second baseline is grounded in the empirical retweeting probability between the target user and the originator, gauging the proportion of microblogs presented to the target user via the originator that are retweeted by the target user. Furthermore, to evaluate the effectiveness of their proposed ensemble learning approach, PALRSF, they contrasted PALRSF with three traditional methods—Naive Bayes (NB), Logistic Regression (LR), and Support Vector Machine (SVM)—utilizing the same features. Their proposed model RBMHDRN attained the most favorable performance, with metrics indicating precision of 0.9296, recall of 0.9120, and an F1 score of 0.9190 [2].

Roy et al. (2020) used the final retweet count as one of the popularity metrics to examine the likely popularity of a tweet. Researchers aims to predict online post popularity using the RNN-HMFC model, which blends Recurrent Neural Networks with Feature Concatenation. It was tested on two datasets: TwitterViolence (900.000 tweets from 2018-2019 and 500.000 tweets from 2010-2018, focusing on violence-related content) and TwitterSEISMIC (166.076 tweets with at least 50 retweets in 15 days). The model was trained on 7 days of data and tested on 8 days, with additional information collected for comprehensive analysis and evaluation. The RNN-HMFC model comprises three feature extraction sub-modules and a final module: Textual Feature Extractor: Cleans tweets and converts words into embedding vectors. Retweet Count Feature Extractor: Transforms retweet counts into sequences of vectors over time using a one-hot encoder.

Explicit Feature Extractor: Gathers additional features that influence tweet popularity, such as mention count, URL count, media count, tweet length, follower count, total tweet count, and favorites count. Final Module: Merges the latent feature vectors from the first two sub-modules to create the final feature set for prediction. They achieved best results under time window 10 sec and precision 0.9521, recall 0.9569, F1 score 0.9545 [3].

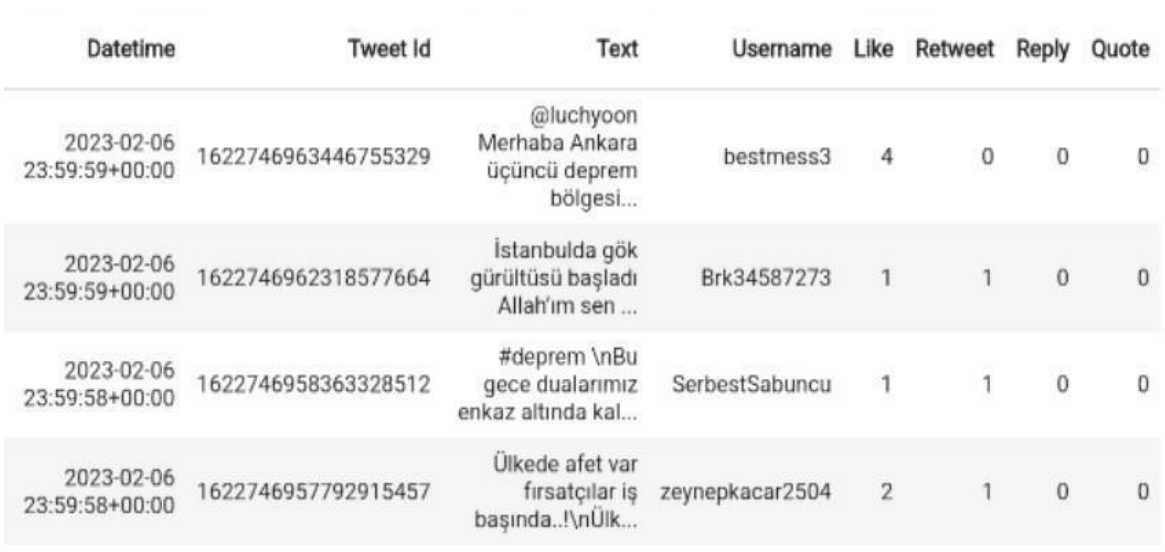
Wang and Yang (2021) defined retweet prediction as a binary classification problem. The researchers introduced an innovative deep multitask learning model for predicting retweets. Initially, they extracted numerical features to represent both tweet information and social aspects. These numerical features were then combined with textual features. Following feature extraction, they generated embeddings for these features and integrated them into the model. They used data gathered by Chris Albon during the 2016 U.S. election day, consisting of 393.764 total tweets, 338.331 processed in English, across 34 data categories, with 76.157 tweets shared. They classified the features into two types: Numerical Features: Include word count, favourites tweet count, hashtag count, tweet length, and mention count, which affect tweet popularity. Social metrics such as user activity (ratio of user listed count to days on Twitter), user status count, favourites user count, user listed count, friend count, days spent on Twitter and follower count are vital for forecasting retweets. Textual Features: Extracted and deduplicated each post's textual content to create a corpus. Instead of basic one-hot encoding, they utilized the word2vec model to train a neural network language model on this corpus. This approach allowed them to extract textual features using semantic embeddings, enhancing analysis of text-related characteristics. They introduced a new multitask learning approach named CH-Transformer for predicting retweets. The CH-Transformer model integrates Transformer architecture with hard sharing modes to forecast the likelihood of content forwarding. Their system comprises three main components: multitask prediction , model training, and feature representations. To assess their model's performance, they also evaluated several other methods on their dataset, including LSTM, CNN-LSTM, CRNN, and MRNN (multi-tasking RNN). In single task learning, SC-Transformer model has the best performance scores. SC-Transformer's scores are follows: precision 0.769, recall 0.733, F1-score 0.75. In multi task learning, their proposed model hit the highest scores under Task 1. Those scores are follows: precision 0.847, recall 0.735, F1 score 0.787 [4].

Sharma and Gupta (2022) aimed to predict retweets from a single user's perspective using only Twitter API data and a minimal feature set, offering a novel approach to retweet prediction. They approached the problem of estimating retweets through three distinct research questions: 1. Determining whether a tweet post will be shared. 2.Predicting the exact number of retweets for a tweet. 3.Classifying tweets into multiple categories based on labels. It examines the influence of numerical user profile features derived from real-time tweets. The study employs regression and classification algorithms for its analyses. The feature set is evaluated on a dataset of 100 million random tweets gathered through the Twitter API. To tackle the first question, they analyzed the Logistic Regression model using Logistic Regression and Logistic Model Tree on the three primary feature groups: Author Features, Proposed Combined Features, and Tweet Features. The Logistic Model Tree model provided the best results for this question, with an accuracy of 0.97 for the 'Tweet Features' feature set, 1.0 for the 'Author Features' feature set, and 1.0 for the 'Proposed Combined Features' feature set. For the second question, it was observed that all models produced weak results. Therefore, it was concluded that predicting the exact number of retweets for a tweet is impossible. In the research results for the third question, retweet counts were categorized into specific ranges. Five machine learning models ((1) Decision Tree, (2) Gradient Boosted Tree, (3) Random Forest, (4) SVM, (5) KNN) and three main feature sets (Author Features, Tweet Content Features, Combined Features) were analysed with these models to achieve the best results [5].

Forecasting whether a tweet will be shared by a user has attracted substantial interest recently. Fu and Cheng (2022) devised the MDF-RP (multidimensional feature-based retweeting prediction) model, which incorporates personality traits. The model first extracts features related to the author, user, and tweet from the aboriginal dataset. It subsequently amalgamates these features with a classifier to ascertain whether the user will re-share the tweet. The dataset was amassed from Weibo. They utilized the C4.5 algorithm, Logistic Regression, and SVM as prediction methodologies. The C4.5 algorithm exhibited the most superior prediction performance. With 2-fold and 10-fold cross-validation, the F1-scores of MDF-RP features increased by 6.03% and 6.63% respectively, compared to basic features. Specifically, under 2-fold cross-validation, the F1-score of MDF-RP features reached 0.809. Moreover, from 2-fold to 10-fold cross-validation, the F1-scores of MDF-RP features were augmented by 1.36%. [6].

3. DATA COLLECTION

After the February 6, 2023 earthquake centered in Kahramanmaraş, tweets containing the keyword 'deprem' (means 'earthquake' in Turkish) were extracted. Python's social network scrape (snscrape) module was used for this. The data was obtained on March 26, 2023, covering the period from February 6, 2023, 00:00 to March 15, 2023, 23:59. The features obtained with the data are as follows; date of tweet (Datetime), tweet id, tweet text (Text), name of the user who sent the tweet (username), number of likes (Like), number of retweets (Retweet), number of replies (Reply), number of quotes (Quote). The attained data exemplified in Figure 3.1.



Datetime	Tweet Id	Text	Username	Like	Retweet	Reply	Quote
2023-02-06 23:59:59+00:00	1622746963446755329	@luchyoon Merhaba Ankara üçüncü deprem bölgesi...	bestmess3	4	0	0	0
2023-02-06 23:59:59+00:00	1622746962318577664	İstanbulda gök gürültüsü başladı Allah'ım sen ...	Brk34587273	1	1	0	0
2023-02-06 23:59:58+00:00	1622746958363328512	#deprem \nBu gece dualarımız enkaz altında kal...	SerbestSabuncu	1	1	0	0
2023-02-06 23:59:58+00:00	1622746957792915457	Ülkede afet var fırsatçılar iş başında..!\nÜlk...	zeynepkacar2504	2	1	0	0

Figure 3.1. View of dataset

Accordingly, 4.490.167 tweets containing the word 'deprem' were captured in a 38-day period. There are 4.069.313 unique words in these tweets. The distributions of the number of tweets by days are given in Figure 3.2. In the graph, the numbers on the x-axis represent the number of days, and the y-axis represents the number of tweets. Accordingly, the number 1 on the x-axis refers to day 1, i.e., the tweets of February 6, 2023, and the number 38 refers to day 38, i.e. the tweets of March 15, 2023.

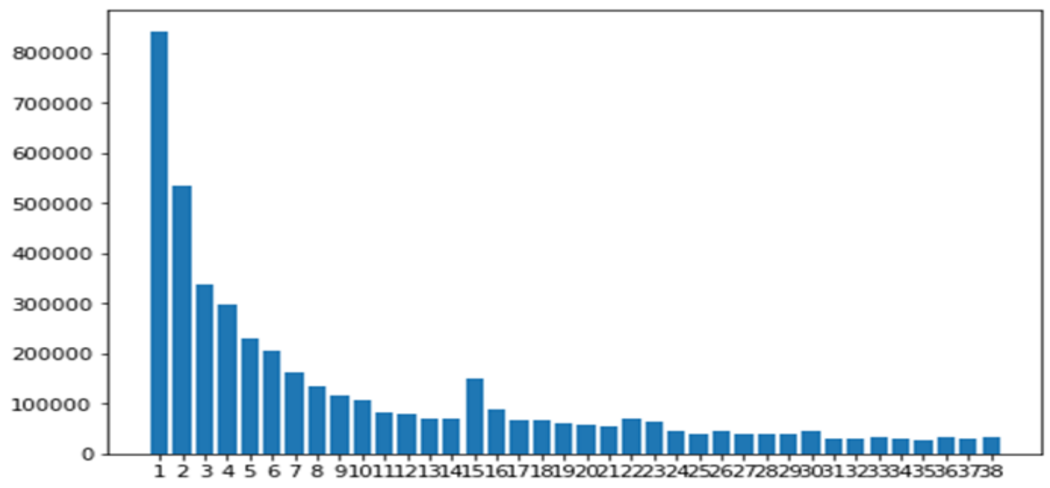


Figure 3.2. Distribution of daily tweet counts by days

During 38 days, there were 1.076.270 different usernames who tweeted tweets containing the word 'deprem'. The maximum tweet length in the entire dataset was 929 characters. Tweet length is 280 tokens or emojis. However, since Unicode characters and the code of emojis can be represented by many characters, it is difficult to define the maximum number of characters in a tweet as a fixed value. Emojis are represented by codes. The class to which each emoji belongs has a Unicode value range. Unicode value ranges can be found on Unicode's website [7]. The standard deviation in the character length of the entire tweet dataset was calculated as 81,690. For 38 days, the distribution of unique usernames tweeted were given in Figure 3.3:

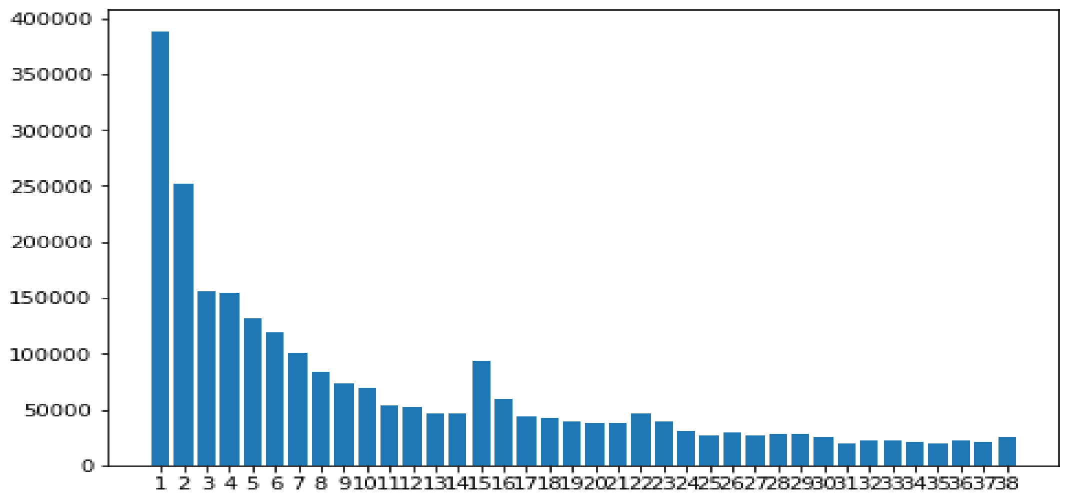


Figure 3.3. Distribution of the number of different users tweeting

Although emojis do not take up 1 unit of space, the Unicode value representing the emoji consists of more than 1 character. For example, the emoji example taken from Unicode.org and the corresponding Unicode's code was given in Figure 3.4:

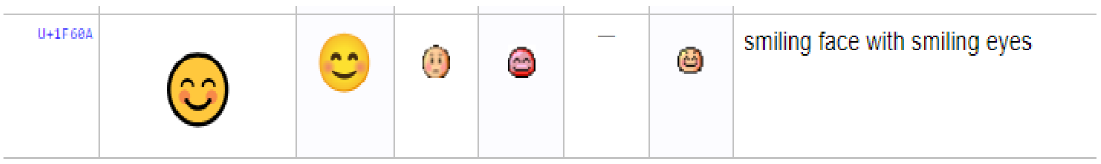

U+1F60A
—
smiling face with smiling eyes

Figure 3.4. Unicode value equivalent of emoji, emoji image and definition [7]

The expression U+1F60A here corresponds to the relevant emoji. The reason why the maximum number of characters is 929 even when emojis are removed can be explained by the presence of characters such as Turkish characters in the tweet text.

In similar studies, information about tweeting users was also captured. In these studies, the features of the tweeting users were also used as features, such as the total number of posts, the number of followers, the number of followers, and the total number of likes. We investigated whether we could get social features by extracting data related to the user feature. When we tried to extract the data, we received an error message. With the sale of Twitter, company policies have been updated. Restrictions and charges were applied to the Twitter API.

4. DATA PREPROCESSING

Data pre-processing processes are summarized in Figure 4.1. First, date data was simplified. 06.02.2023 was assigned the value '1', 07.02.2023 was assigned the value '2', and so on. Lastly all tweets on 15.03.2023 were assigned the value '38'. The dataset before implementing simplification on date time information was exemplified in Figure 4.2. The dataset after implementing simplification on date time information was exemplified in Figure 4.3.

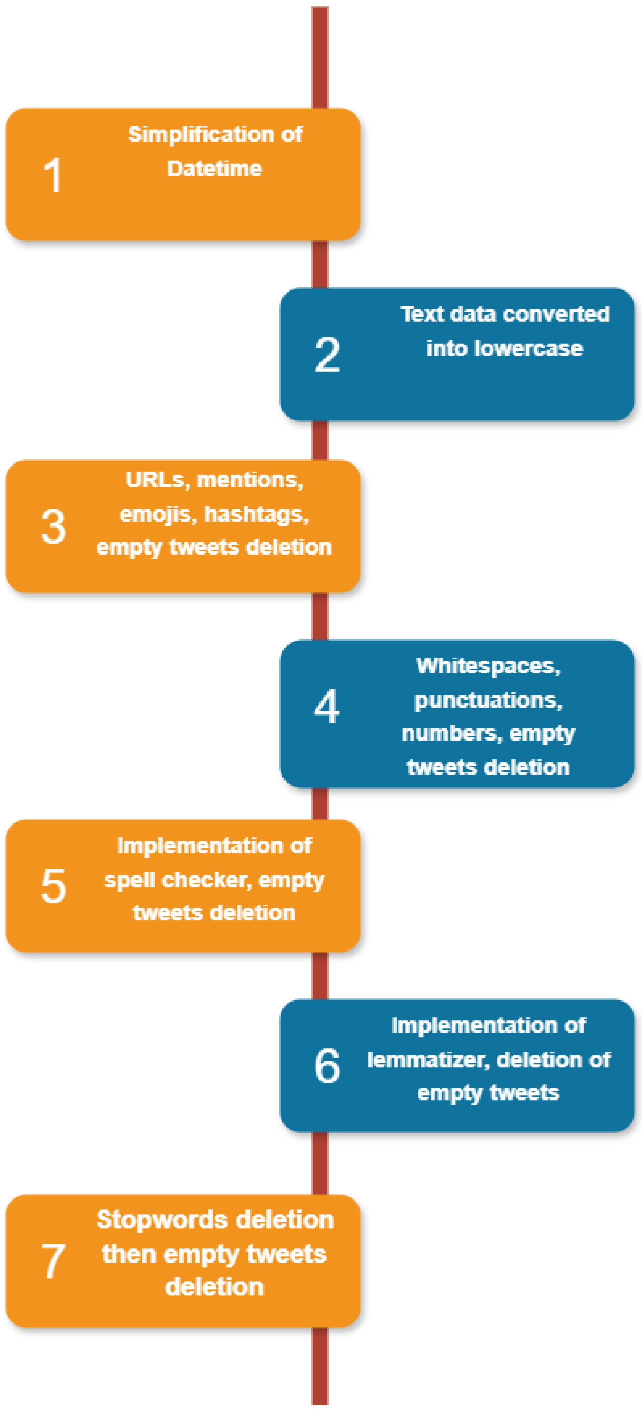
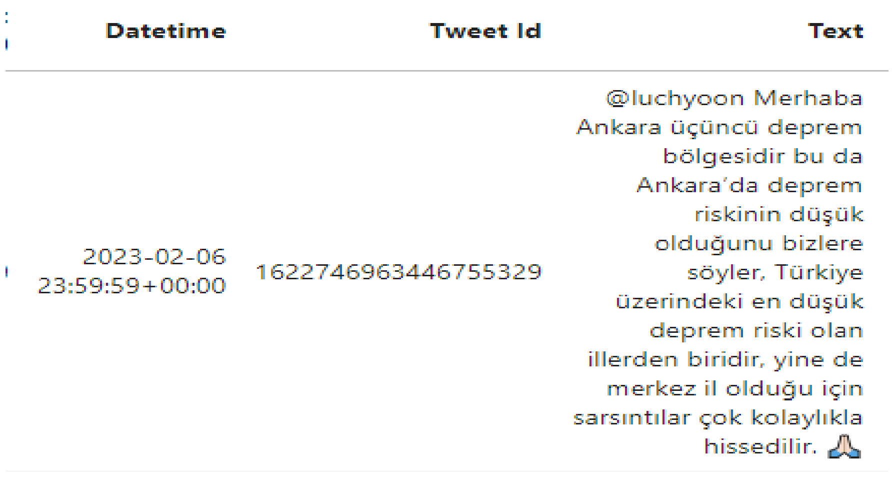
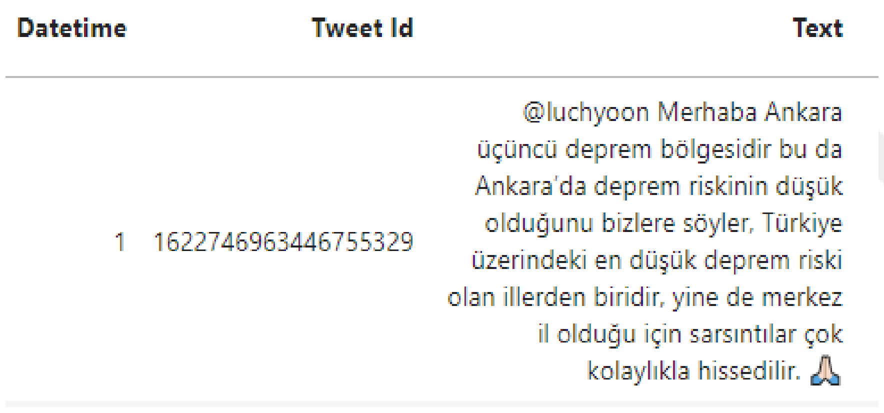

1
Simplification of Datetime
2
Text data converted into lowercase
3
URLs, mentions, emojis, hashtags, empty tweets deletion
4
Whitespaces, punctuations, numbers, empty tweets deletion
5
Implementation of spell checker, empty tweets deletion
6
Implementation of lemmatizer, deletion of empty tweets
7
Stopwords deletion then empty tweets deletion

Figure 4.1. Summary of the data pre-processing steps



Datetime	Tweet Id	Text
2023-02-06 23:59:59+00:00	1622746963446755329	@luchyoon Merhaba Ankara üçüncü deprem bölgesidir bu da Ankara'da deprem riskinin düşük olduğunu bizlere söyler, Türkiye üzerindeki en düşük deprem riski olan illerden biridir, yine de merkez il olduğu için sarsıntılar çok kolaylıkla hissedilir. 🙏

Figure 4.2. Dataset before simplification of date time information



Datetime	Tweet Id	Text
1	1622746963446755329	@luchyoon Merhaba Ankara üçüncü deprem bölgesidir bu da Ankara'da deprem riskinin düşük olduğunu bizlere söyler, Türkiye üzerindeki en düşük deprem riski olan illerden biridir, yine de merkez il olduğu için sarsıntılar çok kolaylıkla hissedilir. 🙏

Figure 4.3. Dataset after simplifying date time information

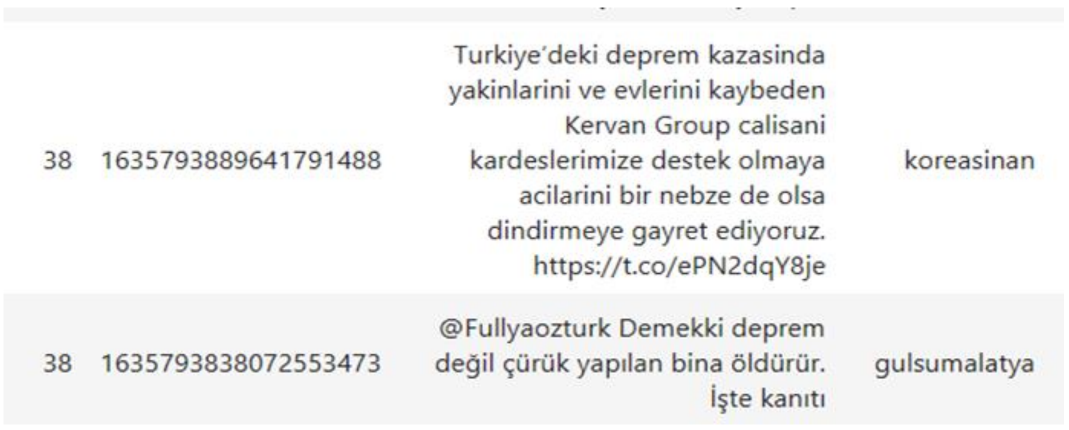
In the next step, the number of hashtags, number of emoji, number of mentions and whether there is any url in the tweet text were processed as separate features for each tweet. Python's built-in regex library that called as ‘re’, was used to search for the number of hashtags, mentions, emoji and url presence.

After processing the hemu features into the dataset, hashtags, mentions, emoji and urls were removed from the tweet text. Punctuation marks, new line characters and numbers were also removed from the entire dataset.

- Removal of URL and Mentions: The urls in the dataset can correspond to different things on twitter. For example, when a user shares a video or a photo, the equivalent in the dataset is a URL link. Likewise, if a user shares an address referring to a different site, it is also observed as a URL link in the dataset. These URLs start with 'https://t.co'. Using this pattern, it was removed from the tweet text through the python regex module.

Mentions, on the other hand, are made by addressing a person or an organization by adding the username of the addressee to the tweet text. Usernames start with the '@' symbol. The python regex library was used to detect the mention pattern and hashtag pattern.

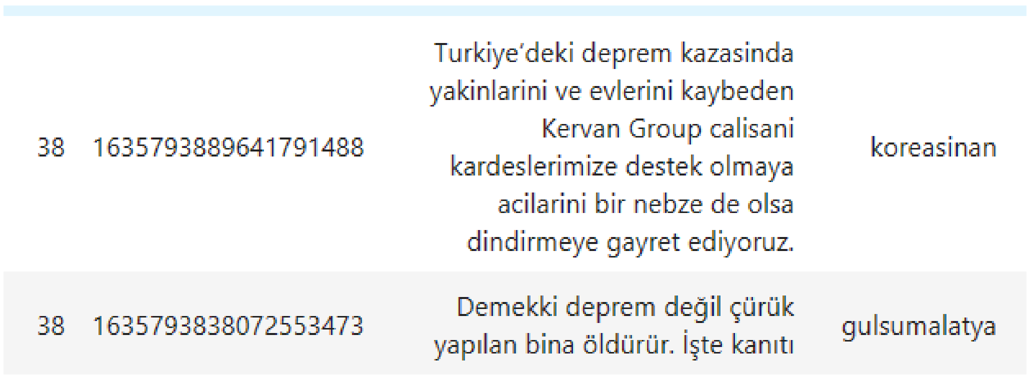
Two tweets example from day 38, before URL and mentions were removed exemplified in Figure 4.4 and Figure 4.5:



38	1635793889641791488	Turkiye'deki deprem kazasinda yakinlarini ve evlerini kaybeden Kervan Group calisani kardeslerimize destek olmaya acilarini bir nebze de olsa dindirmeye gayret ediyoruz. https://t.co/ePN2dqY8je	koreasinan
38	1635793838072553473	@Fullyaozturk Demekki deprem değil çürük yapılan bina öldürür. İşte kanıtı	gulsumalatya

Figure 4.4. Dataset before Mentions and URLs removal

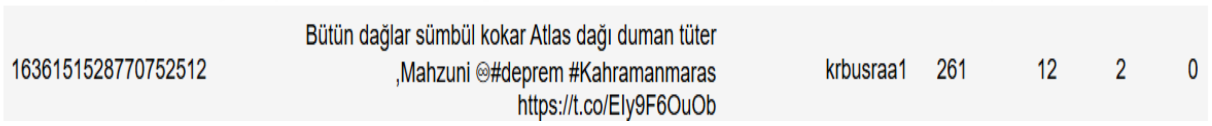
As can be seen, there is a URL in the first tweet text and a mention in the second tweet text. After the URL and mention are removed from the dataset, the relevant tweets appear as follows:



38	1635793889641791488	Turkiye'deki deprem kazasinda yakinlarini ve evlerini kaybeden Kervan Group calisani kardeslerimize destek olmaya acilarini bir nebze de olsa dindirmeye gayret ediyoruz.	koreasinan
38	1635793838072553473	Demekki deprem değil çürük yapılan bina öldürür. İşte kanıtı	gulsumalatya

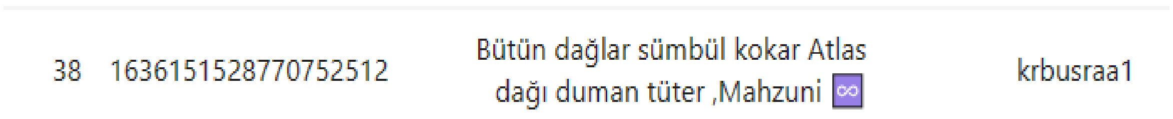
Figure 4.5. Dataset after Mentions and URLs removal

- **Removal of hashtags:** Hashtags are a tool for creating an discussion topic on social media. A hashtag is created by adding the symbol '#' to the beginning of a topic discussed in society on social media. For example, the hashtag of word 'deprem' is expressed as '#deprem'. Examples of tweets before and after hashtags are removed exemplified in Figure 4.6 and Figure 4.7.



1636151528770752512	Bütün dağlar sümbül kokar Atlas dağı duman tüter ,Mahzuni ♾#deprem #Kahramanmaras https://t.co/Ely9F6OuOb	krbusraa1	261	12	2	0

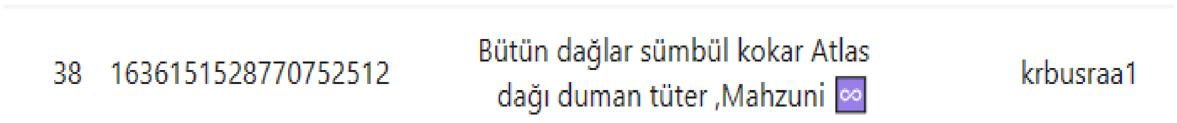
Figure 4.6. Dataset before removing hashtags



38	1636151528770752512	Bütün dağlar sümbül kokar Atlas dağı duman tüter ,Mahzuni ♾	krbusraa1

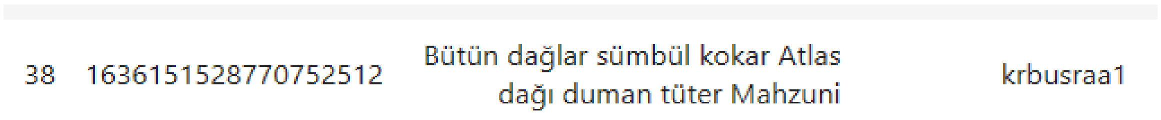
Figure 4.7. Dataset after removing hashtags

- **Removal of Emojis:** Emojis are symbols used by people to express emotions and emphasize statements. Each emoji has a unique unicode value which represents them. Emojis were removed using the Python 'emoji' module. Appearance of the tweet before the removal of emojis exemplified in Figure 4.8 and Figure 4.9:



38	1636151528770752512	Bütün dağlar sümbül kokar Atlas dağı duman tüter ,Mahzuni ♾	krbusraa1

Figure 4.8. View of dataset before removing emojis

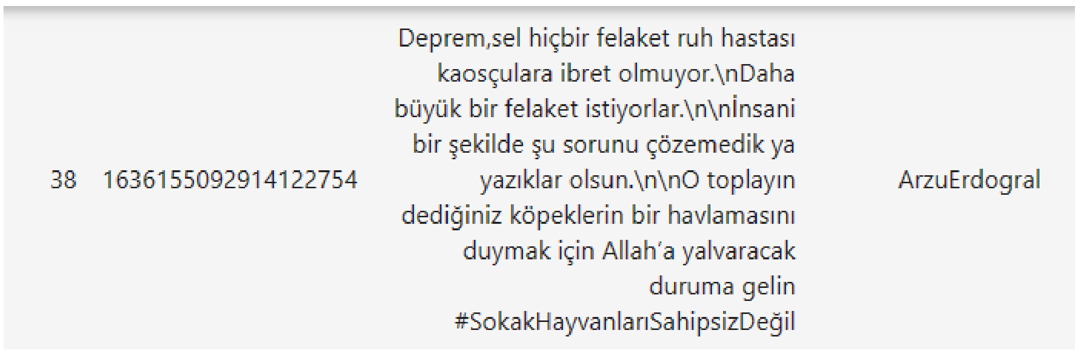


38	1636151528770752512	Bütün dağlar sümbül kokar Atlas dağı duman tüter Mahzuni	krbusraa1

Figure 4.9. View of dataset after removing emojis

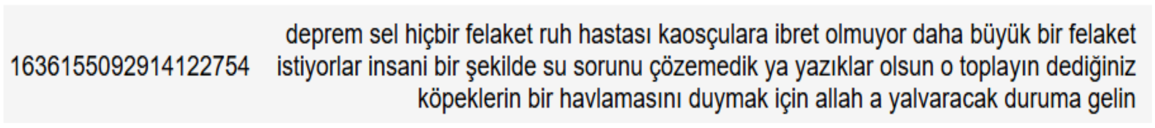
- **Removal of white space characters**

White space characters are characters used to leave a space in the text. They can be observed as '\n' and '\t' in the tweet text. These characters need to be removed to clean the data. View of the dataset before white space characters are removed exemplified in Figure 4.10 and Figure 4.11:



38	1636155092914122754	Deprem,sel hiçbir felaket ruh hastası kaosçulara ibret olmuyor.\nDaha büyük bir felaket istiyorlar.\n\nİnsani bir şekilde şu sorunu çözemedik ya yazıklar olsun.\n\nO toplayın dediğiniz köpeklerin bir havlamasını duymak için Allah'a yalvaracak duruma gelin #SokakHayvanlarıSahipsizDeğil	ArzuErdogral

Figure 4.10. Dataset before removing whitespace characters

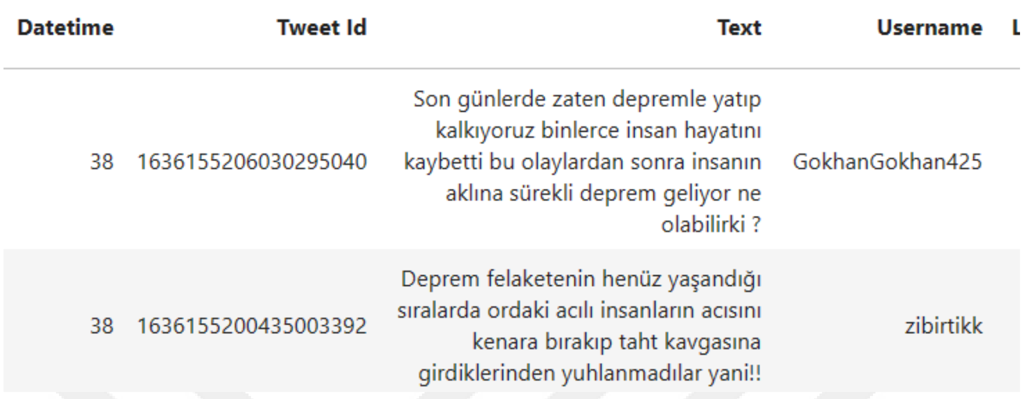


1636155092914122754	deprem sel hiçbir felaket ruh hastası kaosçulara ibret olmuyor daha büyük bir felaket istiyorlar insani bir şekilde su sorunu çözemedik ya yazıklar olsun o toplayın dediğiniz köpeklerin bir havlamasını duymak için allah a yalvaracak duruma gelin

Figure 4.11. Dataset after removing whitespace characters

- **Removal of punctuations**

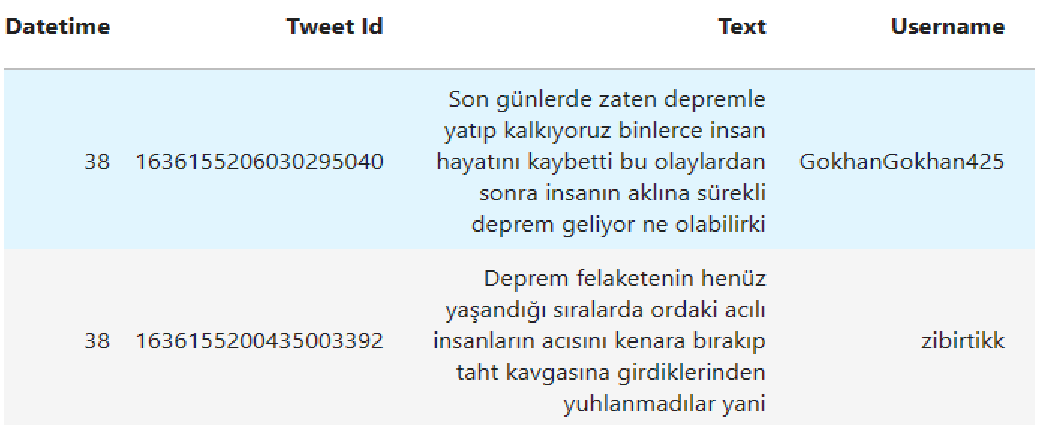
In order to fully clean the text data, punctuation marks must also be removed. Examples of tweets where punctuation marks have not been removed are exemplified in Figure 4.12.



Datetime	Tweet Id	Text	Username	L
38	1636155206030295040	Son günlerde zaten depremle yatıp kalkıyoruz binlerce insan hayatını kaybetti bu olaylardan sonra insanın aklına sürekli deprem geliyor ne olabilirki ?	GokhanGokhan425	
38	1636155200435003392	Deprem felaketenin henüz yaşandığı sıralarda ordaki acılı insanların acısını kenara bırakıp taht kavgasına girdiklerinden yuhlanmadılar yani!!	zibirtikk	

Figure 4.12. Dataset before the removal of punctuations

The appearance of the tweets after the removal of punctuation marks exemplified in Figure 4.13.



Datetime	Tweet Id	Text	Username
38	1636155206030295040	Son günlerde zaten depremle yatıp kalkıyoruz binlerce insan hayatını kaybetti bu olaylardan sonra insanın aklına sürekli deprem geliyor ne olabilirki	GokhanGokhan425
38	1636155200435003392	Deprem felaketenin henüz yaşandığı sıralarda ordaki acılı insanların acısını kenara bırakıp taht kavgasına girdiklerinden yuhlanmadılar yani	zibirtikk

Figure 4.13. Dataset after the removal of punctuations

The Python regex module was used to remove punctuations.

- **Dataset Status After Text Cleaning Process**

Before and after data cleaning, number of tweets, number of words, number of unique words were listed by days. Data cleaning means that punctuation, emojis, hashtags, mentions, URLs, numbers are removed, then empty tweets and tweets 3 characters or less in length are removed. 3 characters was set as the limit because the word 'NaN' consists of 3 characters. The change in the dataset can be observed in Table 4: Distribution of the change of processed data according to days after cleaning of raw data and clean tweet text" in Appendix C.

As the noise level in tweets increases as we approach the early days, it has been observed that the reason for this is the inclusion of media, news, and aid organization links in the tweet text, which are expressed using URLs. This is because even if we have posted a media file or URL link, it will still be represented as a URL link in the data set.

As exhibited in Table 4 which can be found in Appendix C, the term "#words (raw data)" refers to all the tokens that can be separated and extracted based on the spaces, including numbers, emojis, and URL links. Additionally, the entire dataset has been cleaned, and the total number of tweets, words, and unique words has been investigated: Over a period of 38 days, there are a total of 4.074.073 tweets, 76.509.960 words, and 1.824.300 unique words, which will be corrected for spelling errors using the Zemberek module. Foreign language words and misspellings were found in the data which observed with low frequency. This situation causes increasing the number of unique words. This causes a problem in terms of data quality.

- **R^2 Scores Calculation**

R-Squared (R^2) is a statistical metric utilized to assess the percentage of variability in a dependent variable that can be anticipated or clarified by an independent variable [27]. The R-squared values between dependent variables in the dataset before correcting spelling errors have been calculated. The r^2 score between the dependent features was calculated. For this, firstly, r^2 scores of like, retweet, reply and quote features with like, retweet, reply and quote features were calculated. Then, r^2 scores between like, retweet, reply and quote features and hashtag, emoji, mention and url features obtained using tweet texts were calculated. Finally, r^2 scores between hashtag, emoji, mention, url features and hashtag, emoji, mention, url features were calculated. The results were tabulated. Like, Retweet, Reply, Reply, Quote, Hashtag, Mention, Emoji features are recorded according to the quantity and the URL feature is recorded according to whether it is found in the tweet text or not.

Table 4.1. R^2 scores between the features 'Like', 'Retweet', 'Reply', 'Quote' and the features 'Like', 'Retweet', 'Reply', 'Quote'

	like	**retweet**	**reply**	**quote**
like	**1**	0.2441	0.0473	0.0125
retweet	0.2441	**1**	0.1825	0.0776
reply	0.0473	0.1825	**1**	0.2349
quote	0.0125	0.0776	0.2349	**1**

Table 4.2. R^2 scores between the features 'Like', 'Retweet', 'Reply', 'Quote' and the features 'mention', 'hashtag', 'emoji', 'URL'

	mention	**hashtag**	**emoji**	**URL**
like	-0.00174	-0.00169	-0.00171	-0.00170
retweet	-0.00283	-0.00211	-0.00252	-0.00237
reply	-0.00502	-0.00393	-0.00161	-0.00041
quote	-0.05965	-0.05004	-0.01573	7.155e-06

Table 4.3. R^2 scores between the features 'mention', 'hashtag', 'emoji', 'URL' and the features 'mention', 'hashtag', 'emoji', 'URL'

	mention	**hashtag**	**emoji**	**URL**
mention	**1**	-1.3510	-3.5277	-33.2746
hashtag	-1.3510	**1**	-0.5395	-0.1606
emoji	-3.5277	-0.5395	**1**	-0.0660
URL	-33.2746	-0.1606	-0.0660	**1**

The dataset consists of 4.490.167 tweets, which were analyzed for various characteristics. It was observed that 539.302 tweets contained emojis, 1.398.916 tweets included URLs, 2.163.703 tweets had hashtags, and 1.914.897 tweets had mention content.

The Quote feature is represented in the dataset only by a numeric value. A quote interaction is when a user adds a comment to another user's tweet while sharing it on their profile. The numeric values of the quote feature in the dataset show how many different users have engaged in this type of interaction. If the comment contains the word 'deprem', the comment is in our dataset.

After the R^2 coefficients were calculated, the values of the quote feature were embedded into the retweet feature. Quote values were summed with Retweet values. Thus, it was added into the retweet feature. The quote feature was then removed from the dataset. In addition, new output features called 'Retweet+Like', 'Retweet+Reply' and 'Retweet+Like+Reply' were created. Accordingly, Retweet+Like feature is the sum of the amount of Retweets and the amount of Likes in the related row in the dataset, The Retweet+Reply feature refers to the sum of the amount of Retweets and the amount of Replies in the relevant row in the dataset, and the Retweet+Like+Reply feature refers to the sum of the amount of Retweets, Likes and Replies in the relevant row in the dataset.

In addition, we calculated statistics of interaction features, such as ‘Retweet’, ‘Reply’, ‘Like’, ‘Retweet+Like’ etc. We draw each interaction type’s distribution through 38 days according to which criteria is calculated i.e., ‘maximum value’ of Like feature through 38 days. Minimum value of these features always equals zero. So we didn’t give the minimum value of the feature table. Parameters that were calculated through 38 days are ‘maximum value’, ‘mean value’, ‘standard deviation’. Statistics of interaction features can be seen in Appendix A.

- **Spell Checking**

After the tweet text was cleaned of everything except the words, we continued with the misspelling phase. Tweet data contains many words with misspellings. The caliber of the data provided to the model influences the model's efficacy. There are a limited number of libraries for spelling correction in the Turkish language. The most recent and comprehensive of these is the Zemberek library. There are other libraries, but examples such as hunspell are outdated. In addition, they are insufficient against the rich structure of Turkish language. In this study, we used the Zemberek library as a spell checker.

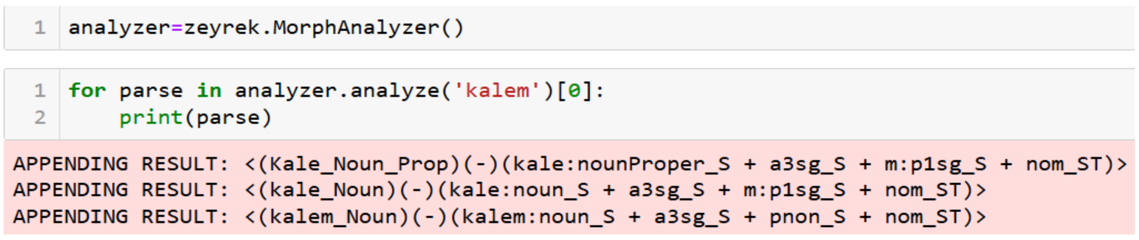
Zemberek is a freely available Natural Language Processing (NLP) library created specifically for the Turkish language. Its aim is to provide a broad spectrum of tools and features for managing and analyzing Turkish text proficiently. Zemberek made its debut in 2007 by Akın et al. [8]. The Zemberek library is available on the website [9] or via pypi. Zemberek is developed using the Java language. Some of the tasks that Zemberek performs are: Turkish Morphological Analysis, Turkish Tokenization, Turkish Spell Checker etc. For example, "iyi" and 'îyi' are the same word, but because the letters 'î' and 'i' are different, they are perceived as two different words by the computer. This makes the number of word types seem much higher than it actually is. Zemberek can correct a word written as 'îyî' as 'iyi'. However, there are also weaknesses in Zemberek. This is discussed in the Discussion and Conclusion section.

After Zemberek was applied, the change in the number of tweets per day in the dataset was tabulated. Another result was a decrease in the number of words. For this reason, the presence of empty tweets was investigated. The state of the dataset after applying zemberek and then removing empty tweets is listed in Table 6. Table 6 can be found in Appendix C. The expressions in the table and their meanings are as follows zemberek b: before zemberek, zemberek a: after zemberek.

After Zemberek, it may be necessary to do lowercase conversion and remove punctuation marks. This is because some words may be capitalized after updating. Zeyrek, a Turkish lemmatizer library, was used to lower the word size even further after spell check with Zemberek. The data outputs processed with the Zemberek module were used as input in the lemmatization process with the Zeyrek module.

- **Lemmatization**

After the spelling mistakes in the words were eliminated, the process of separating the words into their roots was initiated. This is because words with misspellings causes unclearity and causes increase the model size and make the analysis more difficult. In order to simplify the words in the data set, the words should be separated into their roots after the misspellings are fixed. For this, the python Zeyrek module was used. Zeyrek module is the most common and comprehensive lemmatizer module for Turkish language. Zeyrek module is available through pypi or through its developer [10]. Zeyrek was developed as a part of the zemberek library. Zeyrek takes each word, analyzes it, lists the candidate root words. It selects one of the candidate words. An example of Zeyrek can be seen in Figure 4.14.



```
analyzer=zeyrek.MorphAnalyzer()
```

```
for parse in analyzer.analyze('kalem')[0]:
    print(parse)
```

```
APPENDING RESULT: <(Kale_Noun_Prop)(-)(kale:nounProper_S + a3sg_S + m:p1sg_S + nom_ST)>
APPENDING RESULT: <(kale_Noun)(-)(kale:noun_S + a3sg_S + m:p1sg_S + nom_ST)>
APPENDING RESULT: <(kalem_Noun)(-)(kalem:noun_S + a3sg_S + pnon_S + nom_ST)>
```

Figure 4.14. Zeyrek lemmatizer example

Each tweet in the dataset has been tokenized separately. Subsequently, each word has been passed as a parameter to Zeyrek's lemmatization function to obtain the possible root of the word. The changes in the dataset after lemmatization have been calculated separately for each day and presented in a table. The relevant data are located in Table 7 of Appendix C.

During lemmatization, the longest root word suggestion provided by Zeyrek has been selected. Prior to removing stop words, it was observed that some root word suggestions from Zeyrek's library began with uppercase letters. Therefore, all letters were converted to lowercase before removing stop words. After all these processes, the dataset contained 4,057,521 tweets. These tweets included 609,035 unique words, totaling 56,740,073 words.

Subsequently, stop words were removed from the dataset. After removing stop words, tweets with a length of less than 3 characters were also removed. The general status of the data was then analyzed. The status of the dataset after removing stop words and the details of the process can be seen in Appendix B.

After removing stop words, the remaining data contained the following information:

Total number of tweets: 4,050,677

Total number of unique words: 507,178

Total number of words: 49,901,250

Subsequently, the average word counts and standard deviations within each tweet have been calculated separately for each day and presented in a table. This table is located in Table 8 of Appendix C. The distribution of average word counts and their standard deviations over the 38 days is presented in the relevant table.

5. FEATURE SELECTION

The dataset used is a complex dataset consisting of millions of tweets and tweet text in many languages. Meaningful information should be extracted from this dataset. Additionally, words cannot be given to the machine learning model as is. Words must be represented by numbers, otherwise the model cannot detect the words. Various information retrieval methods have been used to extract and vectorize meaningful information from this large and complex dataset.

5.1 Terminology

5.1.1 Information Retrieval

Information retrieval involves finding documents that are pertinent to a specific information requirement. More advanced methods in information retrieval seek to determine not only if a document is pertinent to the user's information requirement, but also how it compares to other documents in terms of relevance. Traditional approaches use the concepts of term frequency and inverse document frequency. The principle behind term frequency is that the more frequently a keyword appears in a document, the more pertinent that document is to the user. Conversely, the principle behind inverse document frequency is that the more documents a keyword appears in, the less useful it is for determining relevance, and therefore, it should be given less weight in relevance calculations [19].

- **TF-IDF**

Term Frequency (TF)

Term frequency (TF) is the number of times a term occurs in a document. It is calculated as:

$$tf(t, d) = count\ of\ t\ in\ d\ /\ number\ of\ words\ in\ d \qquad (1)$$

Inverse Document Frequency (IDF)

The concept of IDF is based on Karen Spärck Jones's PhD thesis called "Synonymy and Semantic Classification" [21]. Inverse document frequency (IDF) is a metric of how rare or common a term is across throughout the corpus. It is computed as:

$$idf(t) = log(N / (df + 1)) \tag{2}$$

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical metric utilized to assess the significance of a word to a document within a collection or corpus. It is a widely used technique in information retrieval, text mining, and user modeling. The significance of a word is determined by multiplying the term frequency (TF) by the inverse document frequency (IDF) [20]. The foundations of this field were laid by various researchers in the 1960s [22] [23].

The formula for TF-IDF is:

$$tf\text{-}idf(t, d) = tf(t, d) * log(N/(df + 1)) \tag{3}$$

where:

t is the term (word).

d is the document (set of words). Document is tweet in this scenario.

N is the total number of documents in the corpus.

df is the document frequency of the term, which is the number of documents that contain the term.

- **Bag of Words**

The bag-of-words model is a text representation approach that views a text as an unstructured compilation of words. It is often employed in natural language processing and information retrieval, where it captures the frequency of word occurrences. In document classification, the bag-of-words model is used as a feature for training classifiers by considering the frequency of each word. The concept of "bag of words" was first mentioned in a linguistic context by Zellig Harris in 1954 [25]. The bag-of-words model can serve as a crucial element in word embeddings like Word2Vec and GloVe. These models utilize the bag-of-words representation as a beginning point and after that apply extra strategies to capture more nuanced semantic connections between words [24].

The bag-of-words model serves a pivotal role in word embeddings, including Word2Vec and GloVe. These models begin with the bag-of-words representation and then use additional techniques to capture more intricate semantic associations between words.

Applications and Areas of Use: The bag-of-words model is extensively applied in several fields, including:

- Document Classification: Used to categorize documents based on their content.
- Information Retrieval: Employed to find relevant documents within a large corpus.
- Text Mining: Utilized to extract significant features from textual data.
- Computer Vision: Applied to represent images using their visual features.

- **Binary Bag-of-Words:** Binary bag-of-words is a variant of the standard bag-of-words model where each word is marked as either presence (1) or absence (0) in a document. This method is particularly useful in binary classification tasks where the presence or absence of a word is more significant than its frequency [26]. This technique has helped manage large volumes of text data by providing an interpretable representation of text data. Also easy to implement and easy to understand. Binary Bag of Words implementation is exemplified in Figure 5.1.

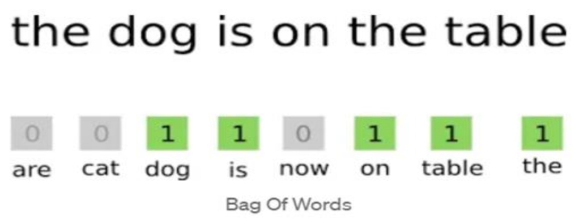


Figure 5.1. Exemplary representation of a binary bag of words [26]

In this study, to elicitation of worthwhile features from tweet data, initially only the data from the first week was considered. This dataset was cleaned by removing unique words with a frequency of 2 or less, as these generally consisted of misspelled or irrelevant words. The IDF values of the unique words in the cleaned data were then calculated. These IDF values were divided into intervals, each consisting of a range of 1. Subsequently, these intervals were sorted from the smallest to the largest IDF values. A table was created to show the number of unique words that fell into each interval. The relevant table is delineated in Table 5.1.

Table 5.1**.** I IDF value dispersal of unique words in the cleaned data of the first week

IDF value range:	Unique word count in range
14<=X<15	15.569
13<=X<14	32.499
12<=X<13	13.055
11<=X<12	7.416
10<=X<11	4.570
9<=X<10	2.791
8<=X<9	1.616
7<=X<8	893
6<=X<7	471
5<=X<6	178
4<=X<5	69
3<=X<4	10
2<=X<3	1
1<=X<2	1

The data for the first week is as follows:

- Number of tweets in the first week: 2.208.321
- Number of unique words: 308.083
- Number of unique words with a frequency of 2 or less: 228.908
- Number of unique words with a frequency above 2: 79.175
- Total number of words: 26.206.196

After removing words with a frequency of 2 or less:

NOTE: After removing unnecessary words, the empty tweets were also removed.

After applying the TF-IDF vectorizer:

- Number of unique words: 79.139
- Total number of tweets: 2.204.141

The study examined the feasibility of adding varying numbers of words as features to a dataset. Words were added in increments of 100 to find the maximum number of features that could be incorporated, with .csv format allowing up to 100 features and .gzip. parquet format allowing up to 400 features. Consequently, 400 words from specific IDF intervals were added to the dataset as binary bag-of-words features, based on their presence or absence in tweets. Initially, 400 words with the lowest IDF values those represented idf range between 1 to 7 were added. New dataset versions were then created using randomly selected two versions of 400 words with IDF values between 7-8, 8-9, and 9-10, resulting in a total of seven dataset versions. Each word was encoded as 1 if present in the tweet text and 0 if absent. Firstly, 400 unique words those has minimum IDF value added beside initial input features such as 'hashtag', 'emoji', 'mention'. Thus we created input feature set. Then it was investigated which feature can be the best class label. The output features in our dataset are as follows; ‘Retweet’, ‘Like’, ‘Reply’, ‘Retweet+Like’, ‘Retweet+Reply’, ‘Retweet+Like+Reply’. Each output feature analyzed seperately. In each experiment, we also investigated what is the best class discrimination boundary from 1 to 5. For example, if the boundary limit for the 'Retweet' feature is 5, Retweet>5 is a moderate-high class and Retweet<=5 is a non-low class. Thus, it was determined which feature can be the best class label and at what threshold it lies. The XGBoost model was utilized in all analyses. Each output feature is discussed separately. Each class separation boundary was analyzed with the XGBoost model for each output feature and the results were tabulated.

Since the best performance was observed when the test set size was chosen as 33%, the test set size was chosen as 33%. The remaining hyper parameters were left by default. Machine learning algorithms assume that each class has a homogeneous distribution and analyze accordingly. Therefore, when working with imbalanced data, the model tends to learn the dominant class and not the minority class. Since we are working with real-world data, this dataset also consists of imbalanced distributed data. There are different solutions for imbalanced data. Since the dataset is noisy and complex, we limit the number of tweets of the dominant class to the number of tweets of the minority class. We randomly select the tweets of the dominant class according to this limit. For this reason, the sample size of both classes was taken equal. In tables, NL stands for "non-low" class, MH stands for "moderate-high" class. Binary classification results and confusion matrices are given in tables.

5.2. XGBOOST RESULTS OF OUTPUT FEATURES

5.2.1 Results of Output Feature 'Like'

XGBoost classification results of output feature 'Like' was exhibited in Table 5.2. For each boundary limit between 1 and 5 learning rate is 0.1, max depth is 6, and test set size is 0.33. Confusion matrix bred from classification when the Like feature threshold value is selected as 5 was exhibited in Table 5.3 Confusion matrix bred from classification when the Like feature threshold value is selected as 4 was exhibited in Table 5.4. Confusion matrix bred from classification when the Like feature threshold value is selected as 3 was exhibited in Table 5.5. Confusion matrix bred from classification when the Like feature threshold value is selected as 2 was exhibited in Table 5.6. Confusion matrix bred from classification when the Like feature threshold value is selected as 1 was exhibited in Table 5.5.

Table 5.2. XGBoost classification results according to threshold values of the 'Like' feature

Boundary Limit:	5	4	3	2	1
NL precision	0.67	0.67	0.67	0.67	0.67
MH precision	0.65	0.65	0.64	0.63	0.63
NL recall	0.63	0.62	0.61	0.59	0.58
MH recall	0.69	0.70	0.70	0.71	0.71
NL F1 score	0.65	0.64	0.64	0.63	0.62
MH F1 score	0.67	0.67	0.67	0.67	0.67
Accuracy:	0.6601	0.6574	0.6535	0.6480	0.6445
Total tweets:	2.204.141	2.204.141	2.204.141	2.204.141	2.204.141
NL size	1.735.588	1.678.566	1.600.082	1.487.336	1.300.067
MH size	468.553	525.575	604.059	716.805	904.074
Subsampled total tweet data	937.106	1.051.150	1.208.118	1.433.610	1.808.148
epoch	default	default	default	default	default
learning rate	0.1	0.1	0.1	0.1	0.1
max depth	6	6	6	6	6
test set size	0.33	0.33	0.33	0.33	0.33

Table 5.3. Confusion matrix as a result of classification when the Like feature threshold value is selected as 5

	Non-low	Moderate-high
Non-low	97221	57458
Moderate-high	47647	106919

Table 5.4. Confusion matrix as a result of classification when the Like feature threshold value is selected as 4

	Non-low	Moderate-high
Non-low	107306	66053
Moderate-high	52800	120721

Table 5.5. Confusion matrix as a result of classification when the Like feature threshold value is selected as 3

	Non-low	Moderate-high
Non-low	121399	78390
Moderate-high	59746	139144

Table 5.6. Confusion matrix as a result of classification when the Like feature threshold value is selected as 2

	Non-low	Moderate-high
Non-low	139756	96977
Moderate-high	69570	166789

Table 5.7. Confusion matrix as a result of classification when the Like feature threshold value is selected as 1

	Non-low	Moderate-high
Non-low	173407	125414
Moderate-high	86730	211138

5.2.2. Results of Output Feature ‘Retweet’

XGBoost classification results of output feature ‘Retweet’ was exhibited in Table 5.8. For each boundary limit between 1 and 5 learning rate is 0.1, max depth is 6, and test set size is 0.33. Confusion matrix as a result of classification when the threshold value of the 'Retweet' feature is selected as 5 was exhibited in Table 5.9. Confusion matrix as a result of classification when the threshold value of the 'Retweet' feature is selected as 4 was exhibited in Table 5.10. Confusion matrix as a result of classification when the threshold value of the 'Retweet' feature is selected as 3 was exhibited in Table 5.11. Confusion matrix as a result of classification when the threshold value of the 'Retweet' feature is selected as 2 was exhibited in Table 5.12. Confusion matrix as a result of classification when the threshold value of the 'Retweet' feature is selected as 1 was exhibited in Table 5.13.

Table 5.8. XGBoost classification results according to threshold values of the 'Retweet' feature

Boundary Limit:	5	4	3	2	1
NL precision	0.74	0.74	0.74	0.73	0.73
MH precision	0.70	0.70	0.70	0.71	0.71
NL recall	0.67	0.68	0.68	0.69	0.70
MH recall	0.77	0.76	0.76	0.75	0.74
NL F1-score	0.71	0.71	0.71	0.71	0.72
MH F1-score	0.73	0.73	0.73	0.73	0.72
Accuracy:	0.7198	0.7202	0.7200	0.7202	0.7198
Total tweets:	2.204.141	2.204.141	2.204.141	2.204.141	2.204.141
NL size	1.929.197	1.892.923	1.843.361	1.769.539	1.648.532
MH size	274.944	311.218	360.780	434.602	555.609
Subsampled total tweet data	549.888	622.436	721.560	869.204	1.111.218
epoch	Default	Default	Default	Default	Default
learning rate	0.1	0.1	0.1	0.1	0.1
max depth	6	6	6	6	6
test set size	0.33	0.33	0.33	0.33	0.33

Table 5.9. Confusion matrix as a result of classification when the threshold value of the 'Retweet' feature is selected as 5

	Non-low	Moderate-high
Non-low	61242	29830
Moderate-high	21013	69379

Table 5.10. Confusion matrix as a result of classification when the threshold value of the 'Retweet' feature is selected as 4

	Non-low	Moderate-high
Non-low	69661	32877
Moderate-high	24605	78261

Table 5.11. Confusion matrix as a result of classification when the threshold value of the 'Retweet' feature is selected as 3

	Non-low	Moderate-high
Non-low	80615	38190
Moderate-high	28489	90821

Table 5.12. Confusion matrix as a result of classification when the threshold value of the 'Retweet' feature is selected as 2

	Non-low	Moderate-high
Non-low	99356	44233
Moderate-high	36038	107211

Table 5.13. Confusion matrix as a result of classification when the threshold value of the 'Retweet' feature is selected as 1

	Non-low	Moderate-high
Non-low	128928	54461
Moderate-high	48307	135006

5.2.3. Results of Output Feature 'Reply'

XGBoost classification results of output feature 'Reply' was delineated in Table 5.14. For each boundary limit between 1 and 5 learning rate is 0.1, max depth is 6, and test set size is 0.33. Confusion matrix as a result of classification when Reply feature threshold value is selected as 5 was delineated in Table 5.15. Confusion matrix as a result of classification when Reply feature threshold value is selected as 4 was delineated in Table 5.16. Confusion matrix as a result of classification when Reply feature threshold value is selected as 3 was delineated in Table 5.17. Confusion matrix as a result of classification when Reply feature threshold value is selected as 2 was delineated in Table 5.18. Confusion matrix as a result of classification when Reply feature threshold value is selected as 1 was delineated in Table 5.19.

Table 5.14. XGBoost classification results according to threshold values of the Reply feature

Boundary Limit:	5	4	3	2	1
NL precision	0.70	0.70	0.69	0.67	0.65
MH precision	0.67	0.66	0.66	0.65	0.64
NL recall	0.63	0.63	0.64	0.63	0.62
MH recall	0.73	0.73	0.71	0.69	0.66
NL F1-score	0.67	0.67	0.66	0.65	0.64
MH F1-score	0.70	0.69	0.68	0.67	0.65
Accuracy:	0.6834	0.6807	0.6745	0.6619	0.6417
Total tweets:	2.204.141	2.204.141	2.204.141	2.204.141	2.204.141
NL size	2.151.569	2.141.072	2.125.005	2.096.542	2.031.430
MH size	52.572	63.069	79.136	107.599	172.711
Subsampled total tweet data	105.144	126.138	158.272	215.198	345.422
epoch	Default	Default	Default	Default	Default
learning rate	0.1	0.1	0.1	0.1	0.1
max depth	6	6	6	6	6
test set size	0.33	0.33	0.33	0.33	0.33

Table 5.15. Confusion matrix as a result of classification when Reply feature threshold value is selected as 5

	Non-low	Moderate-high
Non-low	11008	6344
Moderate-high	4643	12703

Table 5.16. Confusion matrix as a result of classification when Reply feature threshold value is selected as 4

	Non-low	Moderate-high
Non-low	13266	7676
Moderate-high	5617	15067

Table 5.17. Confusion matrix as a result of classification when Reply feature threshold value is selected as 3

	Non-low	Moderate-high
Non-low	16779	9482
Moderate-high	7518	18451

Table 5.18. Confusion matrix as a result of classification when Reply feature threshold value is selected as 2

	Non-low	Moderate-high
Non-low	22612	13029
Moderate-high	10981	24394

Table 5.19. Confusion matrix as a result of classification when Reply feature threshold value is selected as 1

	Non-low	Moderate-high
Non-low	35540	21544
Moderate-high	19304	37602

5.2.4. Results of Output Feature 'Retweet+Like+Reply'

XGBoost classification results of output feature 'Retweet+Like+Reply' was delineated in Table 5.20. For each boundary limit between 1 and 5 learning rate is 0.1, max depth is 6, and test set size is 0.33. Confusion matrix as a result of classification when Retweet+Like+Reply feature threshold value is selected as 5 was delineated in Table 5.21. Confusion matrix as a result of classification when Retweet+Like+Reply feature threshold value is selected as 4 was delineated in Table 5.22. Confusion matrix as a result of classification when Retweet+Like+Reply feature threshold value is selected as 3 was delineated in Table 5.23. Confusion matrix as a result of classification when Retweet+Like+Reply feature threshold value is selected as 2 was delineated in Table 5.24. Confusion matrix as a result of classification when Retweet+Like+Reply feature threshold value is selected as 1 was delineated in Table 5.25.

Table 5.20. XGBoost classification results according to threshold values of the 'Retweet+Like+Reply' feature

Boundary Limit:	5	4	3	2	1
NL precision	0.69	0.68	0.68	0.67	0.66
MH precision	0.65	0.65	0.65	0.65	0.64
NL recall	0.62	0.62	0.62	0.62	0.61
MH recall	0.71	0.71	0.70	0.70	0.69
NL F1 score	0.65	0.65	0.65	0.65	0.64
MH F1 score	0.68	0.68	0.67	0.67	0.67
Accuracy:	0.6685	0.6650	0.6617	0.6591	0.6511
Total tweets:	2.204.141	2.204.141	2.204.141	2.204.141	2.204.141
NL class size	1.598.548	1.529.822	1.436.928	1.305.794	1.093.844
MH class size	605.593	674.319	767.213	898.347	1.110.297
Subsampled total tweet data	1.211.186	1.348.638	1.534.426	1.796.694	2.204.141 (left same)
epoch	Default	Default	Default	Default	Default
learning rate	0.1	0.1	0.1	0.1	0.1
max depth	6	6	6	6	6
test set size	0.33	0.33	0.33	0.33	0.33

Table 5.21. Confusion matrix as a result of classification when the threshold value of the 'Retweet+Like+Reply' feature is selected as 5

	Non-low	Moderate-high
Non-low	124316	75422
Moderate-high	57079	142857

Table 5.22. Confusion matrix as a result of classification when the threshold value of the 'Retweet+Like+Reply' feature is selected as 4

	Non-low	Moderate-high
Non-low	138730	84383
Moderate-high	64696	157242

Table 5.23. Confusion matrix as a result of classification when the threshold value of the 'Retweet+Like+Reply' feature is selected as 3

	Non-low	Moderate-high
Non-low	157264	96249
Moderate-high	75078	177770

Table 5.24. Confusion matrix as a result of classification when the threshold value of the 'Retweet+Like+Reply' feature is selected as 2

	Non-low	Moderate-high
Non-low	184722	111768
Moderate-high	90337	206083

Table 5.25. Confusion matrix as a result of classification when the threshold value of the 'Retweet+Like+Reply' feature is selected as 1

	Non-low	Moderate-high
Non-low	221241	139498
Moderate-high	114315	252313

5.2.5. Results of Output Feature ‘Retweet+Like’

XGBoost classification results of output feature ‘Retweet+Like’ was delineated in Table 5.26. For each boundary limit between 1 and 5 learning rate is 0.1, max depth is 6, and test set size is 0.33. Confusion matrix as a result of classification when Retweet+Like feature threshold value is selected as 5 was delineated in Table 5.27. Confusion matrix as a result of classification when Retweet+Like feature threshold value is selected as 4 was delineated in Table 5.28. Confusion matrix as a result of classification when Retweet+Like feature threshold value is selected as 3 was delineated in Table 5.29. Confusion matrix as a result of classification when Retweet+Like feature threshold value is selected as 2 was delineated in Table 5.30. Confusion matrix as a result of classification when Retweet+Like feature threshold value is selected as 1 was delineated in Table 5.31.

Table 5.26. XGBoost classification results according to threshold values of the 'Retweet+Like' feature

Boundary Limit:	5	4	3	2	1
NL precision	0.69	0.69	0.68	0.68	0.68
MH precision	0.66	0.65	0.65	0.65	0.64
NL recall	0.62	0.62	0.61	0.61	0.60
MH recall	0.72	0.72	0.72	0.71	0.72
NL F1 score	0.65	0.65	0.65	0.64	0.64
MH F1 score	0.69	0.68	0.68	0.68	0.68
Accuracy:	0.6708	0.6674	0.6650	0.6633	0.6609
Total tweets:	2.204.141	2.204.141	2.204.141	2.204.141	2.204.141
NL class size	1.615.876	1.550.197	1.461.296	1.339.018	1.144.344
MH class size	588.265	653.944	742.845	865.123	1.059.797
Subsampled total tweet data	1.176.530	1.307.888	1.485.690	1.730.246	2.119.594
epoch	Default	Default	Default	Default	Default
learning rate	0.1	0.1	0.1	0.1	0.1
max depth	6	6	6	6	6
test set size	0.33	0.33	0.33	0.33	0.33

Table 5.27. Confusion matrix as a result of classification when the threshold value of the 'Retweet+Like' feature is selected as 5

	Non-low	Moderate-High
Non-low	120356	73552
Moderate-High	54268	140079

Table 5.28. Confusion matrix as a result of classification when the threshold value of the 'Retweet+Like' feature is selected as 4

	Non-low	Moderate-high
Non-low	132745	83017
Moderate-High	60554	155288

Table 5.29. Confusion matrix as a result of classification when the threshold value of the 'Retweet+Like' feature is selected as 3

	Non-low	Moderate-high
Non-low	150293	94806
Moderate-high	69436	175743

Table 5.30. Confusion matrix as a result of classification when the threshold value of the 'Retweet+Like' feature is selected as 2

	Non-low	Moderate-high
Non-low	174443	110812
Moderate-high	81457	204270

Table 5.31. Confusion matrix as a result of classification when the threshold value of the 'Retweet+Like' feature is selected as 1

	Non-low	Moderate-high
Non-low	211039	138392
Moderate-high	98797	251239

5.2.6. Results of Output Feature ‘Retweet+Reply’

XGBoost classification results of output feature ‘Retweet+Reply’ was shown in Table 5.32. For each boundary limit between 1 and 5 learning rate is 0.1, max depth is 6, and test set size is 0.33. Confusion matrix as a result of classification when Retweet+Reply feature threshold value is selected as 5 was delineated in Table 5.33. Confusion matrix as a result of classification when Retweet+Reply feature threshold value is selected as 4 was delineated in Table 5.34. Confusion matrix as a result of classification when Retweet+Reply feature threshold value is selected as 3 was delineated in Table 5.35. Confusion matrix as a result of classification when Retweet+Reply feature threshold value is selected as 2 was delineated in Table 5.36. Confusion matrix as a result of classification when Retweet+Reply feature threshold value is selected as 1 was delineated in Table 5.37.

Table 5.32. XGBoost classification results according to threshold values of the 'Retweet+Reply' feature

Boundary Limit:	5	4	3	2	1
NL precision	0.73	0.73	0.72	0.72	0.70
MH precision	0.70	0.70	0.70	0.70	0.70
NL recall	0.68	0.68	0.68	0.69	0.70
MH recall	0.75	0.74	0.74	0.73	0.70
NL F1 score	0.70	0.70	0.70	0.70	0.70
MH F1 score	0.72	0.72	0.72	0.71	0.70
Accuracy:	0.7118	0.7106	0.7091	0.7067	0.6978
Total tweets:	2.204.141	2.204.141	2.204.141	2.204.141	2.204.141
NL class size	1.896.463	1.855.496	1.799.007	1.712.725	1.564.410
MH class size	307.678	348.645	405.134	491.416	639.731
Subsampled total tweet data	615.356	697.290	810.268	982.832	1.279.462
epoch	Default	Default	Default	Default	Default
learning rate	0.1	0.1	0.1	0.1	0.1
max depth	6	6	6	6	6
test set size	0.33	0.33	0.33	0.33	0.33

Table 5.33. Confusion matrix of the classification result when the threshold value of the 'Retweet+Reply' feature is selected as 5

	Non-low	Moderate-high
Non-low	68640	32560
Moderate-high	25967	75901

Table 5.34. Confusion matrix of the classification result when the threshold value of the 'Retweet+Reply' feature is selected as 4

	Non-low	Moderate-high
Non-low	77877	37193
Moderate-high	29407	85629

Table 5.35. Confusion matrix of the classification result when the threshold value of the 'Retweet+Reply' feature is selected as 3

	Non-low	Moderate-high
Non-low	90672	42822
Moderate-high	34974	98921

Table 5.36. Confusion matrix of the classification result when the threshold value of the 'Retweet+Reply' feature is selected as 2

	Non-low	Moderate-high
Non-low	111266	50823
Moderate-high	44318	117928

Table 5.37. Confusion matrix of the classification result when the threshold value of the 'Retweet+Reply' feature is selected as 1

	Non-low	Moderate-high
Non-low	146671	64001
Moderate-high	63609	147942

As can be seen from the results of the analysis, the most successful results were achieved when the class boundaries 2 and 4 were selected in the Retweet label. Both results are similar to each other. In order to find out which is the best option, both label and threshold results were investigated on all 7 versions datasets for retweet class boundary 2 and retweet class boundary 4. Both scenarios were also analyzed with XGBoost. In these tables, IDF Range min 400 refers to the analysis performed with the dataset created with 400 words with minimum IDF value. Similarly, IDF Range 7-8 v1 refers to version 1 of the dataset containing a feature set of 400 words randomly selected from words with IDF values between 7 and 8. IDF Range 7-8 v2 refers to version 2 of the dataset containing a feature set of 400 words randomly selected from words with IDF values between 7 and 8. In other words, the random word selection process was repeated twice on the same range and each feature set was added to the dataset separately, resulting in two different dataset versions. The results are as follows.

5.3. RETWEET BOUNDRY = 2, IDF FEATURE SETS ANALYSIS

XGBoost results with Retweet Class Limit 2, IDF value minimum 400 terms: Class distribution 50% moderate-high retweet class, 50% non-low retweet class. All hyper parameters were kept the same and constant in all analyses. When retweet class limit 2 is assigned (rt>2 means moderate-high class (MH), rt<=2 means non-low class (NL)). Class distributions are as follows: Number of non-low retweet class tweets: 1.769.539 Number of moderate-high retweet class tweets: 434,602 Total number of subsampled tweets included in the analysis (50% NL retweet class, 50% MH retweet class) 869.204 tweets. The number of tweets of the MH class was equalized to the NL class. In other words, the distribution of NL and MH in the subsampled dataset was 50%-50%. The results and confusion matrices are given below.

Classification results of feature sets in different IDF value ranges with XGBoost model retweet class label threshold value 2 is delineated in Table 5.38. Confusion matrix of IDF value minimum 400 words with retweet class boundary equals to 2 is delineated in Table 5.39. Confusion matrix of words those IDF values between 7 and 8, taken random 400 words as a version 1 with retweet class boundary equals to 2 is delineated in Table 5.40. Confusion matrix of words those IDF values between 7 and 8, taken random 400 words as a version 2 with retweet class boundary equals to 2 is delineated in Table 5.41. Confusion matrix of words those IDF values between 8 and 9, taken random 400 words as a version 1 with retweet class boundary equals to 2 is delineated in Table 5.42. Confusion matrix of words those IDF values between 8 and 9, taken random 400 words as a version 2 with retweet class boundary equals to 2 is delineated in Table 5.43. Confusion matrix of words those IDF values between 9 and 10, taken random 400 words as a version 1 with retweet class boundary equals to 2 is delineated in Table 5.44. Confusion matrix of words those IDF values between 9 and 10, taken random 400 words as a version 2 with retweet class boundary equals to 2 is delineated in Table 5.45.

Table 5.38. Classification results of feature sets in different IDF value ranges with XGBoost model retweet class label threshold value 2

IDF Range	min 400	7-8 v1	7-8 v2	8-9 v1	8-9 v2	9-10 v1	9-10 v2
accuracy	0.7212	0.6965	0.6937	0.6941	0.6940	0.6918	0.6926
NL precision	0.74	0.73	0.73	0.73	0.73	0.72	0.72
MH precision	0.71	0.67	0.67	0.67	0.67	0.67	0.67
NL Recall	0.69	0.63	0.62	0.62	0.62	0.62	0.62
MH recall	0.76	0.77	0.77	0.77	0.77	0.76	0.76
NL F1-score	0.71	0.67	0.67	0.67	0.67	0.67	0.67
MH F1-score	0.73	0.72	0.71	0.71	0.71	0.71	0.71
max depth	6	6	6	6	6	6	6
learning rate	0.1	0.1	0.1	0.1	0.1	0.1	0.1
seed	42	42	42	42	42	42	42
epoch	Default	Default	Default	Default	Default	Default	Default
test size	0.33	0.33	0.33	0.33	0.33	0.33	0.33

Confusion Matrices of Analyzes in case Retweet class boundary equals to 2 confusion matrix of retweet class boundary=2, IDF min 400

Table 5.39. Confusion matrix of IDF value minimum 400 words with retweet class boundary equals to 2.

	Non-low	Moderate-high
Non-low	98546	45006
Moderate-high	34977	108309

Table 5.40. Confusion matrix of words those IDF values between 7 and 8, taken random 400 words as a version 1 with retweet class boundary equals to 2.

	Non-low	Moderate-high
Non-low	89534	53579
Moderate-high	33463	110262

Table 5.41. Confusion matrix of words those IDF values between 7 and 8, taken random 400 words as a version 2 with retweet class boundary equals to 2

	Non-low	Moderate-high
Non-low	88919	54617
Moderate-high	33238	110064

Table 5.42. Confusion matrix of words those IDF values between 8 and 9, taken random 400 words as a version 1 with retweet class boundary equals to 2

	Non-low	Moderate-high
Non-low	89209	54059
Moderate-high	33674	109896

Table 5.43. Confusion matrix of words those IDF values between 8 and 9, taken random 400 words as a version 2 with retweet class boundary equals to 2

	Non-low	Moderate-high
Non-low	89408	54210
Moderate-high	33549	109671

Table 5.44. Confusion matrix of words those IDF values between 9 and 10, taken random 400 words as a version 1 with retweet class boundary equals to 2

	Non-low	Moderate-high
Non-low	88730	54448
Moderate-high	33966	109694

Table 5.45. Confusion matrix of words those IDF values between 9 and 10, taken random 400 words as a version 2 with retweet class boundary equals to 2

	Non-low	Moderate-high
Non-low	89129	54125
Moderate-high	34056	109528

5.4. RETWEET BOUNDRY = 4, IDF FEATURE SETS ANALYSIS

XGBoost results with Retweet Class Discrimination Threshold 4, IDF value minimum 400 terms: Class distribution 50% moderate-high class, 50% non-low class. All hyperparameters were kept the same and constant in all analyses. When the retweet class limit is set to 4 (rt>4 means moderate-high (MH) class, rt<=4 means non-low (NL) class) Class distributions are as follows: non-low retweet class 1,892,923, moderate-high retweet class 311,218, homogeneous distribution retweet count taken into classification after subsample: 622,436. The number of tweet data of the moderate-high class was equalized to the non-low class. In other words, the distribution of NL and MH classes in the subsampled dataset was 50%-50%. The results and confusion matrices are given below.

Classification results of feature sets in different IDF value ranges with XGBoost model retweet class label threshold value 4 was shown in Table 5.46. Confusion matrice of IDF value minimum 400 words with retweet class boundary equals to 4 was delineated in Table 5.47. Confusion matrix of words those IDF values between 7 and 8 were delineated in Table 5.48 and Table 5.49. Confusion matrix of words those IDF values between 8 and 9, taken random 400 words as a version 1 with retweet class boundary equals to 4 was shown in Table 5.50 and taken random 400 words as a version 2 with retweet class boundary equals to 4 was shown in Table 5.51 and taken random 400 words as a version 1 with retweet class boundary equals to 4 was shown in Table 5.52 and taken random 400 words as a version 2 with retweet class boundary equals to 4 was shown in Table 5.53.

Table 5.46. Classification results of feature sets in different IDF value ranges with XGBoost model retweet class label threshold value 4

IDF Range	min 400	7-8 v1	7-8 v2	8-9 v1	8-9 v2	9-10 v1	9-10 v2
accuracy	0.7192	0.6921	0.6918	0.6909	0.6915	0.6893	0.6894
NL precision	0.74	0.73	0.73	0.73	0.73	0.73	0.73
MH precision	0.70	0.66	0.66	0.66	0.66	0.66	0.66
NL recall	0.67	0.60	0.60	0.60	0.60	0.60	0.60
MH recall	0.77	0.78	0.78	0.78	0.78	0.78	0.78
NL F1-score	0.71	0.66	0.66	0.66	0.66	0.66	0.66
MH F1score	0.73	0.72	0.72	0.72	0.72	0.72	0.72
max depth	6	6	6	6	6	6	6
learning rate	0.1	0.1	0.1	0.1	0.1	0.1	0.1
seed	42	42	42	42	42	42	42
epoch	Default	Default	Default	Default	Default	Default	Default
test size	0.33	0.33	0.33	0.33	0.33	0.33	0.33

Confusion Matrices of Analyzes in case Retweet class boundary equals to 4

Table 5.47. Confusion matrix of IDF value minimum 400 words with retweet class boundary equals to 4

	Non-low	Moderate-high
Non-low	69152	33844
Moderate-high	23843	78565

Table 5.48. Confusion matrix of words those IDF values between 7 and 8, taken random 400 words as a version 1 with retweet class boundary equals to 4

	Non-low	Moderate-high
Non-low	61939	40511
Moderate-high	22741	80213

Table 5.49. Confusion matrix of words those IDF values between 7 and 8, taken random 400 words as a version 2 with retweet class boundary equals to 4

	Non-low	Moderate-high
Non-low	61467	41091
Moderate-high	22213	80633

Table 5.50. Confusion matrix of words those IDF values between 8 and 9, taken random 400 words as a version 1 with retweet class boundary equals to 4

	Non-low	Moderate-high
Non-low	61696	40951
Moderate-high	22531	80226

Table 5.51. Confusion matrix of words those IDF values between 8 and 9, taken random 400 words as a version 2 with retweet class boundary equals to 4

	Non-low	Moderate-high
Non-low	61585	40876
Moderate-high	22487	80456

Table 5.52. Confusion matrix of words those IDF values between 9 and 10, taken random 400 words as a version 1 with retweet class boundary equals to 4

	Non-low	Moderate-high
Non-low	61493	41301
Moderate-high	22509	80101

Table 5.53 Confusion matrix of words those IDF values between 9 and 10, taken random 400 words as a version 2 with retweet class boundary equals to 4

	Non-low	Moderate-high
Non-low	61314	41119
Moderate-high	22687	80284

According to the analysis results, the best feature for the label feature is the 'Retweet' feature. The best class boundary value for the retweet feature is retweet =2. The word set that best represents the tweets is the feature set of 400 words with minimum IDF value.

6. METHODOLOGY

6.1. XGBOOST

The XGBoost model was initially introduced in the 2016 article "XGBoost: A Scalable Tree Boosting System" by Tianqi Chen and Carlos Guestrin [13]. XGBoost, which stands for "Extreme Gradient Boosting," is a gradient boosting library that has been optimized and scaled for the efficient training of machine learning models. It combines forecasts from numerous weak models to generate a more robust overall prediction. Recognized for its capability to manage large datasets and deliver top-tier performance in various tasks such as classification and regression, XGBoost has become one of the most favored machine learning algorithms. A remarkable characteristic of XGBoost is its efficient management of missing values, allowing it to function with real-world data without extensive preprocessing. It also supports parallel processing, allowing models to be trained on large datasets within a reasonable timeframe.

XGBoost is used in numerous applications, including Kaggle competitions, recommendation systems. Its high customizability allows for fine-tuning of various parameters to optimize performance. Created with a C++ library that improves the training process for gradient boosting.

In this algorithm, decision trees are built in a sequential manner. The idea of weights is essential in XGBoost, where each independent variable is given a weight and input into a decision tree to generate forecastings. Variables that are incorrectly predicted by the tree are assigned greater weights and passed on to subsequent decision trees. These individual predictors then combine to form a robust and more accurate model. XGBoost is versatile and can perform regression, classification, ranking and specialized prediction tasks [14]. Also XGBoost is also a very suitable model for feature selection.

XGBoost uses the Gradient Boosting algorithm but also has many optimization techniques. For this reason, it is called extreme gradient boosting. It is a collection of many algorithms. Comparing with Gradient Boosting algorithm, XGBoost requires much less resources. If working with billions of samples, XGBoost provides results much faster. XGBoost has L1 and L2 regularization techniques used to prevent overfit. It can also handle missing values by filling automatically missing values. Support for parallel computing allows it to work with large data sets. Contains tree pruning; this makes the tree deeper but more optimized.

The reason we chose this model is that it is one of the models that can analyze the data set best. The capacity of the XGBoost model is higher than models such as SVM. We can think of model capacity as the amount of data the model can learn from. The XGBoost model is more successful than other models in learning complex data. The main reason why we chose this model is that the data set used is large and complex. In addition, the ability to apply regularization to the model, providing GPU support, which allows parallel computing, providing tree pruning, and cache memory optimization are other advantages.

- ***Gradient Boosting Algorithm***

Gradient Boosting first makes initial predictions based on input features. It takes the mean of the errors, which is the difference between these estimates and the ground truth values. Then, it scales the new decision tree model with the learning rate and adds the average of the errors in the previous step to this new model. The learning rate is utilized to prevent the model from overfitting the training data. Then, the errors, which are the difference between the predictions of this model and the ground truth, are averaged and used when creating the next decision tree. These errors from the previous step are taken and added to the new decision tree scaled with the learning rate. Learning rate allows the real value to be approached in small steps. This process is repeated many times. We can think of this as creating a huge model by combining small models. Gradient Boost continues to construct trees in this manner until it reaches the specified number of trees or until any additional trees do not enhance the fit.

New Prediction = Previous Prediction + (Learning Rate) (New Model Prediction)*

6.2. RANDOM FOREST

Random forest is a commonly used machine learning algorithm, introduced by Leo Breiman, that integrates the output of numerous decision trees to reach at a single result [12].

The random forest algorithm consists of multiple decision trees. Each tree is built using a bootstrap sample, which is a subset of the training data selected with replacement. One-third of this sample is reserved as the out-of-bag (oob) sample for testing purposes. Additional randomness is introduced through feature bagging, enhancing diversity and reducing correlation among the trees. Depending on the task, the prediction method differs: for regression, the predictions of individual trees are averaged, while for classification, the most common class is chosen through majority voting. The oob sample is then used for cross-validation to finalize the prediction. Random Forest model structure can be seen in Figure 6.1.

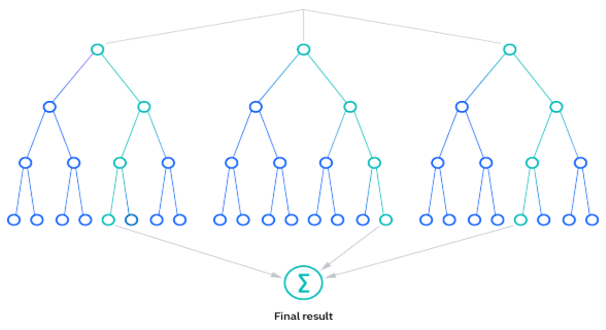


Figure 6.1. Random forest model structure [11]

6.3. LSTM

Sepp Hochreiter and Jürgen Schmidhuber made known Long Short-Term Memory (LSTM) networks in their 1997 paper titled "Long Short-Term Memory" [15]. Long Short-Term Memory (LSTM) is a form of recurrent neural network (RNN) that incorporates specialized units alongside conventional ones. LSTM units feature a "memory cell" capable of retaining information over extended durations, enabling them to grasp longer-term dependencies. LSTMs address the vanishing and exploding gradient problem by introducing new gates, such as input and forget gates, which provide better control over the information flow and enable the preservation of long-range dependencies. The core concept of LSTM is the Cell State, which acts as a communication channel and network memory, carrying meaningful information across cells to make predictions. This helps solve the short-term memory problem and allows old data to be propagated through the network. The information that the Cell State must carry is determined by the gates, which use the sigmoid activation function to decide what information is necessary or unnecessary. The forget gate decides which information to keep or discard by applying the sigmoid function to the previous cell output and current input. The input gate updates the Cell State by determining which previous and current information is important or unimportant, using both the sigmoid and tanh activation functions. The output gate determines the next cell's input and is used for making predictions, combining the sigmoid and tanh outputs. Overall, the LSTM architecture with its gating mechanism allows the network to selectively retain and propagate relevant information, enabling it to effectively capture long-term dependencies in sequential data.

6.4 Large Language Model

6.4.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) was unveiled in 2018 by Google. The research paper introducing BERT, titled "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," was written by Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova [16].

BERT is a unique language model in that it is trained to understand the context of a word by considering the words that come both before and after it. This is a key difference from previous language models, which only looked at the context in a single direction, either from left-to-right or right-to-left. The bidirectional nature of BERT's training allows it to gain a deeper, more nuanced understanding of the meaning and usage of words based on the full surrounding context. This contrasts with unidirectional models that can only leverage the context from one side, potentially missing important information that influences the meaning of a word. By taking into account the words that precede and follow a target word, BERT is able to better capture the true semantics and usage of that word. This bidirectional approach is a significant advancement over prior language models, and is a key factor behind BERT's strong performance on a variety of natural language processing tasks.

6.4.1.1. Key Features of BERT

Bidirectional Contextual Understanding: Unlike previous models that only considered context in one direction, BERT is trained to comprehend the context of a word by examining both the words preceding and following it. This allows it to capture the nuances of language more effectively.

Unsupervised Pre-training: BERT is initially trained on a vast corpus of unlabeled text data, including Wikipedia and BookCorpus. This enables it to learn general language representations that can be adapted for specific tasks.

Transfer Learning: The pre-trained BERT model can be fine-tuned for a wide range of NLP tasks, such as question answering, text classification, and named entity recognition, by adding a small number of task-specific layers. This versatility makes it a powerful tool for various applications.

State-of-the-Art Performance: BERT has achieved state-of-the-art results on various NLP benchmarks, including GLUE, SQuAD, and RACE, surpassing human performance by 2.0% on eleven tasks.

6.4.1.2. DistilBert

DistilBERT was first introduced in 2019 in the paper "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter" by Sanh et al. [17]. DistilBERT is a compact and efficient version of the BERT model, designed to be smaller, faster, and more lightweight while maintaining a significant portion of its performance. DistilBERT is 40% smaller and 60% faster than the original BERT model, while retaining an impressive 97% of its performance. This makes it an attractive option for applications where computational resources are limited or real-time processing is crucial.

6.4.1.2.1 DistilBert Turkish (DistilBERTurk)

The Turkish version of DistilBERT is called DistilBERT-base-Turkish-cased. It is a distilled version of the BERT model, specifically designed for the Turkish language. This model can be used for various NLP tasks such as text classification, named entity recognition, and question answering in Turkish.

According to its source [18], model was trained on 7GB of the original dataset used for training BERTurk, with the cased version of BERTurk serving as the teacher model. The training process utilized the official Hugging Face implementation, running for 5 days on four RTX 2080 TI GPUs.

For vector embeddings, like all of the LLMs, also DistilBERTurk uses attention mechanism. This mechanism turns words into various numbers according to their position in sentence. Because any machine learning model cannot percept words as it was. It is crucial to convert input data into numerical values. This attention mechanism has a very advantageous structure compared to previous methods. Because model can make inferences about the word according to its position in the sentences. Other models such as LSTM and RNN have no capability of achieving this using their own components.

6.5. Evaluation Metrics

The metrics we use to measure model performance are as follows:

- **Precision:** Precision is the ratio of correctly predicted positive observations to the total predicted positives. It measures the accuracy of the positive predictions.

Precision = (True Positives) / (True Positives + False Positives) (4)

- **Sensitivity (Recall)** : (true positive rate) refers to the probability of a positive test, conditioned on truly being positive.

Recall = (True Positives) / (True Positives + False Negatives) (5)

- **Accuracy** Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, calculated as following definition:

Accuracy = (True Positives + True Negatives) / (Total Predictions) (6)

- **F1 Score :** F1 Score is the harmonic mean of precision and recall. It provides a single metric that balances both the concerns of precision and recall. It is especially useful when the class distribution is imbalanced.

F1 Score = 2 X ((Precision X Recall) / (Precision + Recall)) (7)

- **AUC**: AUC stands for ”Area under the ROC Curve.” That is, AUC measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1).

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

- **ROC curve:** An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

$TPR = TP / (TP + FN)$ (8)

False Positive Rate (FPR) is defined as follows:

$FPR = FP / (FP + TN)$ (9)

7. RESULTS

XGBoost, Random Forest, LSTM and DistilBERTurk models were utilized. The XGBoost model gave the best result. XGBoost was also the fastest running model among these models.

7.1 Results of XGBOOST

The hyperparameters were set as follows: test set size 33%, Learning rate 0.2, max depth 20, booster gbtree, device gpu, tree method gpu_hist. The results are as follows*: Non-low class* precision 0.75, recall 0.70, F1-score 0.73, *moderate-high class* precision 0.72, recall 0.77, F1-score 0.74. Average accuracy 0.7340 and ROC-AUC score 0.81. ROC-AUC graph of XGBoost can be seen in Figure 7.1.

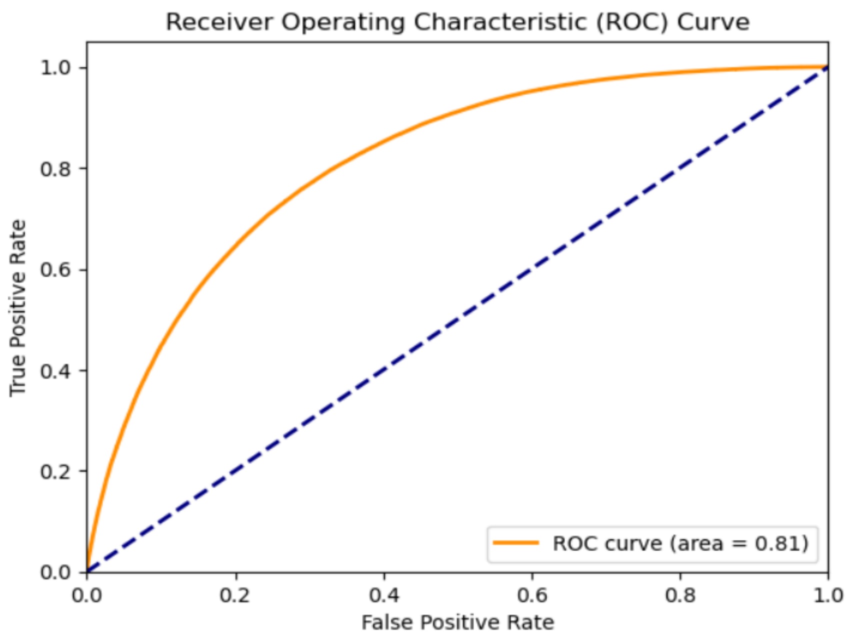


Figure 7.1. ROC-AUC graph of XGBoost

Table 7.1. Confusion matrix of XGBoost

	NL	**MH**
NL	100687	42693
MH	33623	109835

7.2. Results of LSTM

As a dataset, the data with the 400 terms with the lowest IDF value is taken. These 400 terms best represent the tweet text data in the dataset. Since these 400 features are given, the text is not given. Retweet label class discrimination limit is set as 2. The test set size was set as 33%. 20 epochs were applied. Optimizer parameter was set as 'adam' and loss parameter was set as binary cross entropy. The activation function was set as sigmoid. ROC was calculated as 0.80. The classification report according to the classes is as follows; precision 0.75, recall 0.69, F1-score 0.72 for *non-low class*, precision 0.71, recall 0.77, F1 score 0.74 for *moderate-high class*. Model accuracy was calculated as 0.7311 and ROC-AUC Score as 0.8049. Train-Validation accuracy graph of LSTM can be seen in Figure 7.2. ROC-AUC curve of LSTM can be seen in Figure 7.3.

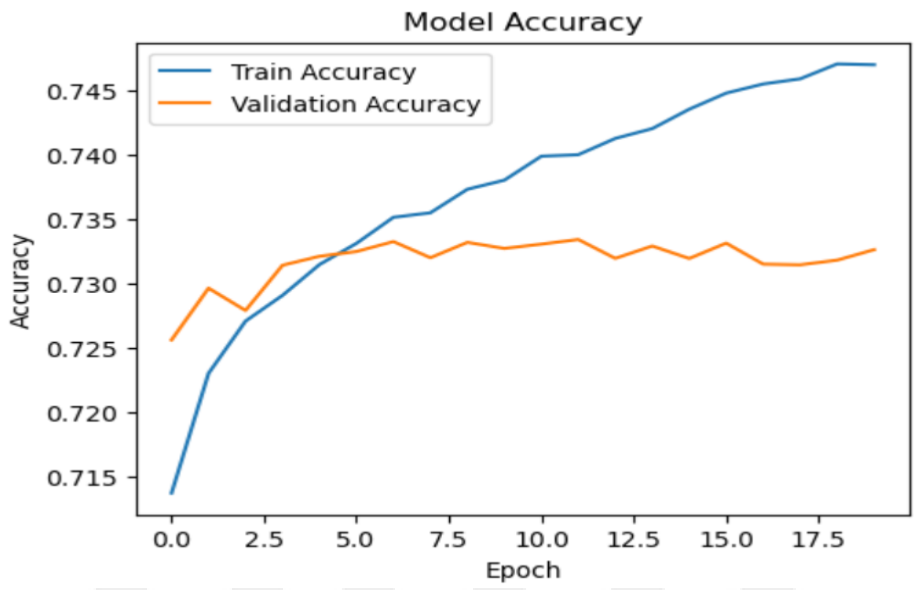


Figure 7.2. Train-Validation accuracy graph of LSTM

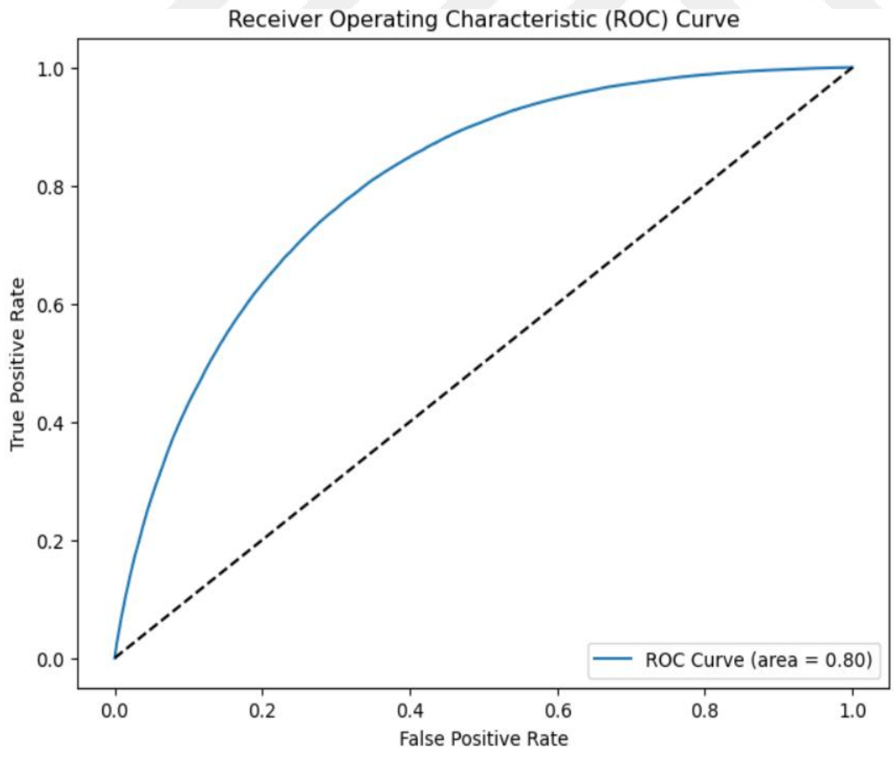


Figure 7.3. ROC-AUC curve of LSTM

Table 7.2: Confusion matrix of LSTM model

	NL	**MH**
NL	99945	43503
MH	33623	109767

7.3. Results of Random Forest

As a dataset, the data with the 400 terms with the lowest IDF value is given. These 400 terms best represent the tweet text data in the dataset. Since these 400 features are given, the Text feature was not given. Retweet class limit is set as 2. The test set size was set as 33%. The number of trees in the forest is set to 20, max depth is set to 6 and other parameters are left as default. The test accuracy value was calculated as 0.6855. The classification report according to classes is as follows. *Non-low* class precision value was 0.67, recall value was 0.75, F1-score value was 0.70. *Moderate-high* class precision value was 0.71, recall value was 0.62, F1-score value was 0.66. ROC-AUC score was calculated as 0.7555. ROC-AUC curve of Random Forest model can be seen in Figure 7.4.

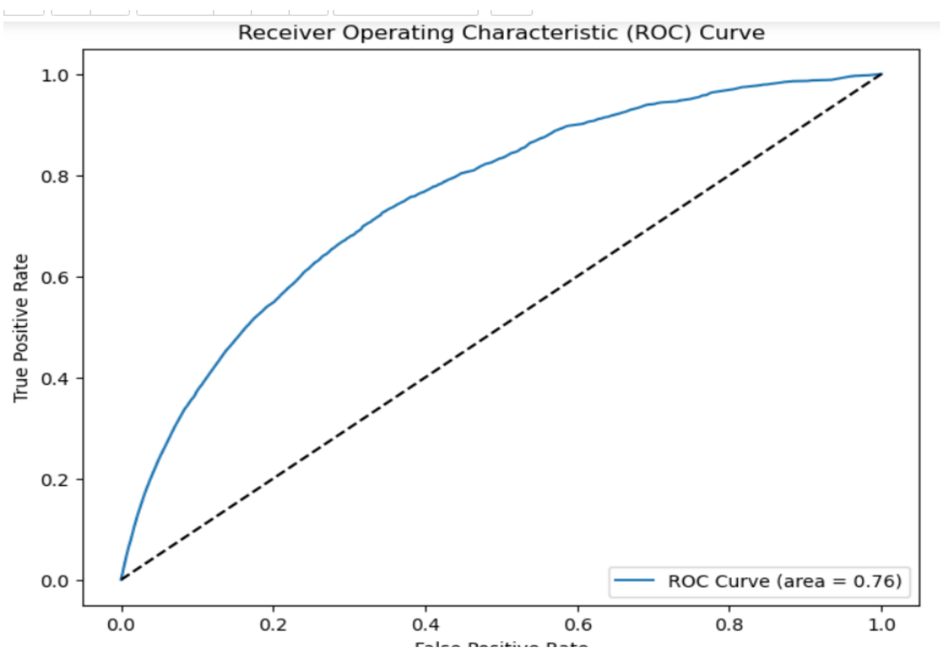


Figure 7.4. ROC-AUC curve of Random Forest model

Table 7.3. Confusion matrix of Random Forest

	NL	**MH**
NL	107245	36203
MH	53995	89395

7.4. Results of DistilBerTurk

In the analysis with LLM, the feature set with a minimum IDF value of 400 words could not be used. 'Text' feature was given instead. LLM vectorizes each word in the text data in the dataset by itself with the attention mask structure. 10 epochs were applied. The test set size was set to 33%. Batch size is 8, max_len is 128, drop out probability is 0.2. The model consists of 6 layers. To improve performance, all 6 layers of this pre-trained model set up frozen. The number of dominant class tweets was limited to the number of minority class tweets. No scaling was applied to the dataset. 'Adam' optimizer was chosen as the optimizer and the learning rate was 2e-5. The data for the first week is given along with the tweet text data. The results are as follows: *Non-low class* precision 0.72, recall 0.70, F1-score 0.71, *Moderate-high class* precision 0.71, recall 0.73, F1 score 0.72. Overall accuracy calculated as 0.71. DistilBERTurk train-validation accuracy graph and ROC-AUC score can be seen in Figure 7.5 and Figure 7.6 respectively.

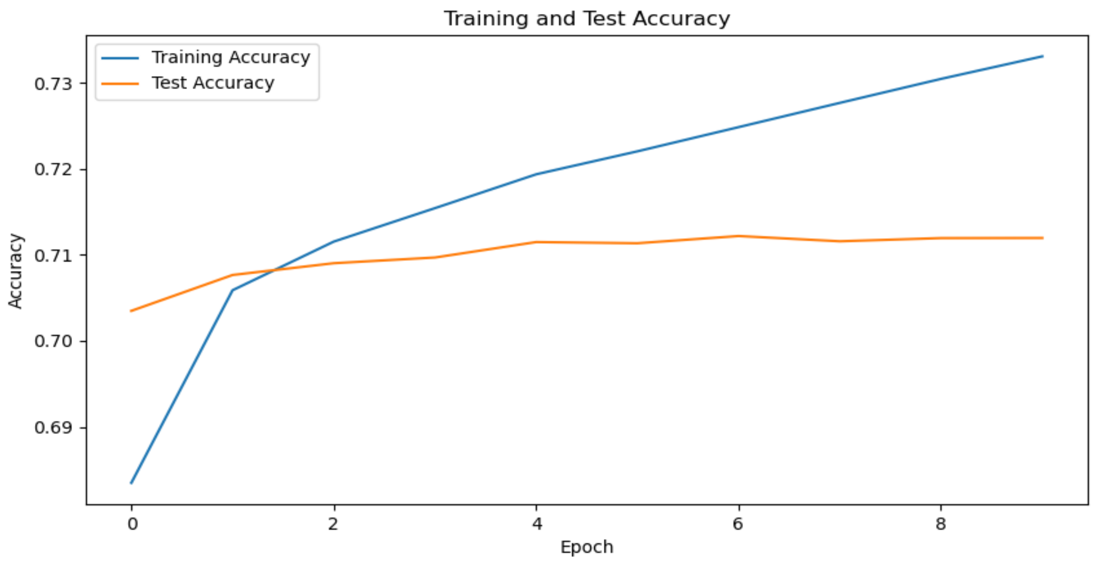


Figure 7.5. DistilBERTurk train-validation accuracy through 10 epochs

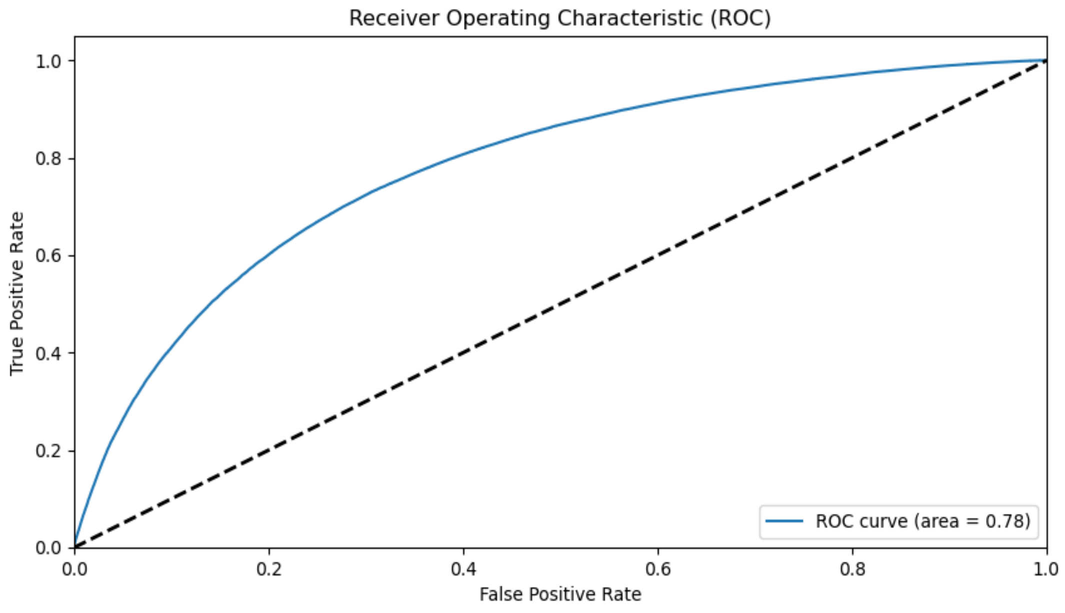


Figure 7.6. ROC-AUC score of the DistilBERTurk model

Table 7.4. Confusion matrix of DistilBERTurk model

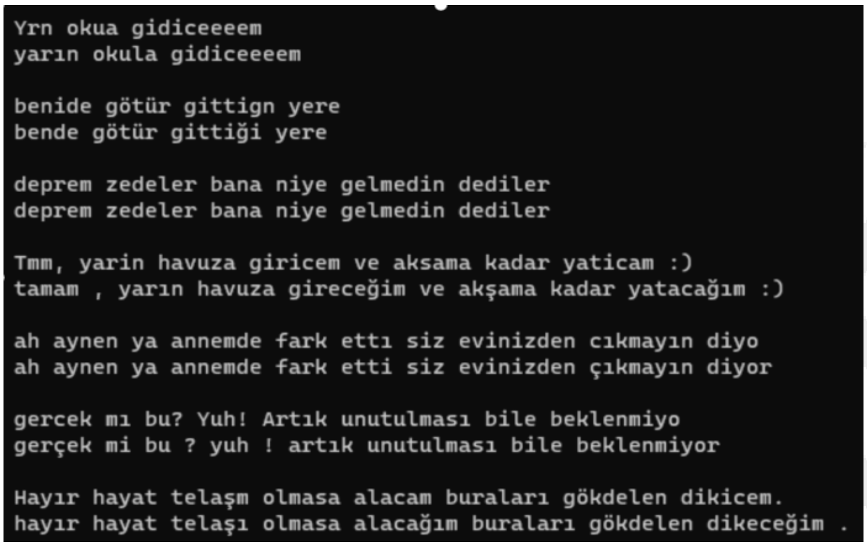
	NL	**MH**
NL	99911	43537
MH	39084	104306

8. DISCUSSION AND CONCLUSION

The earthquakes centered in Kahramanmaras reverberated around the world. The dataset contains tweet text in many different languages, including Turkish. There is no comprehensive library to separate Turkish words from other words during preprocessing of the data. While the data was retrieved, the location where the tweet was shared was also not available. Therefore, it was not possible to completely clean the tweet text from irrelevant words. This means that we have very noisy data. It was also observed that frequency-based word cleaning does not always help model performance. This was especially more noticeable when working with LLMs.

It was observed that Zemberek treated some foreign language words as Turkish language. After the natural disaster, people called for help on social media. When sending a tweet on Twitter, the post has a character limit. If a tweet exceeding this limit then cannot be post. Therefore, some people who wanted to call for help without getting stuck in the character limit wrote their messages without spaces. Zemberek also lacks the functionality to add spaces where necessary in messages written without spaces. It leaves the words it cannot fix as they are. Similarly, Zemberek lacks the ability to spell out words written as abbreviations. In addition, considering the misspellings in the whole text data, misspellings in abbreviations will also be observed.

In addition, it was observed that Zemberek worked incorrectly in some places. The expression 'Ali Sunal'ın' became 'ali sunalın' after Zemberek implemented. For this reason, expressions such as 'ın' and 'in' have been also added to the stop words in the stop words deletion phase to be applied after the lemmatization process. It was also observed that the Zemberek library recognized some foreign words as Turkish and changed them. The Zemberek library does not have a feature that can detect whether the written word belongs to Turkish or not. Although there are modules in Python that can detect this, they are usually designed for English. The module for Turkish is quite weak. In addition, Zemberek is able to correct two repeated letters but not three or more repeated vowels. This can be seen in Figure 8.1.

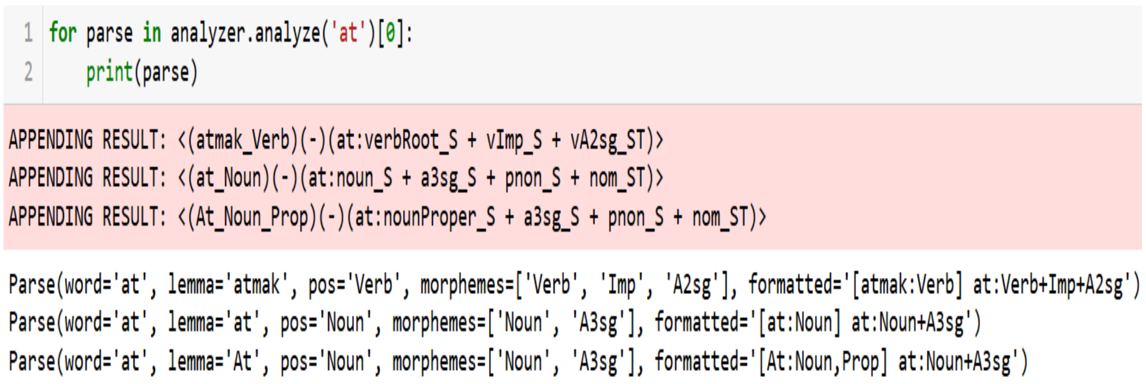


```
Yrn okua gidiceeeem
yarın okula gidiceeeem

benide götür gittign yere
bende götür gittiği yere

deprem zedeler bana niye gelmedin dediler
deprem zedeler bana niye gelmedin dediler

Tmm, yarin havuza giricem ve aksama kadar yaticam :)
tamam , yarın havuza gireceğim ve akşama kadar yatacağım :)

ah aynen ya annemde fark ettı siz evinizden cıkmayın diyo
ah aynen ya annemde fark etti siz evinizden çıkmayın diyor

gercek mı bu? Yuh! Artık unutulması bile beklenmiyo
gerçek mi bu ? yuh ! artık unutulması bile beklenmiyor

Hayır hayat telaşm olmasa alacam buraları gökdelen dikicem.
hayır hayat telaşı olmasa alacağım buraları gökdelen dikeceğim .
```

Figure 8.1. Inabilities of zemberek module

Zemberek has reduced the number of unique words in its database from millions to hundreds of thousands with its spelling correction process. However, zemberek has not a perfect performance. It is not possible to manually process hundreds of thousands of unique words one by one in a very limited time. Therefore, this directly affects the quality of the data. This is affected success of the model. If it was still possible to capture location data as in the past, it would be possible to remove foreign words from dataset completely.

Some weak points were observed while also working with Zeyrek. It may choose the wrong word from the alternatives it generated for possible stemming words. For this reason, we decided to choose the word with the highest length among the alternatives as the candidate root word. However, this decision cannot always be the best decision. In addition, Zeyrek, just like Zemberek, shows similar behavior about words written without spaces. It leaves the word as it is or assigns an incorrect candidate root word. Similarly, it may perceive a word in a tweet written in a foreign language as a Turkish word. For example, the word 'at' in a tweet written as 'we are at Hatay' is analyzed as shown in Figure 8.2.



```
for parse in analyzer.analyze('at')[0]:
    print(parse)
```

```
APPENDING RESULT: <(atmak_Verb)(-)(at:verbRoot_S + vImp_S + vA2sg_ST)>
APPENDING RESULT: <(at_Noun)(-)(at:noun_S + a3sg_S + pnon_S + nom_ST)>
APPENDING RESULT: <(At_Noun_Prop)(-)(at:nounProper_S + a3sg_S + pnon_S + nom_ST)>

Parse(word='at', lemma='atmak', pos='Verb', morphemes=['Verb', 'Imp', 'A2sg'], formatted='[atmak:Verb] at:Verb+Imp+A2sg')
Parse(word='at', lemma='at', pos='Noun', morphemes=['Noun', 'A3sg'], formatted='[at:Noun] at:Noun+A3sg')
Parse(word='at', lemma='At', pos='Noun', morphemes=['Noun', 'A3sg'], formatted='[At:Noun,Prop] at:Noun+A3sg')
```

Figure 8.2. Inabilities of zeyrek module

In this study, we worked with real world data. In real world data, data imbalanced situation is often observed. Existing machine learning methods cannot deal effectively with imbalanced data. The model works with the assumption that the data is homogeneously distributed among classes. Therefore, it tends to learn the dominant class and not the minority class. There are solutions to this problem with different methods. Some researchers have tried to find a balance in the distribution of data between classes by increasing the amount of available data with oversample methods. Some researchers have continued the analysis by taking smaller pieces of the large data set. This situation was also encountered in this study. Since the data set is quite large and unbalanced, we chose the subsample method. In this study, we predicted the level of retweet engagement that a tweet would receive within the tweet ecosystem. We also evaluated the performance of automatic vectorization by large language models (LLMs) and compared it with a dataset manually vectorized using the binary bag of words technique, which represents the tweet corpus with a maximum feature count of 400 words that can be added. Through the classification process, we observed that the model performance of XGBoost remains advantageous even in the era we mention large language models.

9. REFERENCES

[1] Daga, I., Gupta, A., Vardhan, R., Mukherjee, P. (2020). Prediction of Likes and Retweets Using Text Information Retrieval, Procedia Computer Science, Volume 168, Pages 123-128, ISSN 1877-0509, DOI: https://doi.org/10.1016/j.procs.2020.02.273.

[2] Chen, L., Deng, H., (2020) Predicting User Retweeting Behavior in Social Networks with a Novel Ensemble Learning Approach journal name IEEE Access DOI: 10.1109/ACCESS.2020.3015397

[3] Roy, S., Suman, B., K., Chandra, J., Dandapat, S., K. (2020). Forecasting the Future: Leveraging RNN based Feature Concatenation for Tweet Outbreak Prediction. In Proceedings of the 7th ACM IKDD CoDS and 25th COMAD (CoDS COMAD 2020). Association for Computing Machinery, New York, NY, USA, 219–223. https://doi.org/10.1145/3371158.3371190

[4] Wang, J., Yang, Y., Tweet Retweet Prediction Based on Deep Multitask Learning, Neural Processing Letters DOI: https://doi.org/10.1007/s11063-021-10642-3 Year 2021

[5] Sharma, S., Gupta, V., Role of twitter user profile features in retweet prediction for big data streams, Multimedia Tools and Applications (2022) 81:27309–27338 DOI https://doi.org/10.1007/s11042-022-12815-1, 2022

[6] Fu, X., Cheng, S., Zhao, L., Lv, J., Retweet Prediction Based on Multidimensional Features, Wireless Communications and Mobile Computing DOI https://doi.org/10.1155/2022/1863568, 2022

[7] [online]. Available: *(https://unicode.org/emoji/charts/full-emoji-list.html)* *(*Acces Date: April 2023*)*

[8] AA Akın, MD Akın, Zemberek, an open source NLP framework for Turkic languages, Structure 10(2007), 1-5

[9] [online]. Available: https://github.com/ahmetaa/zemberek-nlp (Access Date: November 2023)

[10] [online]. Available https://github.com/obulat/zeyrek (Access Date: November 2023)

[11][online]. Available https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems (Access Date: June 2024)

[12] Random Forests, L. Breiman, Machine Learning, Volume 45, pages 5-32, , October 2001. DOI: https://doi.org/10.1023/A:1010933404324

[13] XGBoost: A Scalable Tree Boosting System, Tianqi Chen, Carlos Guestring, KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining August 2016 Pages 785–794 DOI : https://doi.org/10.1145/2939672.2939785

[14] [online]. Available: https://www.geeksforgeeks.org/xgboost/ (Access Date: April 2024)

[15] Hochreiter, S., Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780.

[16]: Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.

[17] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

[18] [online] Available: https://huggingface.co/dbmdz/distilbert-base-turkish-cased (Access Date: May 2024)

[19] Cohen, K. B. (2014). Chapter 6 - Biomedical Natural Language Processing and Text Mining, Editor(s): Indra Neil Sarkar, Methods in Biomedical Informatics, Academic Press, 2014, Pages 141-177, ISBN 9780124016781, DOI: https://doi.org/10.1016/B978-0-12-401678-1.00006-3.

[20] [online] Available: https://monkeylearn.com/blog/what-is-tf-idf/ (Access Date: June 2024)

[21] Spärck Jones, K. (1972). "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". *Journal of Documentation*. **28** (1): 11–21. CiteSeerX 10.1.1.115.8343. doi:10.1108/eb026526. S2CID 2996187.

[22] Salton, G., Wong, A., Yang, C. S. (1975), "A Vector Space Model for Automatic Indexing," *Communications of the ACM*, vol. 18, nr. 11, pages 613–620.

[23] Sammon, J. W. , SOME MATHEMATICS OF INFORMATION STORAGE AND RETRIEVAL, TECHNICAL REPORT NO. RADC-TR- 68-178 June 1968

[24][online] Available: https://en.wikipedia.org/wiki/Bag-of-words_model#cite_note-3 (Access Date: December 2023)

[25] Zellig H. (1954). "Distributional Structure". *Word*. **10** (2/3): 146–62. doi:10.1080/00437956.1954.11659520.

[26] [online] Available: https://medium.com/algorithms-data-structures/bag-of-words-nedir-32b1f0017a95 (Access Date: May 2024)

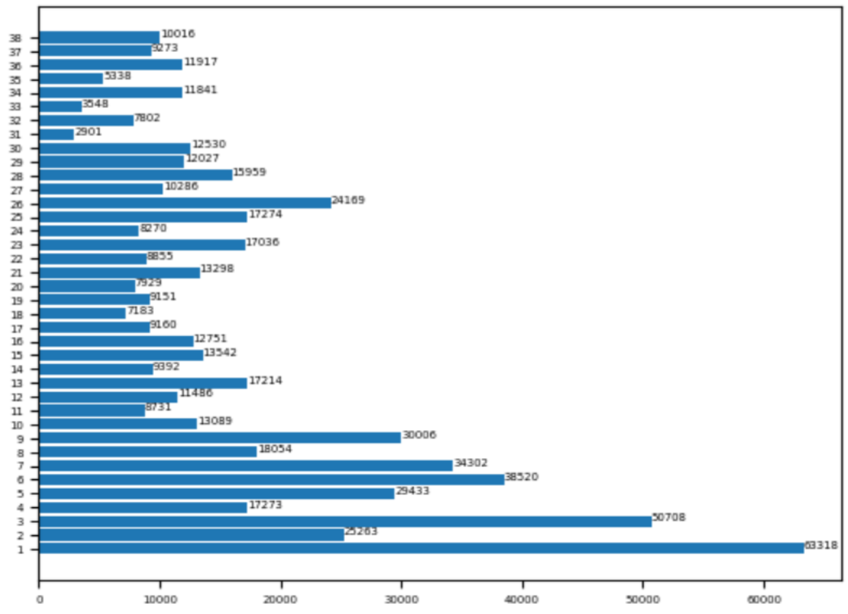
[27][online] Available: What is R Squared? R2 Value Meaning and Definition (freecodecamp.org) (Access Date: June 2023)

APPENDIXES

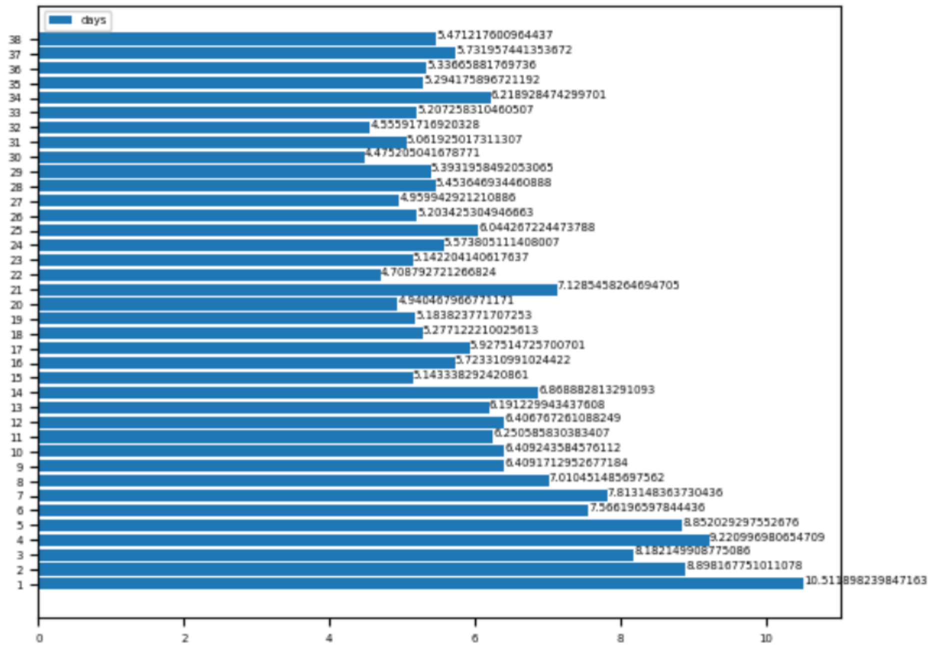
APPENDIX 1 - Interaction Statistics

1. Statistics for the Retweet feature

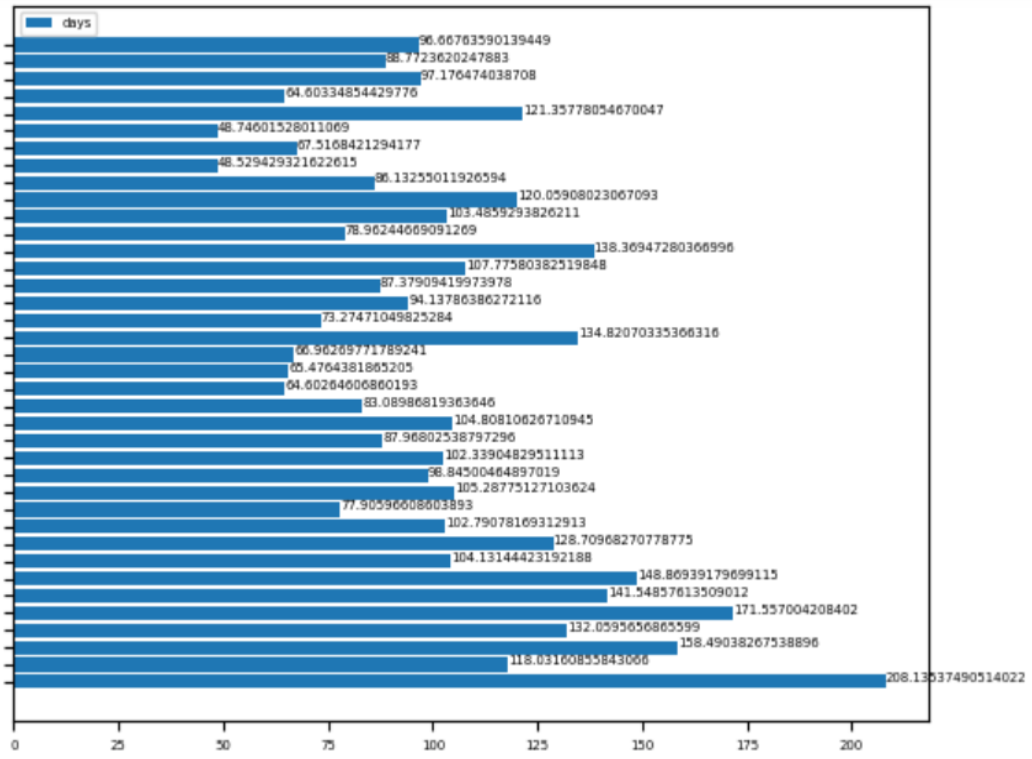
1.a. Distribution of the maximum number of retweets by days



1. b. Distribution of Mean of Retweets by days(retweet.mean())

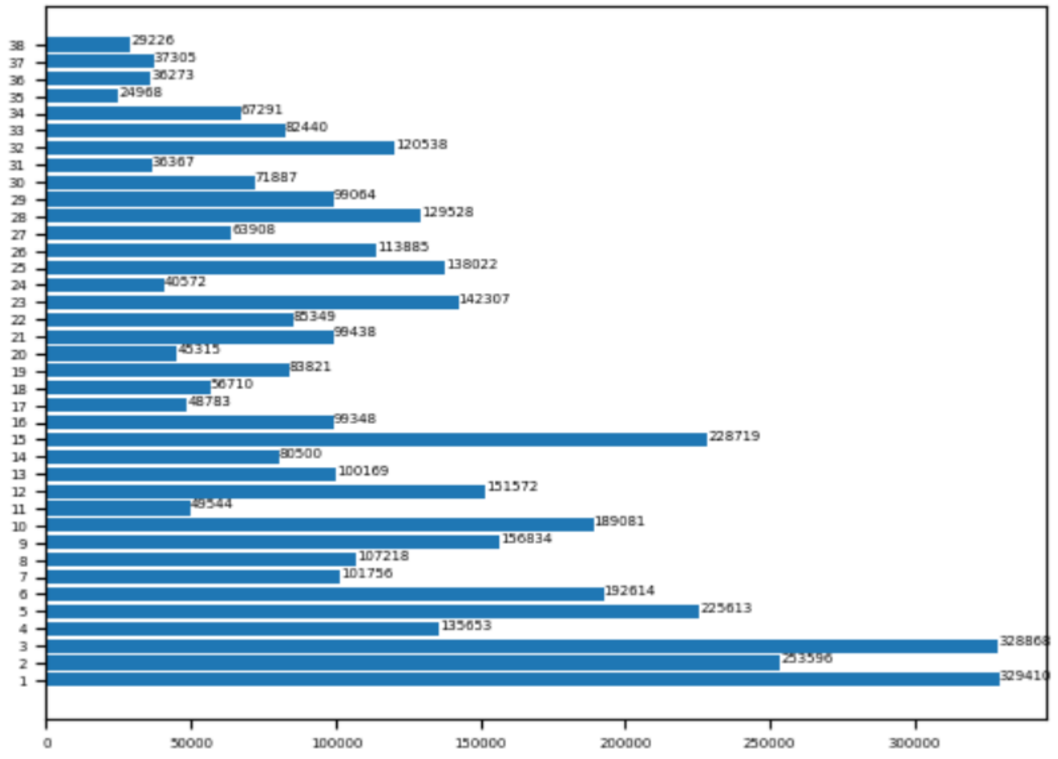


1.c. Distribution of Standard Deviation of retweets by days (retweet.std())

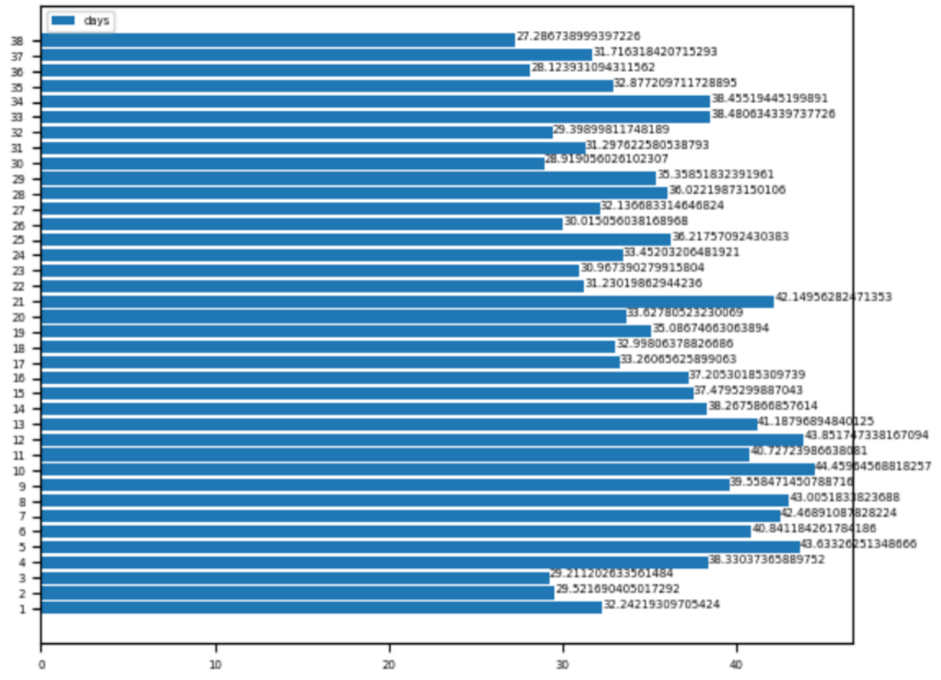


2. Statistics for the ‘like’ feature

2.a. Distribution of maximum like value by days

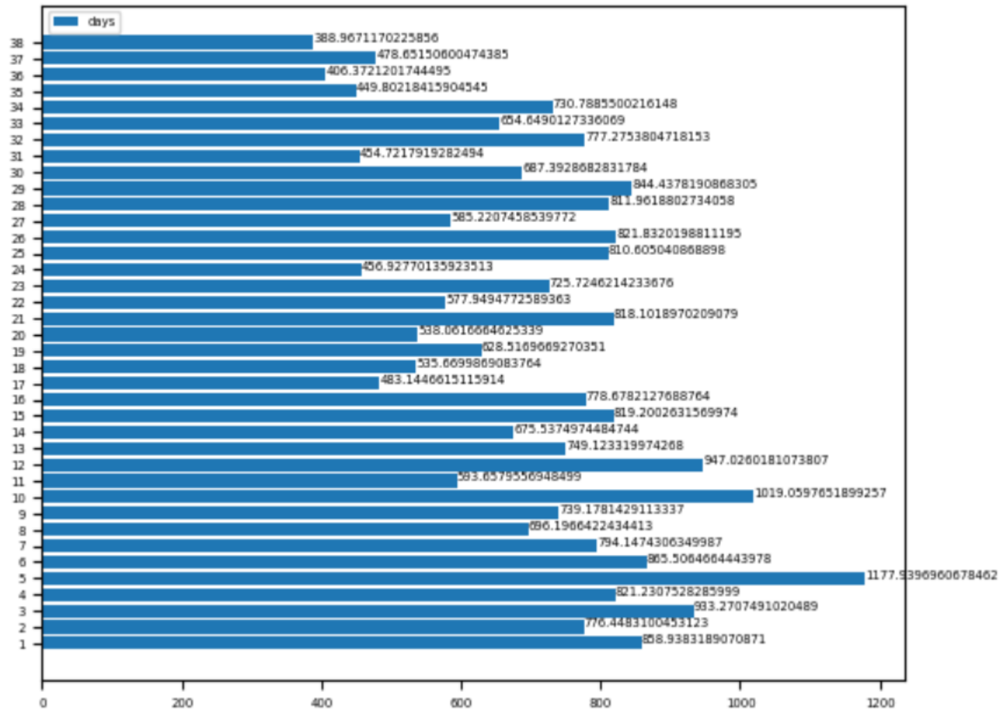


2. b. Distribution of mean value of like by days



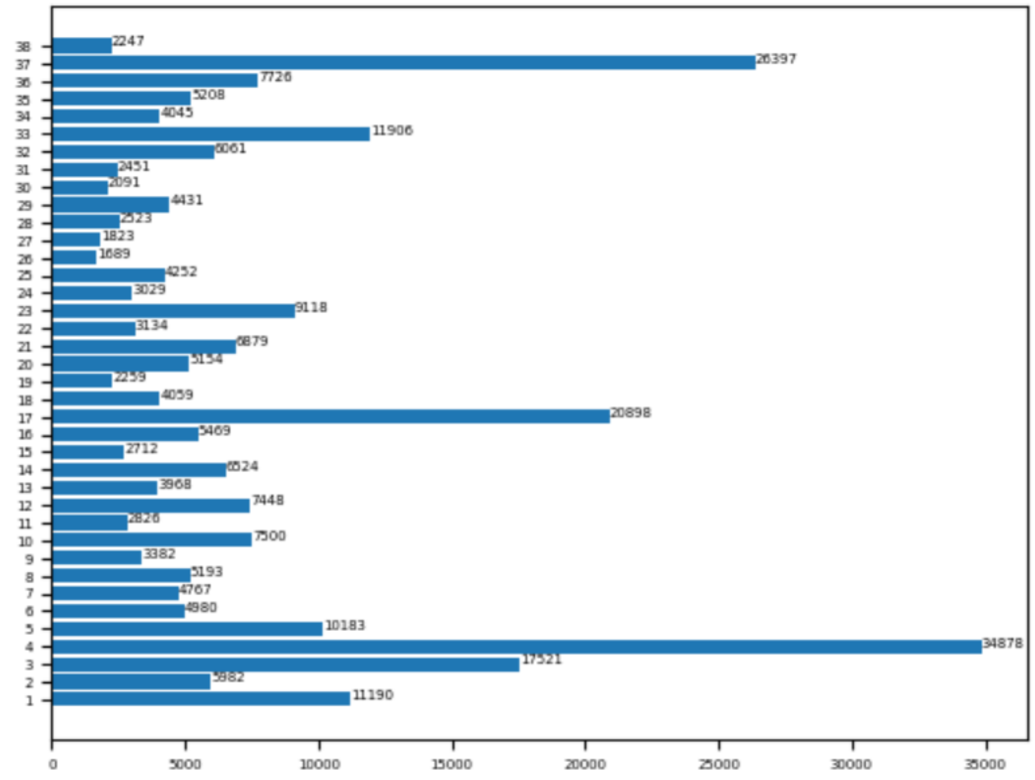
(Note: the reason why the like value for the first day seems to be relatively low is that the number of tweets for that day is much higher compared to other days.)

2.c) Standard deviation of like counts by day:

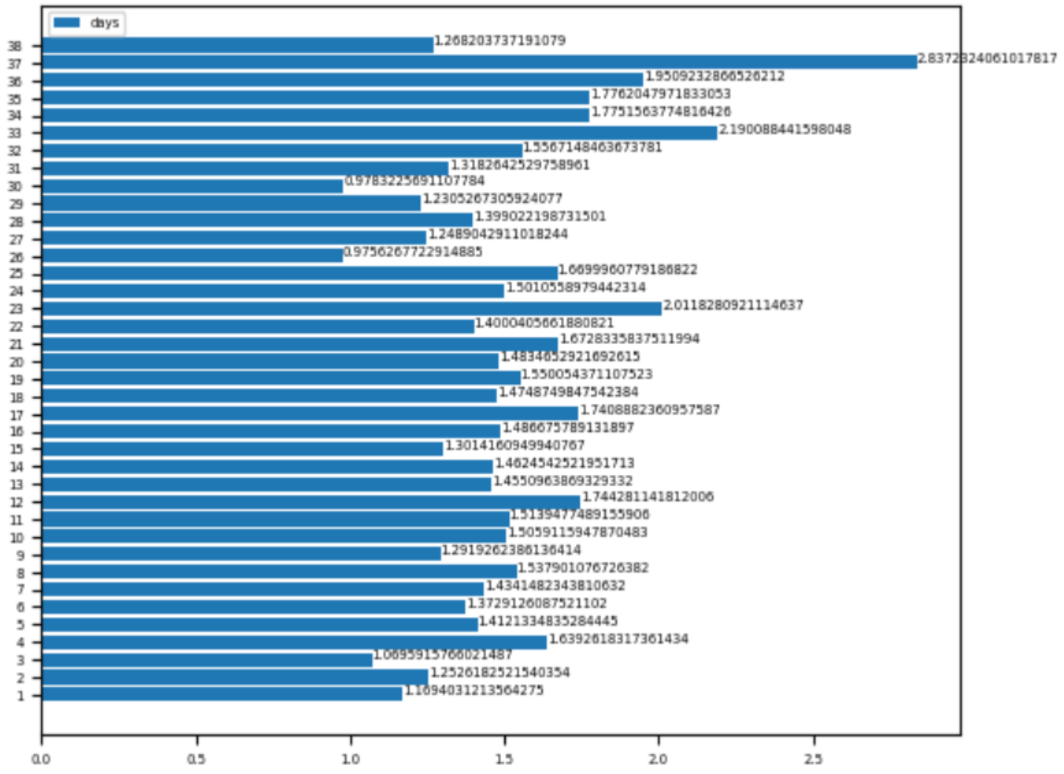


3. Statistics for the Reply feature

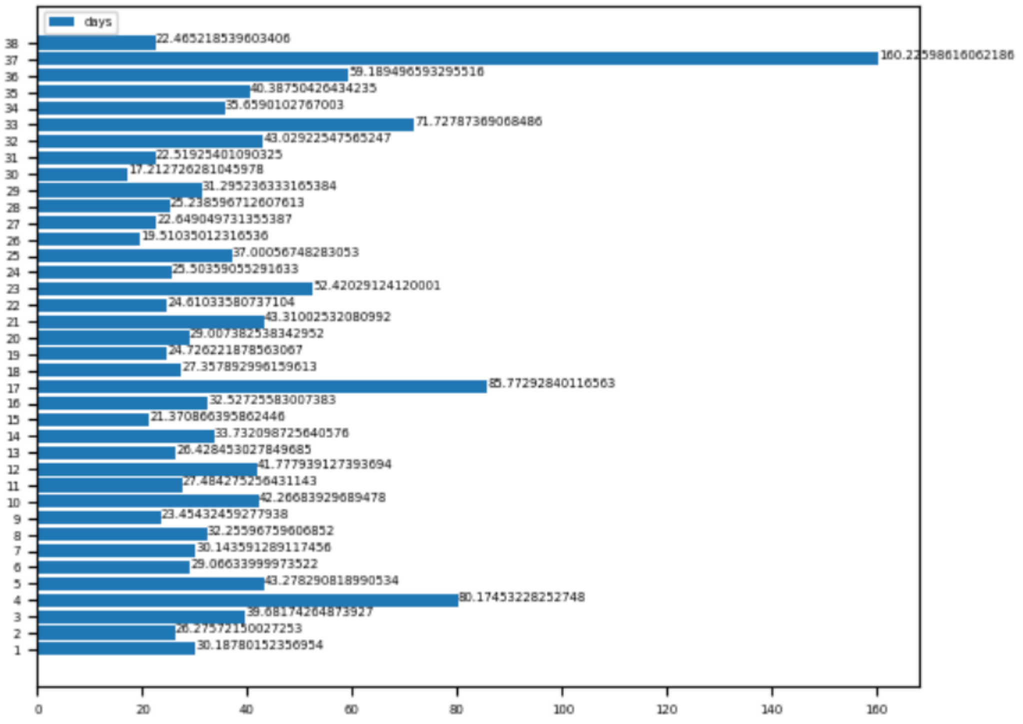
3.a. Distribution of maximum reply value by days



3. b. Distribution of mean value of reply count by days

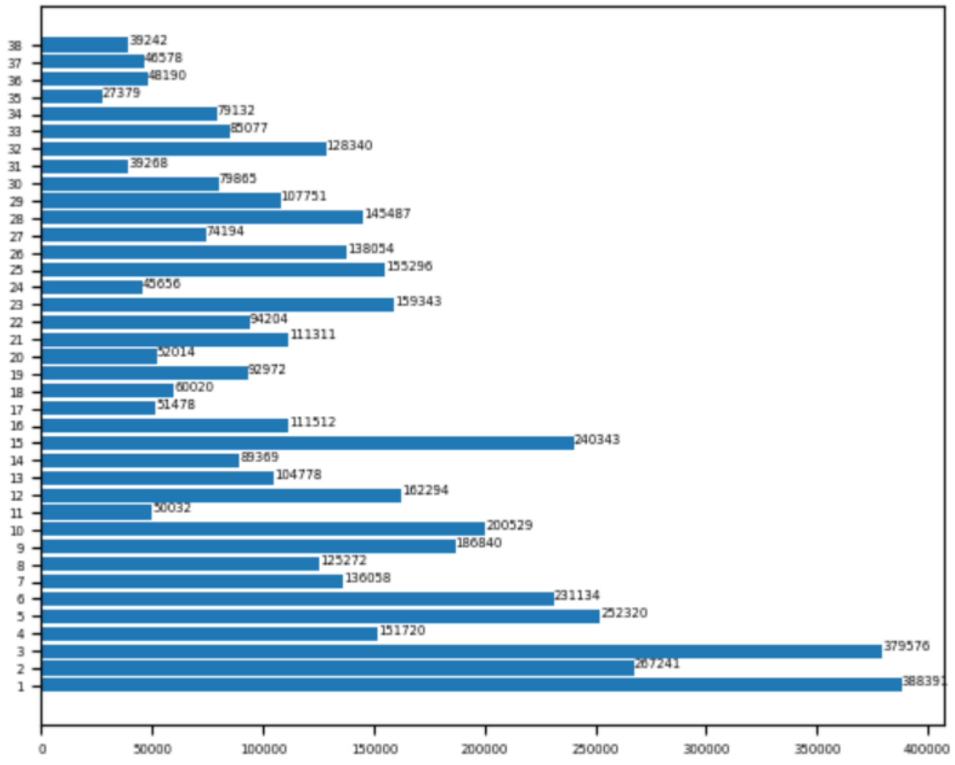


3.c. Distribution of standard deviation value of reply count by days

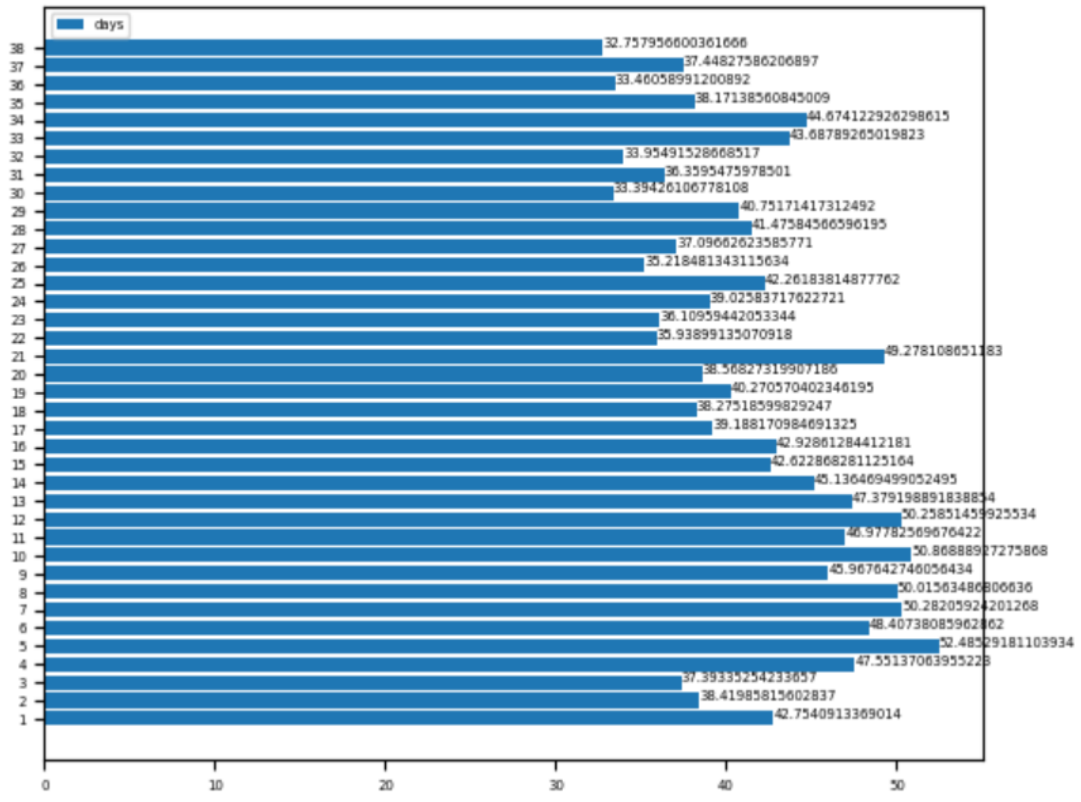


4. Retweet + Like feature statistics

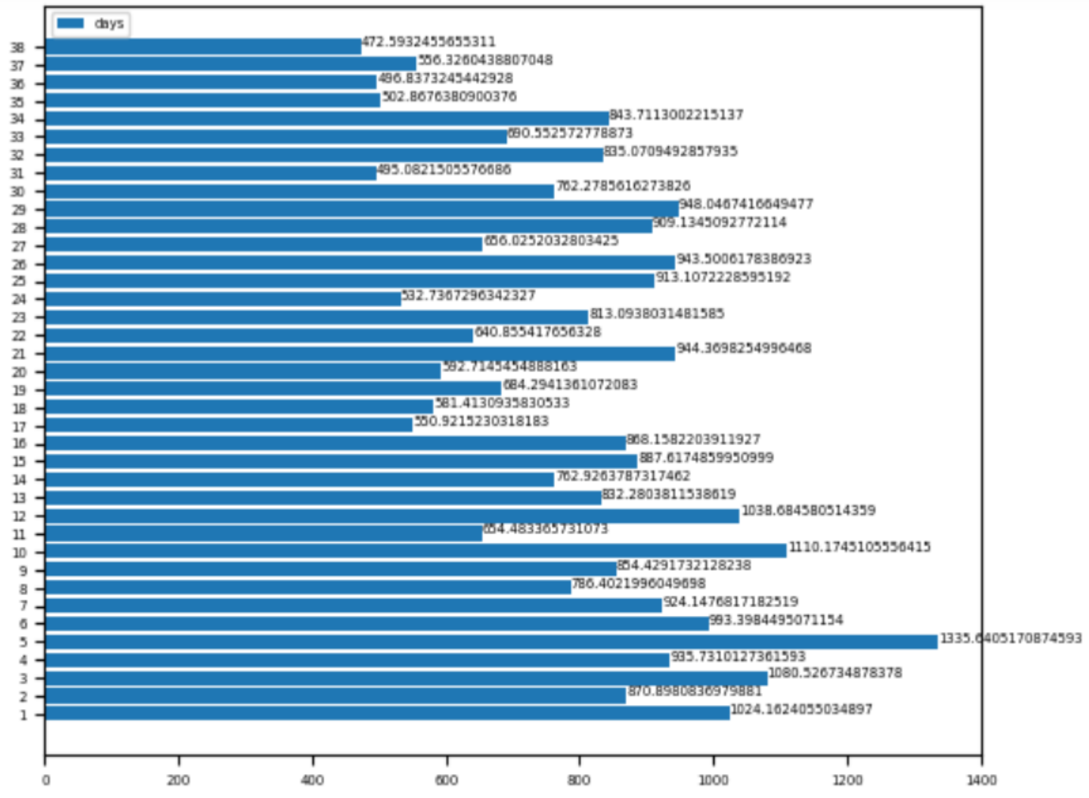
4.a. Distribution of maximum Retweet+Like value by days:



4. b. Distribution of mean value of Retweet+Like feature by days:

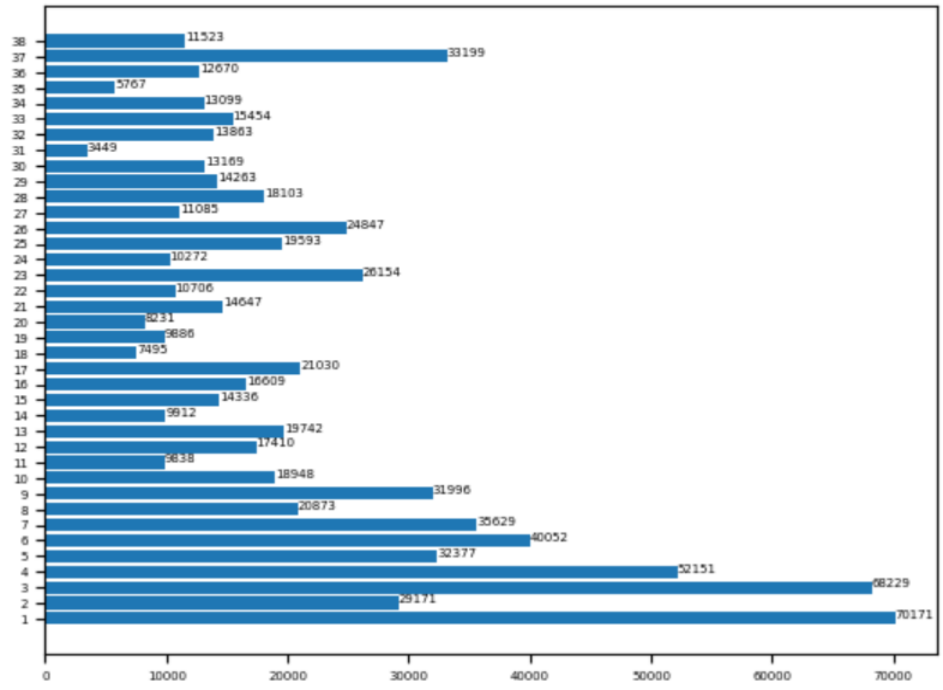


4.c. Distribution of standard deviation value of Retweet+Like feature by days

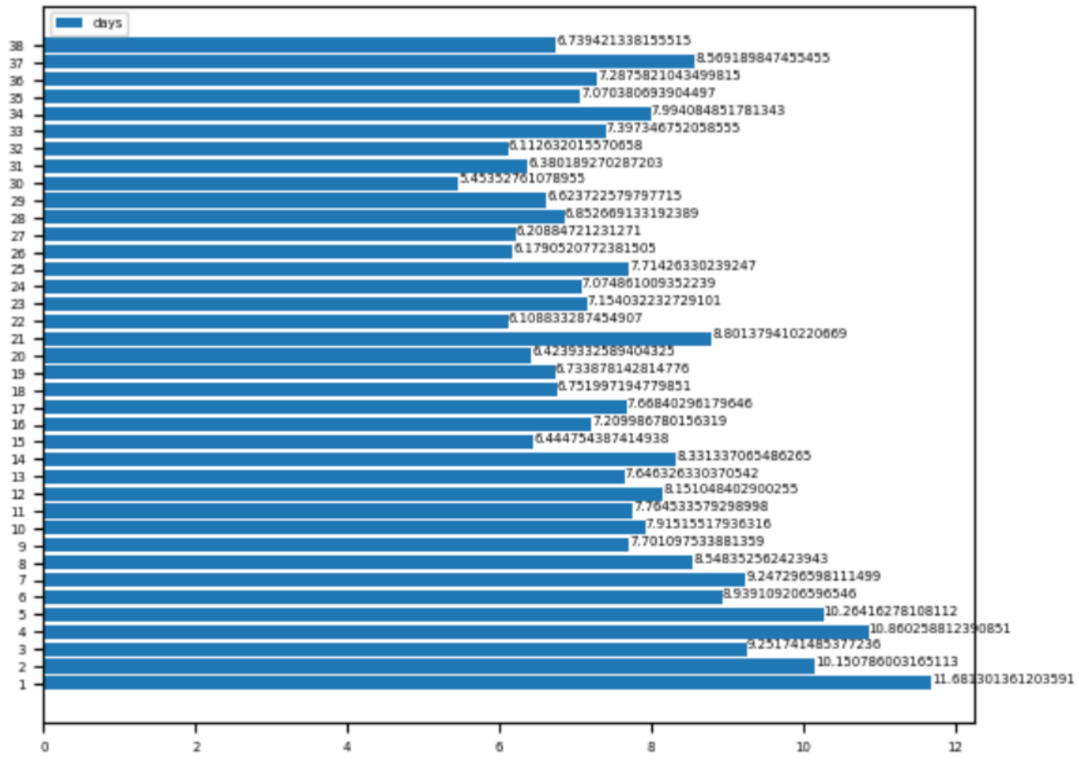


5. Retweet+Reply Feature Statistics

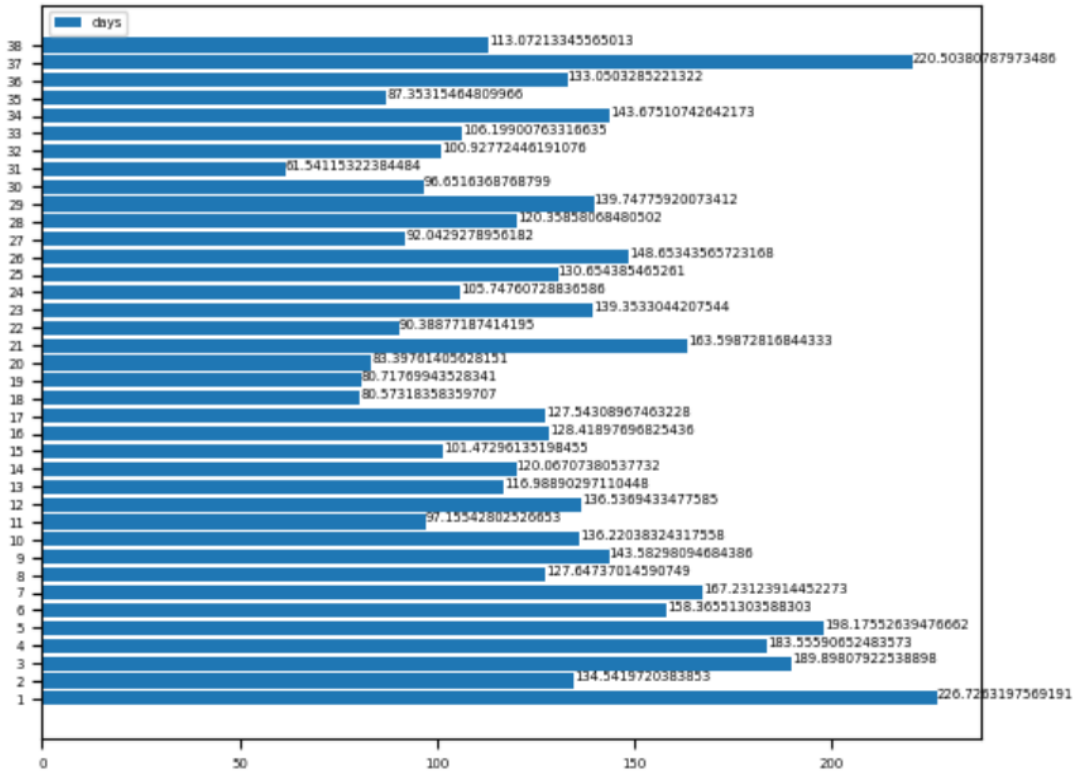
5.a. Distribution of max value of Retweet+Reply feature by days



5.b. Distribution of mean value of Retweet+Reply feature by days:

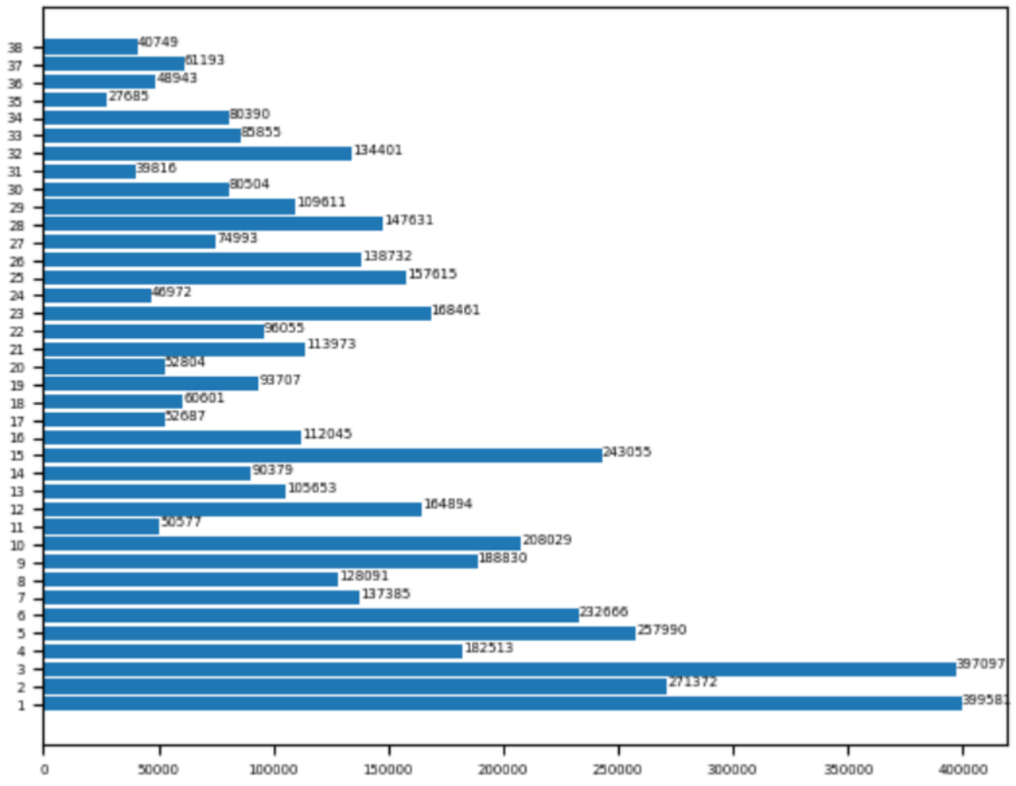


5.c. Distribution of standard deviation value of Retweet+Reply feature by days:

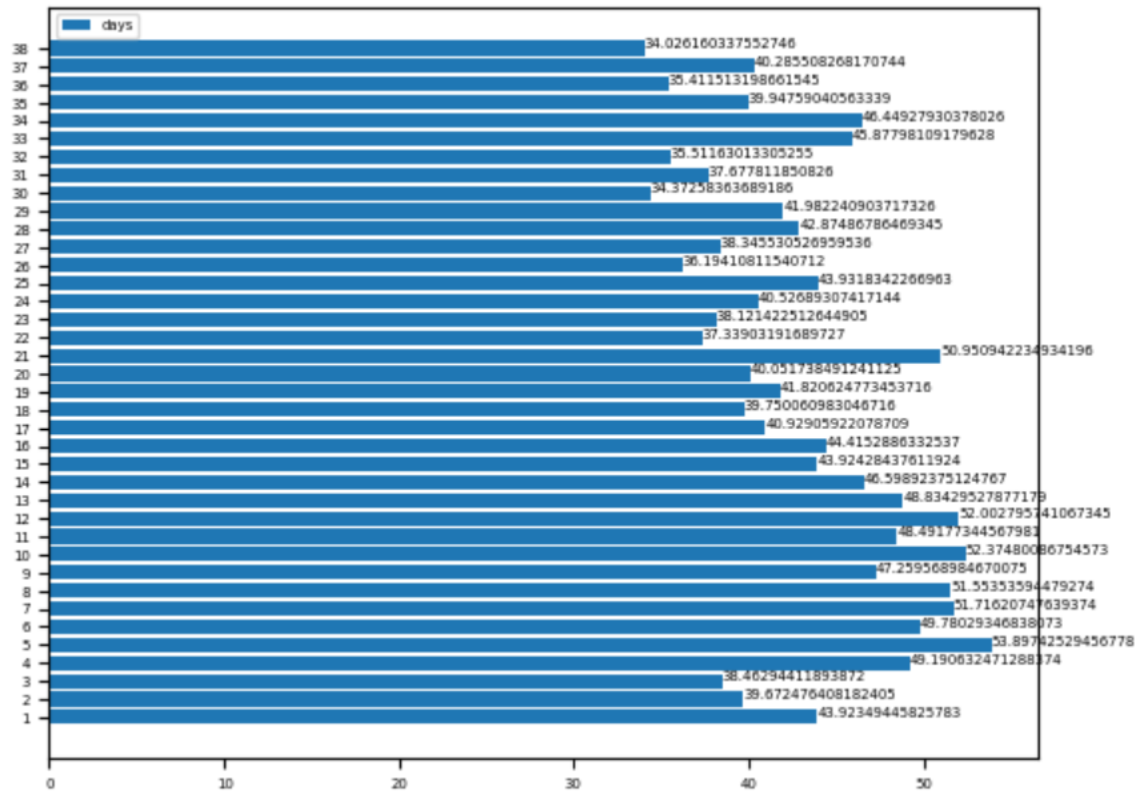


6. Statistics of the Retweet+Like+Reply feature

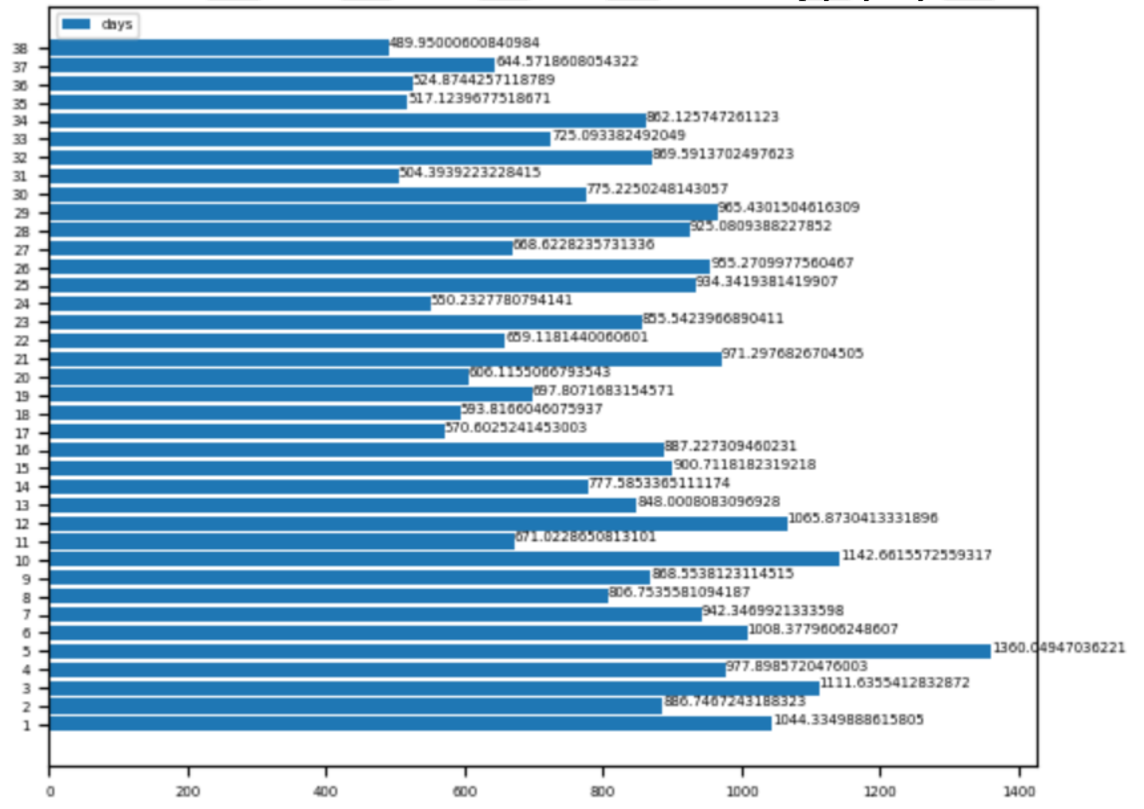
6.a. Distribution of maximum count of retweet+like+reply by days



6.b. Distribution of mean value of retweet+like+reply by days



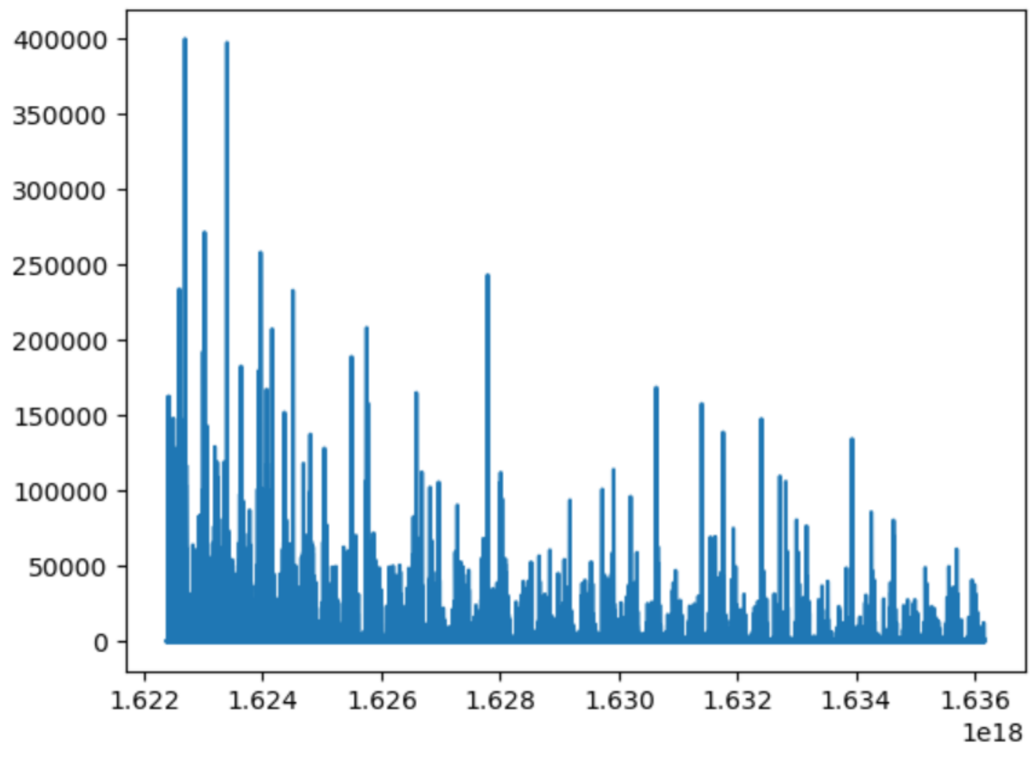
6.c. Distribution of standard deviation value of retweet+like+reply by days



7. Distribution of tweet id and retweet+like+reply count together

First, data from 38 days were combined. Then the retweet+like+reply value was sorted in descending order. Finally, tweet ids were sorted in descending order. Then a graph was drawn:

This graph shows that we are working with a very complex data set.
(y-axis indicates the number of retweets+like+reply, x-axis indicates tweet ids)



source code:

```
depremtwt = [gun1, gun2, gun3, gun4, gun5, gun6, gun7, gun8, gun9, gun10, gun11,
gun12, gun13, gun14, gun15, gun16, gun17, gun18, gun19, gun20, gun21, gun22,
gun23, gun24, gun25, gun26, gun27, gun28, gun29, gun30, gun31, gun32, gun33,
gun34, gun35, gun36, gun37, gun38]
dtweet = pd.concat(depremtwt)
dtweet=dtweet.sort_values('Retweet+Like+Reply', ascending=False).copy()
dtweet=dtweet.sort_values('Tweet Id', ascending=False).copy()
TWEETID=dtweet["Tweet Id"]
RLR=dtweet["Retweet+Like+Reply"]
# Create a line plot
plt.plot(TWEETID, RLR)
```

APPENDIX 2 - STOP WORDS REMOVE PROCESS DETAILS

The earthquake disaster reverberated across many countries and nationalities, and an international mobilization was declared. Since the dataset contains tweets sent from all over the world in many different languages besides Turkish, it became essential to remove stop words in many languages. Since the Twitter API does not have access to the location of the tweets at the time of tweet data extraction, it is not possible to clean tweets in different languages by location. The tweets in Russian, Japanese, Chinese, Arabic and Sanskrit were cleaned with the python regex module based on the unicode values of their alphabets. In addition, for languages using the Latin alphabet, the remaining stop words were cleaned by excluding words that have a Turkish equivalent.

Operations performed:
1- Turkish stopwords such as be, etmek, yap etc. were discarded
2- Stopwords for German, English, French, Slovak, Romanian, Spanish, Portuguese, Italian, Latvian, Irish, Hungarian, Czech, Czech, Croatian, Dutch, Danish, Norwegian, Swedish, Estonian, Indonesian, Polish, Vietnamese, Turkish (another stopwords list was obtained from a different source), Bulgarian were deleted. While deleting these stopwords, care was taken to leave the words in the word list that have Turkish equivalents intact. For example, examples of words common to Turkish in the Indonesian list but removed from the list: "ada, agar, akan, anda, asal, beri, betul, bulan, Cuma, dini, diri, ini, itu, kamu, kan, kapan, kini, kira, kita, masa

English stopwords sources:
https://www.kaggle.com/datasets/rowhitswami/stopwords
not removed: 'et', 'his'
https://github.com/stopwords-iso/stopwords-l/blob/master/stopwords-sl.txt

Slovak stopwords
https://github.com/stopwords-iso/stopwords-k/blob/master/stopwords-sk.txt
not removed: mal, mala, asi, ani, no

Romanian stopwords
https://github.com/stopwords-iso/stopwords-ro/blob/master/stopwords-ro.txt
not removed: acele, al, ala, alt, alta, ar, as, asa, aş, cam, din, incit, insa, isi, iti, parca, şase

Spanish stopwords
https://github.com/Alir3z4/stop-words/blob/master/spanish.txt
not removed: ahi, al, así, el, en, es, haber, han, hasta, has, son, soy, su, sus, suya,un, una, usan, van, ver, no, para, sola, solo, tus

Portuguese stopwords
https://github.com/stopwords-iso/stopwords-pt/blob/master/stopwords-pt.txt
not removed: alem, ali, além, as, dar, deste, deve, devem, diz, dizer, dizem, ela, ele, em, forma, iste, mil, nas, nem, para, ser, sim, tenha, tente, ter, um, ver

Italian stopwords
https://github.com/stopwords-iso/stopwords-it/blob/master/stopwords-it.txt

not removed: ad, al, all, ansa, ben, bene, dagli, dal, dalla, dall, del, dell, deve, ed, eri, fare, feci, in, sara, su, un, una

Latvian stopwords
https://github.com/stopwords-iso/stopwords-lv/blob/master/stopwords-lv.txt
not removed: ar, bet, bez

Irish (ga) stopwords
https://github.com/stopwords-iso/stopwords-ga/blob/master/stopwords-ga.txt
not removed: um, ins, in, gur, dar, ar

Hungarian stopwords
https://github.com/stopwords-iso/stopwords-hu/blob/master/stopwords-hu.txt
not removed: akik, akkor, akár, alá, az, bal, ban, el, ellen, elém, ez,ezen, ezer, hanem, hat, hol, ide, is, ismét, kin, kire, kit, kérem, mert, min, nem, oda, sok, sokan, után, utána, veled, értem, és, év, éve, évi, évvel

Czech stopwords
https://github.com/stopwords-iso/stopwords-cs/blob/master/stopwords-cs.txt
not removed: tim, tema, ten, tam, tak, pod, on, nam, email, bez, beze, az

Croatian-stopwords
https://github.com/stopwords-iso/stopwords-hr/blob/master/stopwords-hr.txt
not removed: ali, ili, iz, on, ona, ova, pak, pod, su, uz

Dutch-stopwords
https://github.com/stopwords-iso/stopwords-nl/blob/master/stopwords-nl.txt
not removed words: af, al, deden, en, er, had, hem, hun, in, is, kan, kon, men, nam, net, onder

Danish stopwords
https://github.com/stopwords-iso/stopwords-da/blob/master/stopwords-da.txt
not removed: ad, af, alt, at, dem, der, din, dine, en, er, et, far, fire,
ham, han, har, her, hun, kan, lav, ses, sin, ser, sine, sit, var

Norwegian-stopwords
https://github.com/stopwords-iso/stopwords-no/blob/master/stopwords-no.txt
not removed: at, av, dem, der, dere, din, en, ene, er, et, han, har, her, hun, med, min, mine, sin, som, var

Swedish-stopwords
https://github.com/stopwords-iso/stopwords-sv/blob/master/stopwords-sv.txt
not removed words: att, av, dag, del, delen, dem, din, en, er, han, har, in, kan, med, min, sade, sin, ur, var

German-stopwords
https://github.com/solariz/german_stopwords/blob/master/german_stopwords_full.txt
https://github.com/stopwords-iso/stopwords-de/blob/master/stopwords-de.txt
not removed: an, bin, dem, denen, der, deren, direkt, er, es, hat, in, ins, ist, mal, mit, nur, tat, total, tut

French-stopwords
https://github.com/stopwords-iso/stopwords-fr/blob/master/stopwords-fr.txt
not removed: as, bas, bat, en, et, es, hem, hep, il, meme, mine, on,
pas, pire, sera, ses, son, stop, tel, telle, tente, un

Estonian-stopwords
https://github.com/kristel-/estonian-stopwords/blob/master/estonian-stopwords-lemmas.txt
not removed: alt, taha, ette, saati, tasa, kus, aga, et, at, taga, mil, ah,
luks, ah, pot, ports, ust, utu, kes, mis

Indonesian -stopwords
https://github.com/stopwords-iso/stopwords-id/blob/master/stopwords-id.txt
not removed: ada, agar, akan, anda, asal, beri, betul, bulan, cuma, dini, diri, ini, itu,
kamu kan kapan kini kira kita masa misal sela tak tanya yakin

Polish- stopwords
https://github.com/bieli/stopwords/blob/master/polish.stopwords.txt
not removed: ani az bez iz jest nam nas pod tak taka tam

Turkish -stopwords- second source except Zemberek
https://github.com/stopwords-iso/stopwords-tr/blob/master/stopwords-tr.txt
There are not common words with Turkish.

Vietnamese Stop words
https://github.com/stopwords/vietnamese-stopwords/blob/master/vietnamese-stopwords.txt
There are not common words with Turkish.

The dataset states after the stop words are removed are as follows:

In this table #tweets: the number of tweets from that day, #words: the total number of words in tweets from that day, #word variations: the number of word variations in tweets from that day. (count of unique words)

#days: the number of the day to which that data belongs. The number 1 refers to the data containing tweets dated February 6, 2023, and the number 38 refers to the data containing tweets dated March 15, 2023.
#words: the total number of words in the data for that day
#word_variations: total number of unique words in the data for that day
#tweets: total number of tweets in the data for that day
NOTE: At this stage, the number of words in tweets and the number of words with repeated letters have not yet been added as features, but they will be added in the next stage. Afterwards, the total number of words, the number of tweets and the number of word variants of the whole dataset were investigated:

Table 5: Distribution of the status of the dataset according to days after the stopwords are removed

Days	#words	#word_variations (#unique words)	#tweets	word count mean	std deviation of word count
1	7.447.017	118.630	702.042	10,60	6,74
2	5.077.794	105.331	422.596	12,01	6,90
3	3.339.712	79.638	278.368	11,99	6,63
4	3.195.528	78.622	249.868	12,78	6,96
5	2.592.907	70.304	209.490	12,37	6,98
6	2.508.691	65.911	191.841	13,07	6,91
7	2.044.547	61.631	154.116	13,26	7,04
8	1.701.136	56.075	129.456	13,14	7,04
9	1.486.393	51.966	112.209	13,24	7,00
10	1.343.949	49.065	103.927	12,93	6,96
11	1.031.775	41.555	80.020	12,89	7,07
12	997.107	39.635	76.357	13,05	7,16
13	897.441	37.951	69.172	12,97	7,31
14	915.210	38.211	69.017	13,26	7,10
15	1.534.669	47.832	144.328	10,63	6,99
16	1.129.114	43.124	86.066	13,11	7,11
17	858.522	36.911	65.932	13,02	7,16
18	851.177	35.188	65.477	12,99	7,26
19	768.280	32.662	60.572	12,68	7,12
20	709.423	32.381	56.355	12,58	7,30
21	773.379	32.818	55.169	14,01	7,13
22	841.596	34.752	68.898	12,21	7,32
23	761.581	33.800	63.573	11,97	7,43
24	608.988	30.252	46.325	13,14	7,33
25	515.522	27.470	38.180	13,50	7,25
26	599.806	27.612	44.393	13,51	6,96
27	520.770	26.002	39.172	13,29	7,14
28	494.822	25.807	37.738	13,11	7,31

29	477.473	24.912	37.987	12,56	7,04
30	605.967	24.249	44.602	13,58	7,30
31	439.270	21.912	30.283	14,50	7,17
32	388.904	22.673	31.300	12,42	7,28
33	417.346	22.990	32.726	12,75	7,12
34	382.198	21.689	29.376	13,01	7,19
35	369.179	21.478	27.241	13,55	7,30
36	428.466	22.469	32213	13,30	7,08
37	397.184	22.271	31159	12,74	7,26
38	448.407	24.031	33133	13,53	7,08

APPENDIX 3 - DATA PREPROCESSING BEFORE AFTER COMPARISON TABLES

Table 4: Distribution of the change of processed data according to days after cleaning of raw data and clean tweet text

	#tweets (raw data)	#words (raw data)	#word_variations (raw data)	#tweets (cleaned)	#words (cleaned)	#word_variations (cleaned)
day 1	844.184	14.230.211	1.099.688	711.870	11.382.495	447.300
day 2	535.763	10.021.869	922.574	426.525	7.667.863	402.830
day 3	336.985	6.451.997	653.592	280.077	4.991.235	315.093
day 4	297.338	6.060.550	661.075	251.379	4.833.321	321.379
day 5	230.574	4.735.968	571.571	210.393	3.933.932	290.133
day 6	204.591	4.446.350	542.600	192.525	3.799.628	280.511
day 7	161.251	3.631.681	494.593	154.620	3.180.490	263.301
day 8	133.759	2.978.686	443.889	129.838	2.654.812	238.798
day 9	115.405	2.591.847	400.670	112.525	2.324.841	219.844
day 10	107.899	2.358.700	375.654	104.202	2.101.266	204.565
day 11	82.066	1.811.689	313.739	80.228	1.621.905	175.037
day 12	77.931	1.721.375	298.281	76.545	1.551.454	166.700
day 13	70.719	1.547.224	279.372	69.304	1.394.266	159.651

day 14	70.429	1.586.584	280.665	69.129	1.433.042	163.111
day 15	148.660	2.789.325	379.275	145.188	2.480.955	203.665
day 16	87.563	1.958.595	325.300	86.234	1.774.426	184.332
day 17	67.046	1.474.965	270.304	66.041	1.324.446	154.055
day 18	66.467	1.452.092	260.013	65.592	1.300.211	147.804
day 19	61.477	1.325.186	241.585	60.694	1.186.997	138.119
day 20	57.244	1.221.246	231.372	56.457	1.100.586	134.881
day 21	55.933	1.319.705	235.856	55.241	1.206.175	140.350
day 22	69.852	1.457.866	253.216	69.023	1.310.299	145.027
day 23	64.351	1.297.842	238.233	63.662	1.162.870	139.263
day 24	46.981	1.035.670	208.743	46.406	927.289	122.650
day 25	38.681	879.303	187.781	38.245	791.237	111.429
day 26	44.847	1.009.755	180.115	44.434	886.057	109.049
day 27	39.748	892.312	173.571	39.244	802.130	106.138
day 28	38.318	855.761	175.035	37.840	772.103	107.767
day 29	38.611	830.372	168.893	38.065	744.484	102.793
day 30	45.262	1.016.396	163.314	44.747	891.086	96.812
day 31	30.610	719.815	142.042	30.327	640.676	86.279
day 32	31.624	666.067	148.905	31.341	593.139	90.621
day 33	33.092	713.015	153.166	32.790	638.674	92.530
day 34	29.725	653.994	141.114	29.416	583.611	88.188
day 35	27.575	628.929	139.986	27.266	566.389	87.579
day 36	32.568	741.224	147.347	32.276	650.250	90.576
day 37	31.527	688.465	146.048	31.204	609.476	88.576
day 38	33.511	771.239	155.473	33.180	695.844	95.384

Table 6: Distribution of the change of the data given as input to Zemberek and the data received as output from Zemberek according to days

	#words (zemberek b)	**#word_variations (zemberek b)**	**#words (zemberek a)**	**#word_variations (zemberek a)**
day 1	11.382.495	447.300	11.343.465	283.481
day 2	7.667.863	402.830	7.654.217	258.525
day 3	4.991.235	315.093	4.979.755	206.649
day 4	4.833.321	321.379	4.822.021	213.301
day 5	3.933.932	290.133	3.923.196	196.689
day 6	3.799.628	280.511	3.790.213	191.755
day 7	3.180.490	263.301	3.172.799	181.217
day 8	2.654.812	238.798	2.646.009	167.119
day 9	2.324.841	219.844	2.317.329	155.756
day 10	2.101.266	204.565	2.093.724	145.314
day 11	1.621.905	175.037	1.616.284	126.133
day 12	1.551.454	166.700	1.545.311	121.015
day 13	1.394.266	159.651	1.389.384	116.221
day 14	1.433.042	163.111	1.429.388	118.003
day 15	2.480.955	203.665	2.469.910	142.890
day 16	1.774.426	184.332	1.768.565	132.339
day 17	1.324.446	154.055	1.319.416	112.523
day 18	1.300.211	147.804	1.294.212	108.394
day 19	1.186.997	138.119	1.181.726	101.454
day 20	1.100.586	134.881	1.095.649	99.629
day 21	1.206.175	140.350	1.201.207	102.290
day 22	1.310.299	145.027	1.304.850	105.467
day 23	1.162.870	139.263	1.157.589	102.171
day 24	927.289	122.650	922.989	90.999
day 25	791.237	111.429	787.514	83.993
day 26	886.057	109.049	882.350	82.172
day 27	802.130	106.138	799.167	79.993
day 28	772.103	107.767	769.651	80.862
day 29	744.484	102.793	741.806	77.091

day 30	891.086	96.812	888.255	73.480
day 31	640.676	86.279	638.526	65.961
day 32	593.139	90.621	590.378	69.127
day 33	638.674	92.530	635.393	70.061
day 34	583.611	88.188	581.370	67.035
day 35	566.389	87.579	564.047	66.555
day 36	650.250	90.576	646.046	68.690
day 37	609.476	88.576	607.084	67.180
day 38	695.844	95.384	693.094	71.960

Table 7: Distribution of the change in the data set according to days after applying the Zeyrek lemmatizer

Days:	**#tweets**	**#words**	**#word variations**
day1	705.696	8.572.581	152.678
day2	423.154	5.771.323	136.222
day3	278.616	3.814.579	104.725
day4	250.138	3.602.137	104.019
day5	209.681	2.927.917	93.544
day6	192.013	2.830.856	87.936
day7	154.261	2.321.491	82.496
day8	129.554	1.932.477	75.531
day9	112.263	1.685.526	69.959
day10	104.001	1.526.177	66.228
day11	80.080	1.173.201	56.460
day12	76.415	1.134.707	53.804
day13	69.196	1.018.914	51.567
day14	69.039	1.040.547	51.728
day15	144.842	1.766.986	65.060
day16	86.101	1.280.712	58.521
day17	65.962	973.664	50.063
day18	65.507	961.288	47.877
day19	60.638	870.815	44.554

day20	56.382	803.438	44.138
day21	55.196	876.803	44.530
day22	68.941	956.940	47.230
day23	63.597	859.305	45.687
day24	46.351	686.803	40.919
day25	38.191	582.379	37.395
day26	44.402	672.386	37.252
day27	39.203	589.196	35.325
day28	37.802	564.032	35.091
day29	38.018	544.399	33.923
day30	44.697	699.854	33.039
day31	30.297	495.278	29.668
day32	31.314	439.420	30.956
day33	32.764	471.360	31.371
day34	29.387	430.849	29.659
day35	27.247	418.201	29.246
day36	32.248	483.996	30.594
day37	31.170	449.398	30.277
day38	33.157	510.138	32.461

# avg word in tweets: the average number of words in the tweets in the data for that day std deviation of word count: refers to the standard deviation of word count in the tweets in the data for that day.

Table 8: Distribution of the average number of words and standard deviations in the tweet according to the days after the stopwords removed

Days	**#avg words in tweets**	**std dev of number of words in tweets**
day 1	12,21956338	7,591213129
day 2	13,7139552	7,803054206
day 3	13,77203391	7,513275877
day 4	14,49163262	7,8717784
day 5	14,03381327	7,934039366
day 6	14,81327306	7,804544003
day 7	15,14112446	7,99275964

day 8	15,03190176	7,973114045
day 9	15,11604001	7,921373286
day 10	14,77531947	7,877127851
day 11	14,74212038	7,996539799
day 12	14,9548518	8,064546321
day 13	14,84426845	8,220972545
day 14	15,17465491	8,017937402
day 15	12,26693224	7,835123164
day 16	14,97466928	8,021298415
day 17	14,86915194	8,10346412
day 18	14,78626712	8,156774291
day 19	14,4593984	8,012537732
day 20	14,36997623	8,205325545
day 21	15,97311399	8,039462527
day 22	14,01050173	8,180136134
day 23	13,60356621	8,416340473
day 24	14,9530323	8,292748748
day 25	15,38838993	8,190245784
day 26	15,21366155	7,706149715
day 27	15,10353799	8,006015829
day 28	15,00079361	8,220892846
day 29	14,38931559	7,922462615
day 30	15,72579815	8,443732129
day 31	16,41835825	8,081240827
day 32	14,11595453	8,165107858
day 33	14,45705653	7,970109779
day 34	14,74206282	8,053944317
day 35	15,42933167	8,230323505
day 36	15,08859464	7,933961401
day 37	14,51103625	8,139123442
day 38	15,47501282	8,021258528