Sentiment Analysis for Amazon Reviews

Huseyin YILMAZ

Summary

Sentiment analysis of product reviews is one of the most popular implementations of NLP (natural language processing). In this analysis, I want to study the correlation between the Amazon product reviews and the rating of the products given by the customers. I used traditional machine learning algorithms and deep neural networks. Seven different traditional machine learning algorithms used with six different bag of words methods (CountVectorizer, TfIdfVectorizer, HashingVectorizer, PCA with SMOTE Combination, Truncated SVD with SMOTE Combination, Word2Vec). Results of methods were compared and visualized. Logistic regression with CouuntVectorizing emerged as the best model.

1. INTRODUCTION

1.a. General

Natural language processing (or NLP) serves numerous use cases when dealing with text or unstructured text data. One of the subtopics of this research is called sentiment analysis or opinion mining, which is, given a bunch of text, we can computationally study peoples opinions, appraisals, attitudes, and emotions toward entities, individuals, issues, events, topics and their attributes. Applications of this technique are diverse. For example, businesses always want to find public or consumer opinions and emotions about their products and services. Potential customers also want to know the opinions and emotions of existing users before they use a service or purchase a product. Last but not least, researchers uses these information to do an in-depth analysis of market trends and consumer opinions, which could potentially lead to a better prediction of the stock market. The average human reader will have difficulty identifying relevant sites and accurately summarizing the information and opinions contained in them. Besides, to instruct a computer to recognize sarcasm is indeed a complex and challenging task given that at the moment, computer still cannot think like human beings.

1.b. Problem

Our goal is to build a sentiment analysis model that predicts whether a user liked a product or not, based on their review on Amazon. Our dataset consists of customers’ reviews and ratings, which we got from Consumer Reviews of Amazon products. We extracted the features of our dataset and built several supervised model based on that. These models not only include traditional algorithms such as Logistic Regression, Linear SVM, Naive Bayes, Kernel SVM, KNN, Random Forest, Gradient Boosting, XGBoost; but also deep learning with Keras. We compared the accuracy of these models and got a better understanding of the polarized attitudes towards the products.

1.c. Data Set

Our dataset comes from consumer reviews of Amazon products which are related with patio, lawn and garden. The data was obtained from Julian McAuley's dataset collection.[1] This dataset has 13,272 data points in total. Each record has below feature:

- Product/productId: asin, e.g. amazon.com/dp/B00006HAXW
- Product/title: title of the product
- Product/price: price of the product
- Review/userId: id of the user, e.g. A1RSDE90N6RSZF
- Review/profileName: name of the user
- Review/helpfulness: fraction of users who found the review helpful
- Review/score: rating of the product
- Review/time: time of the review (unix time)
- Review/summary: review summary
- Review/text: text of the review

2. DATA WRANGLING

2.1. Initial Understanding

The initial look of the data set is as below

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	A1JZFGZEZVWQPY	B00002N674	Carter H "1amazonreviewer@gmail . com"	[4, 4]	Good USA company that stands behind their prod...	4.0	Great Hoses	1308614400	06 21, 2011
1	A32JCI4AK2JTTG	B00002N674	Darryl Bennett "Fuzzy342"	[0, 0]	This is a high quality 8 ply hose. I have had ...	5.0	Gilmour 10-58050 8-ply Flexogen Hose 5/8-Inch ...	1402272000	06 9, 2014
2	A3N0P5AAMP6XD2	B00002N674	H B	[2, 3]	It's probably one of the best hoses I've ever ...	4.0	Very satisfied!	1336176000	05 5, 2012
3	A2QK7UNJ857YG	B00002N674	Jason	[0, 0]	I probably should have bought something a bit ...	5.0	Very high quality	1373846400	07 15, 2013
4	AS0CYBAN6EM06	B00002N674	jimmy	[1, 1]	I bought three of these 5/8-inch Flexogen hose...	5.0	Good Hoses	1375660800	08 5, 2013

[1] http://seotest.ciberius.info/seo--jmcauley.ucsd.edu/data/amazon/

Information – info( )

One of the basic and common way to understand the date is using “info( )” method. It is simple but tells a lot.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13272 entries, 0 to 13271
Data columns (total 9 columns):
reviewerID        13272 non-null object
asin              13272 non-null object
reviewerName      13107 non-null object
helpful           13272 non-null object
reviewText        13272 non-null object
overall           13272 non-null float64
summary           13272 non-null object
unixReviewTime    13272 non-null int64
reviewTime        13272 non-null object
dtypes: float64(1), int64(1), object(7)
memory usage: 1.0+ MB
```

- <u>What we learned from the information:</u>
  - We have the shape, 13272 observations(records or rows) and 9 columns (or variables).
  - There is no missing value.
  - There are two variables related with date but data types are not datetime, one of them is "int64" and the other one is "object". One time related variable will be enough for us, we can drop one of them.
  - We need to figure out that whether the "helpful" variable needs to be converted to numeric type in order to use it.
  - There are two different variables which identify reviewer/user, we can drop one of them.
  - In order to improve practical and readable coding, we need change some of the column names and also we need to convert column names to lowercase.

- <u>Design of reshaping:</u>
  - "reviewerID" --> "customer"
  - "asin" --> "product"
  - "reviewerName" --> column will be droped
  - "reviewText" --> "review_text" (will be merged with “summary”
  - "helpful" --> will be splited in two columns; "pos_feedback" as positive feedback + "neg_feedback" as negative feedback.
  - "overall" --> "rating"
  - "summary" --> as is
  - "unixReviewTime" --> "time"
  - "reviewTime" --> column will be droped

- <u>Issues fixed:</u>
  - 3 new columns created
  - 5 redundant columns dropped
  - Some column names were changed and made lowercase

Statistics summary – describe( )

Numeric features

	rating	time	pos_feedback	neg_feedback
count	13272.000000	1.327200e+04	13272.000000	13272.000000
mean	4.186483	1.358624e+09	3.233424	0.523282
std	1.084114	4.709839e+07	20.279594	2.765096
min	1.000000	9.548928e+08	0.000000	0.000000
25%	4.000000	1.341965e+09	0.000000	0.000000
50%	5.000000	1.370304e+09	0.000000	0.000000
75%	5.000000	1.393546e+09	1.000000	0.000000
max	5.000000	1.405987e+09	923.000000	167.000000

Non-numeric features

- Number of unique customers: 1686
- Number of unique products: 962
- Review per customer: 7.87
- Review per product: 13.79

- Rating:
  - Mean of the ratings is more than 4 out of 5. It means that people are tendentious to giving high ratings. "std" value (1.084) and percentile values show that 1 and 2 star ratings are rare.
  - Small numbers of "ratings under 4" will decrease the predictability of these ratings. To overcome this problem we need to split the ratings in to two groups as "good" and "bad" ratings.

- Total votes (t_votes) and positive votes (p_votes):
  - Their means are more than 3.0 but percentile values shows that more than half of the reviews don't have "helpful"votes.
  - They have outliers and should be cleaned or imputed.

- Non-numeric variables statistics:
  - Some customers have more than one ratings and most probably we have some outliers.
  - All ratings do not belong to diffent different people

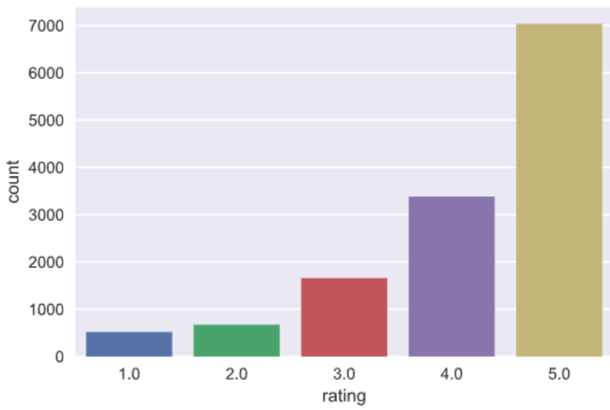
2.2. Text Preprocessing

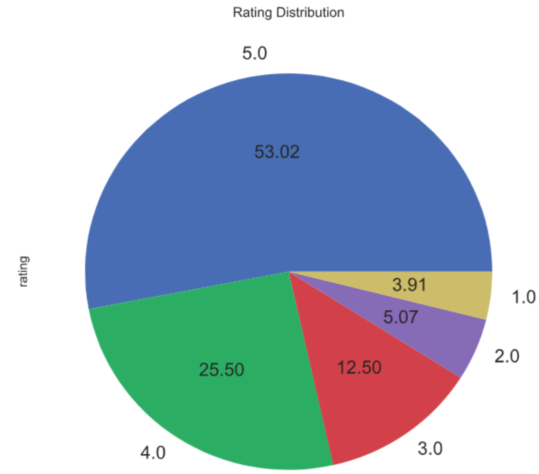
After creating the merged feature, in the context of corpus normalization we applied advanced text cleaning such as:

- Lowercase the text
- Keep only words
- Html removal
- Expanding contractions
- Whitespace and underscores removal
- Special characters removal
- Accent marks removal
- Lemmatization
- Stop words removal

3. EXPLORATORY DATA ANALYSIS

3.1. “Rating” Feature



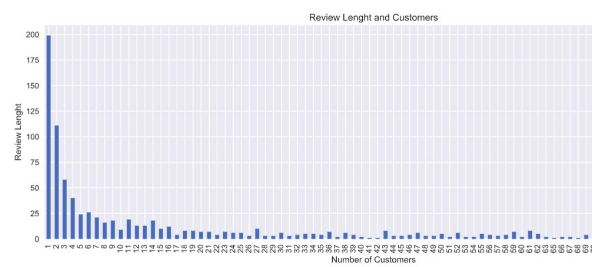


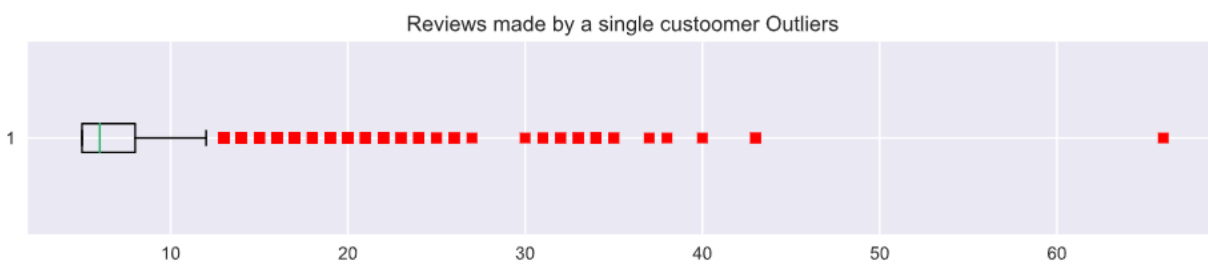
- There is an imbalance between rating classes
- Especially 1 and 2 ratings have small portions according to other classes

3.2. “Customer” feature

Number of reviews and customers

Review length and customers

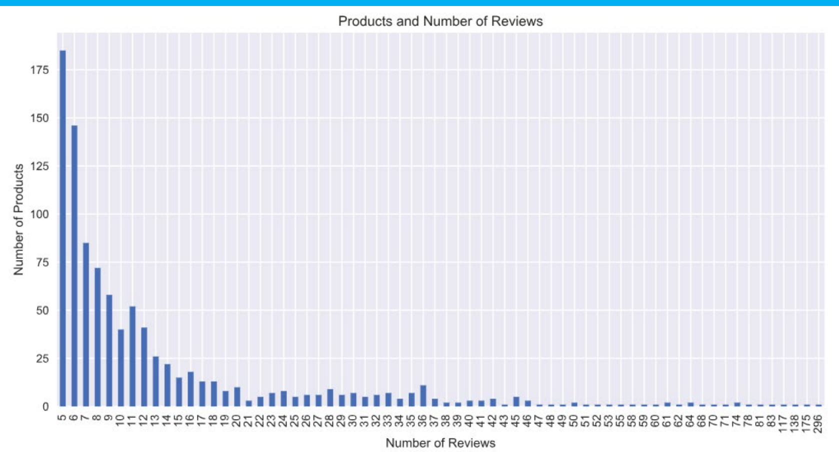


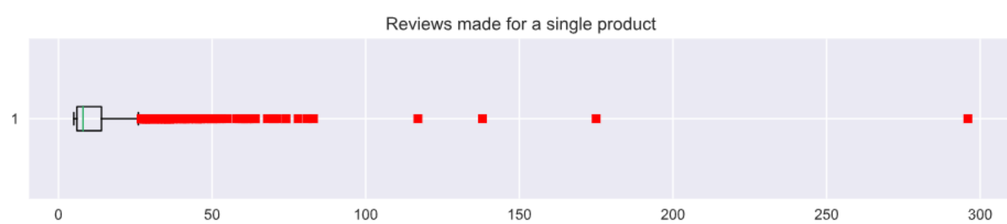


- Most of the customers gave less than 8 reviews (mean = 7.87).
- Giving more than 20 reviews is very rare.
- They are rare but we have customers who made reviews more than 40.
- Customers who has a large number of reviews may affect the objectiveness of the results.
- Here, "customer uniqueness" computed as a metric of "rating class".

- The purpose is understanding "how much are the reviews made by different customers or how much are they populated by same customers.
- For instance, customer uniqueness of "rating class 5" is 0.23, this means 77% of the reviews are given by customers who already made a review before.
- And some customers are populating the review rates which may affect the test scores negatively.
- There are outlier customers in regards of number of reviews.
- This may affect the objectiveness of the rating class.
- So, they may affect the score of the model.
- Most of the customers are tend to make short reviews.

3.3. “Product” feature:

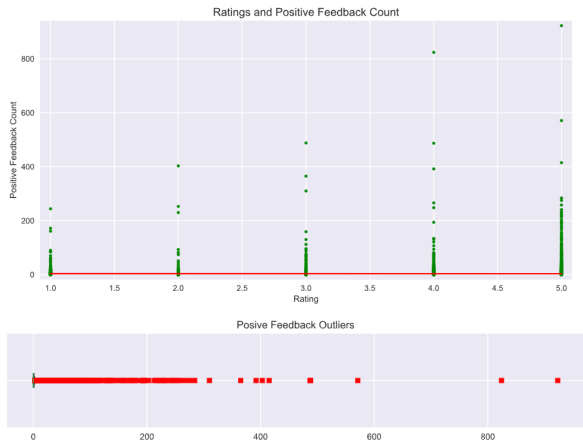




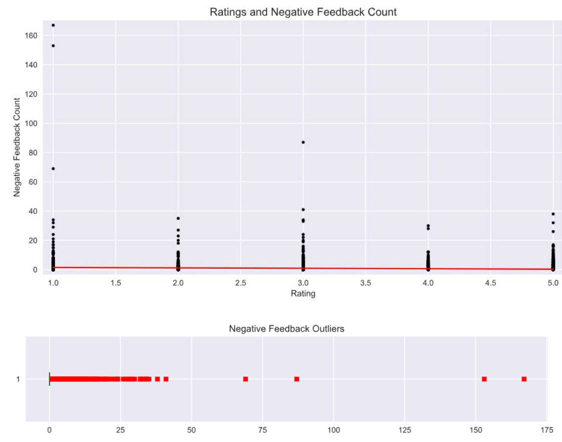
- Most of the products have less than 14 reviews (mean = 13.79)
- Product which have been received more than mean can analysis differently for extracting the strong issues about the products. For instance, cronical problems can be easily detected in this way.
- There are outlier products.
- These outliers can be considered in two different aspects
  - The first one, most probably, reviews which are made for a single product share the same or similar words, and this fact may effect the test score.
  - The second one, reviews of outlier products may give clues about the strong and weak points of the related products.

3.4. “Feedback” feature:

Positive feedback

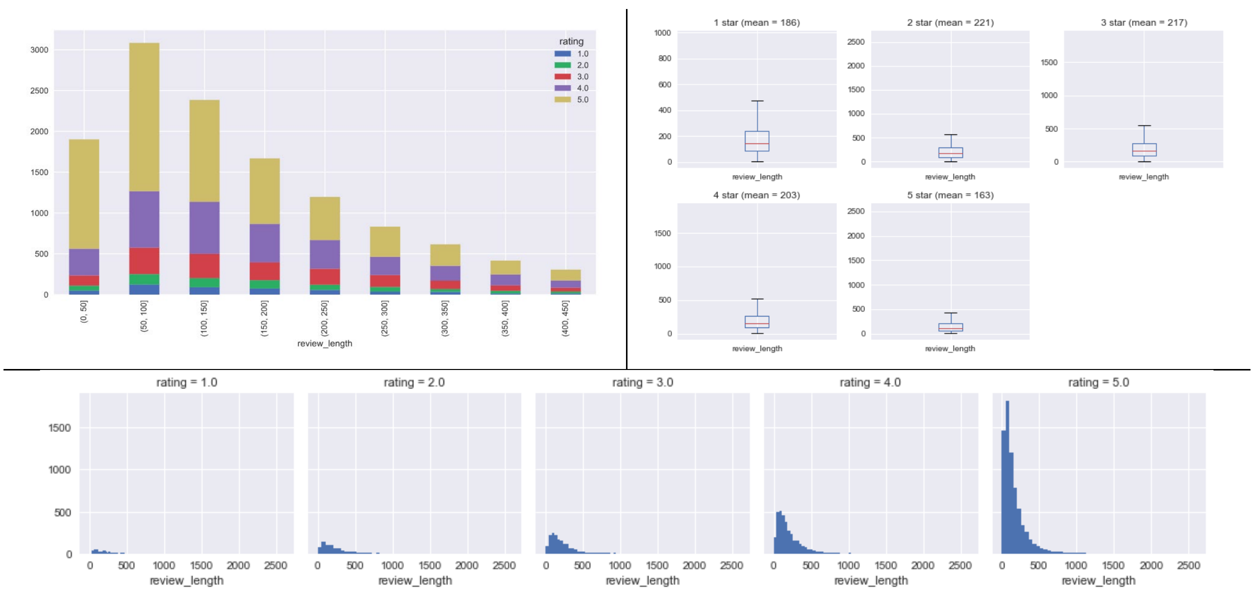


Negative Feedback

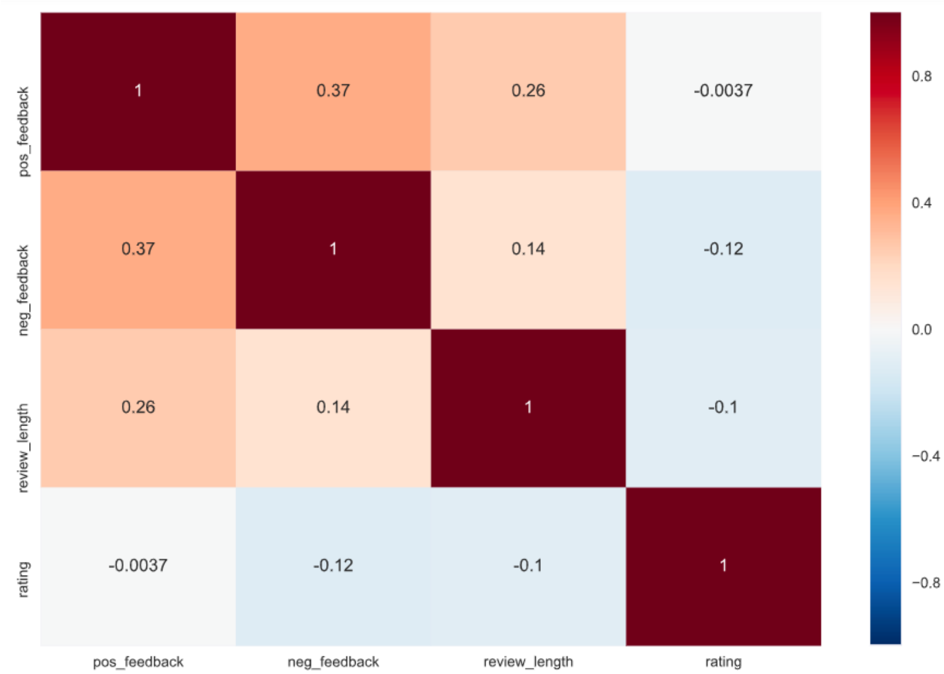


- The correlation (red lines in scatter plots) between feedbacks and ratings are very small and neglectable.
- Feedback outliers can provide the information of what customers like most about a specific product, and company can use this information for further improvements of the related products.

3.5. "Review Length" feature

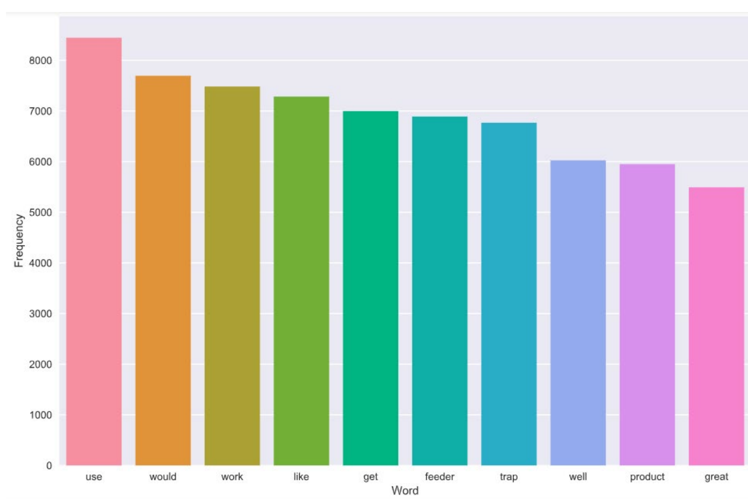


- There is a very slight negative correlation between Review Length and Rating Classes.
- Most of the reviews have less than 200 words.
- Boxplots show us that customers tend to using more words when they give 2 , 3 and 4 ratings.



- As seen above correlation matrix, there is no strong correlation between any two numeric variables.

3.6. “Review text” feature



- Most significant inference of text graphs is that there is a great number of matching words among review texts of different rating classes

3.7. EDA Findings

- There is an imbalance between rating classes, especially 1 and 2 ratings have small portions according to other classes.
- Most of the customers gave less than 8 reviews (mean = 7.87) but there are also outlier customers who have a large number of reviews. A great number of reviews which are made by a single customer may affect the objectiveness and so the test score negatively.
- Customer uniqueness of 4 and 5 ratings are low, it means they are populated by same customers. This may bring the same above drawbacks.
- Most of the products have less than 14 reviews (mean = 13.79). Product which have been received more than mean can be analyzed differently for extracting the strong issues about the products. For instance, chronical problems can be easily detected in this way.
- There is a negative correlation between number of feedbacks and rating classes are not strong. But outlier feedbacks can provide additional information.
- Most of the reviews have less than 200 words. And customers tend to using more words when they give 2, 3 and 4 ratings.
- There is no strong correlation between any two numeric variables.
- Most significant inference of text graphs is that there is a great number of matching words among review texts of different rating classes

- **Recap crucial points:**
  - There is no strong relationship between numeric predictors and target variable
  - Data set is imbalanced in regards of rating classes.
  - There is a great number of matching words among rating classes.

- **Conclusion:**
  - Using numeric variables will not make meaningful contribution to prediction.
  - We can reduce the imbalance with reducing the number of rating classes.

5. FEATURE ENGINEERING AND MODELING

In accordance with EDA Findings, the number classes (ratings) has been reduced. Five classes have been splitted into two group as “bad” (1, 2) and “not bad” (3, 4, 5). Therefore, analysis became a supervised binary-classification problem. We are trying to predict the ratings based on the reviews left by customers who bought patio, lawn or garden products. We used traditional machine learning algorithms and deep neural network with Keras. We implemented seven different traditional algorithms with six different methods. Algorithms:

- Logistic Regression
- Linear SVM
- Naive Bayes
- Kernel SVM
- KNN
- Random Forest
- Gradient Boosting
- XGBoost

In regards of feature engineering, review test data has been vectorized with six different methods. These bag of words methods:

- CountVectorizer
- TfIdfVectorizer
- HashingVectorizer
- PCA with SMOTE Combination
- Truncated SVD with SMOTE Combination
- Word2Vec

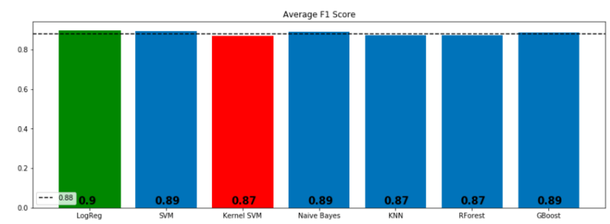
4.1. Modeling with Count-Vectorizing

Eight different machine learning algorithms implemented with Count-Vectorizing method. Uni-gram has been used as the best parameter for ngram_range. Accuracy scores and classification report results have been gathered as a comparison table. Best average f-1 scores and minor class f-1 scores of each model have been plotted

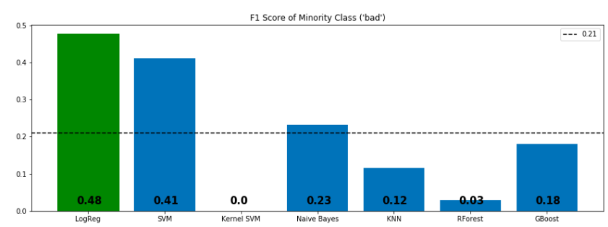
Comparison Table

				precision	recall	f1-score	support
vectorizer	model	accuracy	class				
			bad	0.411009	0.568528	0.477103	394.0
	LogReg	0.889014	not bad	0.956174	0.920347	0.937919	4030.0
			average	0.907622	0.889014	0.896879	4424.0
			bad	0.409091	0.411168	0.410127	394.0
	SVM	0.894665	not bad	0.942403	0.941935	0.942169	4030.0
			average	0.894907	0.894665	0.894786	4424.0
			bad	0.000000	0.000000	0.000000	394.0
	Kernel SVM	0.910940	not bad	0.910940	1.000000	0.953395	4030.0
			average	0.829812	0.910940	0.868486	4424.0
			bad	0.483871	0.152284	0.231660	394.0
CountVect	Naive Bayes	0.910036	not bad	0.922326	0.984119	0.952221	4030.0
			average	0.883277	0.910036	0.888048	4424.0
			bad	0.271028	0.073604	0.115768	394.0
	KNN	0.899864	not bad	0.915451	0.980645	0.946927	4030.0
			average	0.858058	0.899864	0.872904	4424.0
			bad	0.750000	0.015228	0.029851	394.0
	RForest	0.911844	not bad	0.912138	0.999504	0.953824	4030.0
			average	0.897698	0.911844	0.871536	4424.0
			bad	0.661290	0.104061	0.179825	394.0
	GBoost	0.915461	not bad	0.919074	0.994789	0.955434	4030.0
			average	0.896116	0.915461	0.886358	4424.0

Average F-1 Scores



Minor Class F-1 Scores



- With count vectorizing, logistic regression gave the best f-1 scores for both “average” and “minor class”.
- Kernel SVM is the weakest algorithm with count-vectorizing.
- Besides Kernel SVM; KNN, RForest, GBoost and XGboost have remained under the mean of minor class f-1 score.

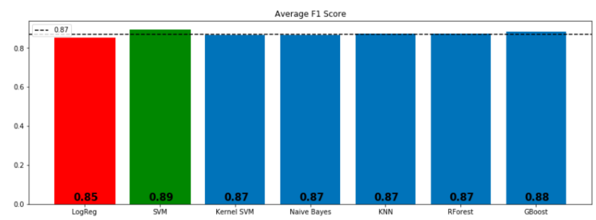
4.2. Modeling with Tfidf – Vectorizing

Eight different machine learning algorithms implemented with Tfidf-Vectorizing method. Uni-gram has been used as the best parameter for ngram_range. Accuracy scores and classification report results have been gathered as a comparison table. Best average f-1 scores and minor class f-1 scores of each model have been plotted

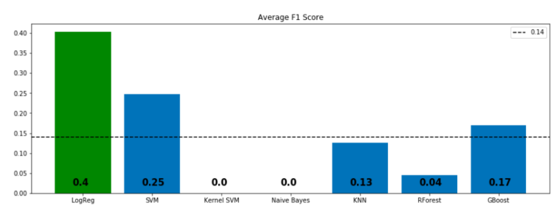
Comparison Table

				precision	recall	f1-score	support
vectorizer	model	accuracy	class				
TfidfVect	LogReg	0.825497	bad	0.290000	0.662437	0.403400	394.0
			not bad	0.962259	0.841439	0.897802	4030.0
			average	0.902388	0.825497	0.853771	4424.0
	SVM	0.920434	bad	0.783784	0.147208	0.247863	394.0
			not bad	0.922759	0.996030	0.957995	4030.0
			average	0.910382	0.920434	0.894751	4424.0
	Kernel SVM	0.910940	bad	0.000000	0.000000	0.000000	394.0
			not bad	0.910940	1.000000	0.953395	4030.0
			average	0.829812	0.910940	0.868486	4424.0
	Naive Bayes	0.910940	bad	0.000000	0.000000	0.000000	394.0
			not bad	0.910940	1.000000	0.953395	4030.0
			average	0.829812	0.910940	0.868486	4424.0
	KNN	0.899864	bad	0.283186	0.081218	0.126233	394.0
			not bad	0.916029	0.979901	0.946889	4030.0
			average	0.859668	0.899864	0.873801	4424.0
	RForest	0.912071	bad	0.692308	0.022843	0.044226	394.0
			not bad	0.912718	0.999007	0.953915	4030.0
			average	0.893089	0.912071	0.872899	4424.0
	GBoost	0.913653	bad	0.590909	0.098985	0.169565	394.0
			not bad	0.918541	0.993300	0.954459	4030.0
			average	0.889362	0.913653	0.884556	4424.0

Average F-1 Scores



Minor Class F-1 Scores



- With count vectorizing, logistic regression gave the best f-1 scores for minor class f-1 score but failed with average f-1 score. Best average f-1 score has been received by Linear SVM.
- Kernel SVM and Naïve Bayes are the weakest algorithms with tfidf-vectorizing.
- Besides Kernel SVM and Naïve Bayes; KNN and RForest have remained under the mean of minor class f-1 score.

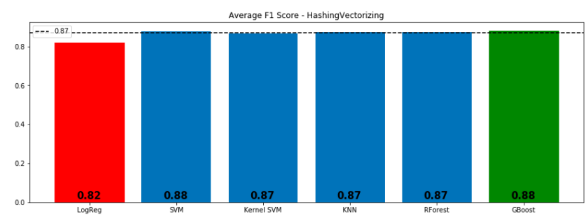
4.3. Modeling with Hashing-Vectorizing

Seven different machine learning algorithms implemented with Hashing-Vectorizing method. Uni-gram has been used as the best parameter for ngram_range. Accuracy scores and classification report results have been gathered as a comparison table. Best average f-1 scores and minor class f-1 scores of each model have been plotted.

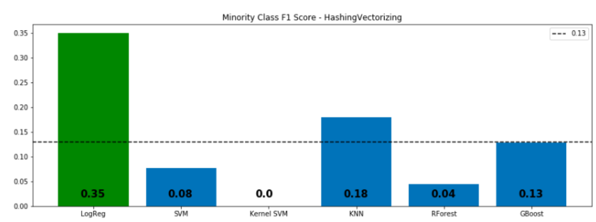
Comparison Table

vectorizer	model	accuracy	class	precision	recall	f1-score	support
HashVect	LogReg	0.775090	bad	0.235708	0.680203	0.350098	394.0
			not bad	0.961667	0.784367	0.864015	4030.0
			average	0.897013	0.775090	0.818246	4424.0
	SVM	0.913427	bad	0.761905	0.040609	0.077108	394.0
			not bad	0.914149	0.998759	0.954583	4030.0
			average	0.900591	0.913427	0.876436	4424.0
	Kernel SVM	0.910940	bad	0.000000	0.000000	0.000000	394.0
			not bad	0.910940	1.000000	0.953395	4030.0
			average	0.829812	0.910940	0.868486	4424.0
	KNN	0.893083	bad	0.284153	0.131980	0.180243	394.0
			not bad	0.919359	0.967494	0.942812	4030.0
			average	0.862787	0.893083	0.874898	4424.0
	RForest	0.912071	bad	0.692308	0.022843	0.044226	394.0
			not bad	0.912718	0.999007	0.953915	4030.0
			average	0.893089	0.912071	0.872899	4424.0
	GBoost	0.913653	bad	0.636364	0.071066	0.127854	394.0
			not bad	0.916438	0.996030	0.954578	4030.0
			average	0.891495	0.913653	0.880950	4424.0

Average F-1 Scores



Minor Class F-1 Scores



- With hashing vectorizing, logistic regression gave the best f-1 scores for minor class f-1 score but failed again with average f-1 score. Best average f-1 score has been received by Gradient Boosting.
- Naïve Bayes is the weakest algorithms with hashing-vectorizing.
- Besides Naïve Bayes; Linear SVM and RForest have remained under the mean of minor class f-1 score.

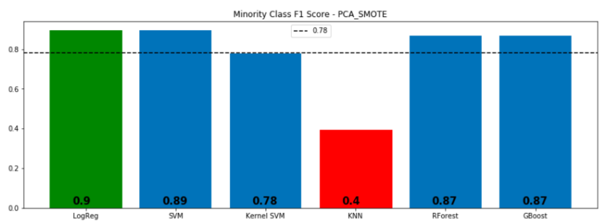
4.4. Modeling with PCA-SMOTE Combination

Seven different machine learning algorithms implemented with PCA-SMOTE combination method. Since we got the best results from, Count-vectorizing based features were used for this combination. Accuracy scores and classification report results have been gathered as a comparison table. Best average f-1 scores and minor class f-1 scores of each model have been plotted.

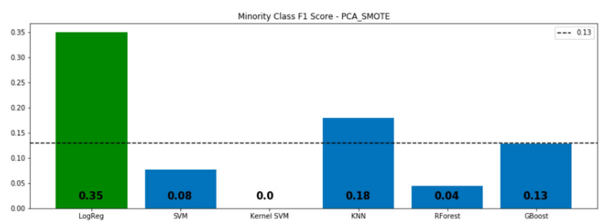
Comparison Table

				precision	recall	f1-score	support
vectorizer	model	accuracy	class				
PCA-SMOTE	LogReg	0.889467	bad	0.407045	0.527919	0.459669	394.0
			not bad	0.952466	0.924814	0.938436	4030.0
			average	0.903891	0.889467	0.895797	4424.0
	SVM	0.893987	bad	0.406015	0.411168	0.408575	394.0
			not bad	0.942360	0.941191	0.941775	4030.0
			average	0.894594	0.893987	0.894289	4424.0
	Kernel SVM	0.720841	bad	0.195069	0.682741	0.303440	394.0
			not bad	0.958949	0.724566	0.825442	4030.0
			average	0.890918	0.720841	0.778952	4424.0
	KNN	0.322559	bad	0.109511	0.926396	0.195868	394.0
			not bad	0.973419	0.263524	0.414763	4030.0
			average	0.896480	0.322559	0.395268	4424.0
	RForest	0.903255	bad	0.160000	0.020305	0.036036	394.0
			not bad	0.911751	0.989578	0.949072	4030.0
			average	0.844801	0.903255	0.867757	4424.0
	GBoost	0.896022	bad	0.176471	0.045685	0.072581	394.0
			not bad	0.913003	0.979156	0.944923	4030.0
			average	0.847408	0.896022	0.867233	4424.0

Average F-1 Scores



Minor Class F-1 Scores



- With count vectorizing, logistic regression gave the best f-1 scores for both "average" and "minor class".
- The weakest algorithms are KNN for average score and Kernel SVM for minor class score.
- Besides Kernel SVM; KNN, Linear SVM and Random Forest have remained under the mean of minor class f-1 score.
- Especially for the minor class, f-1 scores are poor by comparison with so far methods.

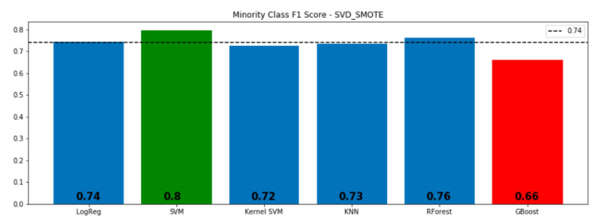
4.5. Modeling with Truncated SVD – SMOTE Combination

Seven different machine learning algorithms implemented with Truncated SVD -SMOTE combination method. Since we got the best results from, Count-vectorizing based features were used for this combination. Accuracy scores and classification report results have been gathered as a comparison table. Best average f-1 scores and minor class f-1 scores of each model have been plotted.

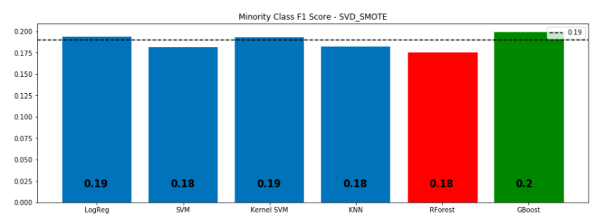
Comparison Table

vectorizer	model	accuracy	class	precision	recall	f1-score	support
SVD-SMOTE	LogReg	0.674729	bad	0.124371	0.439086	0.193838	394.0
			not bad	0.927135	0.697767	0.796262	4030.0
			average	0.855641	0.674729	0.742610	4424.0
	SVM	0.755199	bad	0.129171	0.304569	0.181406	394.0
			not bad	0.921602	0.799256	0.856080	4030.0
			average	0.851029	0.755199	0.795994	4424.0
	Kernel SVM	0.648960	bad	0.121489	0.472081	0.193247	394.0
			not bad	0.928102	0.666253	0.775675	4030.0
			average	0.856266	0.648960	0.723804	4424.0
	KNN	0.661844	bad	0.116295	0.423858	0.182514	394.0
			not bad	0.924029	0.685112	0.786834	4030.0
			average	0.852093	0.661844	0.733013	4424.0
	RForest	0.701854	bad	0.116183	0.355330	0.175109	394.0
			not bad	0.921094	0.735732	0.818044	4030.0
			average	0.849408	0.701854	0.760784	4424.0
	GBoost	0.570976	bad	0.119433	0.598985	0.199156	394.0
			not bad	0.935458	0.568238	0.707008	4030.0
			average	0.862783	0.570976	0.661779	4424.0

Average F-1 Scores



Minor Class F-1 Scores



- With Truncated SVD -SMOTE combination, Gradient Boosting gave the best f-1 scores for minor class f-1 score but failed again with average f-1 score. Best average f-1 score has been received by Linear SVM.
- All scores are poor by comparison with so far methods.

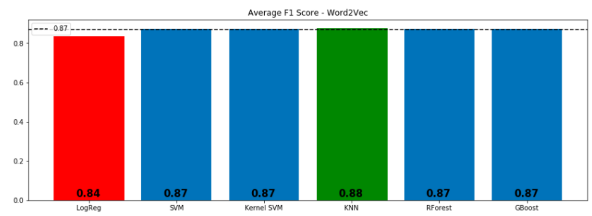
4.6. Modeling with Word2Vec

Seven different machine learning algorithms implemented with Word2Vec method. Accuracy scores and classification report results have been gathered as a comparison table. Best average f-1 scores and minor class f-1 scores of each model have been plotted.

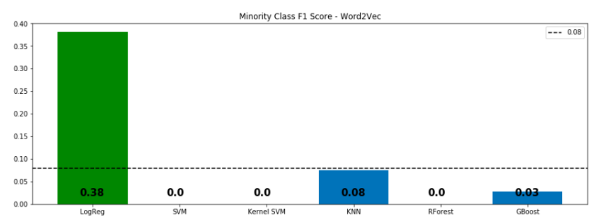
Comparison Table

				precision	recall	f1-score	support
vectorizer	model	accuracy	class				
Word2Vec	LogReg	0.799817	bad	0.261320	0.706294	0.381492	286.0
			not bad	0.966387	0.808774	0.880583	2986.0
			average	0.904758	0.799817	0.836959	3272.0
	SVM	0.912592	bad	0.000000	0.000000	0.000000	286.0
			not bad	0.912592	1.000000	0.954298	2986.0
			average	0.832824	0.912592	0.870885	3272.0
	Kernel SVM	0.912592	bad	0.000000	0.000000	0.000000	286.0
			not bad	0.912592	1.000000	0.954298	2986.0
			average	0.832824	0.912592	0.870885	3272.0
	KNN	0.910147	bad	0.375000	0.041958	0.075472	286.0
			not bad	0.915432	0.993302	0.952779	2986.0
			average	0.868194	0.910147	0.876095	3272.0
	RForest	0.912592	bad	0.000000	0.000000	0.000000	286.0
			not bad	0.912592	1.000000	0.954298	2986.0
			average	0.832824	0.912592	0.870885	3272.0
	GBoost	0.911980	bad	0.400000	0.013986	0.027027	286.0
			not bad	0.913550	0.997991	0.953905	2986.0
			average	0.868661	0.911980	0.872888	3272.0

Average F-1 Scores



Minor Class F-1 Scores



- With Word2Vec, logistic regression gave the best f-1 scores for minor class f-1 score but failed again with average f-1 score. Best average f-1 score has been received by Linear KNN.
- Word2Vec gave the worst mean of minor class f-1 score .

4.7. Modeling with Keras

Embeddings have been created with “keras.preprocessing”. These embeddings have been put in into “Embedding layers” and implemented with two different layers, Conv1D and GRU

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 2263, 100)         4486500
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 2259, 128)         64128
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 1129, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 144512)            0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 144513
=================================================================
Total params: 4,695,141
Trainable params: 208,641
Non-trainable params: 4,486,500
_________________________________________________________________
None
Train on 10618 samples, validate on 2654 samples
Epoch 1/10
 - 263s - loss: 0.3001 - acc: 0.9023 - val_loss: 0.2724 - val_acc: 0.9126
Epoch 2/10
 - 214s - loss: 0.2447 - acc: 0.9107 - val_loss: 0.2685 - val_acc: 0.9077
Epoch 3/10
 - 219s - loss: 0.2196 - acc: 0.9166 - val_loss: 0.2682 - val_acc: 0.9115
Epoch 4/10
 - 215s - loss: 0.1876 - acc: 0.9258 - val_loss: 0.2741 - val_acc: 0.9066
Epoch 5/10
 - 199s - loss: 0.1538 - acc: 0.9391 - val_loss: 0.2905 - val_acc: 0.9077
Epoch 6/10
 - 207s - loss: 0.1232 - acc: 0.9528 - val_loss: 0.3206 - val_acc: 0.9069
Epoch 7/10
 - 197s - loss: 0.0938 - acc: 0.9666 - val_loss: 0.3434 - val_acc: 0.9043
Epoch 8/10
 - 200s - loss: 0.0703 - acc: 0.9782 - val_loss: 0.3901 - val_acc: 0.9069
Epoch 9/10
 - 200s - loss: 0.0530 - acc: 0.9860 - val_loss: 0.4231 - val_acc: 0.9073
Epoch 10/10
 - 203s - loss: 0.0426 - acc: 0.9884 - val_loss: 0.4094 - val_acc: 0.8945
<keras.callbacks.History at 0x117ddf358>
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 2263, 100)         4486500
_________________________________________________________________
gru_1 (GRU)                  (None, 32)                12768
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 4,499,301
Trainable params: 12,801
Non-trainable params: 4,486,500
_________________________________________________________________

Train...
Train on 10618 samples, validate on 2654 samples
Epoch 1/10
 - 276s - loss: 0.3230 - acc: 0.9033 - val_loss: 0.2871 - val_acc: 0.9130
Epoch 2/10
 - 271s - loss: 0.2884 - acc: 0.9100 - val_loss: 0.2733 - val_acc: 0.9130
Epoch 3/10
 - 308s - loss: 0.2734 - acc: 0.9098 - val_loss: 0.2601 - val_acc: 0.9133
Epoch 4/10
 - 289s - loss: 0.2585 - acc: 0.9094 - val_loss: 0.2563 - val_acc: 0.9145
Epoch 5/10
 - 281s - loss: 0.2478 - acc: 0.9097 - val_loss: 0.2548 - val_acc: 0.9148
Epoch 6/10
 - 281s - loss: 0.2435 - acc: 0.9103 - val_loss: 0.2466 - val_acc: 0.9160
Epoch 7/10
 - 280s - loss: 0.2411 - acc: 0.9121 - val_loss: 0.2447 - val_acc: 0.9167
Epoch 8/10
 - 274s - loss: 0.2370 - acc: 0.9130 - val_loss: 0.2426 - val_acc: 0.9160
Epoch 9/10
 - 280s - loss: 0.2309 - acc: 0.9123 - val_loss: 0.2421 - val_acc: 0.9164
Epoch 10/10
 - 294s - loss: 0.2302 - acc: 0.9116 - val_loss: 0.2420 - val_acc: 0.9175
<keras.callbacks.History at 0x1a438786a0>
```

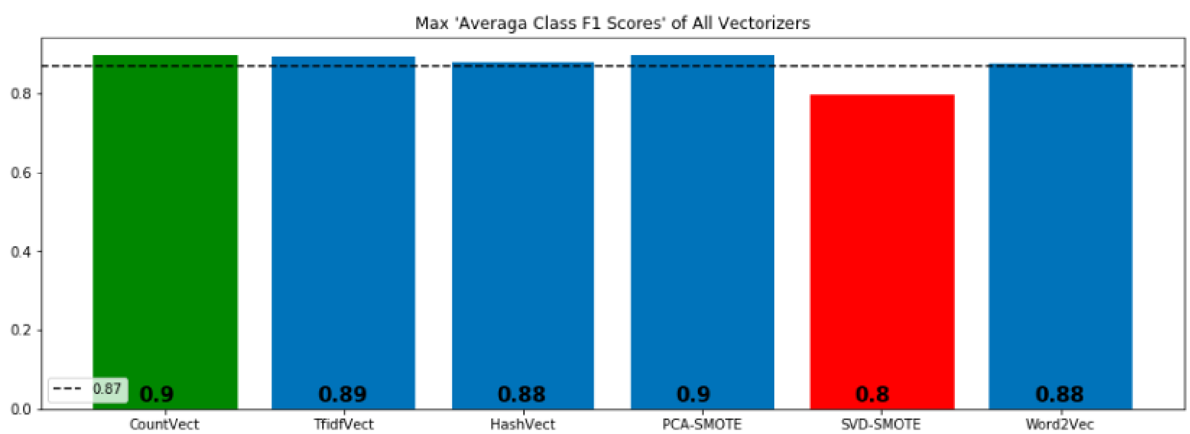
Using Deep neural networks with Keras, didn’t give a better score than the best score of the traditional machine learning algorithms.

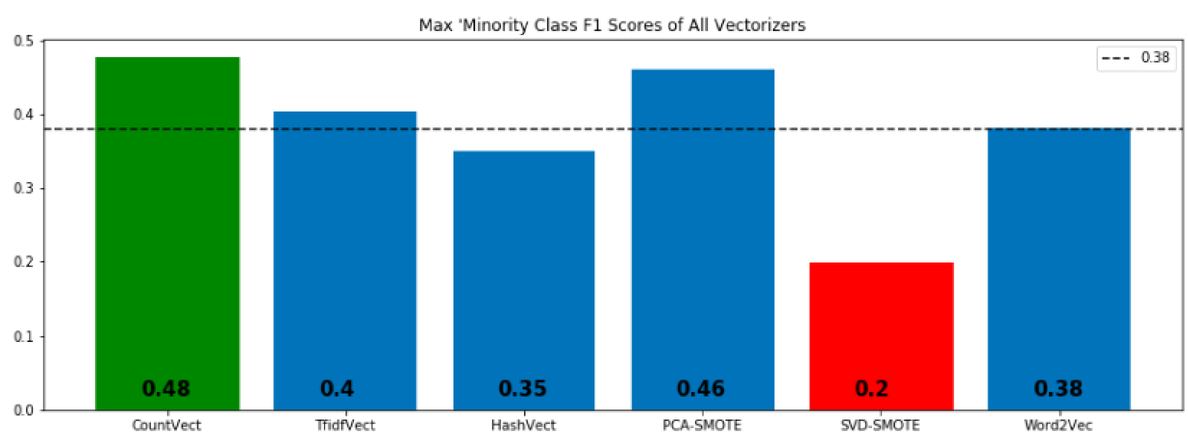
	precision	recall	f1-score	support
0	0.77	0.07	0.13	231
1	0.92	1.00	0.96	2423
micro avg	0.92	0.92	0.92	2654
macro avg	0.85	0.54	0.55	2654
weighted avg	0.91	0.92	0.89	2654

6. CONCLUSION

In this study, we tried to predict the sentiments of customers based on the reviews left by customers. Here are results of modeling:

- As below plots show, best "F1 Scores" have been taken via "Logistic Regression" with CountVectorizer.
  - Best average F1 Score is 90 %.
  - Best minority F1 score is 48 %.
- Kernel SVM and Random Forest showed poor performance over all.
- SMOTE Combinations and Word2Vec Methods didn't work well except Logistic Regression.





Why we couldn't improve our scores? All through the data processing we have seen the effect of the imbalanced data. But another and most effective issue than the imbalanced data were matching words among the classes. we tried find a solution for imbalanced data with oversampling but it didn't work well because of the rate of matching words. Even in the most common 5000, the portion of the matching words is 80%. And deleting all these words didn't solve the problem.

7. FUTURE STUDY

There were two problems with our study. The first one is imbalance of data, and the second one is high percentage of matching words among the classes. Therefore, future studies on the same or any similar data should focus on solving these issues.

In regards of these problems, below points can be considered for the future studies:

- Using different methods in order to minimize the effect of the matching words
- Implementation of deep learning with different neural network types and different layer combinations.
- Using different AutoML tools.
- Implementation of Dask library for parallel processing to decrease run time.