Document Classification with Supervised Learning

Juan A. Campos
Northwestern University, Masters of Science in Data Science

Table of Contents

Abstract

The scale and exponential growth of data generated in recent years has overloaded organizations and has placed a burden on their ability to analyze and extract insight from the data. Efficient and effective methods of retrieving topic relevant documents are needed. A classification solution may allow organizations to accurately group topic related documents together and potentially minimize the need for comprehensive analysis of the data to extract insight. This study will explore several supervised learning machine learning algorithms to determine which is more effective in classifying documents. In addition, the research will examine three feature extraction methods to ascertain which provides the highest accuracy in predicting document classes (i.e. document classification).

1. Introduction

1.1 Statement of Problem

There has been a paradigm shift in data growth. 90% of the world's data was created in short span of two years [1]. "It is estimated that the volume of data that is created in the United States alone is enough to fill 10,000 Libraries of Congress" [1]. And because so much data is being created, companies need to be able to analyze and understand what is in the data. But because the volumes of data are so large, companies only able to analyze 12% of their data [1]. A company's inability to effectively and efficiently examine its data increases the risk of missing out on opportunities, potential growth and revenues.

1.2 Purpose and Scope

This study is being conducted to 1) to explore the accuracy, effectiveness and efficiency of several supervised learning algorithms; and 2) to gain insight and inform on the best practices of developing classification models. The activities that will be addressed include constructing 13 supervised learning and one non-supervised classification models and comparing their performance; and conducting grid search hyper-parameter tuning of selected models. The major questions that will be explored in this paper concern: 1) which learning method is the most effective at classifying documents; 2) which classifier is the most effective and efficient at classifying documents; and 3) which feature extraction method (Count, TF-IDF, Word2Vec) generates the most accurate models.

2. Literature Review

An improved TF-IDF weighting, which replaced the IDF with Binomial Separation (BNS), was found to significantly improve the classification performance of Support Vector Machine models [2]. Achieving similar results to our study, Han et al [3] found that Centroid based classifiers outperform Naïve Bayes and N-nearest Neighbor algorithms. Doc2Vec was evaluated to highlight the impact of document length on model performance by Lau et al [4]. Deka et al [5] found that K-nearest Neighbor outperformed K-means in categorizing text. Xing et al continued their research that demonstrates that Neural Network-Word2Vec based classification approaches perform well in document classification and experimented with Naïve Bayes, K-nearest Neighbor, and Support Vector Machine models [6]. Working with the standard Reuters corpus, Manevitz et al [7] studied and compared the document classification performance of K-nearest Neighbor, Naïve Bayes and a one-class Neural Network. Using the Support Vector Machines algorithm, Lilleberg et al [8] found that stopwords materially reduce the classification accuracy of the models. Using two vectorization methods, 1) Word2Vec word beddings weighted by TF-IDF and 2) traditional TF-IDF models, the research team demonstrated that the classification models with no stopwords outperformed the models that contained stopwords.

3. Methods

The research process for this study involved exploring supervised learning and classification; searching for and gathering general and background information; studying the three vectorization techniques and programming code; and locating current research on classification. For the purpose of this study, limiting the research references to peer-reviewed studies was not necessary.

The Python programming language and the Jupyter Notebook were used exclusively for this analysis. Several Python Index packages were utilized. They included Gensim, the Natural Language Tool Kit (NLTK), RegEx (re), Scikit-Learn, Keras, and Gensim.

An available modified Reuters Corpus containing a total of 7,791 documents was used for the analysis. The corpus is a modified version of the original that contains fewer documents, and is pre-split into test (2,208 docs.) and train (5,583 docs.) datasets. The document dimensionality of the corpus was restricted to 6,500 documents (5,000 train docs. / 1.250 test docs.) for the analysis to remain within the processing capabilities of the computer equipment used in the analysis. The documents in the Reuters corpus were pre-labeled using eight classes (ex. trade, ship, acq, earn, etc.). The eight classes were designated as the classes that would be used in the classification analysis.

The Reuters corpus text was cleaned and normalized using common techniques. Three Reference Term Vector (RTV) matrices were created, one for each of the vectorization methods: Count, TF-IDF, and Doc2Vec. It should be noted the original goal was to use a custom Word2Vec word embedding for the analysis and not the Doc2Vec method. Due to complications that were encountered in constructing the custom Word2Vec matrix that could be resolved before the deadline, the Word2Vec method was replaced with the Doc2Vec method. As regards the removal of terms from the corpus, neither common terms nor uncommon terms were removed from the corpus. The term quantity setting for the three RTVs was set at 1,000 words. Exploratory Data Analysis (EDA) of the corpus was conducted. The EDA results can be viewed in Appendix A.

The three RTV matrices were classified using 11 supervised learning classifiers and two meta-classifiers. Meta-classifiers combine similar or conceptually different machine learning classifiers and classify via a majority or plurality voting. Three of the supervised learning classifiers into each of the meta-classifiers. The classifiers that were used in the study are listed in Appendix B. The metrics used to evaluate the performance of the models were the Accuracy Score, F1 Score, and Processing times. Processing times include model training and model performing model predictions against the test data. The RTV matrix that led to the highest classification accuracy, which was TF-IDF in this case, was then selected to be clustered using the unsupervised K-Means Clustering algorithm. The TF-IDF matrix was also scaled using the t-SNE dimensionality reduction. The cluster count for the analyses was set at eight clusters to match the number of categories that were used in the classification analysis. The clustering results were plotted using scatterplots. Silhouette Statistics analysis was conducted to determine the optimal number of clusters in the corpus. In addition, to supplement the

Silhouette Statistics, an Elbow Graph analysis was performed to obtain a second estimate of the optimal clusters for the corpus. Finally, grid search hyper-parameter tuning using the TF-IDF matrix and the Linear Support Vector Machine (SVM-L) and Random Forest (RF) algorithms was conducted in order to determine the optimal values for the give models.

4. Results

It was discovered that the Multinomial Naïve Bayes (MultiNB) and K-nearest Neighbors (KNN) algorithms could not fit the Doc2Vec matrix because it contained negative values. This issue has three possible solutions: 1) the negative values can be eliminated by normalizing the features so that they fall between the range [0,1]; the second solution is to simply not use these algorithms in the analysis; 3) the final solution is to use other alternatives from the same family that can process negative values, such as Gaussian Naïve Bayes (GNB) or the Bernoulli Naïve Bayes (BNB). As the Naïve Bayes family alternative algorithms were already being used as part of the analysis, and no immediate alternative for KNN could be identified, it was decided to follow option two and the MultiNB and KNN algorithms were removed from the analysis with Doc2Vec. The Voting Classifier (VC1) was also removed from the analysis as it was configured to include the MultiNB and KNN algorithms in its composition for this study. As can be seen in Appendices C through E, although the three algorithms were not used in the analysis with Doc2Vec, the tables and graphs are reflecting that metric performance values were generated for the three models in question. This is curious and no apparent answer came to light explaining why this was occurring.   This matter requires further examination.

The vectorization methods that lead to more accurate models were the Count and the TF-IDF method. Many of the models achieved mid- to high Accuracy and/or F1 scores in the ranges between .85 and .93 using the Count and TF-IDF vectorization methods. Many of the models achieved the same scores when using the Count and TF-IDF matrices. For example, the Logistic Regression model generated the same .9240 Accuracy Score using the Count and the TF-IDF vectors. Taking the averages, the TF-IDF models performed better (.8550) than the Count models (.8539) using the F1 score. On the other hand, on average, the Count models (.8598) managed to edge out the TF-IDF models (.8590) using the Accuracy score. The Doc2Vec matrix performed poorly in classifying documents. Using this vectorization method, the models achieved Accuracy and/or F1 scores in the .50 range. The Accuracy and F1 metric results for each of the three vectorization methods can be found in Appendices C and D, respectively.

When examining the individual models, the SVM-L model outperformed the other models. The SVM-L model generated the highest scores for both the Accuracy (.9368) and F1 (.9362) metrics. The EVC1 achieved the second highest values in both the Accuracy (.9264) and F1 (.9247) scores. The Logistic Regression models were the third best with an Accuracy score of .9240 and an F1 score of .9211. Scoring the lowest in both the Accuracy (.5880) and F1 (.5083) scores, the least accurate model was the KNN model.

The training and test times were combined as the Performance Time metric. This value was calculated to determine which of the vectorization methods provides the most efficient performance. The Doc2Vec method is the costliest in performance time. On average, the Doc2Vec models required processing times 4.50 times greater than that needed by the Count and TF-IDF models. The processing time gap widened only widened as the max features were increased. Taking the averages of the Count (15.2341) and TF-IDF (15.3646) model Processing Times, the Count models performed slightly faster than the TF-IDF models. However, it should be noted than when taking the individual models into account, the TF-IDF models were more efficient than Count models. Referring to the tables in Appendix E, the least efficient models were the Vote Classifier (VC1), the Adaptive Boosting (ADA) and the EVC1 model. It is understandable that the meta-classifiers may require longer Processing Times because they are technically processing several algorithms, they were still almost three times more efficient than the ADA model. The ADA model achieved the worst processing times. The most efficient model was the MultiNB model followed by the BNB model and the SVM-L model being the third fastest model. The Processing times for these models barely register in the graphs when compared to the other 10 models. A second Processing Time graph in Exhibit E is the same graph as the first but without the Doc2Vec model results. This second graph was produced to get a better sense of how the Count and TF-IDF models performed as it is harder to do so with the first graph that includes Doc2Vec.

It was possible to increase the Accuracy of the TF-IDF vector SVM-L and RF models using hyper-parameter turning. As can be seen in Appendix H, The SVM-L Accuracy score increased from .9368 to .9400. The RF model's accuracy increased from .9040 to .9184. It is expected that continued "tweaking" of the grid search parameters might result in even higher performance from the models.

A visual representation of the K-means results showed that the model was not able to adequately cluster the documents into eight classes. In reviewing the K-Means clustering graph (Appendix F), it is apparent that the documents clustered widely, overlapping, and in varying sizes. The cluster to which the most documents were allocated dominated the distributions and appear throughout the graphs comingled with other clusters. A silhouette score analysis was conducted to determine the optimal number of clusters. The statistic suggested that two clusters are ideal for this corpus and this number resulted in the highest score. The text was then graphed using an Elbow graph. Based on the bend in the graph, this approach suggested that five clusters are ideal for this corpus. Appendix G lists the Silhouette Statistic results and the Elbow graph.

5. Conclusions

The study set out to explore supervised learning and non-hierarchical classification methods to determine which is more effective and efficient in classifying documents, and which feature extraction method (Count, TF-IDF, Doc2Vec) provides the highest accuracy in predicting document classes (i.e. text classification) and also generates the most clear-cut interpretable results. Based on the above results, the SVM-L model with the TF-IDF vectorization matrix is the best model to use with this corpus to more accurately classify documents. This model realized the highest Accuracy and F1 scores. In addition, it was amongst the most efficient performing models.

It is believed that the Doc2Vec performed poorly because it is ideally better suited for very large datasets. This study employed a very small dataset. Nevertheless, even with so few documents to process, the models with Doc2Vec were the least efficient and required much more Processing Time than the models with Count or TF-IDF. It is uncertain if the Processing Times of Doc2Vec will improve as the dimensionality of the corpus increases. One another note, although the Word2Vec method was not tested as originally planned, it is expected that it would have performed similarly to the Doc2Vec method as Doc2Vec is an extension of Word2Vec. They share the same operating conditions.

Finally, the results of this study suggest that using unsupervised are best for classifying unstructured data than labeled data. Labeled data should be classified with supervised learning techniques.

6. (Management) Recommendations

Based on the results of this study, it is recommended that the error with Doc2Vec generating performance score values for models that were not test be further researched to determine the cause and a solution. To increase the accuracy of the models, it is recommended that very uncommon and common terms be removed from the corpus using multiple and varying maximum and minimum feature constraints in future tests. The performance results should then be evaluated and compared to previous test results. It is recommended that Corpus Specific Stopwords be removed from the corpus to improve the model classification accuracy. It is also recommended that hyper-parameter tuning to be performed with the SVM-L and other high performing models to find the optimal parameters for the models. It is recommended that a custom Word2Vec using the corpus documents be constructed and tested performing the same tests from this study. It also recommended that a pre-trained Word2Vec word embedding be used in future testing to determine if this method provides better results. The feature dimensionality should be increased in future tests to study the impact of larger datasets on the model performance. Finally, it is recommended that Neural Network methods be introduced in future tests to ascertain their applicability to the document classification problem.

References

[1] NT, Baiju. (2014, June, 12), Exciting facts and finding about Big Data you should know. http://bigdata-madesimple.com/exciting-facts-and-findings-about-big-data/, Retrieved 2019, June 7.

[2] Forman, George. (2008). BNS feature scaling: An improved representation over TF-IDF for SVM text classification. International Conference on Information and Knowledge Management, Proceedings. 263-270. 10.1145/1458082.1458119.

[3] Han EH., Karypis G. (2000) Centroid-Based Document Classification: Analysis and Experimental Results. In: Zighed D.A., Komorowski J., Żytkow J. (eds) Principles of Data Mining and Knowledge Discovery. PKDD 2000. Lecture Notes in Computer Science, vol 1910. Springer, Berlin, Heidelberg.

[4] Lau, J.H., & Baldwin, T. (2016). An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. Rep4NLP@ACL.

[5] Deka, H., & Sarma, P. (2017). A Machine Learning Approach for Text and Document Mining.

[6] Xing, Chao & Wang, Dong & Zhang, Xuewei & Liu, Chao. (2015). Document classification with distributions of word vectors. 10.1109/APSIPA.2014.7041633.

[7] Manevitz, Larry M. and Yousef, Malik. 2002. One-class SVMs for document classification. J. Mach. Learn. Res. 2 (March 2002), 139-154.

[8] Lilleberg, J., Zhu, Y., & Zhang, Y. (2015). Support vector machines and Word2vec for text classification with semantic features. 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), 136-140.

Bibliography

1. Perkins, J., (2010). *Python Processing with NLTK 2.0 Cookbook*. Birmingham: Packt Publishing Ltd.

2. Bird, S., & Loper, E., (2009). *Natural Language Processing with Python.* Sebastopol: O'Reilly Media, Inc.

3. Brownlee, J., (2019). Deep Learning for Natural Language Processing. E-book.

4. Sarkar, D., (2016). Text Analytics with Python. A Practical Real-World Approach to Gaining Actionable Insights from your Data. New York: Apress

5. Richard C. Dubes, How many clusters are best? - An experiment, Pattern Recognition, Volume 20, Issue 6, 1987, Pages 645-663, ISSN 0031-3203, https://doi.org/10.1016/0031-3203(87)90034-3.

6. Pakhira, Malay. (2012). Finding Number of Clusters before Finding Clusters. Procedia Technology. 4. 27–37. 10.1016/j.protcy.2012.05.004.

7. Michael Steinbach and George Karypis and Vipin Kumar, A comparison of document clustering techniques, In KDD Workshop on Text Mining, 2000.

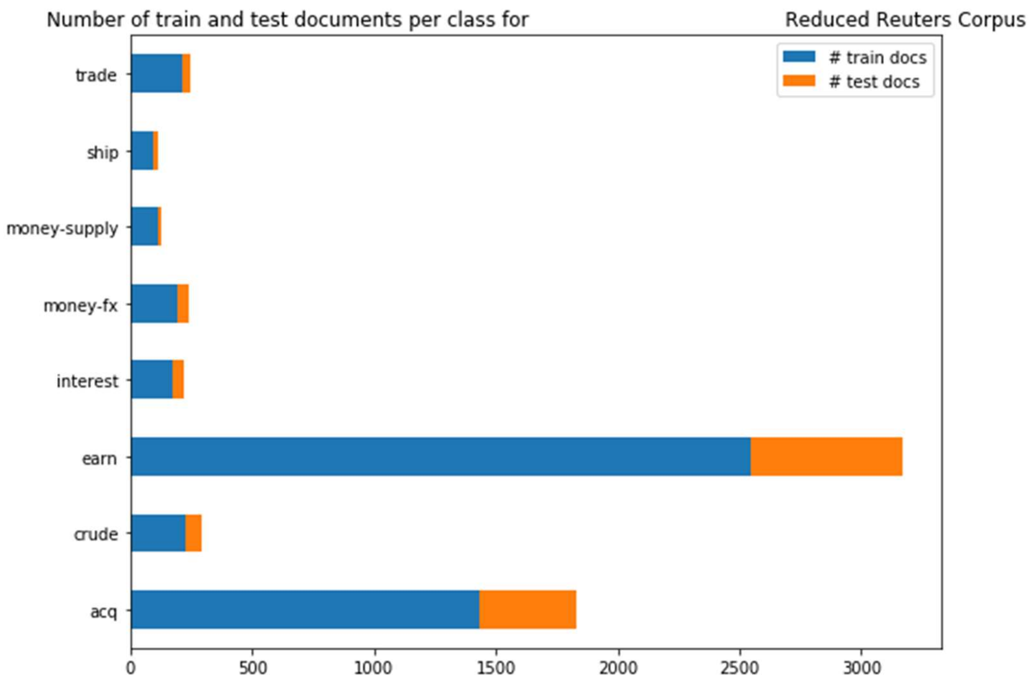
8. Balabantaray, R.C., Sarma, C., & Jha, M. (2015). Document Clustering using K-Means and K-Medoids. CoRR, abs/1502.07938.

APPENDIX A

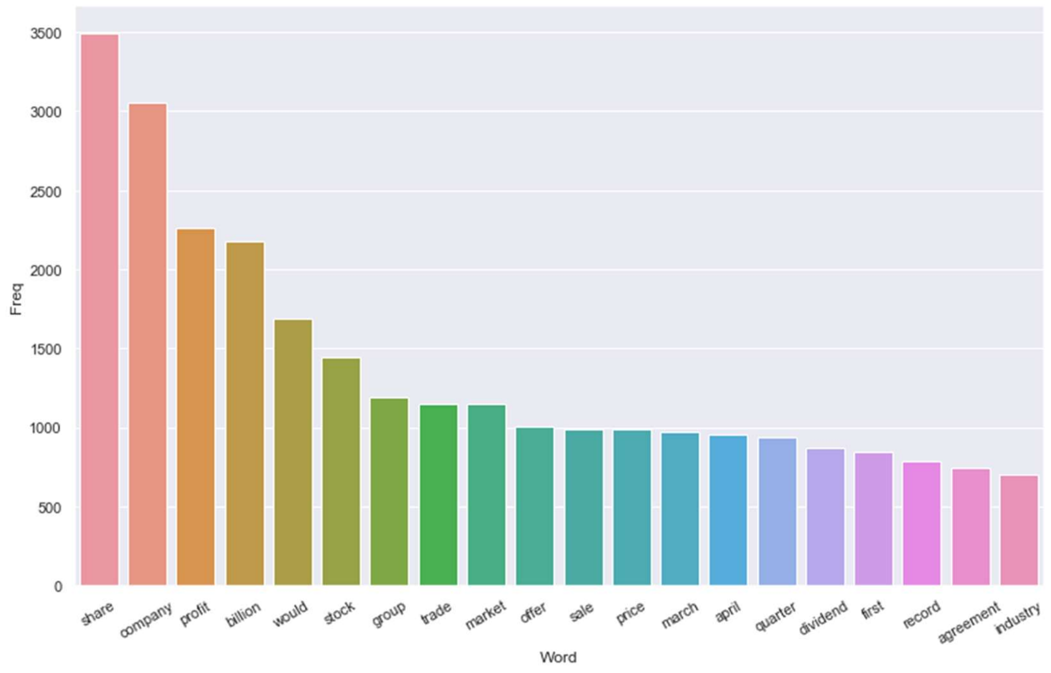
Exploratory Data Analysis Visualizations

Document Counts by Category and Dataset

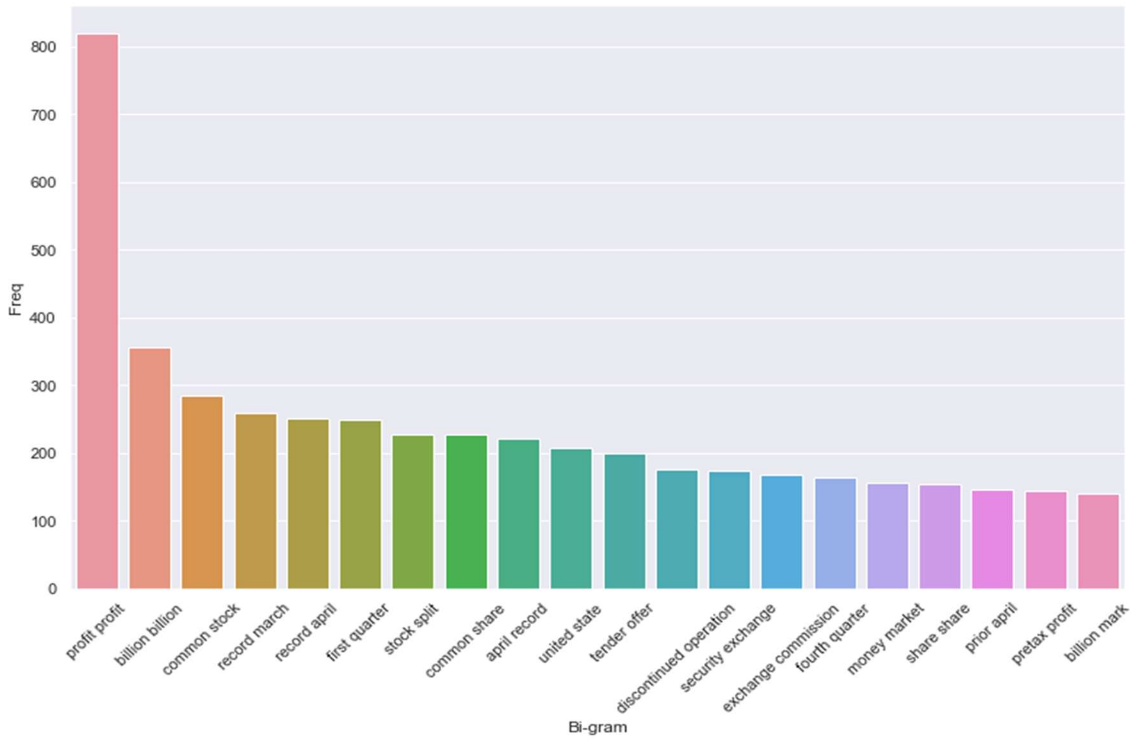
Category	# Train Docs	# Test Docs	Total Docs
earn	2,545	624	3,169
acq	1,436	394	1,830
crude	229	67	296
trade	212	38	250
money-fx	196	47	243
interest	175	45	220
money-supply	112	16	128
ship	95	19	114
Total	**5,000**	**1,250**	**6,250**

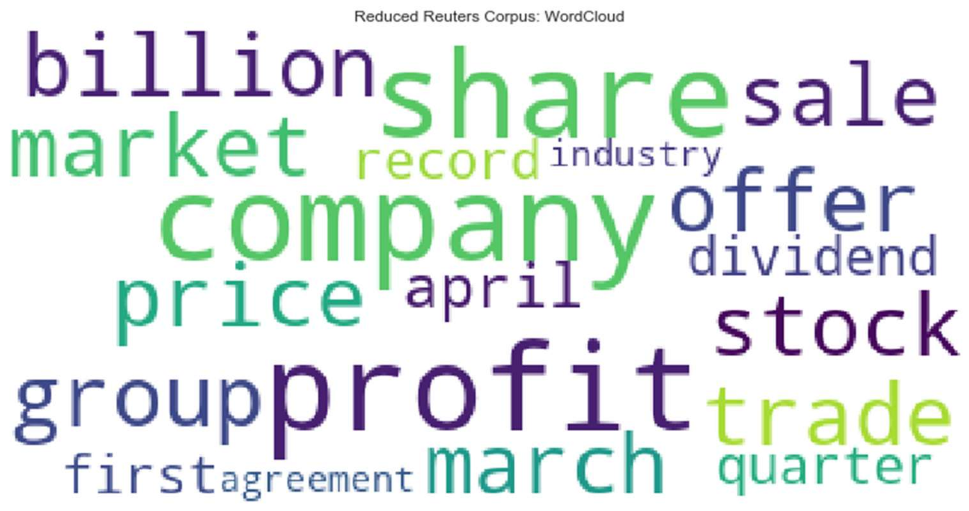


Top 20 Words Frequency



Top 20 Bi-gram Frequency





Top 20 Most Common Words

Word	Frequency
share	3,492
company	3,054
profit	2,259
billion	2,176
would	1,687
stock	1,443
group	1,187
trade	1,150
market	1,147
offer	1,007
sale	988
price	986
march	971
april	952
quarter	938
dividend	873
first	847
record	784
agreement	741
industry	705

APPENDIX B

Supervised Learning Algorithms

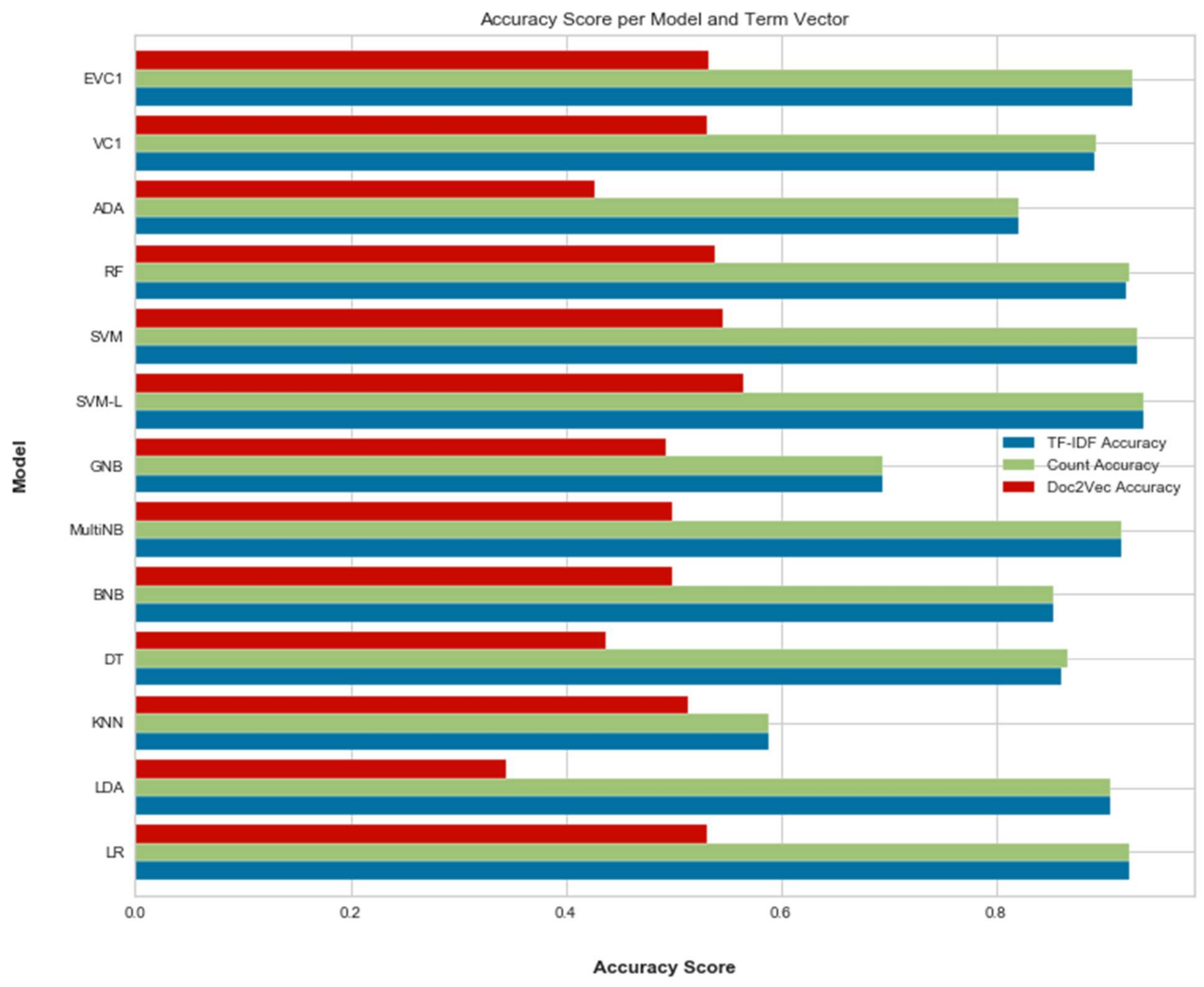
Model	Algorithm Name
LR	Logistic Regression
LDA	Linear Discrimination Analysis
KNN	K-nearest Neighbors
DT	Decision Tree
BNB	Bernoulli Naïve Bayes
MultiNB	Multinomial Naïve Bayes
GNB	Gaussian Naïve Bayes
SVM-L	Linear Support Vector Machine
SVM	Support Vector Machine
RF	Random Forest
ADA	Adaptive Boosting
VC1	Vote Classifier *Comprised of:* *K-nearest Neighbors* *Decision Tree* *Random Forest*
EVC1	Ensemble Vote Classifier *Comprised of:* *Logistic Regression* *Gaussian Naïve Bayes* *Random Forest*

APPENDIX C

Accuracy Score Metric Performance

Accuracy Score per Model and Term Vector

Model	TF-IDF	Count	Doc2Vec
LR	0.9240	0.9240	0.5312
LDA	0.9064	0.9064	0.3448
KNN	0.5880	0.5880	0.5128
DT	0.8600	0.8656	0.4368
BNB	0.8528	0.8528	0.4992
MultiNB	0.9160	0.9160	0.4984
GNB	0.6944	0.6944	0.4936
SVM-L	0.9368	0.9368	0.5656
SVM	0.9312	0.9312	0.5464
RF	0.9200	0.9240	0.5384
ADA	0.8200	0.8200	0.4264
VC1	0.8904	0.8928	0.5304
EVC1	0.9264	0.9256	0.5320
Average	**0.8590**	**0.8598**	**0.4966**

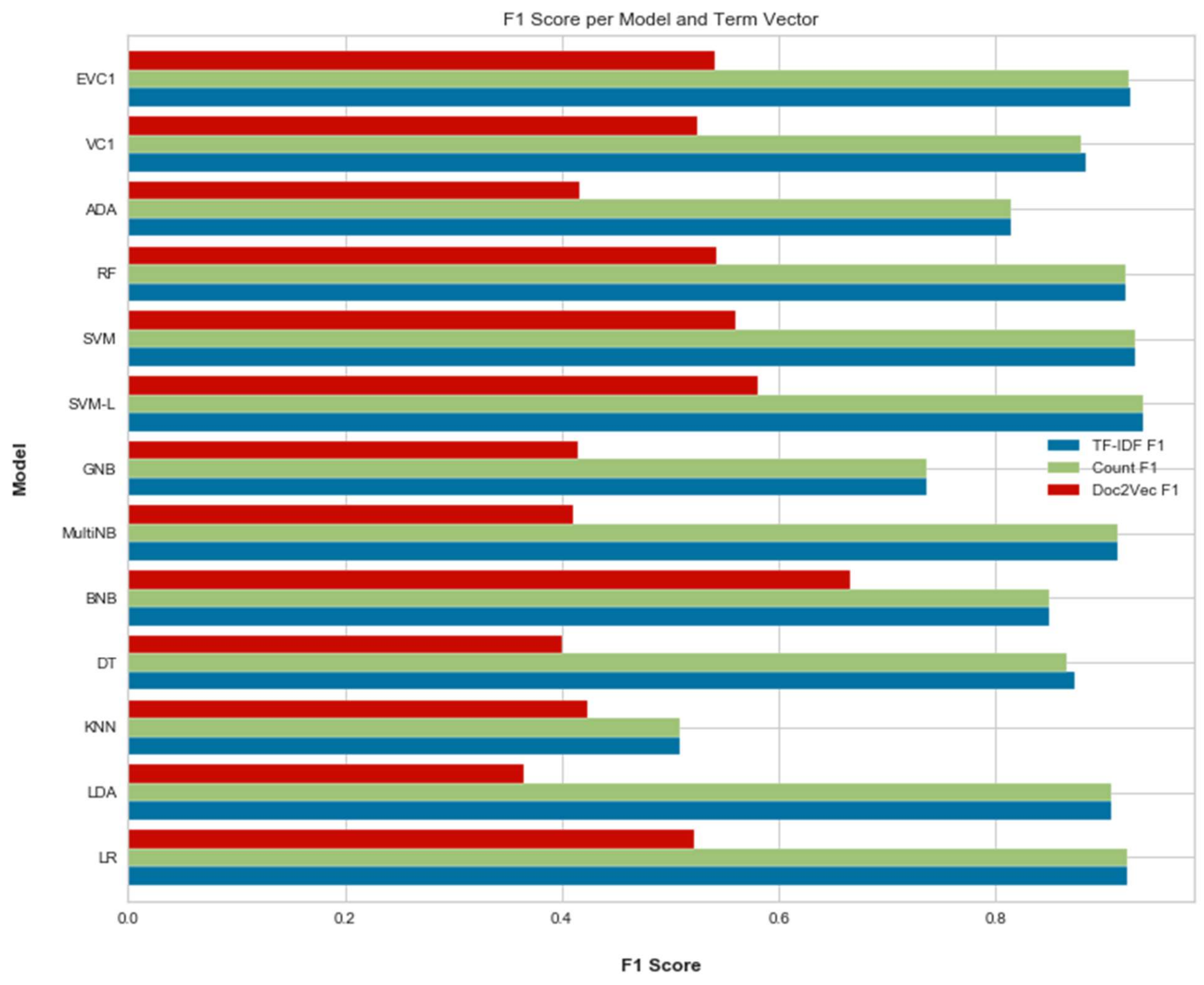

Accuracy Score per Model and Term Vector
EVC1
VC1
ADA
RF
SVM
SVM-L
GNB
MultiNB
BNB
DT
KNN
LDA
LR
Model
0.0
0.2
0.4
0.6
0.8
Accuracy Score
TF-IDF Accuracy
Count Accuracy
Doc2Vec Accuracy

APPENDIX D

F1 Score Metric Performance

F1 Score per Model and Term Vector

Model	TF-IDF	Count	Doc2Vec
LR	0.9211	0.9211	0.5221
LDA	0.9059	0.9059	0.3650
KNN	0.5083	0.5083	0.4238
DT	0.8722	0.8651	0.3999
BNB	0.8497	0.8497	0.6660
MultiNB	0.9130	0.9130	0.4104
GNB	0.7370	0.7370	0.4145
SVM-L	0.9362	0.9362	0.5811
SVM	0.9292	0.9292	0.5607
RF	0.9199	0.9199	0.5426
ADA	0.8145	0.8145	0.4164
VC1	0.8832	0.8783	0.5255
EVC1	0.9247	0.9229	0.5404
Average	**11.1149**	**11.1010**	**6.3682**

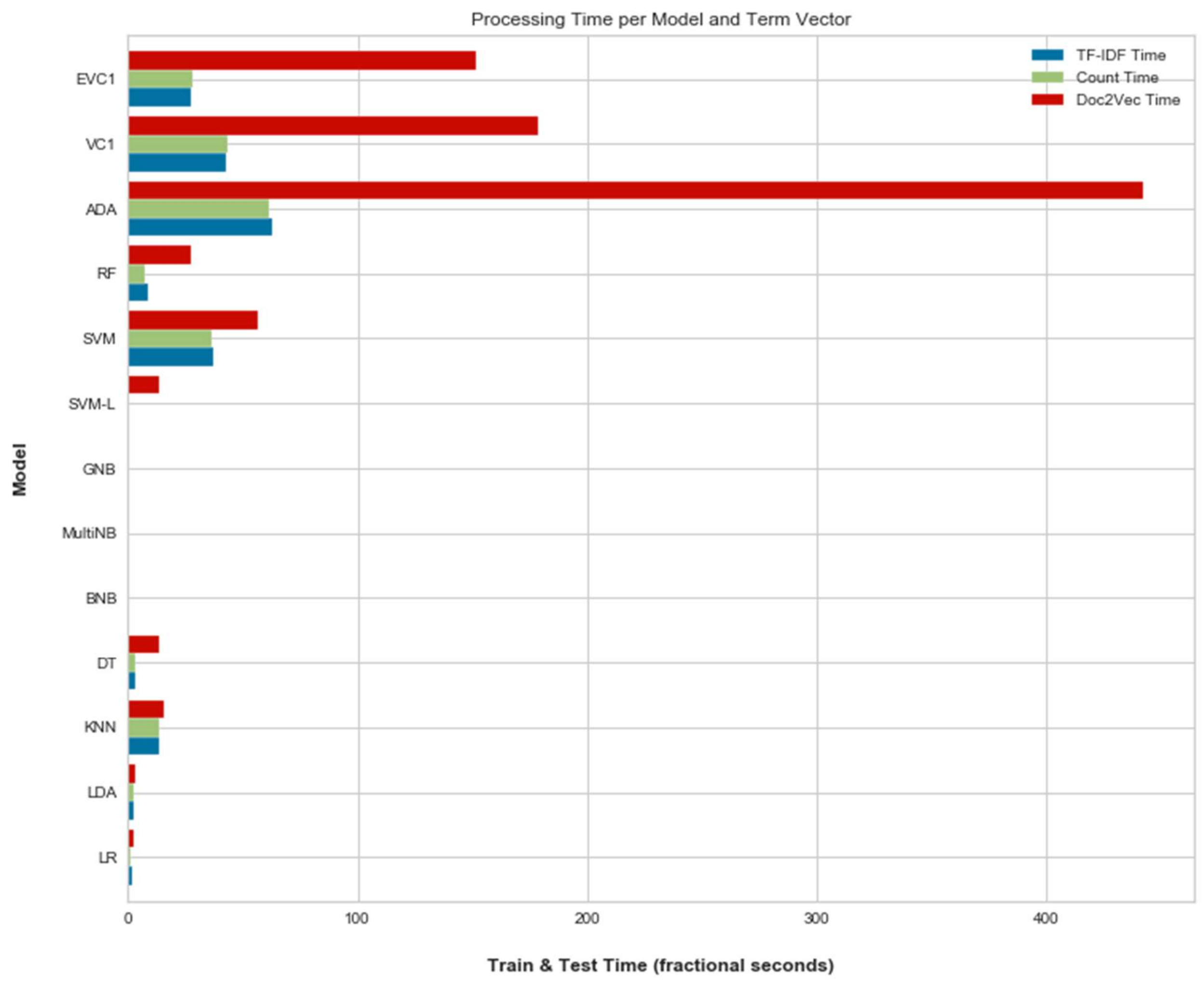

F1 Score per Model and Term Vector
EVC1
VC1
ADA
RF
SVM
SVM-L
GNB
MultiNB
BNB
DT
KNN
LDA
LR
Model
0.0
0.2
0.4
0.6
0.8
F1 Score
TF-IDF F1
Count F1
Doc2Vec F1

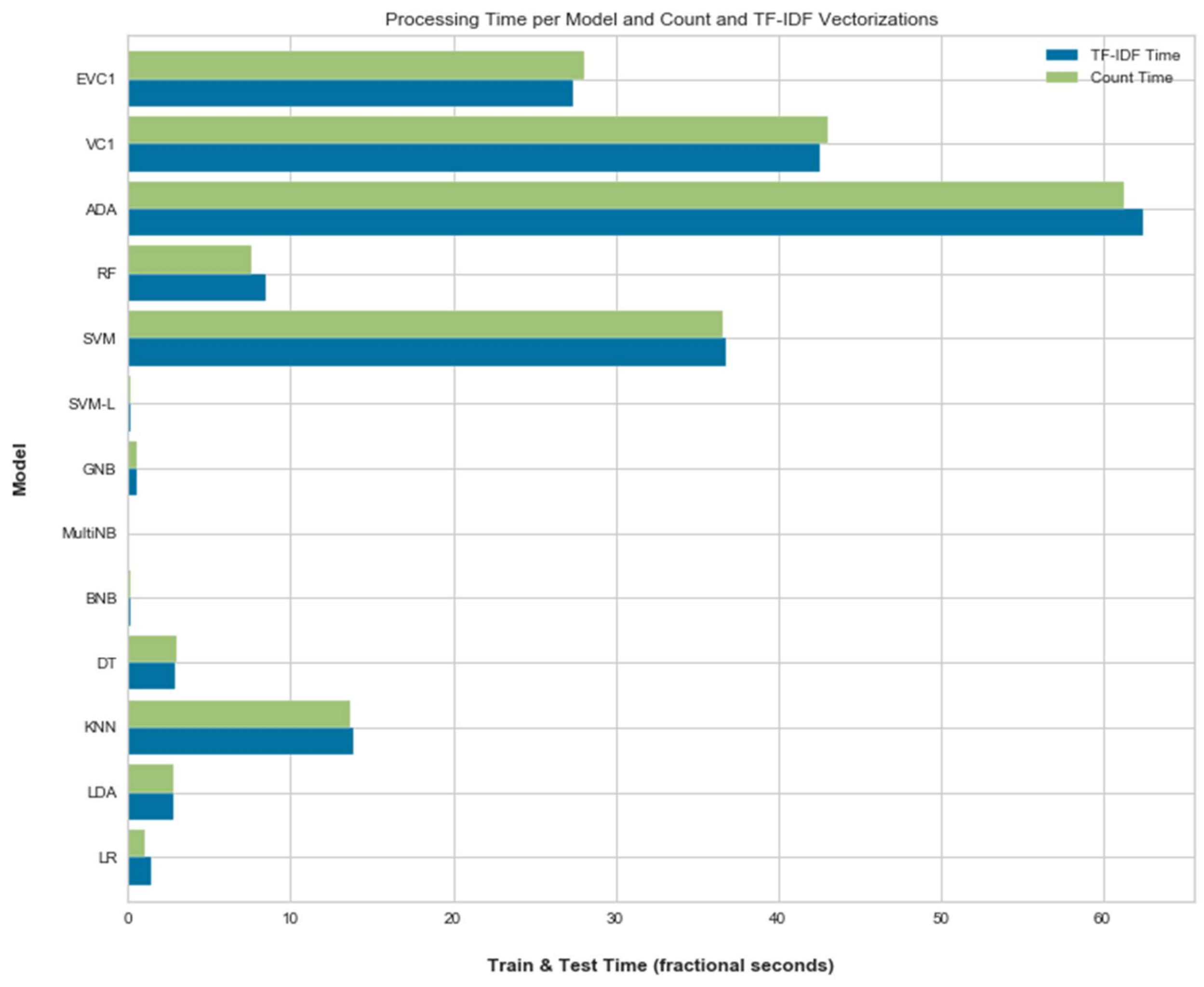
APPENDIX E

Processing Time Performance

Processing Times per Model and Term Vector

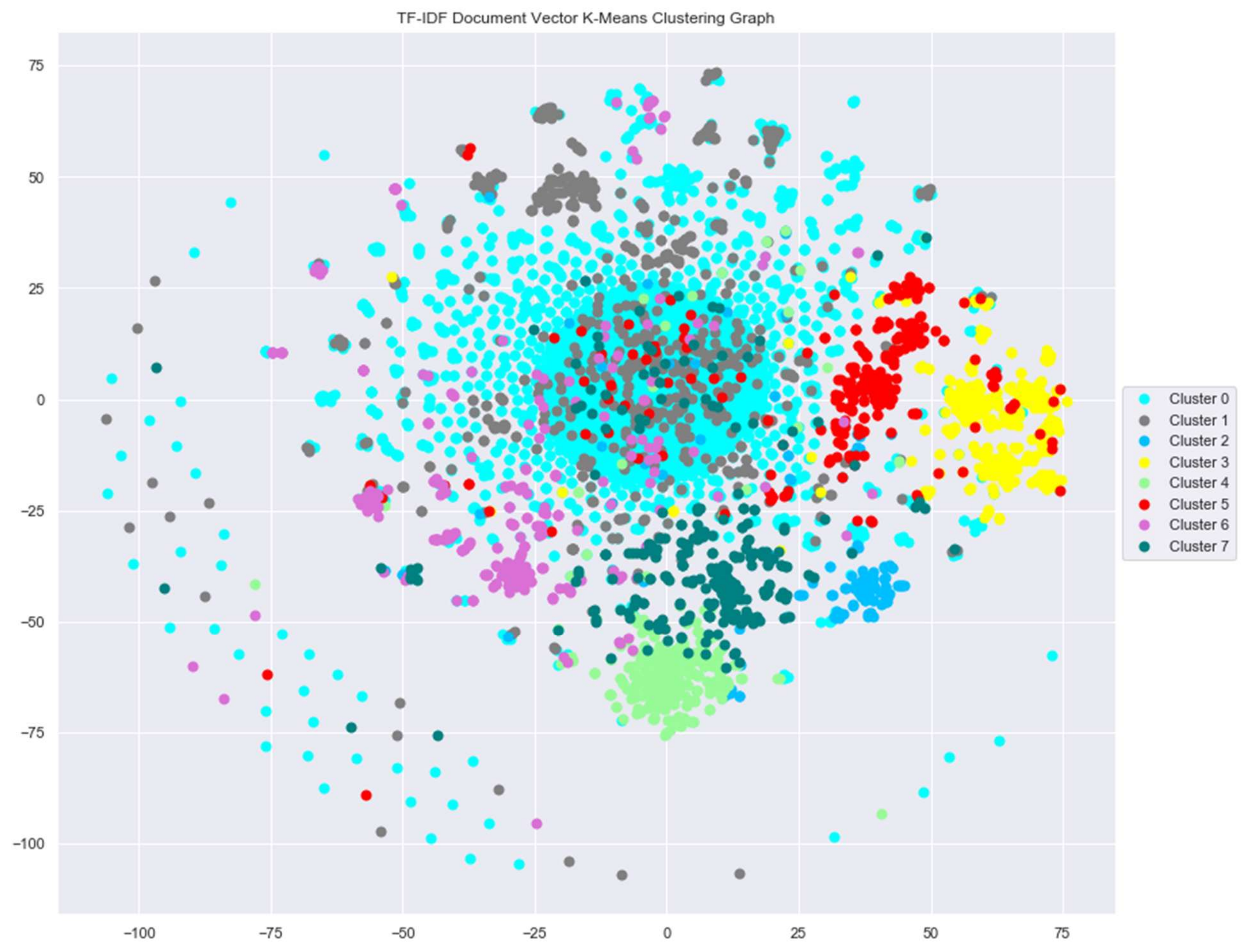
Model	TF-IDF	Count	Doc2Vec
LR	1.4659	1.0779	2.4026
LDA	2.7841	2.7568	2.8253
KNN	13.8897	13.7068	15.6780
DT	2.8812	2.9523	13.6261
BNB	0.1209	0.1289	0.3027
MultiNB	0.0510	0.0370	0.0819
GNB	0.5404	0.5594	0.4665
SVM-L	0.1479	0.1319	13.4955
SVM	36.8483	36.6457	56.4237
RF	8.4563	7.5854	27.3370
ADA	62.4976	61.2905	442.3813
VC1	42.6124	43.1188	178.4922
EVC1	27.4442	28.0514	151.3696
Average	**199.7399**	**198.0428**	**904.8824**

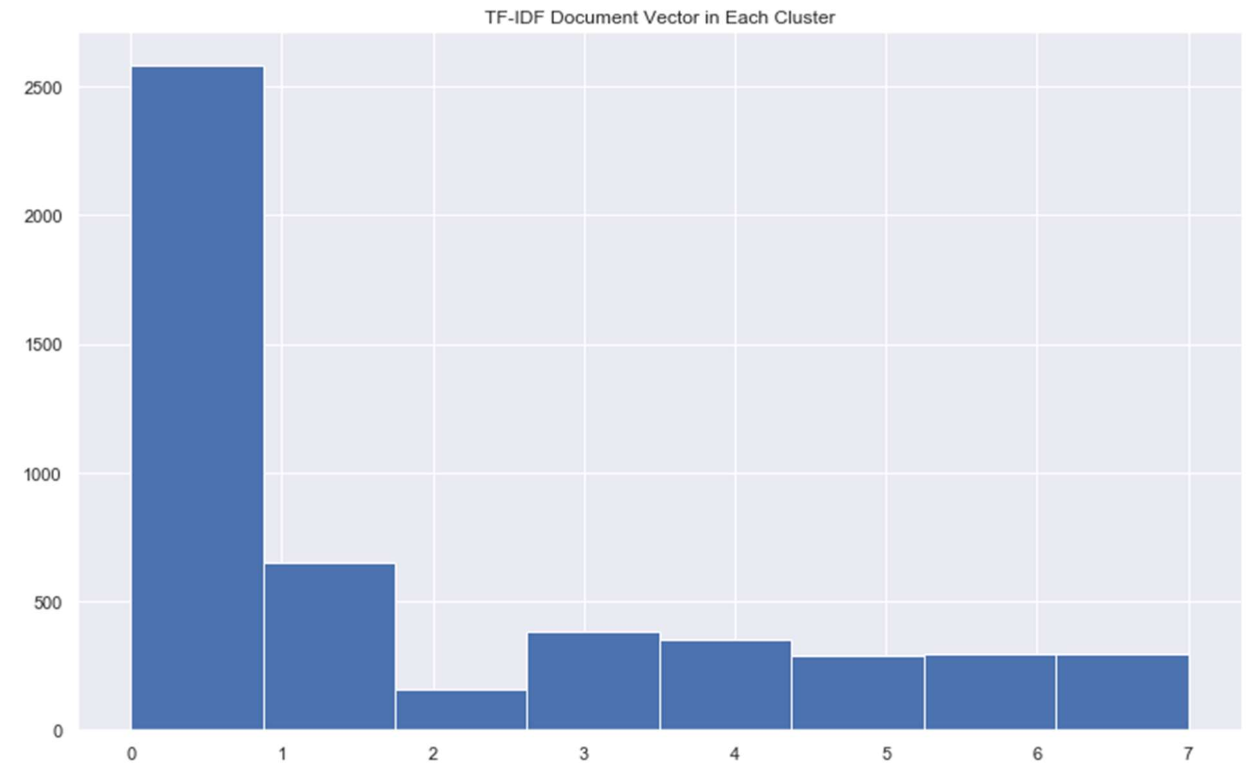

Processing Time per Model and Term Vector
TF-IDF Time
Count Time
Doc2Vec Time
Model
EVC1
VC1
ADA
RF
SVM
SVM-L
GNB
MultiNB
BNB
DT
KNN
LDA
LR
0
100
200
300
400
Train & Test Time (fractional seconds)


Processing Time per Model and Count and TF-IDF Vectorizations
TF-IDF Time
Count Time
EVC1
VC1
ADA
RF
SVM
SVM-L
GNB
MultiNB
BNB
DT
KNN
LDA
LR
Model
0
10
20
30
40
50
60
Train & Test Time (fractional seconds)

APPENDIX F

TF-IDF Vector K-Means Clustering Plots

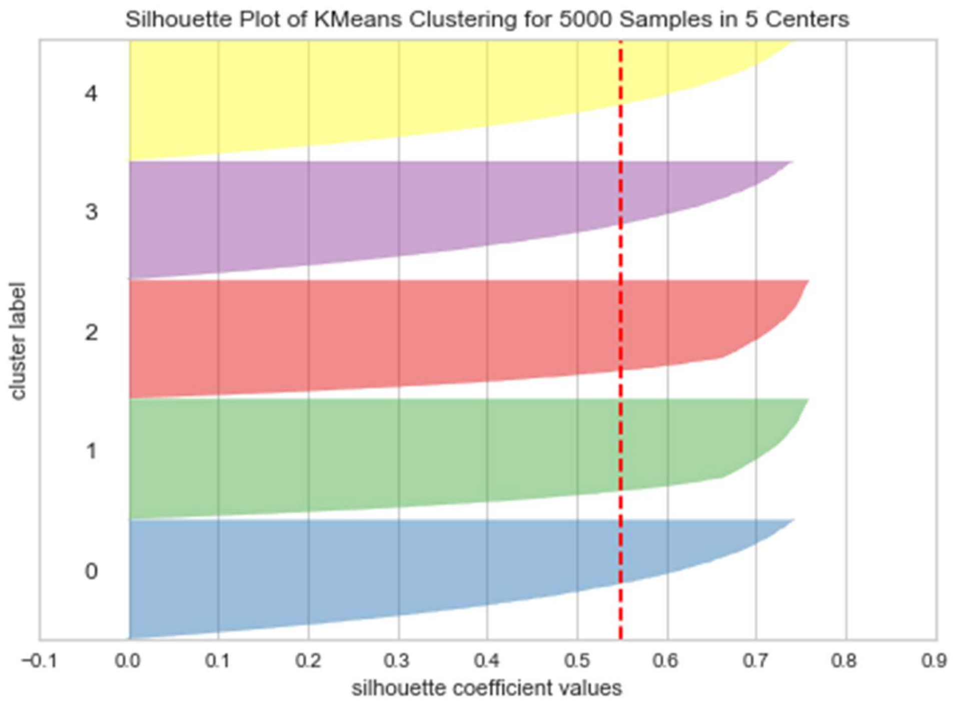


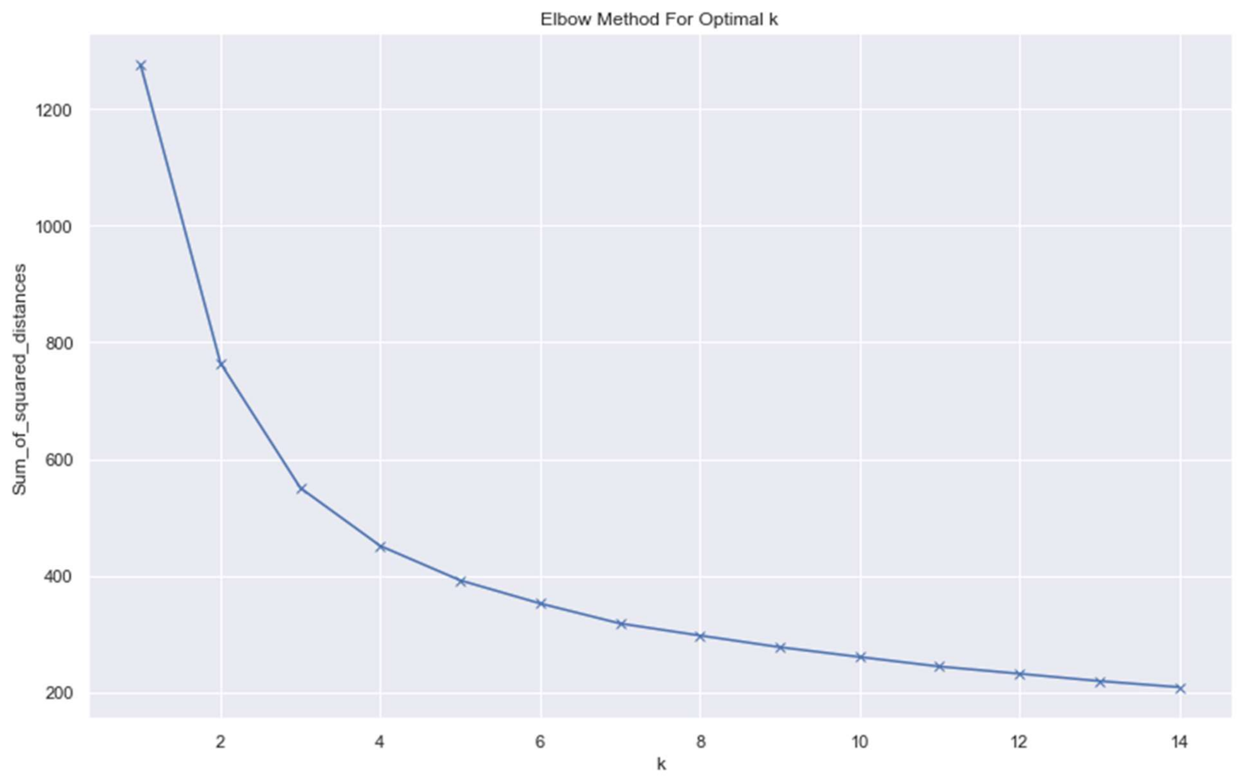

TF-IDF Document Vector in Each Cluster
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
7

APPENDIX G

Optimal Cluster Data and Plots

Number of Clusters	Silhouette Score	
2	0.6243	Optimal Value
3	0.5852	
4	0.5638	
5	0.5492	
6	0.5379	
7	0.5286	
8	0.52	
9	0.5117	
10	0.5032	




Elbow Method For Optimal k
Sum_of_squared_distances
1200
1000
800
600
400
200
2
4
6
8
10
12
14
k

APPENDIX H

Hyper-Parameter Tuning Analysis Results

Model	Base Accuracy	Optimized Accuracy	Best Estimators
SVM-L	0.9368	.9400	{'C': 0.5, 'penalty': 'l2'}
Random Forest	.9040	0.9184	{'criterion': 'gini', 'n_estimators': 100