Clustering for Semi-Supervised Learning on Disasters Tweets

Aran Seth Brian Fernando	100367750
Carlos Alfredo Becerra Torres	100357358
Erich Henrique Silva	100370053
Lucas Daniel Felix	100365550

April 15, 2022

Langara College

DANA-4840 001 – Classification II

Professor **Edward Chiu**

ABSTRACT

Cluster analysis as an unsupervised learning technique is widely implemented throughout many fields of data analytics. When applied to data suited for hierarchical or partitional clustering, it can provide valuable insights into latent groups of the dataset and further improve your understanding of key features that can describe and classify individuals into meaningful clusters for your use case. In this project, we explore an alternative application of partitional clustering to improve the performance of supervised learning classification tasks of text samples when the resources to label training data are limited. By introducing what we call "representative labeling" with K-Means clustering, we show a consistent improvement in classification metrics of Logistic Regression and K-Nearest Neighbors algorithms when compared to naively labeled instances.

1 Introduction

Some of the modern studies that rely on Natural Language Processing (NLP) techniques are often accompanied by data scraping methods that can quickly capture extensive amounts of data to compose large datasets. However, the process of manually labeling a subset of the collected data for training often doesn't have an equivalently scalable and reliable annotating method. Therefore, the approach studied in this project aims to maximize the efficiency of annotating hours by selecting and labeling the most representative candidates, leading up to a training set that is rich and representative of your corpus. Hence, we experiment with another use case of clustering as a semi-supervised learning technique (Géron, A. 2019) [4]. The method will be described in detail in the following sections. The potential audience ranges from machine learning researchers to data analytics students with limited resources to label training data.

2 Dataset

The chosen dataset is from a Kaggle competition named "*Natural language processing with disaster tweets*" [6], where the competitors are tasked with identifying which tweets from the 3,243 public test examples should be defined as a disaster (1) or non-disaster (0). It is a straightforward but challenging binary classification problem, where the model needs to capture the intent of the tweeted message as to whether it referred to an actual disaster event or not. The dataset and more information can be found on the competition website[1].

2.1 Training and validation split

To evaluate the results of the proposed method, we are using the test set provided in the challenge, with 7,503 manually labeled, unique tweets. The original data comes with one identifier column (id), two feature columns (keyword and location), and the uncleaned tweet text (text). Only the tweet text is kept for the following analyses.

Twenty percent (1,523) tweets are kept aside as a hold-out validation set (Validation set). The split is done without shuffling the data, where the top eighty percent (6,090) of the original training set is kept as our object of study (Training set). Keeping the original order of the tweets for training and validation split is fundamental not to introduce bias in candidate sample selection.

[1] https://www.kaggle.com/c/nlp-getting-started/overview

2.2 Pre-processing

Tweets text preprocessing consists of three main steps, namely text cleaning, vectorization, and dimensionality reduction.

2.2.1 Text cleaning

During the cleaning process, user mentions, URLs, and Unicode emoticon characters are removed from the text of the original tweets. To preserve the maximum amount of information, the hashtags are kept unchanged. Our chosen form of vectorization is able to leverage the frequency of important hashtags in the dataset (the ones that appear frequently) and ensure that this information is available to the classification models.

Original tweet examples:

> *"Set our hearts ablaze and every city was a gift And every skyline was like a kiss upon the lips @Â‰ÃÃ›_ https://t.co/cYoMPZ1A0Z"*

> *"@PhDSquares #mufc they've built so much hype around new acquisitions but I doubt they will set the EPL ablaze this season."*

Cleaned tweet examples:

> *"Set our hearts ablaze and every city was a gift And every skyline was like a kiss upon the lips"*

> *"#mufc they've built so much hype around new acquisitions but I doubt they will set the EPL ablaze this season."*

After cleaning, the dataset presented duplicate samples, but they are not removed since the frequency of discussions is essential in the study of social media.

2.2.2 Text vectorization

To convert word strings into a vectorized numeric representation, we opted for a Term Frequency, Inverse Document Frequency (TF-IDF, Sparck Jones, K. 1972) vectorizer [7].

As a well-known and widely implemented vectorization form for text data [5], it measures the term frequency of every observed word in the corpus vocabulary to assign it a weighted metric that is then penalized by the inverse frequency of the documents that contain given word. It is an effective form of driving the attention from frequently used words, terms, and expressions while keeping relevant the important descriptors of a particular group of documents.

The following equations are used to calculate the term frequency (tf), inverse document frequency (idf), and TF-IDF score (tfidf), respectively:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}} \quad (1)$$

$$idf(t,D) = \log\left(\frac{N}{n_t}\right) \quad (2)$$

$$tfidf(t, d, D) = tf(t,d) \cdot idf(t,D) \quad (3)$$

Where:

f: Frequency
t: Term (word)
d: Document
D: Corpus
N: Number of Documents (d) in Corpus (D)
n_t: Number of Documents (d) having Term (t)

The TF-IDF vectorization uses a one-hot encoding form of transformation. Each vocabulary word is assigned to a column, and its' TF-IDF weighted score is stored in the column when the term is present in the original text. As a result, each tweet is now represented in a vector space of N dimensions, where N is the size of the vocabulary present in the corpus.

The TF-IDF vectorizer fitted in our 6,090 training tweets identified 13,053 unique words and was then used to transform the 1,523 observations in the validation set. Table 1 shows the shape of the vectorized subsets.

Table 1. Training and Validation sets shapes after TF-IDF

Subset	Rows	Columns
Training	6090	13053
Validation	1523	13053

2.2.3 Dimensionality reduction

To make the dataset suitable for the clustering analysis, a dimensionality reduction technique is implemented. First, through a Singular Value Decomposition (SVD), the data is reduced to its first "r" components that better capture the variance presence in our TF-IDF vectorized training set. The data is then normalized to unit L2-norm. When applied to text data, the two techniques together are also known as a Latent Semantic Analysis [1] and result in the final form of the training set used for clustering.

To select the ideal number of "r" components, a technique introduced by Gavish and Donoho, 2014 [3] is used. In summary, it finds the optimal Singular Value Hard Threshold (SVHT) based on the distribution of the log of Singular Values (captured variance) in the SVD. By truncating the SVD results on this given value, it is said to effectively remove the components that mainly contain noise from the original data. Figure 1 shows the thresholding point for the training set, where 652 components are found as the optimal "r", explaining 53.75% of the variance in the dataset.

It is important to observe that dimensionality reduction was solely used to improve clustering results and reduce computation time. Therefore, it is only implemented on the training set, where the representative samples are extracted for labeling. The original TF-IDF vectorized training and validation sets are used for training and inference during the classification tasks.

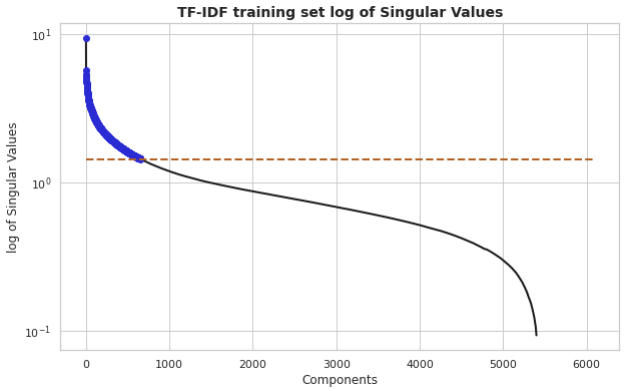

Figure 1. Optimal singular value hard threshold of training set

3 Methods and results

Two distinct classification algorithms are chosen for the classification task of labeling the tweet as disaster (1) or non-disaster (0). Logistic Regression and a K-Nearest-Neighbors Classifier.

The classifiers are exposed to all 6,090 instances of our training set during training. Their performances are then evaluated on the 1,523 samples in the validation set to measure the maximum achievable metrics performance. Subsequently, limited amounts of training samples are used for training and assessed against the same 1,523 validation observations. The training samples are selected via two methods. Naively, taking the first K observations in the training set. And with representative instances, picking the K observations closest to the centroids of K groups clustered with a K-Means algorithm. This process will be covered in detail in the following sections.

3.1 Classification Algorithms

Since the focus of the study is not on the classification task but on the effectiveness of using a partitioning clustering method to select

training samples, the classification algorithms are used in their simplest and standard forms, as they come from their parent libraries.

The Logistic Regressor is present in Python Scikit Learn linear model library, and its default parameters are found in the library documentation[2].

The K-Nearest-Neighbors Classifier is present in Python Scikit Learn neighbors library, and its default parameters are found in the library documentation[3].

3.2 Validation and evaluation metrics

Accuracy and F1 score are the metrics selected to evaluate the classification performance during validation. The F1 is given by the following equation:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \qquad (4)$$

3.3 Baseline performance

After training on the complete training data (6,090 labeled samples), the Logistic Regression achieved 79.0% accuracy in the validation set. Meanwhile, The KNN Classifier achieved 72.4% accuracy. Table 2 shows the metrics for the baseline performance.

Table 2. Validation metrics of classifiers trained on 6,090 labeled tweets

Classifier	Accuracy	F1 Score
Logistic Regression	79.0%	0.754
K-Nearest-Neighbors	72.4%	0.659

3.4 Representative labeling

Looking at our 6,090 training set samples, we propose a hypothetical scenario where one doesn't have labeled instances to perform training. Instead, a small number of tweets is selected to be manually labeled for training on the classification task. This section studies the use of K-Means clustering to segment your corpus into K distinct clusters, from where K representative instances will be selected for labeling. Those instances are the ones closest to the centroid of each cluster, therefore being the best representatives of their surrounding samples. Figure 2 illustrates this process.

The labels for the selected observations are then extracted from the ground-truth present in the original data, and a subset of K samples is used for training. Table 3 shows the results for 300 representative-labeled tweets compared to 300 naively selected ones.

Figure 2. Representative samples

[2] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Table 3. Validation metrics of classifiers trained on 300 representative (R) and naively (N) labeled tweets

Classifier	Accuracy	F1 Score
Logistic Regression (R)	61.7%	0.353
K-Nearest-Neighbors (R)	68.7%	0.648
Logistic Regression (N)	54.0%	0.028
K-Nearest-Neighbors (N)	61.3%	0.464

It is possible to observe that the accuracy increased by an average of 7% on both classifiers. As expected, the 300 representative samples are more valuable to show the model during training than simply selecting the first 300 observations.

3.5 Label propagation

A further performance improvement can be achieved by propagating the labels of the selected representative samples to their surrounding observations [4]. Table 4 shows the results of labels propagated to the 3% surrounding percentile around the previously 300 representative samples. It results in an extended subset of 680 tweets, and the improvement is particularly noticeable on the Logistic Regression classification metrics. Figure 3 illustrates the label propagation process.

Table 4. Validation metrics of classifiers trained on 680 label-propagated tweets

Classifier	Accuracy	F1 Score
Logistic Regression	71.2%	0.649
K-Nearest-Neighbors	68.2%	0.650

The quality of the propagation can explain the improvement. In this example, the propagated samples are 89.85% accurate compared to the ground-truth labels on the original dataset.

3.6 Optimal number of clusters

The ideal number of clusters for text data is challenging to define. For example, a low number of clusters K might lead to training samples that under-represent your dataset, while a high value of K might segment the corpus into an excessive number of groups that start to overlap in meaning and carry too much semantic redundancy.

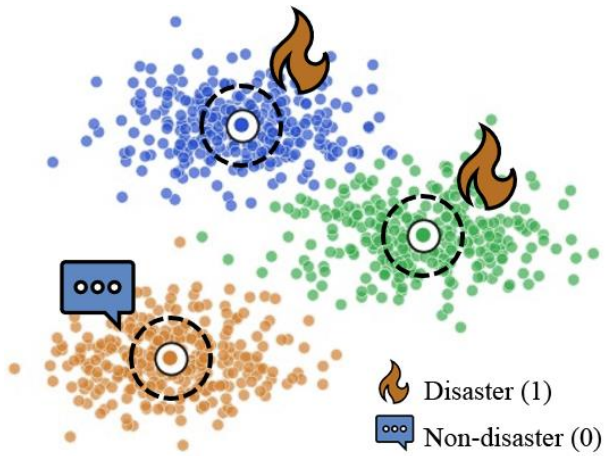

Figure 3. Label propagation

This section shows the results of representative labeling when iterated over different numbers of clusters K.

The metric selected to evaluate clustering performance was the Silhouette Width [2], described by the formula:

$$S(i) = \frac{b(i)-a(i)}{\max\{a(i), b(i)\}} \quad (5)$$

Where:

a(i): Average distance between the i^{th} observation and other observations in the same cluster

b(i): Average distance between the i^{th} observation and other observations in the nearest cluster

S(i): silhouette score of the i^{th} observation.

Table 5 shows the results of the average silhouette width for the increasing number of clusters K and the elapsed time on each step. The algorithm was initialized three times, and for each value of K, ten independent clustering results are stored to diminish the effect of random initialization of cluster centers.

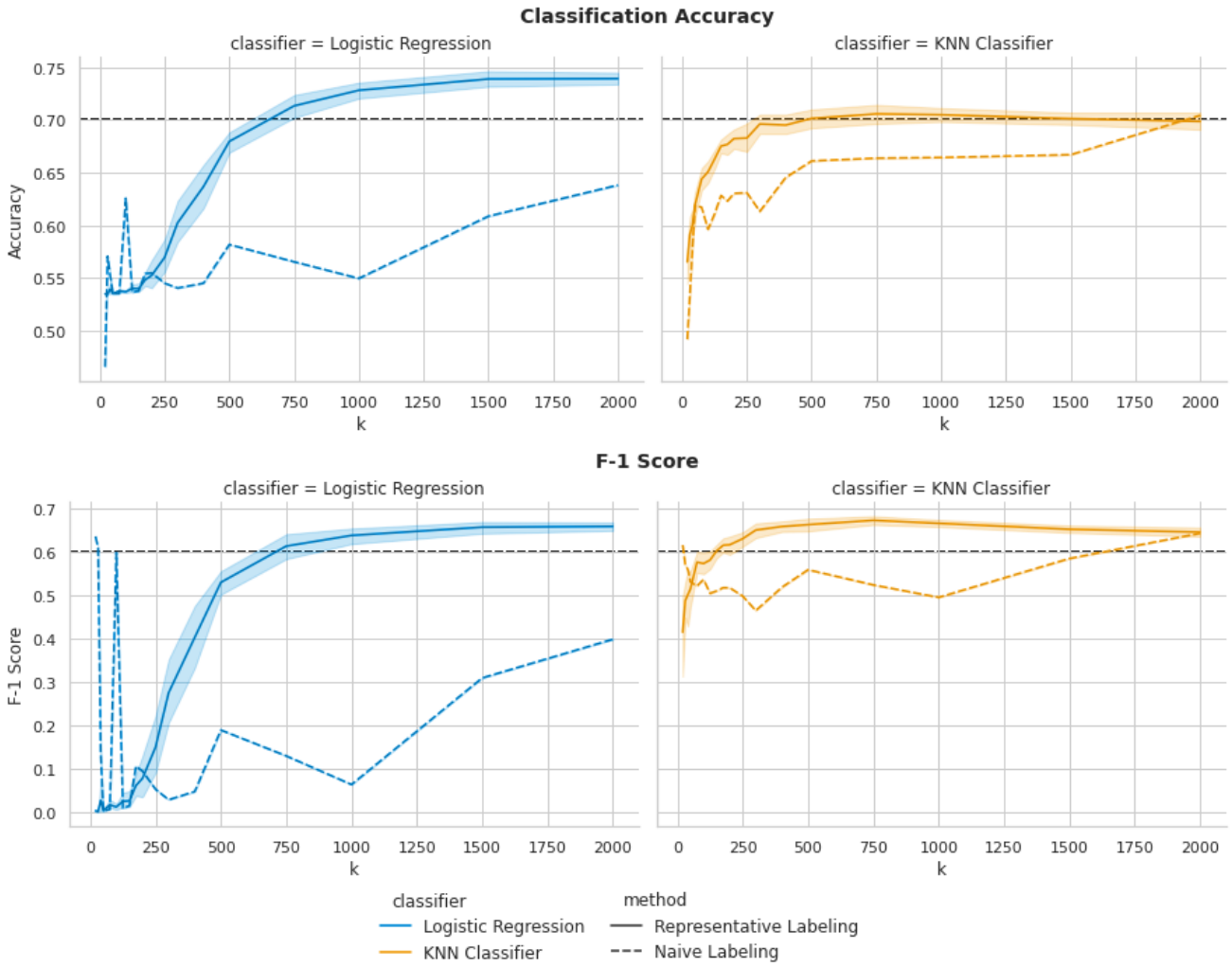

Figure 4. Accuracy and F1 Score of representative and naive samples

Table 5. Average silhouette width for an increasing number of clusters K

K	Avg. Silhouette	Elapsed Time
20	0.0231	19.840s
30	0.0311	21.973s
40	0.0403	23.917s
50	0.0486	26.262s
75	0.0670	30.038s
100	0.0816	39.082s
125	0.0940	42.099s
150	0.1057	47.611s
175	0.1158	50.477s
200	0.1242	53.305s
250	0.1375	59.642s
300	0.1461	63.621s
400	0.1585	75.514s
500	0.1671	88.280s
750	0.1783	113.290s
1000	0.1874	139.351s
1500	0.1983	212.048s
2000	0.2038	273.093s

It is seen that the average silhouette width increases progressively with K. This is a hint at the diversity present in the dataset, a common characteristic when working with text data.

Figure 4 shows the result of K representative samples (selected via K-Means clustering) compared to K naively selected ones. The shaded area around the plot curve illustrates the 95% confidence interval of the ten independent initializations of the clustering algorithm. It is possible to see that the representative labeling

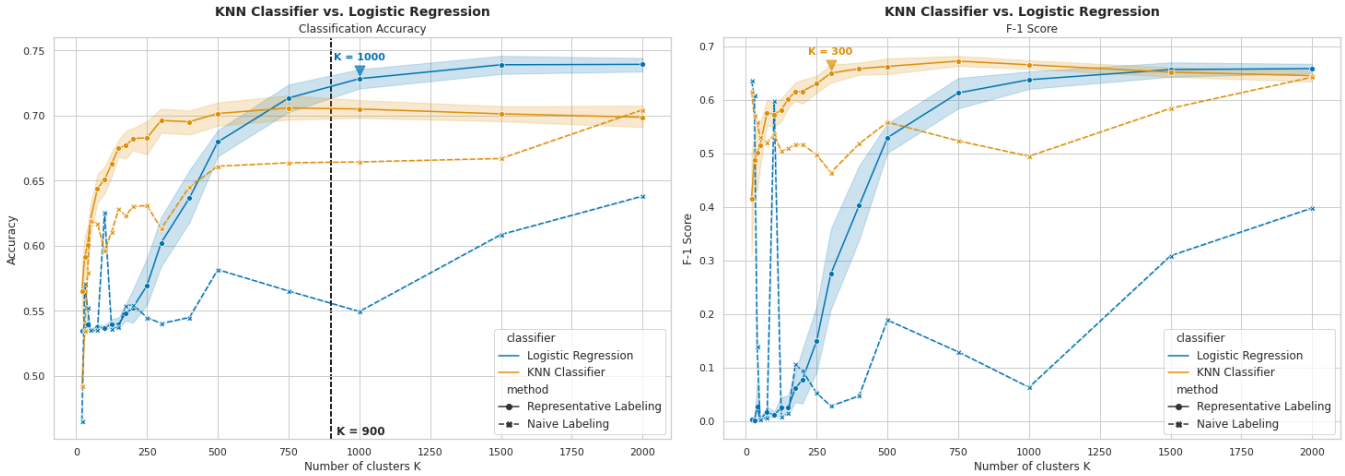

Figure 5. Accuracy and F1 Score of KNN Classifier vs. Logistic Regression

method results in a stable performance increase on both metrics.

Figure 5 compares both classifier results. We can see that the KNN Classifier has an early build-up of the performance metrics with a smaller number of samples, while the Logistic Regression tends to outperform it after being trained on 900 or more observations.

3.7 Cluster characteristics

Clusters are investigated to better understand how the dataset was segmented and which are the representative instances selected in the process. For that, the silhouette scores are plotted to identify the best clusters in terms of clustering effect. Figure 6 illustrates the silhouette plot at K = 50.

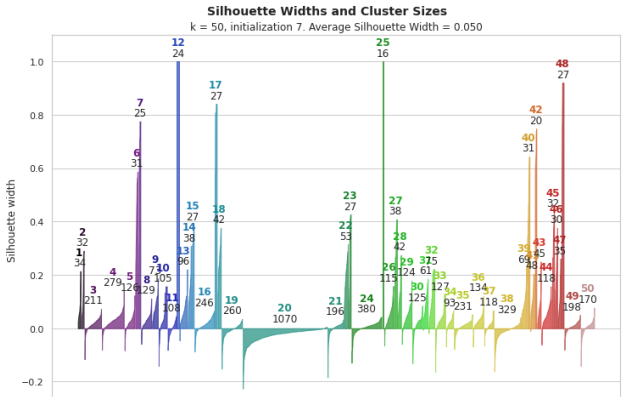

Figure 6. Silhouette plot and cluster sizes

When investigated, clusters with silhouette widths closer to 1.0 captured in the same group similar or identical tweets, which are often related to the same topic or a particular event, such as “Train Derailment in Madhya Pradesh”, “Legionnaires’ disease”, or “Malaysia MH370 flight debris”. Other clusters with average positive silhouette widths also tended to agglomerate tweets with similar words and sentences but resembling more generalized topics such as “Murder”, “Fire”, or “Flooding”.

3.8 Alternative clustering methods

The geometric approach of retrieving the representative individuals based on proximity to the cluster center limits the application of the presented method to spherical, non-hierarchical clustering techniques. As an alternative method, Affinity Propagation can be studied due to its nature of finding representative exemplars to converge the dataset into clusters.

4 Conclusion

Representative labeling with K-Means clustering as a semi-supervised approach to classification tasks, as initially introduced by Géron, A. 2019

[4] in the MNIST dataset, shows promising results also on text data, as seen in this study.

Figure 7 reveals the metric scores for the optimum number of training samples on both classification algorithms. Again, it is possible to observe a substantial increase in performance with the proposed method. It can, therefore, be suggested as a solid alternative to select the best samples for labeling when the resources for data annotation are limited.

4.1 Future research

In future research, some topics can be explored to improve the performance of the clustering method and potentially lead to better results.

1. Implement pre-trained word embeddings to capture a semantic representation of the words. As an alternative to the proposed LSA for dimensionality reduction, word embeddings can vectorize the text into a dense geometric representation that can leverage the semantics of words.
2. Introduce a topic modeling algorithm as a feature engineering task. By assigning samples to topics, they can be compared to clustering results to better assess how effectively your clustering method segmented the dataset.

4.2 Considerations

Some important aspects to consider when applying the proposed method.

1. Representative labeling might induce bias in the classification model if K is too low for the diversity present in the original dataset.
2. Overfitting might occur earlier during the training and validation process when using representative labeled samples since the concept sets aside hard-to-define, diffuse, and less representative samples.
3. Some of the K-Means assumptions for an accurate clustering are not proved during this method due to the high dimensionality and number of clusters. Hence, unequal variance and anisotropicity problems might result in inappropriate cluster centers (and, therefore, inappropriate representative samples).

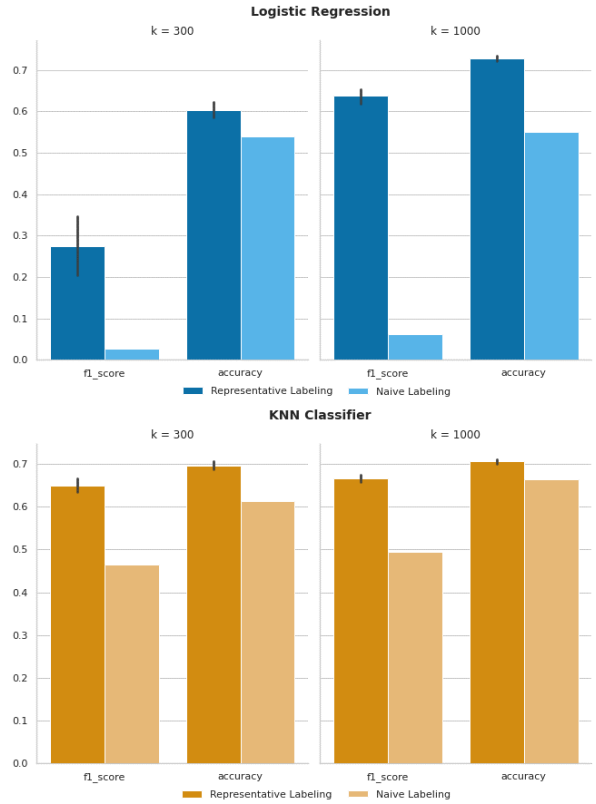

Figure 7. Representative labeling results

5 Acknowledgement

The dataset used in this analysis is from the Kaggle competition "*Natural language processing with disaster tweets*" [6], and the rules of fair use were respected during this study.

The inspiration for the methodology is from the book "*Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*" by Géron, A. 2019 [5].

REFERENCES

[1] Dumais, S. T. (2005). Latent Semantic Analysis. In *Annual Review of Information Science and Technology* (Vol. 38, pp. 188–230). essay, Journal of the American Society for Information Science.

[2] Edward Chiu. (2022). Chapter 2 Partitioning Clustering. In *DANA 4840 Classification II*. essay.

[3] Gavish, M., & Donoho, D. L. (2014, June 4). *The optimal hard threshold for singular values is 4/sqrt(3)*. arXiv.org. Retrieved April 2, 2022, from https://arxiv.org/abs/1305.5870

[4] Géron, A. (2019). Chapter 9. Unsupervised Learning Techniques. In *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd Edition). essay, O'Reilly Media, Inc.

[5] Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras and tensorflow: Concepts, tools and techniques to build Intelligent Systems*. O'Reilly.

[6] *Natural language processing with disaster tweets*. Kaggle. (n.d.). Retrieved April 2, 2022, from https://www.kaggle.com/c/nlp-getting-started/overview

[7] SPARCK JONES, K. A. R. E. N. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11–21. https://doi.org/10.1108/eb026526