Practical Machine Learning

Tomeh Lara – Artificial Intelligence 407

Real / Fake Job Posting Prediction

This project analyzes and classifies data using python language. It is illustrated algorithms, the supervised and the unsupervised ones. The formers are solved by decision trees and support vector machine, while the latter, that is to say, the unsupervised algorithms are solved using KMeans clustering and OPTICS clustering, with the help of various libraries, the most famous of them are (Numpy, Sklearn, nltk, Pandas, etc). Most of the data consists of textual data and a little of it is numeric, which is 18 features and 17,880 samples. It is noteworthy that the attempts with 2 different types of features is not possible because the data is mostly text and there are only three columns which are zeros and ones which are absolutely not enough to represent the algorithms on them.

The dataset consists of several advertisements for job opportunities that have been classified as fake or real. Initially, a report on the data was generated using data prep in order to form the basic idea about it, to be cleaned and arranged to be ready for use and implementation of different algorithms. It was noted that the data is never balanced, thus, three columns containing numerical data that have been abandoned and the text data only used, although, all properties are combined into one text to be processed in a faster and easier way as a single block, and thus the data became only two properties. The second step is the data cleaning and natural language processing, this was done by creating a function that converts text to lowercase letters, removes all symbols, blank spaces, punctuation marks, html tags, special characters and numbers, also, gets rid of abbreviations and words called “stop words”, which are words that are frequently repeated in the language and are eliminated to facilitate training the model and extract more accurate and faster results, in addition, it was deleted all the words that are not found in the English language dictionaries and in the end there are two operations each time only one of them is executed, namely, stemming and lemmatizing, stemming algorithms work by taking a list of common prefixes and suffixes found in inflected words and cutting off the end or beginning of the word. This indiscriminate cutting can work in some cases, but not always, while lemmatization takes into

consideration the morphological analysis of the words, it often takes into account words and their context in a sentence. The text is then prepared to apply the algorithms after this function has been applied once to the full document.

Two very important tools were used for converting text data into vectors as model can process only numerical data, Count Vectorizer which is provided by the scikit-learn (Sklearn) library. It is used to transform a given text into a vector on the basis of the count of each word that occurs in the whole text and it's like making a bag of words, and TF-IDF which is stands for Term Frequency- Inverse Document Frequency, this is a statistic that is based on the frequency of a word in the corpus but it also provides a numerical representation of how important a word is for statistical analysis. The difference between these previous tools is that the first one count the number of times a word appears in the document, while the second one considers overall document weightage of a word. Moreover, a text document that has n consecutive objects, including words, numbers, symbols, and punctuation, is known as an n-gram. In many text analytics applications where word sequences are important, such as sentiment analysis, text categorization, and text production, N-gram models are helpful. In this project, unigram, bigram, and sometimes trigram was used.

Now the data is ready for modeling & model evaluation. Therefore, the data was divided into two sections, the larger one for training the model and the smaller one for testing it after that and calculating the accuracy score. The precision, recall, f1-score, and confusion matrix were calculated as well. The confusion matrix shows how many ads were predicted as it is, and how many ads were predicted incorrectly.

It is worth noting that each model has been tested four times and each time the parameters are changed, for example, the size of the data that is tested after training the model, the size of Ngram, stemming or lemmatizing the text, model parameters itself and others.

As for support vector classifier, the model was fitted for the both bag of words and Tf-Idf features, for normal SCV with kernel “linear”, in addition, Linear SVC was used which is Similar to SVC with parameter linear kernel, but implemented in terms of “liblinear” rather than ”libsvm”. When both were applied to the same data, the Linear SVC achieved higher accuracy results.

Furthermore, decision trees were tested four times and each time produced different results, some of which were considered good and very accurate and others which were not, but each time, Tf-Idf vectorizing method achieves higher results than Count Vectorizer.

As for the unsupervised algorithms, the use of KMeans was started, of course, after preprocessing the data and everything that was previously mentioned, the elbow method was used in order to determine the best number of clusters, from which it was found that the data is around two clusters, but the system was tested on six, four, three, and two clusters as well, respectively. The most words found in each group were extracted and a new phrase was tested to show which group it belonged to, in addition to that, a plotting was made for the clusters.

When making plotting for clusters, two methods were used, namely PCA, which is better at capturing data's global structure, and TSNE which is better at capturing neighborly relationships. Both KMeans and MiniBatchKMeans were used, and while comparing both, it was discovered that the MiniBatchKMeans is faster, but produces slightly different results, claims the official website.

Last but not least, in the OPTICS algorithm, after processing the data it was also applied four times, but each time the application is on a part of the data and not all of it, because this algorithm is very slow to implement and run, and the confusion matrix is computed. This section demonstrates how the OPTICS algorithm discovers high-density areas in a dataset, as well as how it compares to DBSCAN and how it differs from it. OPTICS isn't a clustering algorithm, strictly speaking. Instead, it arranges the data's examples in a way that makes it possible to draw out clusters. In order for OPTICS to recognize which cases are core points, the model plotted the reachability distance, which is the distance between one core point and another core point within its epsilon. In order to determine whether the core distance of each instance in the data is less than epsilon, the algorithm first visits each case in the data. A case is a core point if its core distance is less than or equal to epsilon and a case is not a core point if the core distance is bigger than epsilon. A plotting for DBSCAN also was made with two different values of epsilon, although it is quicker than OPTICS, it does not perform well over clusters with various densities.

In conclusion, we found that the Tf-Idf vectorization in the majority achieved better results than the count vectorizer, in addition to the fact that the highest results were reached using the algorithm of the Linear support vector classifier for supervised algorithms, and unsupervised algorithms take

a long time to execute, but it achieves good results and, in this project, KMeans was better, and its ease of implementation is its best feature.

This table illustrates the supervised model that used with different parameters.

Model	Test size	Stem or Lemm	Ngram Range	Bow Accuracy Score	Tf-Idf Accuracy Score	Model	Model parameters
SVM1	0.2	Lemm	(1,1)	0.9465	0,9832	Linear SVC	Default parameters
SVM2	0.2	Lemm	(1,1)	0.9152	0.9803	SVC	Kernel = linear
SVM3	0.2	Stem	(1,2)	0.9732	0.9857	Linear SVC	Random state = 5 Max iteration = 500
SVM4	0.4	Stem	(1,1)	0.95	0.9519	Linear SVC	Random state =10 Fit intercept = false Max iteration = 1500
DT 1	0.2	Lemm	(1,1)	0.9485	0.9485	----------	Default parameters
DT 2	0.3	Lemm	(1,1)	0.949	0.9497	----------	Random state=0 Splitter = random
DT 3	0.2	Stem	(1,1)	0.9496	0.9496	----------	Default parameters
DT 4	0.2	Stem	(1,2)	0.9067	0.9795	----------	Criterion = entropy Splitter = random

Here are some output images that got from different models.

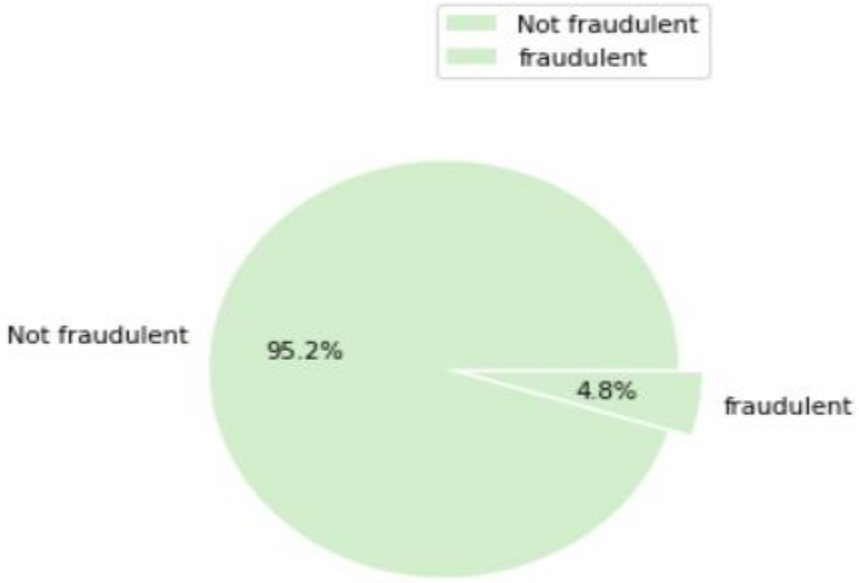


Figure 1 The percentage of real and false ads

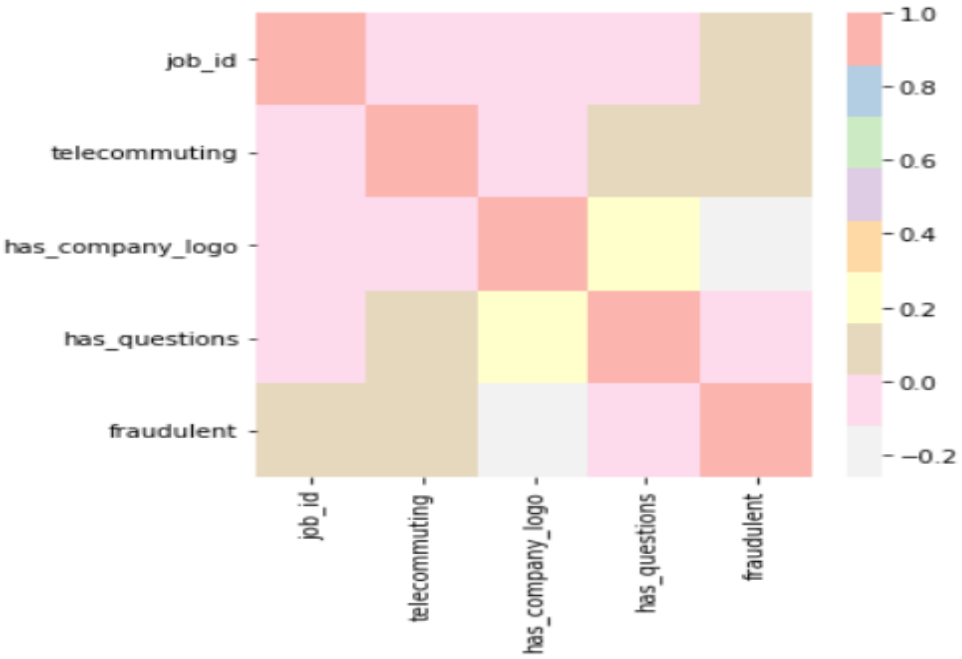


Figure 2 Correlation map

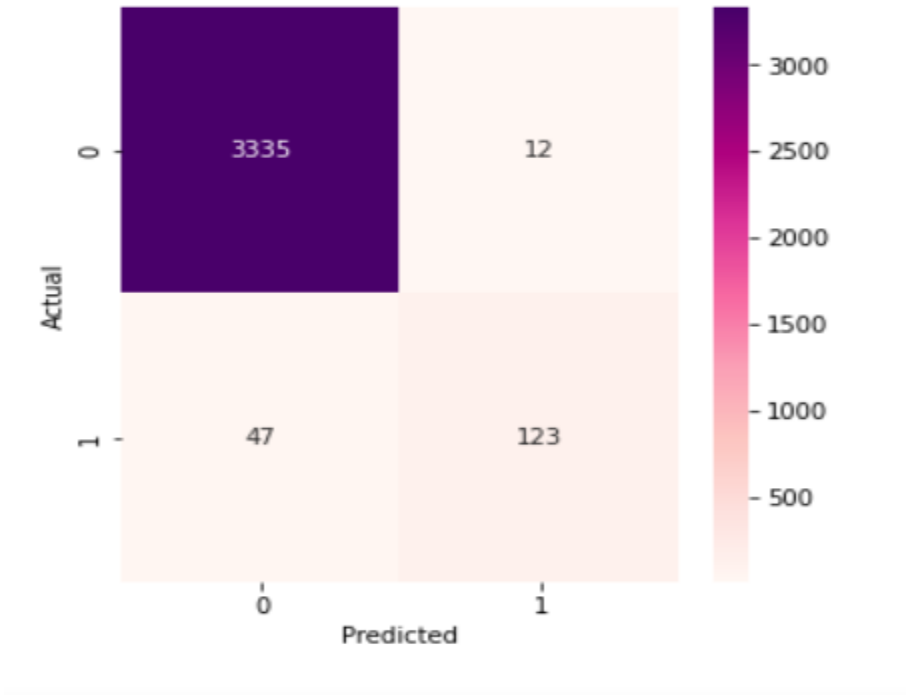


Figure 3 Support Vector Machine Confusion Matrix Tf-Idf Vectorizer

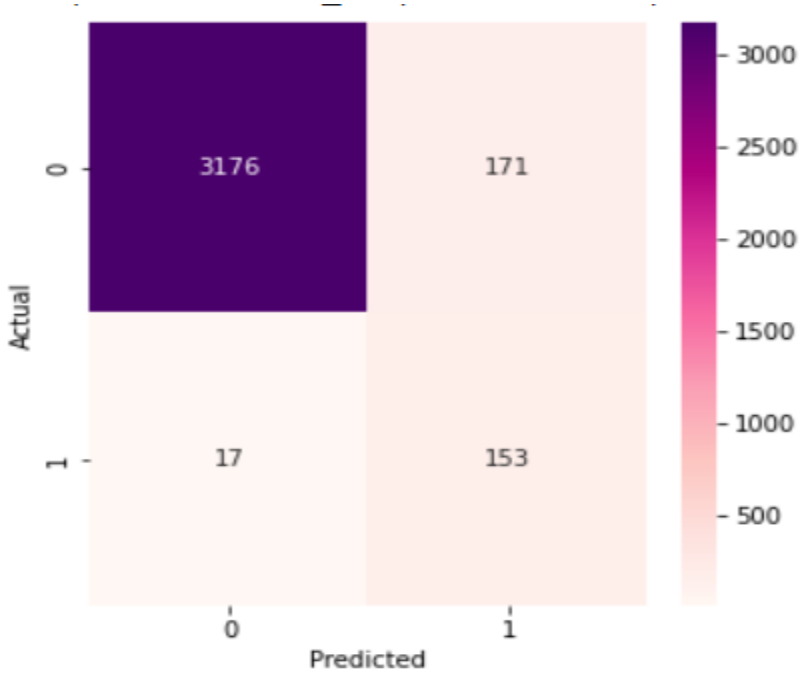


Figure 4 Support Vector Machine Confusion Matrix Count Vectorizer

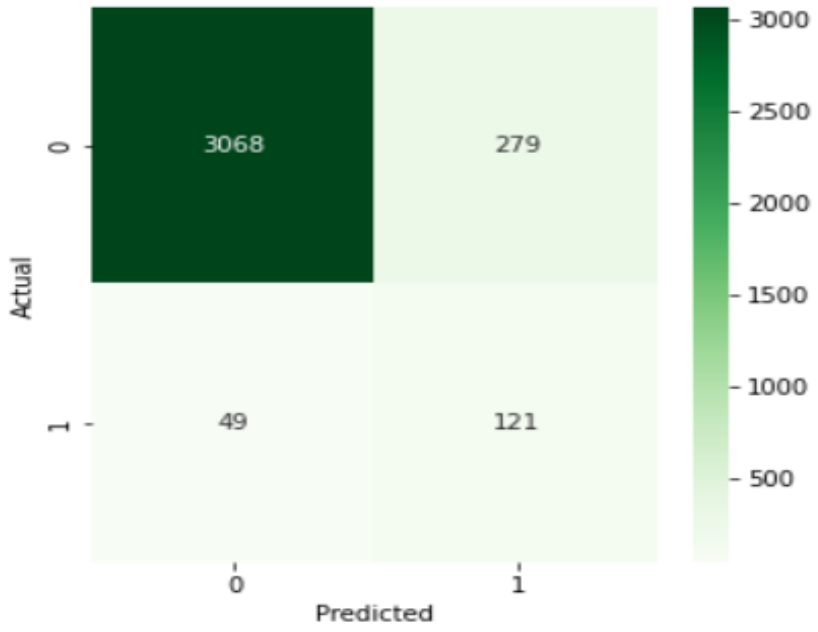


Figure 5 Decision Trees Confusion Matrix Count Vectorizer

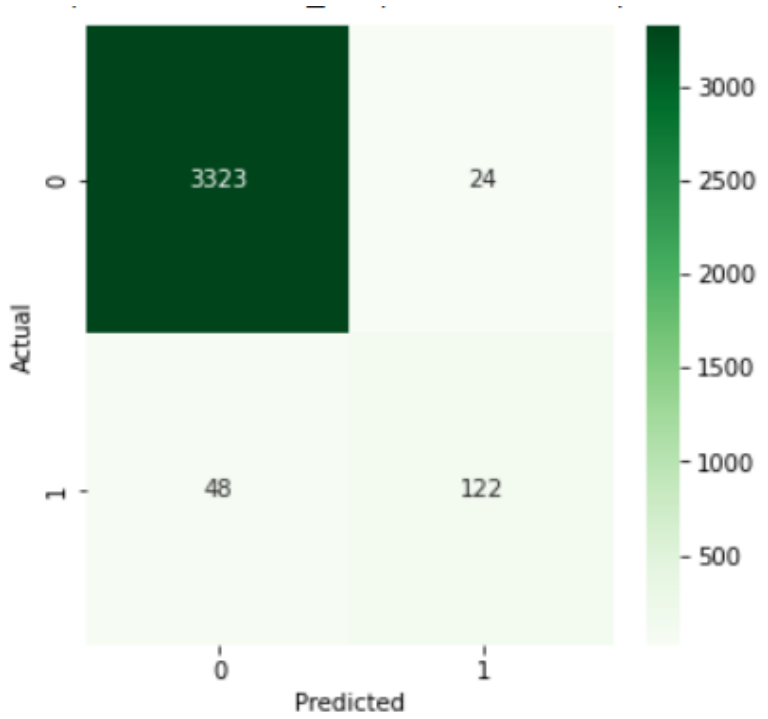
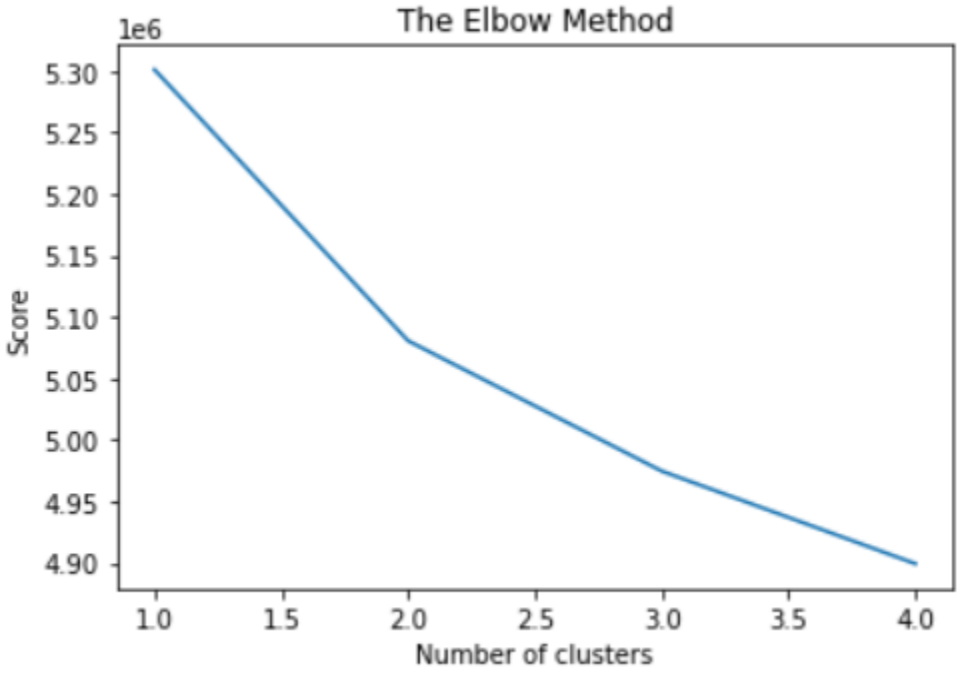
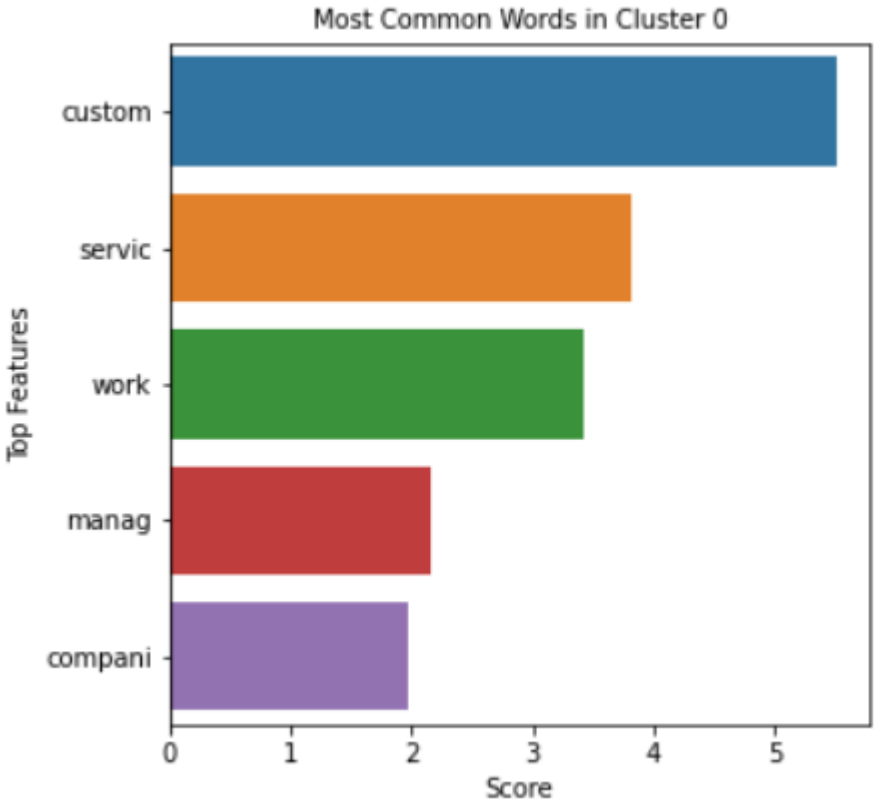
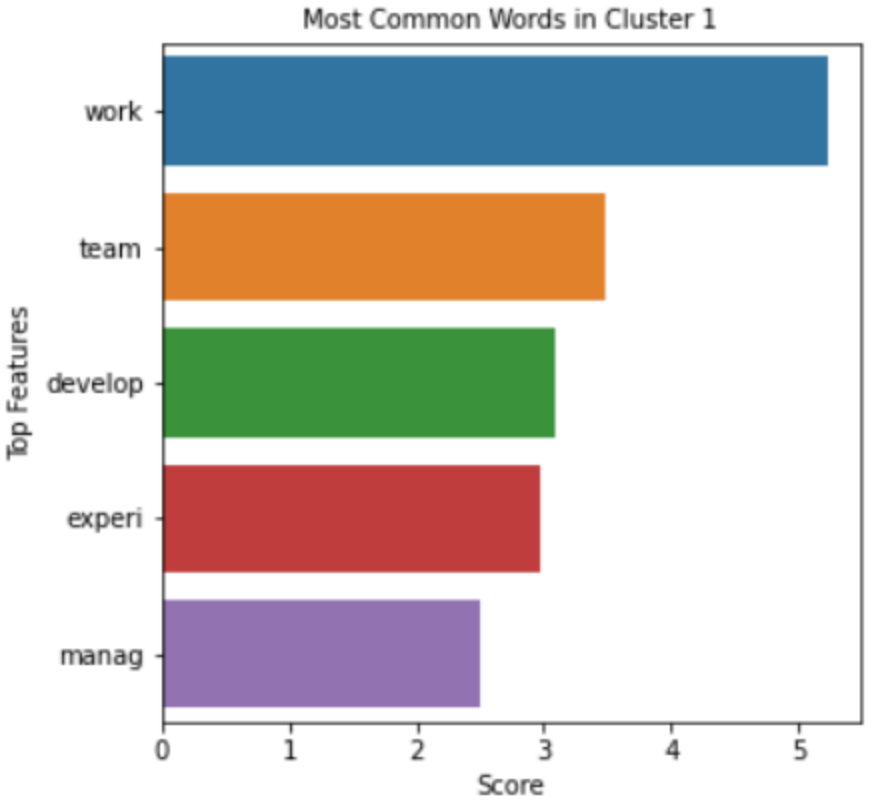


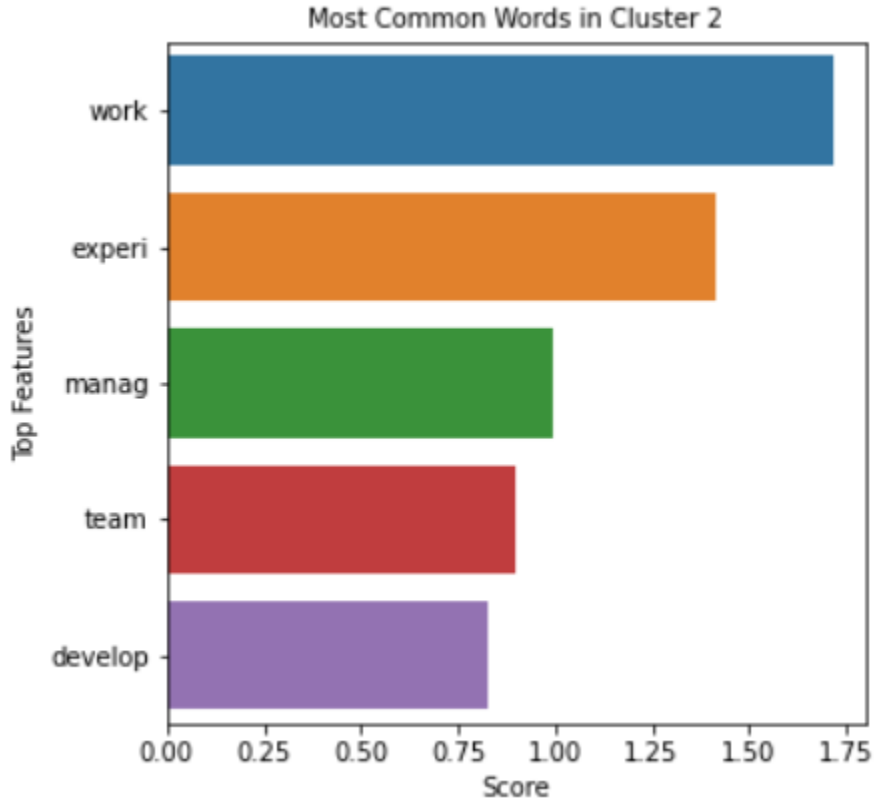
Figure 6 Decision Trees Confusion Matrix Tf-Idf Vectorizer

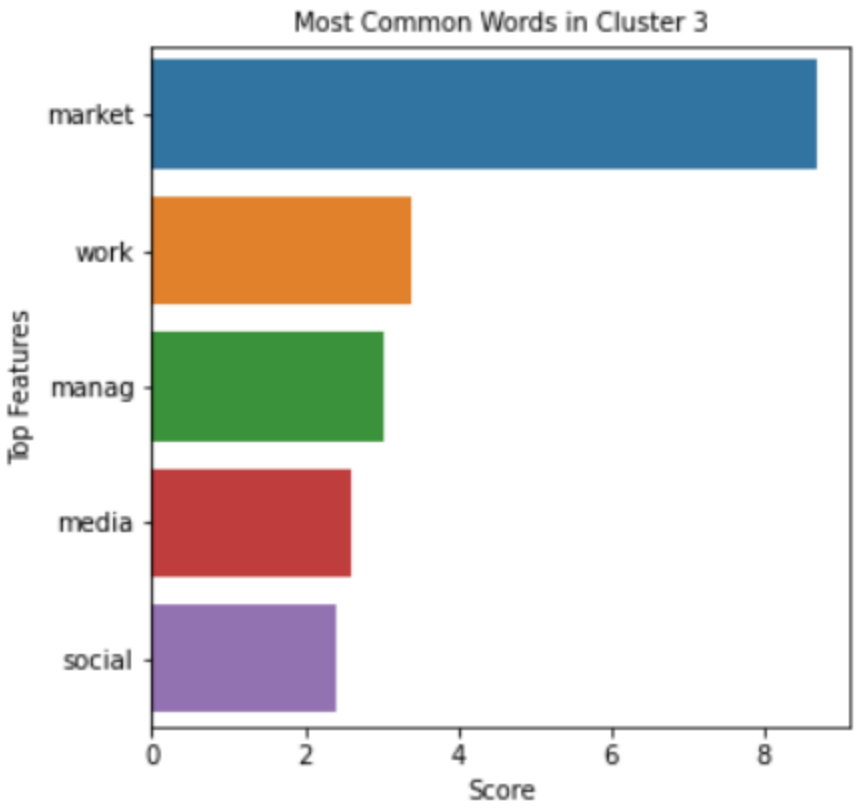
KMeans Clustering output

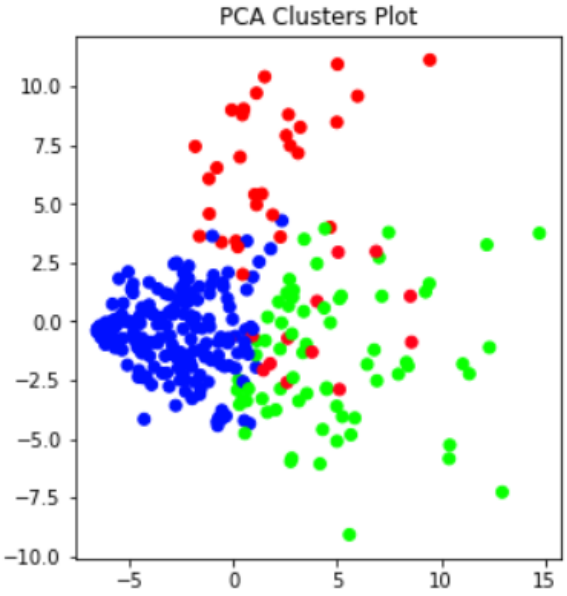


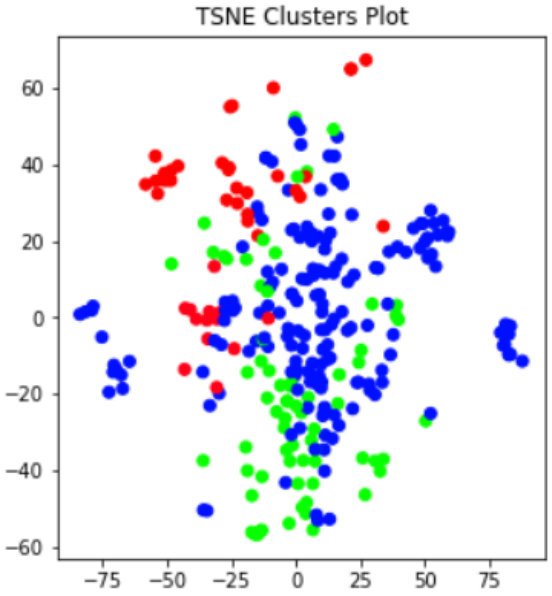


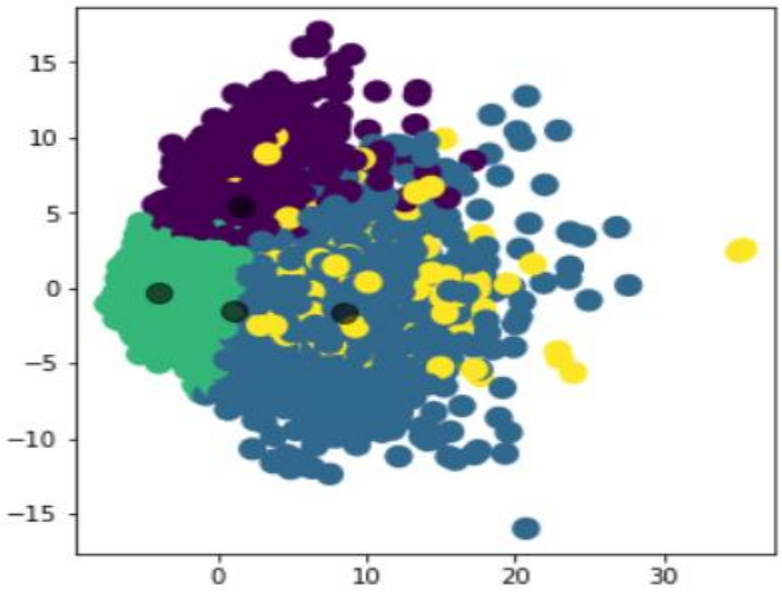

Most Common Words in Cluster 1
work
team
develop
experi
manag
Top Features
0
1
2
3
4
5
Score


Most Common Words in Cluster 2
work
experi
manag
team
develop
Top Features
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Score


Most Common Words in Cluster 3
market
work
manag
media
social
Top Features
0
2
4
6
8
Score


PCA Clusters Plot
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−5
0
5
10
15


TSNE Clusters Plot
60
40
20
0
−20
−40
−60
−75
−50
−25
0
25
50
75



OPTICS Clustering output