Machine Learning for Engineers: Final Project

By: Joseph Edralin, Kyle Lee, Ethan Zang

I. Abstract

Machine learning in its broadest scope builds around the idea of building or using existing algorithms to learn from data. Through machine learning applications, problems can be solved in relation to classification, regression, clustering, etc. Furthermore, the nature of the data can range anywhere between text and images and even to time series data sets. The purpose of this paper is to solve problems relating to machine learning where the task is to solve a problem, particularly clustering and classification. A total of three total data sets (imaging, time-series, and text) were analyzed with the nature of each data set being different from the next. To solve each problem, three different machine learning methods/algorithms were done on the data set and a comparative analysis of the three methods were done to evaluate the best method. Concluding the evaluation of the three methods for each data set, a final recommendation on the method of choice was suggested for the purpose of production.

II. Introduction

The dog breeds data set used was a subset of the Stanford Dogs[1] data set. With 120 total breeds and 20,580 total images, we took 32 of the breeds with 5834 total images. Each of these images contains exactly one dog, with no given standard regarding the size of the dog, the presence of other objects/backgrounds, or the size of the image. Given these images of dogs, the goal was to do a multiclass classification on the data set such that each dog was labeled as its correct breed.

Following a form of image classification, the second data set revolved around the multiclass classification of time-series data. The data set used was taken from a study done by a university in Italy[2] on human activity recognition of data where subjects were engaged in six different physical activities (walking, walking upstairs, sitting, standing, laying). Using this data set, the objective was to identify the physical activity performed by the individual. The embedded accelerometer and gyroscope of a Samsung Galaxy SII smartphone were used to log the recordings of time and frequency domain measurements. In total, the data set had 10,299 samples and 561 features.

The final data set analyzed was a form of text data where the goal was clustering. An overview of the data itself, health related news tweets were collected by a professor from the University of Southern California[3] in 2015 from 15 major different news outlets (BBC, WSJ, CNN, NBC, etc). The data was set up such that each news outlet had its own text file and with a list of samples from the outlet's Twitter account. The sample was formated such that there was an associated tweet ID, date and time, the tweet itself, and the URL.

Through a series of pre-processing, feature extraction, and application of machine learning methods, problems can be solved relating from these data sets.

III. Discussion

Overview

For each of the three data sets, the workflow was: 1) Pre-processing 2) Feature extraction using PCA 3) Implementation of three classifiers 4) Analysis of classifiers' performance. Pre-processing involved unique manipulation of data for standardization and reduction in complexity for simpler application of feature extraction and machine learning algorithms. Feature extraction involved principal component analysis for all three data sets. For classification problems, the machine learning methods used were Perceptron, K-Nearest Neighbors, and Support Vector Machine. For clustering, K-Means was used with euclidean, cosine, and cityblock distances. To evaluate the performance of each algorithm in classification problems, accuracy and f1 score were used to evaluate performance. For clustering, Davies Bouldin and Silhouette scores were used. Sci-kit learn library in Python was primarily used to implement feature extraction and machine learning, along with numpy, pandas, imageio, and a number of other libraries.

Figure 1: Overview of workflow for each data set.

Data Set 1: Dog Breeds

Given the data set of images of thirty-two different dog breeds with roughly 150 images per dog breed, the objective was to train a classification method to be able to predict the breed of a dog given the image.

Pre-processing was necessary to standardize each image before feature extraction so that classification methods could be applied. All images were first resized to be the minimum length in both x and y dimensions. Standardizing the number of pixels in each image allowed each image to have the same number of features. Standardizing specifically to the minimum dimensions among all images as opposed to mean dimensions or maximum dimensions ensured that no image was being enlarged, which would result in loss of image resolution. After finding the minimum height (*y_min*) and length (*x_min*) among all the images, one nuance was added before resizing. Each image was checked whether it was in landscape or portrait mode—essentially, whether the original dimensions were larger in x or y. Then, the smaller dimension of the original image was resized to the minimum of *x_min* and *y_min*, and the larger dimension of the original image as resized to the maximum of *x_min* and *y_min*. This portrait vs. landscape resizing ensured that in the process of resizing images to a standard size, the distortion

was minimized. Images were then turned grayscale by taking the RGB values and converting with the formula:

$$grayscale\ value = 0.2126 * red\ value + 0.7152 * green\ value + 0.0722 * blue\ value$$

Finally, the 2D matrix of grayscale values was flattened, such that each image had a number of features equal to *x_min* * *y_min.* Finally, pixel values were normalized to have zero mean and unit variance. A visualization of pre-processing before flattening of grayscale values can be seen in Figure 2.

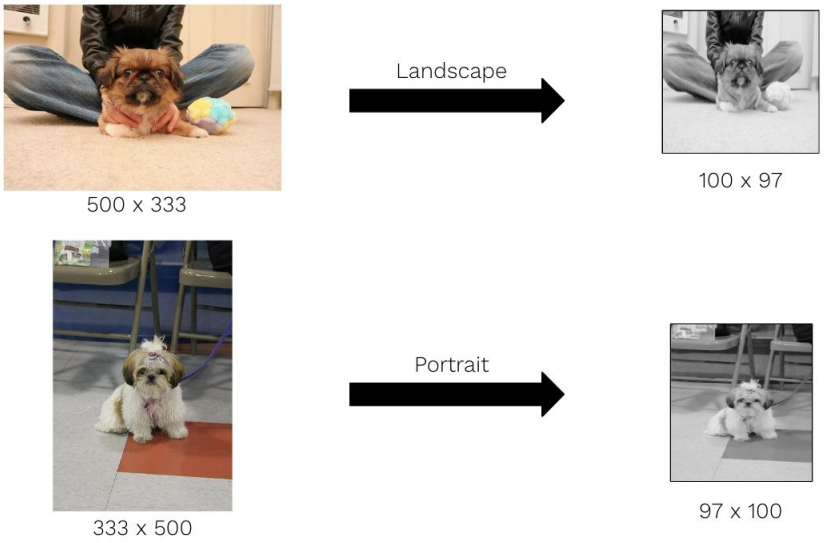

Figure 2: Visualization of pre-processing of images.

PCA was used for feature extraction/ reduction. A 90% variance threshold[9] was used such that number of components selected was the minimum number needed to explain 90% of the variance in the data set. For this data set, 9700 total original features were reduced down to between 751 and 760 depending on the fold of cross-validation.

The three machine learning algorithms used for this classification problem were Perceptron, K-Nearest Neighbors, and Support Vector Machine. Perceptron is a supervised, binary, linear classification algorithm that aims to minimize 0-1 loss. This was combined with a One vs. Rest multi-class learning algorithm for use with all thirty-two classes. K-Nearest Neighbors is a non-parametric method which evaluates the *k* nearest neighbors to the data point of interest and assigns the class according to the class of which most of the *k* nearest neighbors belong. The value of *k* selected was the square root of the number of samples, as this is a general standard/ rule-of-thumb[8]. Support Vector Machine is a linear classifier that moves away from 0-1 loss and uses hinge loss. Since Support Vector Machine does not have a closed form solution, our implementation used gradient descent as an optimization method for minimizing

loss. Stochastic Gradient Descent is an optimization method for minimizing loss. For this implementation, L2 norm loss was used with 100 maximum iterations. Stochastic Gradient Descent essentially used a cost gradient of just one example at each iteration, instead of needing to calculate the sum of the cost of each sample to find the cost gradient. This was once again combined with a One vs. Rest multi-class learning algorithm.

In order to evaluate the performance of each algorithm, a k-fold cross validation was used with five folds. For each fold, the f1 score and accuracy was calculated, as shown in Figure 3. Evidently, Support Vector Machine generated the highest accuracy score, while K-Nearest Neighbors generated the highest F1 score. Thus, these two were the best performing algorithms for the Dog Breeds data set.

	Perceptron		*K-Nearest Neighbors*		*Support Vector Machine*	
Fold Number	**Accuracy Score**	**F1 Score**	**Accuracy Score**	**F1 Score**	**Accuracy Score**	**F1 Score**
1	0.049096	0.047763	0.071490	0.087620	0.068906	0.070649
2	0.062877	0.069296	0.061154	0.082974	0.061154	0.062830
3	0.048276	0.050378	0.059483	0.077116	0.077586	0.077818
4	0.045690	0.049734	0.059483	0.080045	0.081897	0.084242
5	0.066379	0.070108	0.055172	0.070549	0.085345	0.089067
Mean	0.0544636	0.0574558	0.0613564	0.0796608	0.0749776	0.0769212

Table 1: Summary of accuracy and F1 scores for the classification of dog breeds.

```
[array([[2, 0, 0, ..., 0, 0, 0],
[1, 0, 0, ..., 0, 1, 0],
[2, 1, 4, ..., 1, 4, 0],
...,
[2, 1, 0, ..., 3, 0, 2],
[1, 1, 1, ..., 1, 1, 1],
[1, 3, 0, ..., 0, 1, 0]]), array([[5, 3, 5, ..., 3, 0, 0],
[2, 0, 0, ..., 0, 0, 1],
[0, 0, 2, ..., 0, 0, 0],
...,
[1, 0, 0, ..., 1, 0, 0],
[2, 0, 0, ..., 1, 0, 0],
[2, 2, 2, ..., 1, 0, 1]]), array([[2, 2, 3, ..., 0, 0, 0],
[1, 0, 0, ..., 1, 0, 1],
[2, 3, 0, ..., 1, 1, 1],
...,
[0, 0, 1, ..., 0, 1, 2],
[3, 1, 3, ..., 0, 2, 4],
[1, 0, 0, ..., 0, 0, 1]]), array([[4, 5, 0, ..., 1, 3, 3],
[1, 1, 1, ..., 2, 3, 0],
[2, 1, 4, ..., 0, 0, 2],
...,
[1, 1, 1, ..., 1, 0, 0],
[4, 2, 1, ..., 1, 2, 1],
[0, 1, 2, ..., 0, 1, 4]]), array([[3, 1, 0, ..., 0, 0, 0],
[0, 0, 0, ..., 0, 1, 0],
[0, 0, 1, ..., 0, 0, 1],
...,
[0, 0, 0, ..., 1, 2, 0],
[1, 2, 1, ..., 0, 2, 0],
[2, 2, 3, ..., 0, 1, 1]])]
```

Figure 3: Confusion Matrix for Perceptron

```
[array([[ 1,  0,  0, ...,  2,  0,  0],
[ 0,  0,  0, ...,  0,  0,  0],
[ 0,  0,  0, ...,  0,  1,  1],
...,
[ 0,  0,  0, ...,  1,  0,  0],
[ 0,  0,  0, ...,  0,  0,  0],
[22,  8, 14, ...,  7, 12, 22]]), array([[ 1,  2,  0, ...,  0,  0,  0],
[ 0,  1,  0, ...,  0,  0,  0],
[ 0,  0,  0, ...,  0,  0,  0],
...,
[ 0,  0,  0, ...,  1,  0,  0],
[ 0,  0,  1, ...,  0,  1,  0],
[ 3,  4,  9, ...,  7,  3, 10]]), array([[ 1,  0,  1, ...,  1,  0,  0],
[ 1,  0,  0, ...,  0,  0,  0],
[ 0,  0,  0, ...,  0,  0,  0],
...,
[ 0,  0,  0, ...,  0,  0,  0],
[ 0,  0,  0, ...,  0,  0,  0],
[19, 14, 25, ..., 16, 12, 16]]), array([[ 5,  0,  0, ...,  0,  1,  0],
[ 0,  0,  0, ...,  0,  0,  0],
[ 0,  0,  0, ...,  0,  0,  0],
...,
[ 0,  0,  0, ...,  0,  0,  0],
[ 0,  0,  0, ...,  0,  1,  0],
[17, 10, 25, ..., 16, 11, 16]]), array([[ 3,  0,  0, ...,  0,  0,  0],
[ 1,  1,  0, ...,  0,  0,  0],
[ 1,  0,  0, ...,  0,  0,  0],
...,
[ 0,  0,  0, ...,  0,  0,  0],
[ 0,  0,  0, ...,  0,  0,  0],
[23, 15, 17, ..., 12, 20, 18]])]
```

Figure 4: Confusion Matrix for K-NN

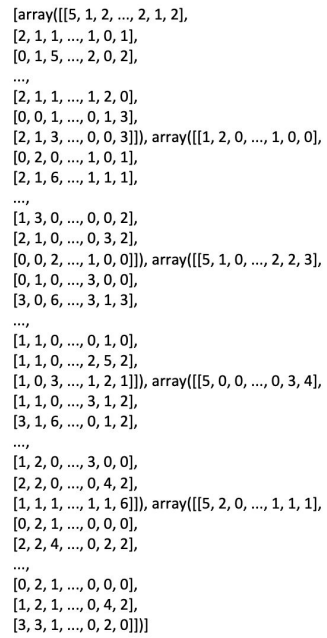

Figure 5: Confusion Matrix for Support Vector Machine

Data Set 2: Human Activity Recognition

For this data set, there were 10,299 human samples with 561 features where each feature represented a time or frequency domain measurement. The data for each sample was given as a single 561 feature vector containing these features (e.g. body acceleration (x, y, z), gravitational acceleration (x, y, z), etc), and each human sample was performing one of the six physical activities. All 561 entries of the vector represent the activity data for each sample. Given the measurement data and the corresponding physical activity (Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing, or Laying), the goal was to classify the physical activity based on those parameters.

Since the data was numerical and was already pre-processed to eliminate noise by the distributors, the only pre-processing that was required was standardization to zero mean and unit variance. Similar to the dog breeds data set, a 90% variance threshold was used for PCA feature extraction. The data set dimensionality was reduced from 561 to 65 for each of the five folds. The three machine learning algorithms used for this classification problem were Perceptron, K-Nearest Neighbors, and Support Vector Machine. All three were implemented in the same way with the same parameters as in the dog breeds implementation. Once again, k-fold cross validation was used with five total folds. The accuracy scores and f1 scores across each of these folds are shown in Figure 7. Evidently, Support Vector Machine led to highest mean accuracy and f1 scores.

	Perceptron		*K-Nearest Neighbors*		*Support Vector Machine*	
Fold Number	**Accuracy Score**	**F1 Score**	**Accuracy Score**	**F1 Score**	**Accuracy Score**	**F1 Score**
1	0.928155	0.928172	0.909709	0.909841	0.945631	0.945439
2	0.935922	0.936034	0.904369	0.904416	0.951456	0.951475
3	0.934466	0.934938	0.913592	0.913800	0.945631	0.945717
4	0.926699	0.926898	0.911650	0.912052	0.939806	0.939815
5	0.929092	0.928865	0.915007	0.914989	0.954832	0.939815
Mean	0.9308668	0.9309814	0.9108654	0.9110196	0.9474712	0.9444522

Table 2: Summary of accuracy and F1 scores for the classification of human activity.

```
[array([[1, 3, 2, ..., 0, 1, 3],
[4, 0, 0, ..., 0, 0, 0],
[2, 1, 6, ..., 2, 3, 4],
...,
[1, 1, 0, ..., 2, 1, 1],
[0, 0, 1, ..., 3, 1, 0],
[2, 0, 0, ..., 2, 2, 2]]), array([[2, 1, 1, ..., 1, 0, 1],
[1, 1, 2, ..., 0, 1, 1],
[0, 1, 2, ..., 0, 1, 3],
...,
[1, 1, 0, ..., 0, 1, 1],
[1, 0, 0, ..., 1, 1, 0],
[0, 1, 0, ..., 0, 1, 1]]), array([[2, 1, 1, ..., 0, 1, 2],
[3, 0, 1, ..., 2, 0, 1],
[0, 0, 3, ..., 1, 2, 3],
...,
[1, 0, 3, ..., 0, 0, 2],
[1, 3, 1, ..., 2, 1, 1],
[0, 1, 1, ..., 2, 1, 2]]), array([[4, 2, 3, ..., 2, 4, 5],
[1, 2, 0, ..., 0, 2, 2],
[1, 3, 0, ..., 1, 0, 0],
...,
[0, 2, 0, ..., 0, 2, 1],
[2, 2, 5, ..., 1, 7, 1],
[3, 2, 0, ..., 1, 0, 1]]), array([[2, 0, 0, ..., 0, 0, 0],
[1, 0, 2, ..., 0, 0, 0],
[0, 0, 1, ..., 1, 0, 2],
...,
[0, 0, 0, ..., 2, 0, 0],
[3, 5, 2, ..., 2, 3, 5],
[1, 1, 3, ..., 2, 1, 2]])]
```

Figure 6: Confusion Matrix for Perceptron

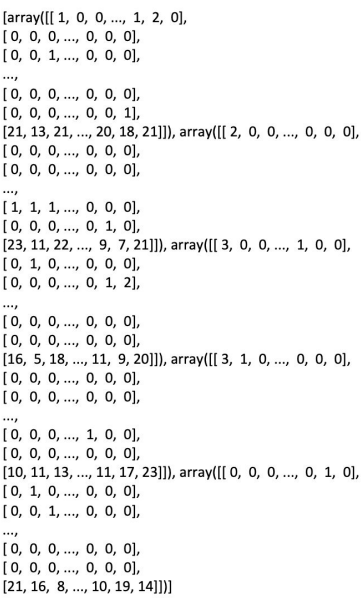

Figure 7: Confusion Matrix for K-NN

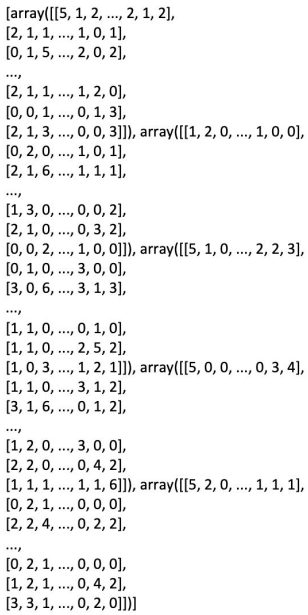

Figure 8: Confusion Matrix for Support Vector Machine

Data Set 3: Health Tweets

For the final data set, data pertaining to health tweets from 15 major health news agencies was used, and each .txt file in the data set contained all the health-related tweets for one specific

news agency. Each tweet was stored as a line within the .txt file, and was formatted as follows: [tweet ID]||[date/time]||[tweet], with '|' as a separator between each field.

With this data set, the goal was to cluster the tweets based on topic. Pre-processing methods for each tweet had to first be applied before clustering methods were used. First, although the input for each tweet was formatted as [tweet ID]||[date/time]||[tweet], only the [tweet] field was kept. Next, the url following each tweet was also removed, as only the text was desired. In order to standardize each tweet, all punctuation was removed and all text was made lowercase. Lastly, only tweets that contained the words from the set ['health','cancer','hospital','care','death'] were kept, as these words were among the most represented topics across all tweets. This final step also provided the desired number of clusters for analysis: 5.

Figure 9: Before and After Pre-Processing

After all the tweets were pre-processed, the TfidfVectorizer was used on each tweet to convert each tweet into a numerical vector. It took in the dataframe of pre-processed tweets as input and returned a 2D matrix of vectors as output. Each tweet gets converted into a vector of products of term frequency (TF) and inverse data frequency (IDF), hence the name TfidfVectorizer. Term frequency is the frequency of words within each tweet and is equal to $\frac{\textit{times a word appears in a tweet}}{\textit{total number of words in the tweet}}$. It is specific to each word within a tweet. For example, given the phrase 'i like apples', the TF for each word is ⅓, and given the phrase 'i like sweet pears', the TF for each word is ¼. Inverse data frequency is the uniqueness of a word across all tweets and is equal to $\log\left(\frac{\textit{total number of tweets}}{\textit{number of tweets the word appears in}}\right)$ [4]. It is specific to each word, but can be applied to all tweets. For example, given those same 2 phrases, the IDF for 'like' is 0 because 'like' appears in both tweets, and log(2/2) is 0. Meanwhile, the IDF for 'apples' is 0.33 because it only appears in 1 out of 2 tweets, and log(2/1) is 0.33. To create each vector, the TF is multiplied by IDF for each word. An example of how the vectors are created is shown in Figure X. After the tweets were converted to a matrix of numerical vectors, standard PCA was applied and the number of features was reduced to 2 to visualize the clusters on a 2 dimensional plot.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

$$idf(w) = log(\frac{N}{df_t})$$

Tweet 1: 'i like apples'
Tweet 2: 'i like sweet pears'

	i	like	apples	sweet	pears
1	0.33*log(2/2)	0.33*log(2/2)	0.33*log(2/1)	0*log(2/1)	0*log(2/1)
2	0.25*log(2/2)	0.25*log(2/2)	0*log(2/1)	0.25*log(2/1)	0.25*log(2/1)

	i	like	apples	sweet	pears
1	0	0	0.10034	0	0
2	0	0	0	0.07526	0.07526

Figure 10: TfidfVectorizer Example

Following PCA, the K-means clustering algorithm as discussed in class was used. For this data set, K-means was applied three different times, but different distance metrics were used for each method. The metrics used were euclidean distance, cosine distance, and cityblock distance. While euclidean distance is the standard distance metric, cosine distance essentially measures the cosine of the angle between two vectors and projects it in a multi-dimensional space and cityblock distance measures rectilinear distance, or the sum of the difference in the x and y coordinates between two points (Figure 11).

$$d(x,c) = (x-c)(x-c)'$$

$$d(x,c) = 1 - \frac{xc'}{\sqrt{(xx')(cc')}}$$

$$d(x,c) = \sum_{j=1}^{p} |x_j - c_j|$$

Figure 11: Distance metric formulas for Euclidean, Cosine, and City-Block

After the K-means algorithm was applied with k = 5, scatter plots for each distance method were created. Additionally, the Davies-Bouldin score and Silhouette score, which are clustering performance metrics, were calculated. The Davies-Bouldin score is the ratio of intra-cluster distances and inter-cluster distances, so clusters that are less dispersed among points in the same cluster and farther apart from other clusters is ideal[6]. The minimum score is zero, and lower scores are ideal. Similarly, the Silhouette score is a measure of how well a sample is matched to its own cluster versus other clusters, and the overall Silhouette score is the average score of all samples. The scores range from -1 to 1, with 1 being ideal and -1 being the worst. A score of zero implies overlapping clusters, and a negative score implies that a sample was assigned incorrectly[7]. The plots and corresponding performance scores are shown below.

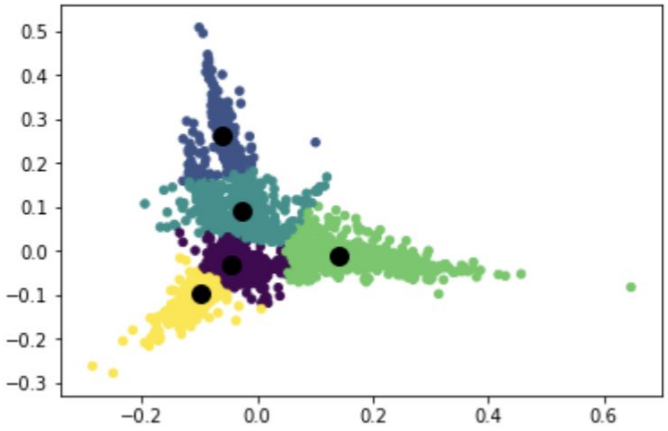

Figure 12: K-Means Clustering - Euclidean Distance

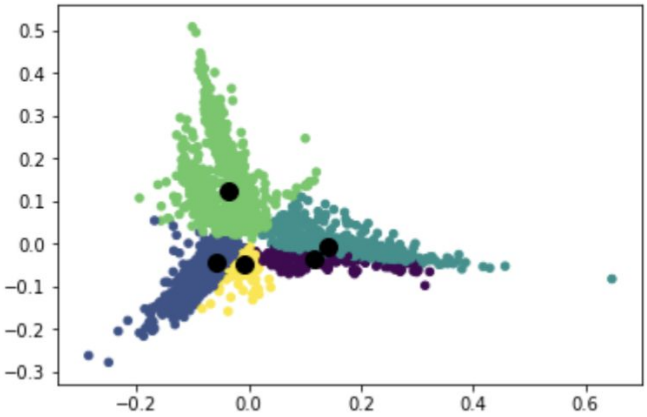

Figure 13: K-Means Clustering - Cosine Distance

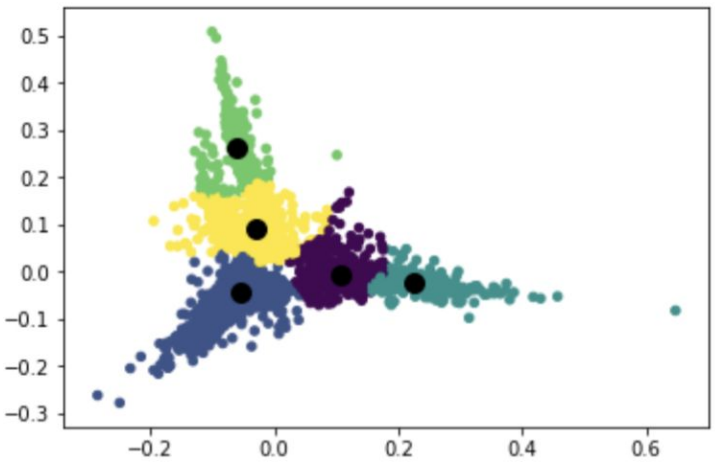

Figure 14: K-Means Clustering - Cityblock Distance

	Davies-Bouldin	**Silhouette Score**
Euclidean	0.0580758	0.576280
Cosine	1.676452	0.804192
Cityblock	0.582980	0.596227

Table 3: Performance Scores for K-Means Clustering by Metric

```
df_cosine = df
df_cosine['cluster'] = y_predicted
print(df_cosine[df_cosine['cluster'] == 0])
```

	data	cluster
0	hamilton police send mental health pros to the...	0
1	wind turbine noise linked to only 1 health iss...	0
2	check expiry dates health canada advises after...	0
3	despite paying top dollar some military mental...	0
6	quadriplegic temporary foreign worker denied h...	0
8	rt healthcanada health canada suspends two nat...	0
9	health canada willing to ease fecal transplant...	0

	data	cluster
53	lung cancer now top cancer killer for women in...	1
66	cancer screening not all its cracked up to be	1
220	even moderate drinkers face elevated breast ca...	1
229	angelina jolie effect on breast cancer screeni...	1
242	rt drkarlkabasele wearing a bra doesnt raise b...	1
250	angelina effect inspiring more women to get br...	1
283	breast cancer risk tied to mutated gene in fam...	1
291	sleeping in darkness improves effectiveness of...	1
294	can aspirin treat breast cancer why arent we t...	1

	data	cluster
4	fake oxycontin suspected in od death of moose ...	2
5	seriously ill senior stuck in us cant find a h...	2
11	how my mothers life and death influenced my di...	2
16	world health organization waited several weeks...	2
17	manitoba to implement all recommendations from...	2
26	hospital patients confusion might ease by seei...	2
28	mds urged to write the rules on assisted death	2
31	hospital suicide so many obstacles to prevention	2
32	first ebolainfected us nurse plans to sue says...	2

	data	cluster
18	toronto hospital investigating possible ebola ...	3
22	ebola american who contracted virus arrives at...	3
79	bc nurse released from hospital after testing ...	3
80	ebola death toll rises to 7588 globally who says	3
97	ebola spread in sierra leone fuels rise in dea...	3
113	ebola bodies dumped at sierra leone hospital b...	3
130	dallas ebola victims family settles with hospital	3
133	new york doctor who contracted ebola to get ou...	3
147	ebola outbreak death toll falls who says	3

	data	cluster
7	rob ford to have cancer surgery may 11	4
13	rob ford set to have cancer surgery in early may	4
25	super seniors research to check if cancer prot...	4
35	adhd diagnosis can double risk of premature de...	4
40	why this researcher wants to end the cancer th...	4
68	cancer survivor dionne warner thrilled by husb...	4
70	breast cancer survivors tattoo artists connect...	4

Figure 15: Tweets Assigned to Five Clusters For Cosine Distance Metric

As seen in Figure 15, the cluster assignments to each tweet are observed for the cosine distance metric, since that was the most successful clustering algorithm. Cluster 0 contained tweets relevant to ‘health’, Cluster 2 contained tweets relevant to ‘death’, and Cluster 3 contained tweets relevant to ‘hospital/ebola’. Interestingly enough, Clusters 1 and 4 both

contained tweets pertaining to ‘cancer’, but Cluster 1’s tweets reported cancer in a negative light, while Cluster 4 reported more positive cancer news. In comparison to the original keywords of [‘health’,’cancer’,’hospital’,’care’,’death’], only ‘care’ was not determined to be a significant clustering word. This cluster was replaced by an additional ‘cancer’ cluster with a more positive context.

IV. Conclusion/Analysis

Overall, across all three data sets, similar methods were used to obtain the desired outputs. All three data sets involved pre-processing, feature extraction using PCA, implementation of three classifiers, and lastly performance analysis. The primary metrics used for classification performance were accuracy and f1-score. In general, accuracy is preferred when correct classifications are emphasized and f1-score is preferred when incorrect classifications are emphasized and classes are imbalanced. Because the classes we analyzed for both data sets were roughly the same size and correct decisions were emphasized, accuracy was the more valid performance metric.

For the Stanford dogs data set, the average accuracy score across the three different classifiers was 0.063599 and the average f1 score was 0.071346, and out of all three classifiers, Support Vector Machine had the highest success while Perceptron had the lowest. There were 32 classes of dogs in our data set, so a random classification would have a 3.125% chance of being correct. However, the average accuracy from the machine learning algorithm was not significantly better, and this can be attributed to pre-processing limitations. Because the inputted images were different sizes, they had to be scaled down to the same image size, so aspect ratios were distorted for some images, which affected the integrity of the original images. For improved results, the algorithm could have identified each dog's face/body and outputted cropped images around that area. This modification could be made in future versions of the algorithm.

The human activity data set achieved much greater success, with an average accuracy of 0.929734 and average f1 score of 0.929823 across all three classifiers. Support Vector Machine had the greatest success and K-nearest Neighbors had the lowest. There were six different classes for activities, so overall this classification was very successful.

Lastly, for the health tweets data set, the cosine distance metric performed much worse than the other two metrics with respect to the Davies-Bouldin score. However, it is important to keep in mind that there was no option to select a desired distance metric when calculating the Davies-Bouldin score, so those scores were calculated assuming Euclidean distances even if the clusters were generated using different metrics. Thus, the Davies-Bouldin scores should not be scrutinized too heavily. In contrast, it performed the best with respect to the Silhouette score metric, since it was closest to 1. This can be attributed to the fact that cosine distance focuses on document similarity, or for our purposes, tweet similarity. Because cosine distance represents the angle between two vectors rather than distance, tweet/document length is devalued and similarity is prioritized. This matches the original intention of clustering tweets by topic, irrespective of tweet length.

All in all, this project was successful, as machine learning algorithms pertaining to clustering and classification were successfully implemented to obtain desired outputs.

V. Bibliography

[1] *Stanford Dogs Dataset for Fine-Grained Visual Categorization*, http://vision.stanford.edu/aditya86/ImageNetDogs/.

[2] *UCI Machine Learning Repository: Human Activity Recognition Using Smartphones Data Set*, https://archive.ics.uci.edu/ml/datasets/Human Activity Recognition Using Smartphones.

[3] *UCI Machine Learning Repository: Health News in Twitter Data Set*, https://archive.ics.uci.edu/ml/datasets/Health News in Twitter.

[4] Maklin, Cory. "TF IDF: TFIDF Python Example." *Medium*, Towards Data Science, 21 July 2019, https://towardsdatascience.com/natural-language-processing-feature-engineering-using -tf-idf-e8b9d00e7e76.

[5] Foley, Daniel. "K-Means Clustering." *Medium*, Towards Data Science, 19 Feb. 2019, https://towardsdatascience.com/k-means-clustering-8e1e64c1561c.

[6] "Sklearn.metrics.davies_bouldin_score." *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html.

[7] "Sklearn.metrics.silhouette_score." *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html.

[8] Subramanian, Dhilip. "A Simple Introduction to K-Nearest Neighbors Algorithm." *Medium*, Towards Data Science, 29 July 2019, https://towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e.

[9] Boyle, Tara. "Feature Selection and Dimensionality Reduction." *Medium*, Towards Data Science, 25 Mar. 2019, https://towardsdatascience.com/feature-selection-and-dimensionality-reduction-f488d1a035de.