UNIVERSITY OF REGINA
FACULTY OF GRADUATE STUDIES AND RESEARCH

DEPARTMENT OF COMPUTER SCIENCE

CS 831 PROJECT REPORT

Stylometric Analysis to Recognize Writing Pattern in a Document

SUSHITHA RAJEEVA (200425621)
TEJASWINI BALGURI(200439882)

Contents

1 Introduction

The use of stylometry aims to extract useful attributes from the writing style of documents. In some cases, it is possible to uncover author names, gender, native language using stylometric techniques. Marketers, analysts, social scientists rely on stylometric analysis because it provides the retrieval of demographic information from the raw text.

The use of related words such as articles, prepositions, and conjunctions by the writers will be different from one another. The diversity of research on cultural questions is studied using stylometric tools by many scholars. For instance, a significant study has been done to find the difference between how men's and women's writing styles. There is research on different writing styles in a single document written by an author indicates plagiarism has been done.

In this report, we start by explaining the problem statement by giving an example. It is followed by the background of stylometric analysis and related work. Later, we start to explain the different methods used for stylometric analysis feature selection and data pre-processing. In the end, the results and limitations of our work have been depicted and the conclusion of the overall work.

2 Problem Statement

Similar to how biometrics like fingerprint and iris are unique from individual to individual, writing style can also be considered as a unique feature. Everyone has their own writing style. The study of literary writing of a document is called Stylometry. This field started as an attempt to analyze the text for proof of authenticity and author identity, but with the development of computers and machine learning algorithms, stylometry is widely used to identify authorship, author gender detection, plagiarism detection. The aim of this project is to develop an intelligent system that takes an input document and classifies the different writing styles found in the document. As previously mentioned, each author has their writing style. Using unsupervised machine learning algorithms like K-Means or DBSCAN, we believe that the different writing styles found in the document could be classified.

Examples: Given an unknown text, it is occasionally feasible to predict who wrote it by calculating certain features, like the average number of words per sentence, average sentence length by word, or the tendency of the author to use "while" instead of "whilst", and comparing the calculations with other texts written by the suspicious author

3 Implementation

3.1 Data Selection

We have selected datasets from the online repository (http://textfiles.com/stories/). The dataset is basically the collection of two different stories written by two different authors. Clustering is carried out on the selected data which consists of different writing styles. Basically, our model could be used for any document and the main goal is to illustrate it.

3.2 Feature Selection

The major part of our system relies on feature extraction. We have to carefully examine the features to retrieve the sensitive information from that text to differentiate the authors. We have selected three major features: Lexical Features, Vocabulary Richness Features, and Readability Scores.

3.2.1 Lexical Features

- Average Word Length
- Average Sentence Length By Word
- Special Character Count
- Punctuation Count

These are the basic feature that we have selected to abstract from the text. Within the context of a sentence, functional words express grammatical relationships among other words, and a word with more consonant is most likely to be a complex word. Punctuation counts are used to differentiating the different genres. Special character count is used to counting the special characters used in the context.

3.2.2 Vocabulary Richness

Vocabulary richness is one of the important features to be considered while extracting the data from the text and much quantitative research depends on methods of vocabulary richness. A vocabulary richness is measured based on the occurrence of a word in the document. If the word occurs repeatedly, it has less vocabulary richness. Vocabulary richness is high if we have new words used frequently in the document.

We have used three measures that tell us the diversity and vocabulary richness in the text.

- Yules characteristic K: It tells us the frequency dependencies of words.

$$K = 10,000 \times (M - N)/(N \times N)$$

$$where\ M = \sum_{i}^{n} i^2.V_i$$

- Shannon Entropy: This formula is used in our work to measure the amount of information a word is offering.

$$E = \sum_{i=0}^{N} P_i.log_{Pi}$$

  P- The probability of a word occurring in the paragraph

- Simpson's Index: This is used to measure the diversity in the paragraph. NLP can use this method to find the diversity in the piece of text. Simpson's index(D) is used to measure the probability that two authors select the document unknowingly that will belong to the same species. We have used this measure in our work to figure out the diversity.

$$SIndex(D) = \sum(n/N^2)$$

  N- Total number of words in a text

  n- Total number of distinctive word

3.2.3 Readability Score

Readability is the measure of human understandable towards reading a text without difficulty. It also shows how a person can differentiate the individual letters or characters from each other. The root of the readability is linguistics. One of the methods used to measure readability is Gunning Fog Index.

$$G = 0.4 \times [(\frac{Words}{Sentences}) + 100 \times (\frac{Complexwords}{words})]$$

Complex words- Has three or more syllables.

Gunning Fog Index: It is the readability test specifically for English writing. The index evaluates the person's ability to understand the text respective to the years of formal education. The formula to measure gunning fox index is given below. For instance, A high school senior around 18 years old needs a 12 fog index reading level.

3.3 Data Pre-Processing

First, we need to download the dataset from textfiles.com. The dataset consists of data files of different authors and different genres. We have selected the children's bedtime stories and we have also provided the proof of concept using research paper. The processing starts by diving the whole context into chunks of small data. We faced some difficulties while determining the size of the chunks. The essence of the passage would not be able to extract if the document was too large. Efficiency would have been lost if the document is too small. After a small research, we decided to use an average of 10 per sentence. We first applied lexical features were computed on those sentences, later removal of punctuation and special characters.

3.4 K-means Algorithm

We are using unlabelled data which needs to be clustered to determine which author has written the possibly the text. Unsupervised learning works well for the data without defined categories. We used a few algorithms for our work but the K-means algorithm turned out to be the best fit for our requirement among all.

3.5 PCA and Data visualization

We have computed 10 features to work on text files. After proceeding with our work by using the K-means algorithm, we used this algorithm to run on all the vectors of all blocks of data and to find the centroids. At this point, the number of centroids correlates with the number of different writing styles found in the text files. We converted a 10-dimensional vector to 2D using Principal Component Analysis to visually see those clusters. Conversion from 10D to 2D vector requires the extraction of essence using PCA. In order to visually see these vectors and the color of the same chunks which were clustered together under a centroid using the K-means algorithm. By using this method, we can see visualize the different writing styles to make our result robust.

4 Experimental Approach

We have started our experiment by choosing two different documents and combined them as one single document. As this merged document has different writing patterns, our algorithm should distinguish this difference. For this purpose, we have employed the K-means algorithm to compute the k value which implies different centroids where each centroid corresponds to a different writing pattern.

The next step was to find an optimal value of k. One way for computing k is to determine it by manual inspection. This can be accomplished by analyzing the various data point in the feature vector. This can be less accurate or the resulting k value might be less efficient since the data is very large. So, we have decided to employ the elbow method to compute the optimal k value.

4.1 Elbow Method

An elbow method is one of the widely used methods for determining the optimal value of k. K-means can be described as an unsupervised learning algorithm that crates k number of clusters by grouping the data. Now, the elbow method can be used to run this k-means clustering on a dataset that has a different range of k values. Then, an average score is computed for each value of k. This can be described as - calculate the sum of squared error (SSE) for k values and choose k such that the SSE starts diminishing.

SSE can be described as the sum of the squared distance between the centroid and each cluster.

$$E = \sum_{i=1}^{k} \sum_{P \in C_i} (p - m_i)^2$$

where k is total clusters, C is the set of objects in each cluster, m can be described as the center point of a cluster.

After plotting the values, it is possible to determine the best value of k. If the line chart represents an arm, then, the diminishing point also called elbow, is the best value of k. If the inflection is strong, then it indicates that it fits best at that value of k. In the above graph, we can see that as k increases, the error decreases. The reason for this is the distortion gets smaller as clusters increases. In this scenario, SSE decreases abruptly at k = 2. So, we can conclude that the optimal value of k is 2.

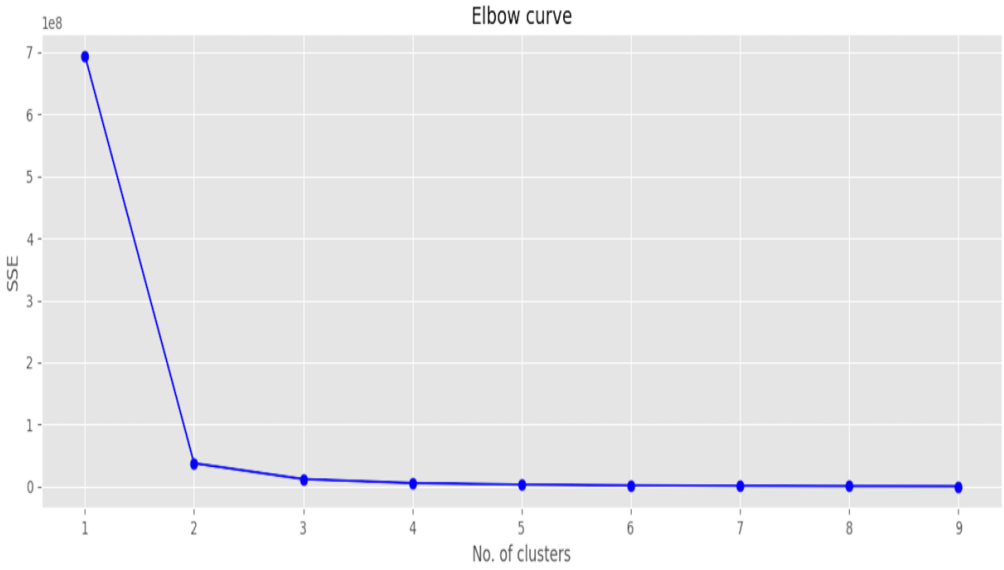


Figure 1: Elbow Curve

5 Test Results

Our experiment was performed by merging two different documents with two writing styles. The merged document has almost 600 lines and 60 paragraphs. As a result, we were able to distinguish two different works of two authors. Also, in order to clearly identify the writing styles, we colored similar chunks with the same color.

Our aim was to achieve a k value of 2, which was successfully returned by the elbow method.

The below figure clearly displays the results of a single document with two different writing approaches of two authors respectively.

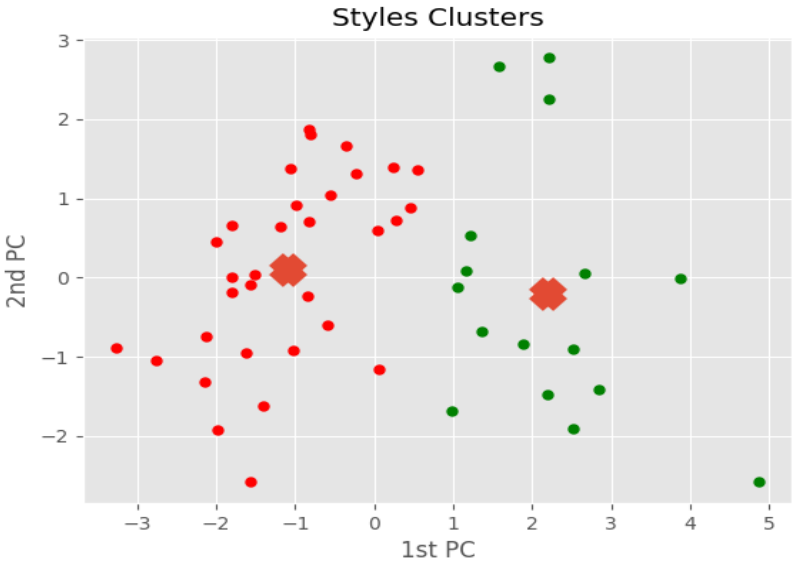


Figure 2: PCA of Style Clusters

6 Limitations

We opted Principal Component Analysis method for converting the 9D feature vector into a 2D vector. PCA is one of the oldest methods used to reduce the dimensionality of the larger dataset and represent it visually. During this conversion process, there is a high possibility of loss of information. But this is not a major challenge for our approach since we have computed the optimal value of k using the elbow method to determine the different works of authors in the document. This proves the accuracy of our experiment.

Let us consider a case where the document written by only one writer is fed as input to our system. The expected result, in this case, will be only one cluster as it is written by a single author. But our approach will create different clusters even when a single author's document is considered. The reason being each and every paragraph might have some differences like lexical features and some paragraphs might have formulas or some paragraphs might be poetic. Since the main goal of our experiment is to produce clusters based on writing styles but not the number of authors, it might result in creating multiple clusters for a single author too. However, the distance between the clusters might be small for the text document with a similar writing style. Hence, our approach not only helps in detecting the writing patterns for different authors but also succeeds in determining the difference between those patterns by computing the distance between the clusters.

7 Possible Extension

In our approach, we have computed only 9 features that include 5 lexical features, 3 vocabulary richness features, and only one readability score. This algorithm can be extended in the future to compute more of these features for example computing the average sentence length by character, number of functional words, Hapax Legomena, Flesch Reading Ease. Including all these features in our algorithm will improve the accuracy of our results by extracting additional features from the input text document.

8 Conclusion

The main aim of our experiment is to determine the different writing styles in a single document. To achieve this, we chose the average size of 10 for each chunk and then divided the documents into chunks. As a next step, we computed the stylometric features for each of these blocks. We used k-means algorithm to cluster the data and then determined the optimal value of k using the elbow method. PCA method was used to convert the 9D feature to a 2D vector and visually represent the result. Finally, all those blocks of data with the same writing pattern are colored the same and under the same centroid which implies the different writing styles in the document. Since this experiment determines the different writing styles in a document, this can also be employed to expose plagiarism, verify the authorship and also evaluate scientific writing.

9 References

[1] Stylometry. (2021, October 13). In Wikipedia. https://en.wikipedia.org/wiki/Stylometry.

[2] Laramée, F. D. (2018a). Introduction to stylometry with Python. The Programming Historian, 7. https://doi.org/10.46430/phen0078

[3] Bergsma, Shane and Post, Matt and Yarowsky, David. (2012). Stylometric analysis of scientific articles. 327-337.

[4] Honore, A. (1979), Some Simple Measures of Richness of Vocabulary. In: Association for Literary and Linguistic Computing Bulletin 7(2), 172177.

[5] J. Kincaid, R.P. Fishburne, R.L. Rogers, and B.S. Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel.

[6] Graeme Hirst and David St-Onge. 1997. Lexical chains as representation of context for the detection and correction of malapropisms. In C. Fellbaum, editor, WordNet:An electronic lexical database and some of its applications. The MIT Press, Cambrige, MA.

[7] Rodríguez Timaná L.C., Saavedra Lozano D.F., Castillo García J.F. (2020) Software to Determine the Readability of Written Documents by Implementing a Variation of the Gunning Fog Index Using the Google Linguistic Corpus. In: Botto-Tobar M., Zambrano Vizuete M., Torres-Carrión P., Montes León S., Pizarro Vásquez G., Durakovic B. (eds) Applied Technologies. ICAT 2019. Communications in Computer and Information Science, vol 1193. Springer, Cham. https://doi.org/10.1007/978-3-030-42517-3_31

[8] Yule, G. Udny. "On Sentence-Length as a Statistical Characteristic of Style in Prose: With Application to Two Cases of Disputed Authorship." Biometrika 30, no. 3/4 (1939): 363–90. https://doi.org/10.2307/2332655.

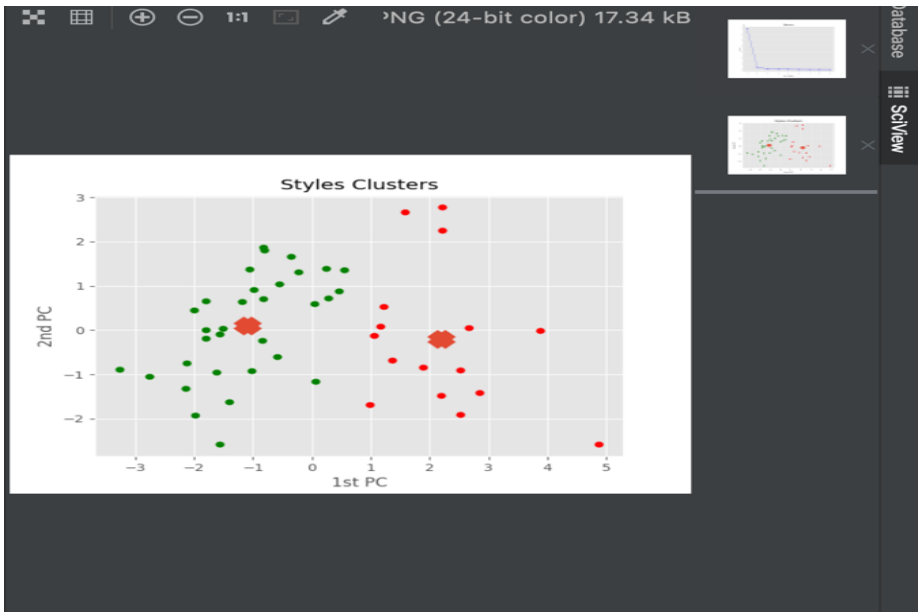
10 Appendices

10.1 User Manual

Below are the steps for running this program
Step 1: Unzip the file CS831_Project and import it in pycharm (any python IDE)
Step 2: Run the checker.py file

```
/Users/momo/Downloads/PlagarismChecker/venv/bin/python /Users/momo/Documents/Assignments/KDD/CS831_Project/checker.py
[nltk_data] Downloading package cmudict to /Users/momo/nltk_data...
[nltk_data]   Package cmudict is already up-to-date!
[nltk_data] Downloading package stopwords to /Users/momo/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Labels :  [1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 0 1 1 0
 0 0 0 0 1 1 1 1 0 0 1]
```

Step 3: View the elbow graph and final PCA.



10.2 Implementation Manual

After understanding the uses of distinguishing the different writing styles in a single document in detecting plagiarism, evaluating scientific writing etc. The overview of the implementation is as follow:

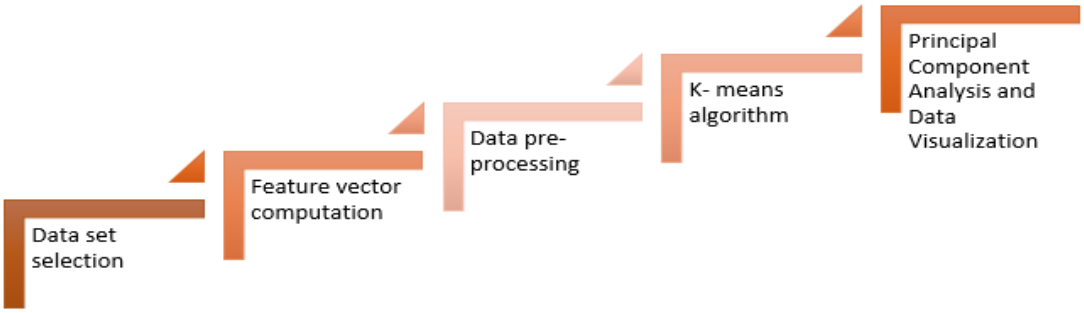


Figure 3: Implementation Overview

We came up with this approach for analyzing the writing styles. Initially, we selected the data from online and merged two documents. Now, we computed a feature vector.

This feature vector comprises of various lexical features, readability scores and vocabulary richness features. Next, data pre-processing is performed by cleaning the punctuation, special characters.

We have selected K-means to cluster the data and the elbow method to determine optimal value of k. Finally, used Principal Component Analysis method to convert the 9D feature vector to a 2D vector and visually represented the results.

10.3 Source Code

Below are the code snippets that demonstrates the main aspects of the algorithm Computation of the lexical features:

Sentence Length By Word Avg method

Returns avg number of words in a sentence

```
def sentence_length_by_word_avg(text):
    tokens = sent_tokenize(text)
    return np.average([len(token.split()) for token in tokens])
```

Word Length Avg method

Removes stop words plus punctuation

```
def word_length_avg(str):
    str.translate(string.punctuation)
    tokens = word_tokenize(str, language='english')
    st = [",", ".", "'", "!", '"', "#", "$", "%", "&", "(", ")", "*", "+", "-", ".", "/", ":", ";", "<",
          "@", "[", "\\", "]", "^", "_", '`', "{", "|", "}", '~', '\t', '\n']
    stop = stopwords.words('english') + st
    words = []
    i = 0
    while i < len(tokens):
        if tokens[i] not in stop:
            words.append(tokens[i])
        i += 1
    return np.average([len(word) for word in words])
```

Count Syllable method

counts number of syllables

```
def count_syllable(word):
    global cmuDictionary
    d = cmuDictionary
    try:
        syl = [len(list(y for y in x if y[-1].isdigit())) for x in d[word.lower()]][0]
    except:
        syl = manual_count_syllable(word)
    return syl
```

Special Character Count method

Counts special characters normalized over length of chunk

```
def special_character_count(text):
    st = ["#", "$", "%", "&", "(", ")", "*", "+", "-", "/", "<", "=", '>',
          "@", "[", "\\", "]", "^", "_", '`', "{", "|", "}", '~', '\t', '\n']
    count = 0
    i = 0
    while i < len(text):
        if text[i] in st:
            count = count + 1
        i += 1

    return count / len(text)
```

Punctuation count method

Counts punctuation

```
def punctuation_count(text):
    st = [",", ".", "'", "!", '"', ";", "?", ":", ";"]
    count = 0
    i = 0
    while i < len(text):
        if text[i] in st:
            count = count + 1
        i += 1
    return float(count) / float(len(text))
```

Computation of the vocabulary rich features:

Yules Characteristic

K = 10,000 * (M - N) / N2 where M Sigma i2 * Vi

```
def yules_characteristic(text):
    words = special_characters_removal(text)
    N = len(words)

    freqs = coll.Counter()
    freqs.update(words)

    vi = coll.Counter()
    vi.update(freqs.values())

    M = sum([(value * value) * vi[value] for key, value in freqs.items()])

    K = 10000 * (M - N) / math.pow(N, 2)
    return K
```

Shannon Entropy method

Shannon Entropy and Simpsons Index are basically diversity indices for any community

$$-1 * \sum \frac{p_i}{n_{p_i}}$$

```
def shannon_entropy(text):
    words = special_characters_removal(text)
    length = len(words)

    freqs = coll.Counter()
    freqs.update(words)

    arr = np.array(list(freqs.values()))

    distribution = 1. * arr
    distribution /= max(1, length)

    H = sc.stats.entropy(distribution, base=2)
    return H
```

Computation of the vocabulary rich features:

Simpsons Index method

$$1 - \sum \frac{n * (n - 1)}{N * (N - 1)}$$

N is total number of words

n is the number of each type of word

```
def simpsons_index(text):
    words = special_characters_removal(text)

    freqs = coll.Counter()
    freqs.update(words)

    N = len(words)

    n = sum([1.0 * i * (i - 1) for i in freqs.values()])

    D = 1 - (n / (N * (N - 1)))

    return D
```

Gunning Fox Index method

```
def gunning_fox_index(text, NoOfSentences):
    words = special_characters_removal(text)

    NoOFWords = float(len(words))
    complexWords = 0


    for word in words:
        if (count_syllable(word) > 2):
            complexWords += 1

    gun = 0.4 * ((NoOFWords / NoOfSentences) + 100 * (complexWords / NoOFWords))

    return gun
```

```
# Using unsupervised K-means to determine centroids
kmeans_algo = KMeans(n_clusters=K)
kmeans_algo.fit_transform(pc_components)
print("Labels : ", kmeans_algo.labels_)
centers = kmeans_algo.cluster_centers_
```

Elbow Routing method

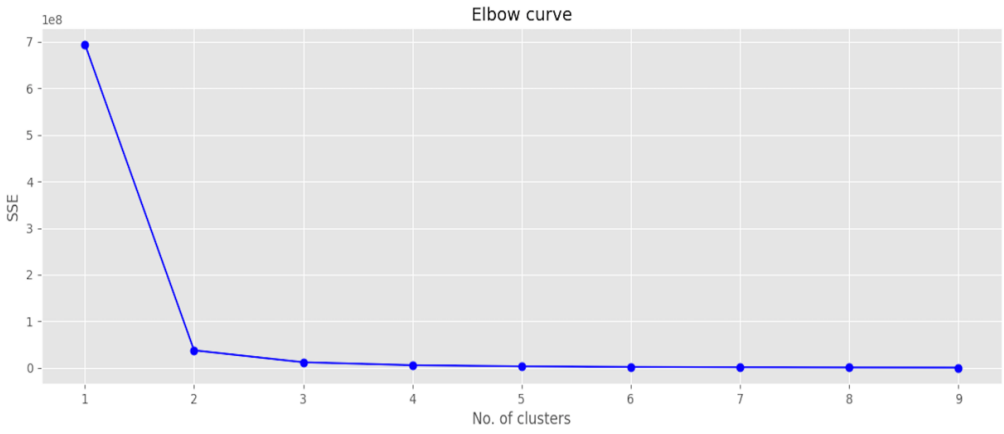
```
def ElbowRoutine(data):
    X = data
    distorsions = []

    for k in range(1, 10):
        kmeans_algo = KMeans(n_clusters = k)
        kmeans_algo.fit(X.reshape(-1,1))
        distorsions.append(kmeans_algo.inertia_)

    fig = plt.figure(figsize = (15, 5))
    plt.plot(range(1, 10), distorsions, 'bo-')
    plt.grid(True)
    plt.ylabel("Square Root Error")
    plt.xlabel("Number of Clusters")
    plt.title('Elbow curve')
    plt.savefig("ElbowCurve.png")
    plt.show()
```

10.4 Experimental Results

10.4.1 Elbow Method

An elbow method is one of the widely used methods for determining the optimal value of k. K-means can be described as an unsupervised learning algorithm that crates k number of clusters by grouping the data. Now, the elbow method can be used to run this k-means clustering on a dataset that has a different range of k values. This can be described as - calculate the sum of squared error (SSE) for k values and choose k such that the SSE starts diminishing.

The result of elbow method is as follows:



10.4.2 Principal Component Analysis

We opted Principal Component Analysis method for converting the 9D feature vector into a 2D vector. PCA is one of the oldest methods used to reduce the dimensionality of the larger dataset and represent it visually.

Result of PCA is as follows: