Dataset Health News in Twitter Data Set

Data Set dapat dikatakan berkualitas jika didalam Data Set tersebut tidak terdapat noise dan outliers, missing values, duplicate data, wrong data, dan fake data (Slide 2 Data dan Eksplorasi Data-IMD).

Praproses data

Isi dari Data Set Health News in Twitter berupa data tweet(tekstual) dari berbagai agensi berita. Karena isi dari data set ini berupa data tweet(tekstual) maka pasti di dalam data tersebut terdapat karakter '@' (mention: untuk menunjukan user yang melakukan tweet), # (hashtag: menunjukan kata kunci didalam platform tiwitter), dan lain-lain yang merupakan karakter yang tidak diperlukan untuk proses clustering. Maka dilakukan **Normalization** yang menjadi Langkah pertama dalam melakukan praproses data.

```
#Text Cleaning

# Removing Mentions
import re
list2=[]
for i in list1:
    list2.append(re.sub(r'@[A-Za-z0-9]+','', i))

# Removing Hashtags
list3=[]
for i in list2:
    list3.append(re.sub(r'#([^\s]+)', r'\1', i))

# Removing Hyperlinks
list4 = []
for i in list3:
    list4.append(re.sub(r'http\S+', '', i))
```

Selanjutnya, dilakukan **Expand Contractions** yaitu memperluas kombinasi kata-kata yang dipersingkat dengan menghilangkan huruf dan menggantinya dengan tanda penyingkat. Contoh "i've" menjadi "i have". Dan menyamaratakan karakter (semua menjadi huruf kecil).

```
# Expanding contractions
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
v = ' \n '.join(list4)
list5 = decontracted(v)
list6 = list5.split(" \n ")
```

```
# Removing punctuations and numbers
list7 = []
for i in list6:
    list7.append(re.sub(r"[^a-zA-Z]", ' ', i))

# Upper Case to Lower Case
list8 =[x.lower() for x in list7]
list8
```

Setelah melakukan expand contraction, Langkah selanjutnya **stopwords removal**. Stop removal adalah menghilangkan kata stopwords (list stopwords dapat dilihat di google). Kami melakukan stopwords removal ketika dilakukan feature extraction.

Lalu Langkah selanjutnya yaitu **tokenization** merupakan proses yang dilakukan untuk memisahkan kalimat menjadi kata-perkata.

```
# Tokenization
tknzr = TweetTokenizer()
alltokens = []
for i in list8:
    tokens = tknzr.tokenize(i)
    alltokens.append(tokens)
alltokens
```

Terakhir, **stemming/lemmatization** digunakan untuk mengubah kata yang sudah diberikan imbuhan ke kata aslinya.

```
#Lemmatization
wordnet_lemmatizer = WordNetLemmatizer()
lemmatize_tweets = []
for i in alltokens:
    lemmatize_words = []
    for j in i:
        lemmatized = wordnet_lemmatizer.lemmatize(j)
        lemmatize_words.append(lemmatized)
    lemmatize_tweets.append(lemmatize_words)
lemmatize_tweets

# Joining Tweets
processed_tweets = []
for i in lemmatize_tweets:
    joined_tweets = ' '.join(map(str, i))
    processed_tweets.append(joined_tweets)
processed_tweets
```

Feature Extraction using TF-IDF

TF-IDF merupakan suatu cara dalam menentukan bobot hubungan suatu kata (term) terhadap dokumen. Terdapat 2 konsep dalam perhitungan TF-IDF, yaitu perhitungan tf dan idf. Term Frequency (Tf) adalah frekuensi kemunculan kata di dalam sebuah dokumen dan inverse document frequency (idf) adalah inverse frekuensi dokumen yang mengandung kata tersebut. Secara sederhana TF-IDF digunakan untuk mengetahui seberapa sering sebuah kata muncul didalam dokumen. Namun, dalam menentukan bobot hubungan suatu kata dalam dokumen,

TF-IDF memperhatikan seberapa penting kata tersebut dalam dokumen dan seberapa umum kata tersebut digunakan di berbagai dokumen. Tf dengan bobot nilai memiliki hubungan positif, dengan kata lain semakin sering kata tersebut muncul maka bobot kata tersebut akan semakin besar.

2.3.1. Term Frequency (TF)

Term Frequency merupakan frekuensi kemunculan term i pada dokumen j dibagi dengan total *term* pada dokumen j. Ditulis dalam bentuk,

$$TF_{i,j} = \frac{f_d(i)}{\max_{j \in d} f_d(j)} \quad (1)$$

2.3.2. Inverse Document Frequency (IDF)

IDF berfungsi mengurangi bobot suatu *term* jika kemunculannya banyak tersebar diseluruh dokumen. Dituliskan dalam bentuk,

$$IDF_t = \log(\frac{D}{df_t}) \quad (2)$$

dimana df_t merupakan jumlah dokumen yang mengandung *term* t dan D merupakan jumlah total dokumen dalam korpus.

2.3.3. Pembobotan

Setelah mendapatkan nilai TF dan IDF maka kedua nilai tersebut dikalikan value by value, Dituliskan sebagai berikut,

$$W_{i,j} = TF_{i,j} \times IDF_j \quad (3)$$

Dimana $W_{i,j}$ merupakan bobot *term* i pada dokumen j.

Kami menggunakan tools tfidfvectorizer yang disediakan oleh sklearn.feature_extraction.text. Kami membatasi kata sebanyak 1000 kata dan dilakukan stopwords removal.

```
# define vector

foovec = TfidfVectorizer(min_df = 5, max_df = 0.99, tokenizer=nltk.word_tokenize, ngram_range = (1,1), stop_words = 'english', ma
```

Lalu kami menomori dari 1000 kata tersebut, lalu diurutkan dari terkecil hingg terbesar.

```
# train tweets

processed_tweets_tfidf = foovec.fit_transform(processed_tweets)
processed_tweets_tfidf.shape
```

```
# Tweets Number

x = foovec.vocabulary_
x
```

```
{'breast': 104,
 'cancer': 121,
 'risk': 740,
 'test': 880,
 'gp': 360,
 'care': 123,
 'poll': 654,
 'short': 783,
 'people': 625,
 'heart': 392,
 'new': 576,
 'approach': 45,
 'hiv': 402,
 'nh': 579,
 'doctor': 233,
 'review': 735,
 'case': 125,
 'video': 945,
 'day': 200,
```

```
# Sort the tweets

import operator
sorted_x = sorted(x.items(), key=operator.itemgetter(1))
sorted_x
```

```
[('ab', 0),
 ('abortion', 1),
 ('abuse', 2),
 ('aca', 3),
 ('access', 4),
 ('act', 5),
 ('action', 6),
 ('active', 7),
 ('activity', 8),
 ('actually', 9),
 ('ad', 10),
 ('add', 11),
 ('addiction', 12),
 ('adhd', 13),
 ('administration', 14),
 ('adult', 15),
 ('advice', 16),
 ('affect', 17),
 ('affordable', 18),
```

Selanjutnya kami menyimpan nilai dari TF-IDF ke dalam file csv dengan format nama tfidfvectors.csv. Agar memudahkan dalam proses clustering.

```
# Make a .csv file as a place to save the tf-idf score for each word

f2=(open(r'D:\\SEMESTER 7\\Damin\\Health-Tweets\\tfidfvectors.csv', 'w', newline=''))
writer = csv.writer(f2)
header_row = new_list
writer.writerow(header_row)
p = processed_tweets_tfidf.toarray()
for i in range(len(p)):
    writer.writerow(p[i])
```

Scalling with StandardScaler

Melakukan Scalling agar dataset memiliki rentang nilai yang sama, sehingga memudahkan dalam proses clustering.

```
# read the scored dataset that has been saved in a .csv file

df = pd.read_csv('D:\\SEMESTER 7\\Damin\\Health-Tweets\\tfidfvectors.csv', encoding = "ISO-8859-1")
df.head()
```

	ab	abortion	abuse	aca	access	act	action	active	activity	actually	...	worst	worth	wrong	year	yes	yoga	york	young	younger	youth
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
# Scale the dataset for a better data distribution

X = df.iloc[:,0:999]
X = StandardScaler().fit_transform(X)
```

Dimention Reduction using PCA

karena feature yang dimiliki oleh dataset sangat banyak, yaitu sebanyak 1000 jenis features, maka kami perlu mereduksi dimensi sehingga dapat dilakukan clustering.

```
# Train PCA with 90% of datasets trained

pca = PCA(0.90)
pca.fit(X)
pca.n_components_
```

Setelah dilakukan reduksi dimensi, masih terdapat sebanyak 431 komponen. Ini masih terbilang cukup banyak untuk dilakukan klastering. maka kami membatasi hanya sebanyak 2 komponen saja

```
# We define only 5 components PCA

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
pca.explained_variance_ratio_
```

```
array([0.003862  , 0.00377559])
```

```
# datasets after PCA

principalDf = pd.DataFrame(data = principalComponents)
principalDf.head()
```

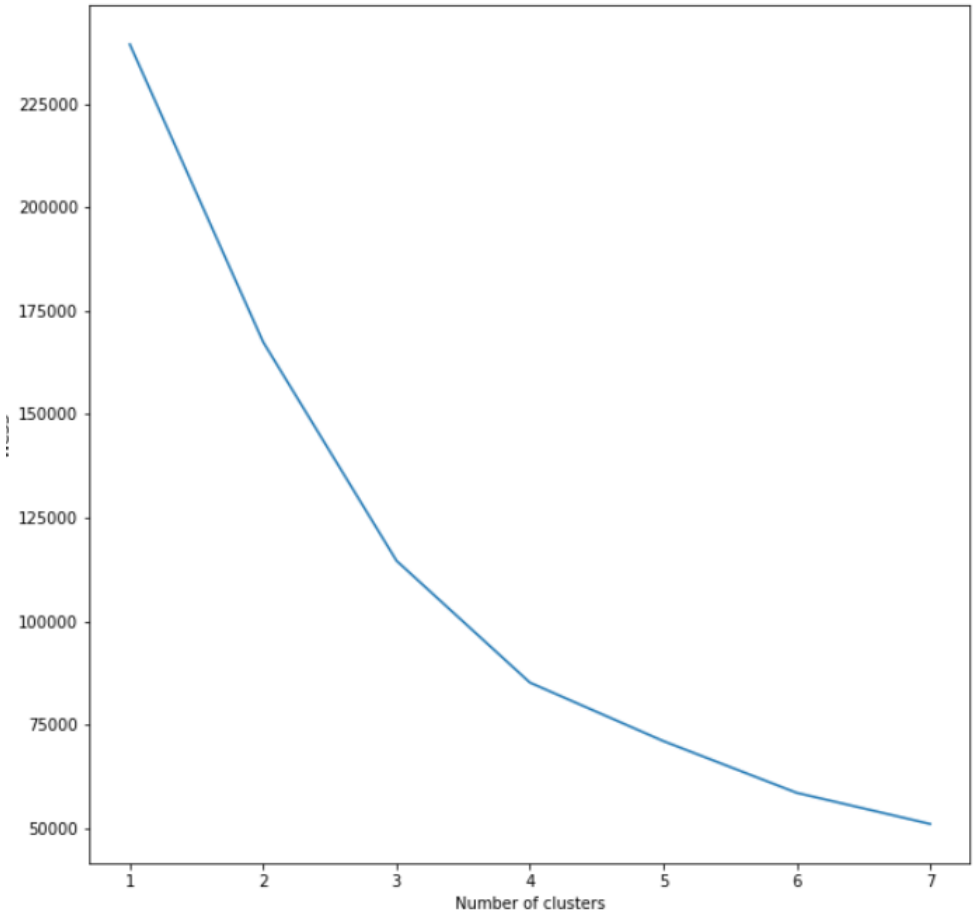
	0	1
0	0.273637	-1.239786
1	-0.887286	1.222685
2	0.598481	-0.097237
3	-0.920236	-0.612111
4	-0.456399	-0.261077

Clustering

Kami melakukan clustering menggunakan metode KMeans karena menurut kami metode ini adalah metode yang paling mudah digunakan didalam python karena sudah disediakan library untuk menggunakan KMeans. Sebelum dilakukan clustering kami melakukan elbow method untuk menentukan jumlah cluster terbaik yang akan digunakan nanti.

```
# finding best n clusters using elbow method

wcss = []
for i in range(1,8):
    model = KMeans(n_clusters = i, init = "k-means++")
    model.fit(principalDf)
    wcss.append(model.inertia_)
plt.figure(figsize=(10,10))
plt.plot(range(1,8), wcss)
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

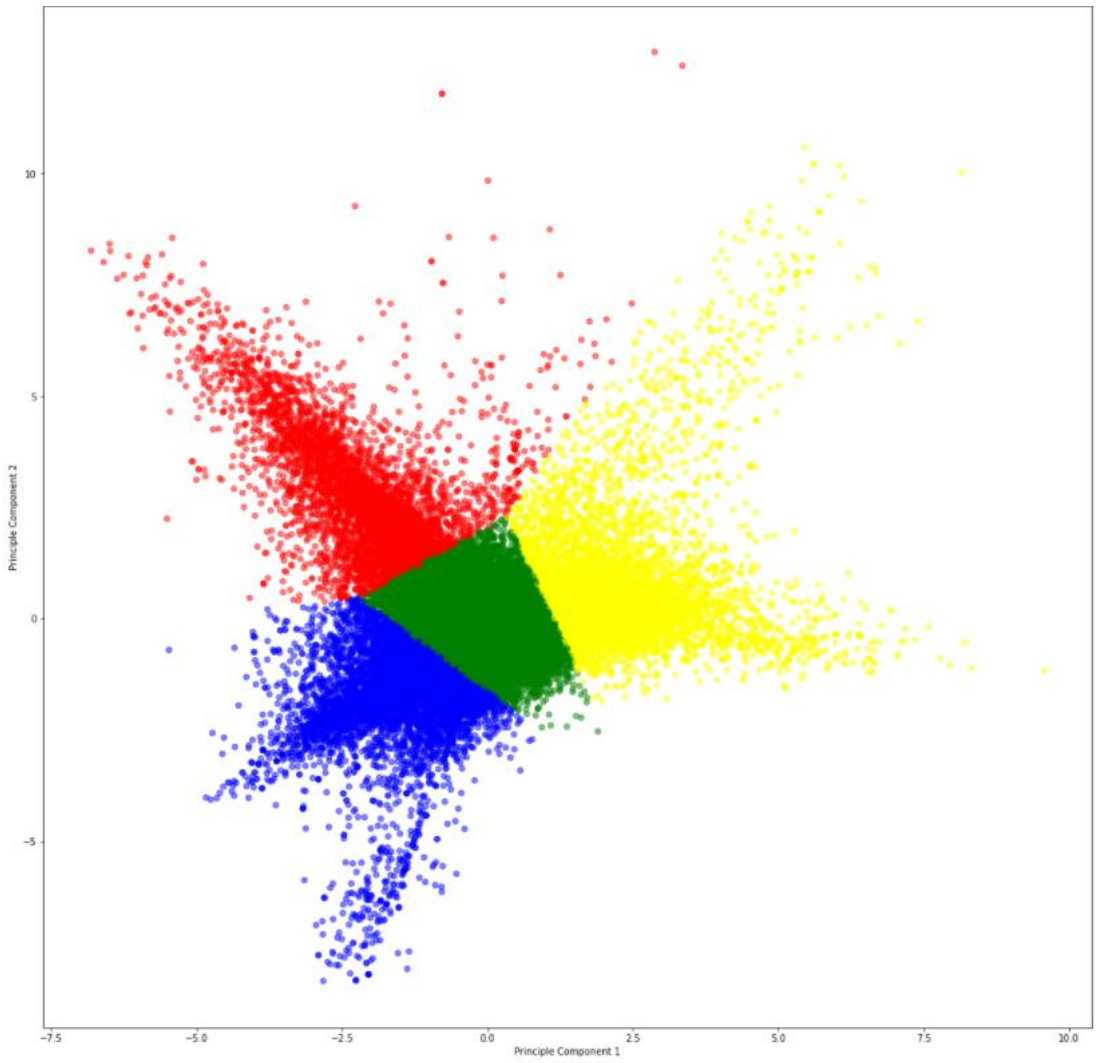

225000
200000
175000
150000
125000
100000
75000
50000
1
2
3
4
5
6
7
Number of clusters

Dalam menentukan cluster terbaik menggunakan elbow method, kita harus melihat persentase hasil perbandingan antara jumlah cluster yang akan membentuk siku pada suatu titik. Maka dari grafik kami mendapat kesimpulan bahwa jumlah cluster terbaik adalah 4 cluster.

```
kmeans = KMeans(n_clusters = 4)
#Compute cluster centers and predict cluster indices
X_clustered = kmeans.fit_predict(principalComponents)

#Define our own color map
LABEL_COLOR_MAP = {0:'red', 1: 'green', 2: 'blue' , 3:'yellow', 4: 'violet' , 5: 'black', 6:'darkorange', 7: 'olive',
                   8: 'chocolate', 9:'magenta', 10: 'maroon', 11: 'cyan', 12:'khaki', 13: 'grey', 14: 'wheat', 15: 'red'}
label_color = [LABEL_COLOR_MAP[l] for l in X_clustered]

# Plot the scatter diagram
plt.figure(figsize = (20,20))
plt.scatter(principalComponents[:,0],principalComponents[:,1], c= label_color, alpha=0.5)
plt.xlabel('Principle Component 1')
plt.ylabel('Principle Component 2')
plt.show()
```



Kesimpulan

Data Set Health News in Twitter awalnya tidak berkualitas, terbukti karena data set ini berupa data tweet(tekstual) yang sudah pasti terdapat karakter-karakter yang tidak diperlukan untuk proses clustering (@, #, ;. dll), maka dari itu perlu dilakukan praproses data. Mulai dari Normalization, expand contraction, stopwords removal, tokenization, terakhir stemming/lemmatization. Setelah praproses data selanjutnya proses clustering dimulai dari scalling data agar data memiliki rentang nilai yang sama. Setelah itu dilakukan elbow method agar mendapatkan jumlah cluster terbaik lalu melakukan clustering menggunakan metode KMeans. Hasilnya berdasarkan grafik data set Health News in Twitter memiliki tingkat kesamaan yang tinggi dalam satu kelas dan memiliki tingkat kesamaan yang rendah antar kelasnya.