APPENDIX

Coding dibawah ini digunakan untuk memproses tf-idf

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
import re
def tokens(text):
    return re.findall('[\w]+', text.lower())

alltok = []
filtertrain = []
for t in tweet:
    wordstok = tokens(t)
    alltok.append(wordstok)
    filter_huruf = []
    for w in wordstok:
        if w.lower() not in Stopwords:
            re.sub('\W', '', w)
            filter_huruf.append(w.lower())

    filtertrain.append(filter_huruf)
print(filtertrain)
from nltk.stem import PorterStemmer
porter = PorterStemmer()
stemtrain = []
for stem in filtertrain:
    wordstem = []
    for word in stem:
        wordstem.append(porter.stem(word))
    stemtrain.append(wordstem)

print(stemtrain)
import numpy as np
listvectorword = [];
for a in stemtrain:
    listvectorword += a

kataunik  = np.unique(listvectorword)
for u in kataunik:
```

```
        print(u)
    list_jumhuruf = [];

    for a in stemtrain:
        jumlah = dict.fromkeys(kataunik, 0)
        for huruf in a:
            jumlah[huruf] += 1

        list_jumhuruf.append(jumlah)

    print(jumlah)
    def computeTF(wordDict, bagOfWords):
        tfDiction = {}
        bagOfWordsCount = len(bagOfWords)
        for word, count in wordDict.items():
            tfDiction[word] = count / float(bagOfWordsCount)
        return tfDiction


    listtf = []
    i = 0
    for doc in stemtrain:
        tf = computeTF(list_jumhuruf[i], doc)

        listtf.append(tf)
        i += 1
    def computeIDF(documents):
        import math
        N = len(documents)

        idfDiction = dict.fromkeys(documents[0].keys(), 0)
        for document in documents:
            for word, val in document.items():
                if val > 0:
                    idfDiction[word] += 1

        for word, val in idfDiction.items():
            idfDiction[word] = math.log(N / float(val))
        return idfDiction

    idf = computeIDF(list_jumhuruf)
    def computeTFIDF(tfBagOfWords, idf):
        tfidf = {}
        for word, val in tfBagOfWords.items():
```

```
            tfidf[word] = val * idf[word]
        return tfidf

list_tfidf = []
for tf in listtf:
    tfidf = computeTFIDF(tf, idf)
    list_tfidf.append(tfidf)

df = pd.DataFrame(list_tfidf)
```

PLAGIARISM CHECK.ORG

2.13% PLAGIARISM APPROXIMATELY

Report #11819752

Introduction Background With the development of technology today it makes it easier for us to exchange opinions or opinions through social media. This will certainly cause a problem, namely differences of opinion or opinion towards fellow social media users. Therefore this project was created to make it easier to analyze the opinions of social media users by using sentiment analysis. Sentiment Analysis itself is the process of analyzing various data in the form of opinions or views so that conclusions are generated from various existing opinions. The problem to be resolved in this project is how the opinion of Twitter social media users on the current social distancing. This problem occurs because of the Covid-19 virus which is currently endemic, one way to avoid or reduce victims of this virus, the government is implementing social distancing to reduce or prevent the spread of the Covid-19 virus. The way to solve this problem is by using the Vector Space Model algorithm and Naive Bayes algorithm. Which is then implemented into Sentiment Analysis to get maximum results. The first step of data is taken by scraping from Twitter first. Then the data will be processed by the preprocessing method before being completed using the Vector Space Model algorithm which will produce positive or negative final results. After that, it will be evaluated by calculating accuracy, precision,

REPORT	CHECKED	AUTHOR	PAGE
#1181975217	NOV 2020, 10:00 AM	UNIKA SOEGIJAPRANATA	**1** OF 23