LAMPIRAN

// Proses install library yang dibutuhkan

```
!pip install textblob
!pip install sastrawi
!pip install emoji
!pip install install PySastrawi
!pip install tweet-preprocessor
!pip install nltk
!pip install seaborn
!pip install numpy


import pandas as pd
import numpy as np
import re
from textblob import TextBlob
import string
import preprocessor as p
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import emoji
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory
import seaborn as sns
from preprocessor.api import clean,tokenize,parse
```

//Proses Read Data

```
def loaddata():
    dataset = pd.read_csv('gametweet.csv')
    return dataset
tweet_df=loaddata()
tweet_df.head(10)
tweet_df.shape
```

//Proses Clean Data

```
def clean_data(tweet):
    tweet = re.sub("RT @[\w]*:", "", str(tweet))
    tweet = re.sub("@[\w]*", "", tweet)
    tweet = re.sub("https://[A-Za-z0-9./]", "", tweet)
    tweet = re.sub("\n", "", tweet)
    tweet = re.sub("&amp", "", tweet)
    tweet = re.sub("#", "", tweet)
    tweet = re.sub(r"[^\w]", ' ', tweet )
    return tweet
tweet_df['Remove_Pattern'] = tweet_df['Tweet'].apply(lambda x: clean_data(x))
tweet_df.sort_values("Remove_Pattern", inplace = True)
tweet_df.head(10)
```

//Proses Stemming dan Stopword

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwordindo = stopwords.words('indonesian')
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory, StopWordRemover,ArrayDictionary


stop_factory = StopWordRemoverFactory().get_stop_words()
```

```
more_stopwords =
['tu','dkk','bikin','dah','Minji','Cooking','pkv','krn','lg','jantanslot88','ceri388','wuih','
bgt','RPG','parada4d','raja12shio','jantanslot88','gw','rt','jaya365','prediksijaya365','
3','cp','danaqq','yg','wkt','dtng','gk','Skrng','pdhl','aelah','jd','bs','nga','nekobet99','ak'
,'LOL','naga303','whay','rek','Lyto','ny','hyatus','tinggl','Maxwin','WKWKW','Freef
orm','we','utk','je','taknak', 'nder','hode','typing'
,'pen','trs','gue','TTM','Owalaa','tmn','udh','knlan','wkwk','kgk','ak',
'kgk','gtw','blom','rekom','jaya365','loe','wkt','gw','si','jd','bs','fatwatarjih','lyto','slotj
aya365','pls',  'apk','ho','nder','https t co jtnyaij','rem','anjay','hiatus','vxpaqups',' t
co','https t co nuzsjznvk','http','jpmaxwin','https', 't co rxhbnxzw', 't co
mktpcq','kekekeke','yg',]

data = stop_factory + more_stopwords

dictionary = ArrayDictionary(data)

str =StopWordRemover(dictionary)

print(data)

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

factory = StemmerFactory()

stemmer = factory.create_stemmer()
```

#Proses Tokenize

```
from nltk.tokenize import TweetTokenizer

#Emoticon

emo=set([':)','XD','=D','X-D','8D',':-D',':P',':>',':o','X-P',':D',':-/',':-(',':s',':(',':@'])

def clean(tweet):
   tokenizer    = TweetTokenizer(preserve_case=False, strip_handles=True,
reduce_len=True)
   tweet_tokens = tokenizer.tokenize(tweet)
   tweets_clean = []
   for word in tweet_tokens:
      if (word not in data and
            word not in emo and
               word not in string.punctuation):
         stem_word = stemmer.stem(word)
         tweets_clean.append(stem_word)
```

```
    return tweets_clean
tweet_df['Hasil'] = tweet_df['Remove_Pattern'].apply(lambda x:clean(x))
tweet_df.head(10)
```

// Proses remove punctuation

```
def removepunc(tanda):
    tanda = " ".join([char for char in tanda if char not in string.punctuation])
    return tanda
tweet_df['Hasil']=tweet_df['Hasil'].apply(lambda x: removepunc(x))
tweet_df.to_csv('Hasilbersih1.csv',encoding='utf8',index=False)
```

// Proses remove tweet kosong

```
tweet_df = tweet_df[tweet_df['Hasil']!='']
tweet_df.head(10)
```

// Proses Rearange data

```
tweet_df.drop_duplicates(subset="Hasil", keep='first',inplace=True)
tweet_df.head(10)
```

// Proses reset index

```
tweet_df=tweet_df.reset_index(drop=True)
tweet_df.head(10)
```

// Proses Remove Colom

```
tweet_df.drop(tweet_df.columns[[0,1,2,3]],axis= 1,inplace = True)
tweet_df.head(10)
```

// Proses Simpan data bersih

```
tweet_df.to_csv('Databersihgame1.csv',encoding='utf8',index=False)
```

// Proses installing google translate

```
!pip3 install googletrans==3.1.0a0
```

// Import library yang kita butuh

```
import pandas as pd
from googletrans import Translator
```

```
dframe = pd.read_csv('Databersihgame1.csv')
dframe.head(10)
//Proses translasi data
translator = Translator()
translations ={}
for column in dframe.columns:
    unique_elements = dframe[column].unique()
    for element in unique_elements:
        translations[element] = translator.translate(element).text
translations
// Proses Simpan data transate
dframe.replace(translations, inplace = True)
dframe.head(10)
dframe.to_csv('hasilgogle1.csv',encoding='utf8',index=False)
// Proses instal library
!pip install wordcloud
!pip install matplotlib
!pip install tweepy
!pip install sklearn
!pip install scikit-learn
!pip install estimator
!pip install seaborn
from wordcloud import WordCloud
import csv
import nltk
from nltk.classify import NaiveBayesClassifier
import matplotlib.pyplot as plt
import seaborn as sns
import preprocessor as p
```

```
from nltk.tokenize import word_tokenize
from textblob import TextBlob
from nltk.stem import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn import model_selection
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels
from sklearn.preprocessing import
LabelBinarizer,OrdinalEncoder,OneHotEncoder
from sklearn import metrics
import joblib
import pickle
dataframe = pd.read_csv('hasilgogle1.csv')
dataframe.head(15)
#Proses labeling menggunakan Textblob
from textblob import TextBlob
def getSubjectivity(text):
    return TextBlob(text).sentiment.subjectivity
def getPolar(text):
    return TextBlob(text).sentiment.polarity
dataframe['Subjektif'] = dataframe ['Hasil'].apply(getSubjectivity)
dataframe['Polaritas'] = dataframe ['Hasil'].apply(getPolar)
dataframe.head(10)
def getAnalysis(score):
```

```
    if score<0:
        return 'Negative'
    elif score==0:
        return 'Neutral'
    else:
        return 'Positif'


dataframe['Label']=dataframe['Polaritas'].apply(getAnalysis)
#show dataframe
dataframe.head(10)
dataframe.to_csv('hasildataaktual1.csv',encoding='utf8',index=False)
// Proses membuat word cloud
semua=''.join([tweet for tweet in dataframe['Hasil']])
wc=WordCloud(width=500,height=300,random_state=21,max_font_size=119).ge
nerate(semua)
plt.imshow(wc,interpolation="bilinear")
plt.axis('off')
plt.show()
// Proses membuat diagram batang
dataframe['Label'].value_counts()
plt.title('Sentimen Analisis Game Online')
plt.xlabel('Sentiment')
plt.ylabel('Counts')
dataframe['Label'].value_counts().plot(kind='bar')
plt.show()
// Proses membuat diagram lingkaran
dataframe['Label'].value_counts()
plt.title('Sentimen Analisis Game Online')
plt.xlabel('Sentiment')
plt.ylabel('Counts')
```

```
dataframe['Label'].value_counts().plot(kind='pie')
plt.show()
// Proses melihat value
dataframe['Label'].value_counts()
dataframe = dataframe.astype({'Hasil' : 'category'})
dataframe = dataframe.astype({'Label' : 'string'})
dataframe.dtypes
X = dataframe[['Hasil']]
y = dataframe ['Label']
print(X.shape)
print(y.shape)
// Proses membuat TF-IDF
trans= CountVectorizer()
print(dataframe['Hasil'].shape)
X=trans.fit_transform(dataframe['Hasil'])
print(X.toarray())
print('Shape of Sparse Matrix:',X.shape)
print('Amount of Non-Zero occurences : ',X.nnz)
#TFIDF Transform
tftransform =TfidfTransformer(use_idf=False).fit(X)
X= tftransform.transform(X)
print(X.shape)
// Proses membuat vectorizer
from sklearn.feature_extraction.text import CountVectorizer
bt = CountVectorizer().fit(dataframe['Hasil'])
bt.vocabulary_
// Proses membuat data training dan testing
from sklearn.model_selection import train_test_split
X_train, X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=42)
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

// Proses algoritma Naïve Bayes

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB().fit(X_train,y_train)
pred = model.predict(X_test)
predict = pd.Series(pred)
print(predict.to_string())
true= pd.Series(y_test)
print(true.to_string())
```

//Proses membuat membuat *matrix confussion* NB

```
from time import time
from pandas import DataFrame
import seaborn as sn
t = time()
y_pred    = model.predict(X_test)
test_time = time() - t
print(" test time : %.3fs" % test_time)
scores    = metrics.accuracy_score(y_test,y_pred)
print("Accuracy :  %.3fs" % scores)

print(metrics.classification_report(y_test, y_pred,
target_names=['Negatif','Neutral','Positif']))
column    = ['Negatif','Neutral','Positif']
confirm   = confusion_matrix(y_test,y_pred)
df_data   = DataFrame(confirm, index=column,columns=column)
ax        = sn.heatmap(df_data,cmap='Oranges',annot=True)
ax.set_title("Confusion Matrix")
```

```
ax.set_xlabel("Label Prediksi")
ax.set_ylabel("Label Sebenarnya")
```

// Proses instal algoritma *vader lexicon*

```
!pip install vaderSentiment
```

// Proses algoritma *vader lexicon*

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
score = [analyzer.polarity_scores(x) for x in databaru['Hasil']]
databaru['Compound_Score'] = [x['compound'] for x in score]
#compoun Lexicon
databaru.loc[databaru['Compound_Score'] <0,'Sentiment'] = 'Negative'
databaru.loc[databaru['Compound_Score'] ==0,'Sentiment'] = 'Neutral'
databaru.loc[databaru['Compound_Score'] >0,'Sentiment'] = 'Positif'
databaru.head(20)
x=databaru['Hasil']
Y=databaru['Sentiment']
dataframe = databaru.astype({'Hasil' : 'string'})
dataframe = dataframe.astype({'Sentiment' :'string'})
dataframe.dtypes
from sklearn.model_selection import train_test_split
x_train, x_test,Y_train,Y_test =
train_test_split(X,Y,test_size=0.2,random_state=42)
print(x_train.shape)
print(x_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

//Proses membuat membuat *matrix confussion* VL

```
import seaborn as sn
t = time()
Y_pred    = model.predict(x_test)
test_time = time() - t
print(" test time : %.3fs" % test_time)
scores    = metrics.accuracy_score(Y_test,Y_pred)
print("Accuracy :  %.3fs" % scores)
print(metrics.classification_report(Y_test, Y_pred,
target_names=['Negatif','Neutral','Positif']))
column    = ['Negatif','Neutral','Positif']
confirm   = confusion_matrix(Y_test,Y_pred)
df_data   = DataFrame(confirm, index=column,columns=column)
ax        = sn.heatmap(df_data,cmap='Blues',annot=True)
ax.set_title("Confusion Matrix")
ax.set_xlabel("Label Prediksi")
ax.set_ylabel("Label Sebenarnya")
```