LAMPIRAN

1. Scraping Data Shopee

```
!pip install google-play-scraper

from google_play_scraper import app

import pandas as pd

import numpy as np

from google_play_scraper import Sort, reviews

result, continuation_token = reviews(
    'com.shopee.id',
    lang='id'
    country='id',
    sort=Sort.MOST_RELEVANT,
    count=8229,
    filter_score_with=None
)

df_busu = pd.DataFrame(np.array(result),columns=['review'])

df_busu = df_busu.join(pd.DataFrame(df_busu.pop('review').tolist()))

df_busu.head()
new_df = df_busu[['userName', 'score','at', 'content']]
sorted_df = new_df.sort_values(by='at', ascending=False).
sorted_df.head()

my_df = sorted_df[['content','score']]

my_df.to_csv("data_ulasan_shopee.csv", index = False)
```

2. Pelabelan Data Shopee

```
def pelabelan(score):
  if score < 3:
    return 'Negatif'
  elif score == 4 :
    return 'Positif'
  elif score == 5 :
    return 'Positif'
my_df['Label'] = my_df ['score'].apply(pelabelan)
```

```
my_df.head(50)

# info() digunakan untuk menampilkan informasi detail tentang dataframe,
#seperti jumlah baris data, nama-nama kolom berserta jumlah data dan tipe
datanya, dan sebagainya.
my_df.info()

#Tampilkan setiap baris yang memiliki nilai null (NaN) pada kolom apapun
#Gunakan fitur isna() yang disediakan library pandas
my_df.isna()

my_df.isna().any()

my_df.describe()

#mencari jumlah baris data yang bernilai null
#terdapat kolom label memiliki nilai kosong
my_df.isnull().sum()

my_df.dropna(subset=['Label'],inplace = True)

my_df.isnull().sum()

my_df.to_csv("shopee_pelabelan.csv", index = False)
```

3. *Case Folding* dan *Cleaning*

```
import re
def  clean_text(df, text_field, new_text_field_name):
my_df[new_text_field_name] = my_df[text_field].str.lower()
    my_df[new_text_field_name] = my_df[new_text_field_name].apply(lambda
elem: re.sub(r"(@[A-Za-z0-9]+)|([^0-9A-Za-z
\t])|(\w+:\/\/\S+)|^rt|http.+?", "", elem))
    # remove numbers
    my_df[new_text_field_name] = my_df[new_text_field_name].apply(lambda
elem: re.sub(r"\d+", "", elem))
    return my_df

my_df['text_clean'] = my_df['content'].str.lower()
my_df['text_clean']
data_clean = clean_text(my_df, 'content', 'text_clean')
data_clean.head(10)
```

4. *Stopword*

```
import nltk.corpus
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('indonesian')
data_clean['text_StopWord'] = data_clean['text_clean'].apply(lambda x:'
'.join([word for word in x.split() if word not in (stop)]))
data_clean.head(10)
```

5. *Tokenizing*

```
import nltk
nltk.download('punkt')
from nltk.tokenize import sent_tokenize, word_tokenize
data_clean['text_tokens'] = data_clean['text_StopWord'].apply(lambda x:
word_tokenize(x))
data_clean.head()
```

6. *Stemming*

```
!pip install Sastrawi

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
factory = StemmerFactory()
stemmer = factory.create_stemmer()

#-----------------STEMMING -----------------
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
#import swifter

# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# stemmed
def stemmed_wrapper(term):
    return stemmer.stem(term)

term_dict = {}
hitung=0

for document in data_clean['text_tokens']:
    for term in document:
        if term not in term_dict:
            term_dict[term] = ' '
```

```
print(len(term_dict))
print("------------------------")
for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)
    hitung+=1
    print(hitung,":",term,":" ,term_dict[term])

print(term_dict)
print("------------------------")

# apply stemmed term to dataframe
def get_stemmed_term(document):
    return [term_dict[term] for term in document]

#script ini bisa dipisah dari eksekusinya setelah pembacaaan term selesai
data_clean['text_steamindo'] = data_clean['text_tokens'].apply(lambda x:'
'.join(get_stemmed_term(x)))
data_clean.head(10)

my_df.to_csv("textPrepocessing_shopee.csv", index = False)
```

7. *Decision Tree* tanpa SMOTE dan *Decision Tree* menggunakan SMOTE

```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score,f1_score, classification_report, confusion_matrix

# Membagi data
X_train, X_test, y_train, y_test = train_test_split(data_clean['content'],
data_clean['Label'],
                                                    test_size=0.20,
random_state=0)

# Vektorisasi menggunakan CountVectorizer
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

# Fungsi untuk mencetak metrik
def print_metrics(y_true, y_pred, title):
    print(f"\n{title}")
    print("Accuracy:", accuracy_score(y_true, y_pred))
```

```
    print("Precision:", precision_score(y_true, y_pred,
average="weighted", zero_division=0))
    print("Recall:", recall_score(y_true, y_pred, average="weighted",
zero_division=0))
    print("F1-score:", f1_score(y_true, y_pred, average="weighted",
zero_division=0))
    print("\nConfusion Matrix:")
    print(confusion_matrix(y_true, y_pred))
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred, zero_division=0))

# 1. Decision Tree tanpa SMOTE
dt_model = DecisionTreeClassifier(random_state=0)
dt_model.fit(X_train_vectorized, y_train)
y_pred_dt = dt_model.predict(X_test_vectorized)

# 2. Decision Tree dengan SMOTE
smote = SMOTE(random_state=0)
X_train_smote, y_train_smote = smote.fit_resample(X_train_vectorized,
y_train)

dt_model_smote = DecisionTreeClassifier(random_state=0)
dt_model_smote.fit(X_train_smote, y_train_smote)
y_pred_dt_smote = dt_model_smote.predict(X_test_vectorized)

# Mencetak hasil
print_metrics(y_test, y_pred_dt, "Decision Tree tanpa SMOTE - Metrics:")
print_metrics(y_test, y_pred_dt_smote, "Decision Tree dengan SMOTE -
Metrics:")
```