LAMPIRAN

1. Scraping data

```
result_1, _ = reviews(
    'com.bibit.bibitid', lang='id', country='id',
sort=Sort.NEWEST, count=1000
)

result_2, _ = reviews(
    'ajaib.co.id', lang='id', country='id', sort=Sort.NEWEST,
count=1000
)

result_3, _ = reviews(
    'com.stockbit.android', lang='id', country='id',
sort=Sort.NEWEST, count=1000
)

all_reviews = result_1 + result_2 + result_3
df_final = pd.DataFrame(all_reviews)
df_final.to_csv(file_path, index=False)
```

2. Data preparation

```
# Lower Case
def lowercase(review_text):
    text_lower = review_text.lower()
    return text_lower

df_dataset['lowercase'] =
df_dataset['content'].apply(lowercase)
df_dataset.head()
# Menghilangkan Emoji
def remove_emojis(text):
    emoji_pattern = re.compile(
        "["
        "\U0001F600-\U0001F64F"
        "\U0001F300-\U0001F5FF"
        "\U0001F680-\U0001F6FF"
        "\U0001F700-\U0001F77F"
        "\U0001F780-\U0001F7FF"
        "\U0001F800-\U0001F8FF"
        "\U0001F900-\U0001F9FF"
        "\U0001FA00-\U0001FA6F"
        "\U0001FA70-\U0001FAFF"
        "\U00002702-\U000027B0"
        "\U000024C2-\U0001F251"
```

```
        "]+",
        flags=re.UNICODE)

    return emoji_pattern.sub(r'', text)

df_dataset['clean_review'] =
df_dataset['lowercase'].apply(lambda x: remove_emojis(x))
df_dataset.head(5)
# Menghilangkan Number
def remove_number(review_text, default_replace=" "):
  num = re.sub(r'\d+', default_replace, review_text)
  return num

df_dataset['clean_review'] =
df_dataset['clean_review'].apply(lambda x: remove_number(x))
df_dataset.head(5)
# Menghilangkan tanda baca
def remove_punctuation(review_text, default_text=" "):
    list_punct = string.punctuation
    delete_punct = str.maketrans(list_punct, ' ' *
len(list_punct))

    review = review_text.translate(delete_punct)
    return review

df_dataset['clean_review'] =
df_dataset['clean_review'].apply(lambda x:
remove_punctuation(x))
df_dataset.head(5)
# Menghilangkan Whitespaces

def remove_whitespaces(review_text):
  review = re.sub(r'\s+',' ', review_text)
  review = review.strip()
  return review

df_dataset['clean_review'] =
df_dataset['clean_review'].apply(remove_whitespaces)
df_dataset.head(5)
# Normalization
def normalisasi_review(teks):
    kata_kata = teks.split()

    kata_normalisasi = [kamus_dict.get(kata, kata) for kata in
kata_kata]

    return ' '.join(kata_normalisasi)
```

```
df_dataset['normalization_review'] =
df_dataset['clean_review'].apply(normalisasi_review)
df_dataset.head()
def stopword_remove(review_text):
    words = review_text.split()
    filtered_words = [word for word in words if word not in
stop_words]
    return ' '.join(filtered_words)

df_dataset['stopwords_review'] =
df_dataset['normalization_review'].apply(stopword_remove)

df_dataset.head(5)
# Stemming
import swifter
from tqdm import tqdm

tqdm.pandas()

def stem_review(review_text):
    return stemmer.stem(review_text)

df_dataset['stemming_review'] =
df_dataset['stopwords_review'].swifter.progress_bar().apply(st
em_review)

df_dataset.head()
```

3. Training model

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(
    ngram_range=(1, 2),
    max_df=0.95,
    min_df=5,
    sublinear_tf=True
)

X_tfidf = tfidf.fit_transform(X)

print(X_tfidf.shape)
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X,
```

```
    y,
    test_size=0.3,
    random_state=42,
    stratify=y
)

print("Train size:", X_train.shape[0])
print("Test size :", X_test.shape[0])
# 10 Iterasi x 5
kernels = ['sigmoid']
n_splits = 10
n_repeats = 5
```

4. Evaluation SCFV

```
# Menyimpan akurasi per fold per repeat
fold_accuracies = np.zeros((n_repeats, n_splits))

for repeat in range(n_repeats):
    print(f"\n Repeat ke-{repeat+1}")

    skf = StratifiedKFold(
        n_splits=n_splits,
        shuffle=True,
        random_state=42 + repeat
    )

    svm = SVC(
        kernel='sigmoid',
        C=16.2,
        gamma=0.02
    )

    for fold, (train_idx, test_idx) in
enumerate(skf.split(X_tfidf, y)):
        X_train = X_tfidf[train_idx]
        X_test  = X_tfidf[test_idx]
        y_train = y.iloc[train_idx]
        y_test  = y.iloc[test_idx]

        svm.fit(X_train, y_train)
        y_pred = svm.predict(X_test)

        acc = accuracy_score(y_test, y_pred)
        fold_accuracies[repeat, fold] = acc

        print(f"   Fold {fold+1:02d} | Acc: {acc:.4f}")
```

```
mean_fold_acc = np.mean(fold_accuracies, axis=0)
std_fold_acc  = np.std(fold_accuracies, axis=0)

folds = np.arange(1, n_splits + 1)

plt.figure(figsize=(9, 5))
plt.plot(folds, mean_fold_acc, marker='o', linewidth=2)
plt.xticks(folds)
plt.ylim(0.65, 0.95)
plt.xlabel("Iteration (Fold)")
plt.ylabel("Accuracy")
plt.grid(True, linestyle='--', alpha=0.6)

# Tampilkan persentase di setiap titik
for x, y in zip(folds, mean_fold_acc):
    plt.text(x, y + 0.005, f"{y*100:.2f}%", ha='center',
fontsize=9)

plt.title("Hasil 10-fold Stratified Cross Fold Validation")
plt.tight_layout()
plt.show()

from scipy.stats import friedmanchisquare

stat, p_value = friedmanchisquare(
    df_results['linear'],
    df_results['rbf'],
    df_results['sigmoid']
)

print("Friedman statistic:", stat)
print("p-value:", p_value)
```


5. Testing prediction

```
# Transform & predict
X_tfidf = tfidf.transform(df['stemming_review'].astype(str))
predicted_labels = model.predict(X_tfidf)

# Simpan hasil
df['predicted_label'] = predicted_labels
df.to_csv(output_csv, index=False)

print(f"Hasil prediksi disimpan di: {output_csv}")
valid_mask = (
    ((df_predict_label['score'].isin([4, 5])) &
(df_predict_label['predicted_label'] == 1)) |
```

```
    ((df_predict_label['score'].isin([1, 2, 3])) &
(df_predict_label['predicted_label'] == 0))
)

mismatch_mask = ~valid_mask

print(f"Jumlah prediksi valid    : {valid_mask.sum()}")
print(f"Jumlah prediksi mismatch : {mismatch_mask.sum()}")
print(f"Total data               : {len(df_predict_label)}")
sentiment_counts = df['label'].value_counts()
colors = sns.color_palette("pastel", 2)
explode = (0, 0.1)
labels = ['Positive', 'Negative']

ax = sentiment_counts.plot(kind='pie', fontsize=12,
colors=colors, explode=explode, autopct='%1.1f%%',
labels=labels)
ax.set_ylabel('')
plt.xlabel('Sentiment Label', weight = "bold", fontsize = 14,
labelpad = 20)
plt.axis('equal')
plt.legend(labels=labels, loc='best')
plt.show()
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Filter sentimen positif
df_sentiment_positif = df[df['label'] == 1]

# Gabungkan teks
text_sentiment_positif = ' '.join(
    df_sentiment_positif['stemming_review'].astype(str)
)

# Cleaning ringan
text_sentiment_positif = text_sentiment_positif.replace("'",
"").replace('"', "")

# Generate wordcloud
wordcloud = WordCloud(
    max_words=500,
    background_color='white',
    width=800,
    height=400
).generate(text_sentiment_positif)

# Tampilkan
```

```
plt.figure(figsize=(10,5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title("WordCloud Sentimen Positif")
plt.show()
```