Source Code Data Preprocessing

```
import pandas as pd
import re

df = pd.read_csv('bpjs.csv', delimiter=';')
print(df.head())
print(df.columns)

df = df[['full_text']]
df = df.rename(columns={'full_text': 'tweet'})
print(df.head())

def clean_text(text):
    text = re.sub(r'@\w+', '', text)  # Hapus mention
    text = re.sub(r'http\S+|www\S+|https\S+', '', text,
flags=re.MULTILINE)  # Hapus tautan URL
    text = re.sub(r'#\w+', '', text)  # Hapus hashtag
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)  # Hapus karakter
non-alfanumerik
    text = re.sub(r'\s+', ' ', text).strip()  # Hapus spasi
ekstra
    text = text.lower()  # Ubah teks menjadi lowercase
    return text

def preprocess_text(text):
    cleaned_text = clean_text(text)
    return cleaned_text

# Lakukan preprocessing untuk setiap teks di kolom 'tweet'
df['processed_tweet'] = df['tweet'].apply(preprocess_text)

print(df.head())

#drop kolom tweet dan ubah kolom processed_tweet  jadi tweet
df = df.drop('tweet', axis=1)
df = df.rename(columns={'processed_tweet': 'tweet'})
print(df.head())

df.to_csv('bpjs_clean.csv', index=False)
```

Source Code Modeling Transformers

```
!pip3 install torch --index-url
https://download.pytorch.org/whl/cu118
!pip install accelerate
!pip install datasets

import pandas as pd
import torch
from sklearn.model_selection import train_test_split
from transformers import BertTokenizer,
BertForSequenceClassification, Trainer, TrainingArguments ,
BertConfig, EarlyStoppingCallback, TrainerCallback
from sklearn.metrics import classification_report
from datasets import load_metric
import gdown
import matplotlib.pyplot as plt

# Download the dataset from Google Drive
url =
"https://drive.google.com/uc?id=1B_RImDfeWu1plpStl903tVEt1ms4RKB
x"
output = "bpjs.csv"
gdown.download(url, output, quiet=False)

# Memuat dataset
df = pd.read_csv('bpjs.csv')

# Memastikan fitur (teks) adalah string
df['tweet'] = df['tweet'].astype(str)

# Memastikan label adalah integer
df['sentiment'] = df['sentiment'].astype(int)

# Memisahkan data menjadi fitur (X) dan label (y)
X = df['tweet'].values
y = df['sentiment'].values

# Membagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Memuat tokenizer lBERT
tokenizer = BertTokenizer.from_pretrained('ayameRushia/bert-
base-indonesian-1.5G-sentiment-analysis-smsa')

# Tokenisasi data latih dan uji
train_encodings = tokenizer(list(X_train), padding=True,
truncation=True, max_length=128)
test_encodings = tokenizer(list(X_test), padding=True,
truncation=True, max_length=128)

# Mempersiapkan dataset dalam format PyTorch
class SentimentDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels
```

```
    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in
self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)

train_dataset = SentimentDataset(train_encodings, y_train)
test_dataset = SentimentDataset(test_encodings, y_test)

# Memuat model DistilBERT untuk klasifikasi
model =
BertForSequenceClassification.from_pretrained('ayameRushia/bert-
base-indonesian-1.5G-sentiment-analysis-smsa', num_labels=3)

# Fungsi compute metrics
def compute_metrics(p):
    predictions, labels = p
    preds = torch.tensor(predictions).argmax(dim=1)
    report = classification_report(labels, preds,
output_dict=True)
    accuracy = report['accuracy']
    precision = report['weighted avg']['precision']
    recall = report['weighted avg']['recall']
    f1 = report['weighted avg']['f1-score']
    return {
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'f1': f1
    }

# Mendefinisikan argumen pelatihan dengan hyperparameter tuning
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=64,
    warmup_steps=200,
    weight_decay=0.1,
    logging_dir='./logs',
    logging_steps=10,
    eval_strategy="epoch",
    save_strategy="epoch",
    learning_rate=3e-5,
    load_best_model_at_end=True,
    metric_for_best_model="accuracy"
)

# Mendefinisikan pelatihan
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics,
```

```
    callbacks=[EarlyStoppingCallback(early_stopping_patience=3)]
)

# Melatih model
trainer.train()

# Evaluasi model
results = trainer.evaluate()
print("Akurasi :", results['eval_accuracy'])
```

Source Code Modeling Naïve Bayes & Support Vector Machine

```
import pandas as pd
import gdown
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,
classification_report, accuracy_score, precision_score,
recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns

# Fungsi untuk mengunduh dataset menggunakan gdown
def download_dataset(url, output):
    gdown.download(url, output, quiet=False)

# URL dataset
url_A = 'https://drive.google.com/uc?id=1-
b2JNs3t8pkWOS6x9hf1zghoAfmtWhHd'
url_B =
'https://drive.google.com/uc?id=1B_RImDfeWu1plpStl903tVEt1ms4RKB
x'
url_C = 'https://drive.google.com/uc?id=1np12EGOUGPBKVazfhnFYfW-
bYLYi9KvV'
url_D =
'https://drive.google.com/uc?id=1P1IZszCl7kakSXghbbmxjiYqfah2yC5
d'

# Output file paths
file_path_A = 'dataset_A.csv'
file_path_B = 'dataset_B.csv'
file_path_C = 'dataset_C.csv'
file_path_D = 'dataset_D.csv'

# Mengunduh dataset
download_dataset(url_A, file_path_A)
download_dataset(url_B, file_path_B)
download_dataset(url_C, file_path_C)
download_dataset(url_D, file_path_D)

# Fungsi untuk memuat dataset
def load_dataset(file_path):
    return pd.read_csv(file_path)

# Fungsi untuk memproses data dan melatih model
def train_and_evaluate_model(dataset, model, vectorizer):
    X = dataset['tweet']
    y = dataset['sentiment']

    # Memisahkan data latih dan data uji
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
    # Mengubah teks menjadi fitur TF-IDF
    X_train_tfidf = vectorizer.fit_transform(X_train)
    X_test_tfidf = vectorizer.transform(X_test)

    # Melatih model
    model.fit(X_train_tfidf, y_train)

    # Memprediksi data uji
    y_pred = model.predict(X_test_tfidf)

    # Menghitung confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Menghitung metrik evaluasi
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred,
average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')

    return cm, accuracy, precision, recall, f1

# Memuat dataset
dataset_A = load_dataset(file_path_A)
dataset_B = load_dataset(file_path_B)
dataset_C = load_dataset(file_path_C)
dataset_D = load_dataset(file_path_D)

# Membuat model Naive Bayes dan SVM
nb_model = MultinomialNB()
svm_model = SVC()

# Membuat vectorizer
vectorizer = TfidfVectorizer()

# Melatih dan mengevaluasi model untuk setiap dataset
datasets = {'A': dataset_A, 'B': dataset_B, 'C': dataset_C, 'D':
dataset_D}
models = {'Naive Bayes': nb_model, 'SVM': svm_model}
results = {'Naive Bayes': {}, 'SVM': {}}

for model_name, model in models.items():
    for dataset_name, dataset in datasets.items():
        cm, accuracy, precision, recall, f1 =
train_and_evaluate_model(dataset, model, vectorizer)
        results[model_name][dataset_name] = {
            'confusion_matrix': cm,
            'accuracy': accuracy,
            'precision': precision,
            'recall': recall,
            'f1': f1
        }

# Menampilkan heatmap dan tabel untuk Naive Bayes
fig, axes = plt.subplots(2, 2, figsize=(15, 10))
fig.suptitle('Confusion Matrices for Naive Bayes')

for ax, (dataset_name, metrics) in zip(axes.flatten(),
results['Naive Bayes'].items()):
```

```
    sns.heatmap(metrics['confusion_matrix'], annot=True,
fmt='d', cmap='Blues', ax=ax)
    ax.set_title(f'Dataset {dataset_name}')
    ax.set_ylabel('Actual')
    ax.set_xlabel('Predicted')

plt.show()

print("Naive Bayes Evaluation Metrics:")
nb_results = []
for dataset_name, metrics in results['Naive Bayes'].items():
    nb_results.append([dataset_name, metrics['accuracy'],
metrics['precision'], metrics['recall'], metrics['f1']])
nb_df = pd.DataFrame(nb_results, columns=['Dataset', 'Accuracy',
'Precision', 'Recall', 'F1-Score'])
print(nb_df)

# Menampilkan heatmap dan tabel untuk SVM
fig, axes = plt.subplots(2, 2, figsize=(15, 10))
fig.suptitle('Confusion Matrices for SVM')

for ax, (dataset_name, metrics) in zip(axes.flatten(),
results['SVM'].items()):
    sns.heatmap(metrics['confusion_matrix'], annot=True,
fmt='d', cmap='Blues', ax=ax)
    ax.set_title(f'Dataset {dataset_name}')
    ax.set_ylabel('Actual')
    ax.set_xlabel('Predicted')

plt.show()

print("SVM Evaluation Metrics:")
svm_results = []
for dataset_name, metrics in results['SVM'].items():
    svm_results.append([dataset_name, metrics['accuracy'],
metrics['precision'], metrics['recall'], metrics['f1']])
svm_df = pd.DataFrame(svm_results, columns=['Dataset',
'Accuracy', 'Precision', 'Recall', 'F1-Score'])
print(svm_df)
```