Laporan ETS BDA - Case D: Sentiment Analysis IMDB

1. Identitas Mahasiswa

Nama: [Riki Raja purnama]

NIM: [20123004]

Kelas: [a23]

Case: D - IMDB Sentiment Analysis

Link GitHub: [Isi Link Repo]

2. Dataset & Deskripsi Masalah

Dataset IMDB memiliki 50.000 review film dengan dua label sentimen: positive dan negative. Tujuan dari studi kasus ini adalah membangun model machine learning yang mampu mengklasifikasikan sentimen berdasarkan teks ulasan film.

3. Python Modeling

Tahapan yang dilakukan:

- Import dataset dan eksplorasi awal

- Cleaning teks (lowercase, hapus tanda baca, hapus karakter non-huruf)

- Ekstraksi fitur menggunakan TF-IDF dengan 5000 fitur

- Split dataset (80% train, 20% test)

Model baseline: Multinomial Naive Bayes

Model pembanding: Logistic Regression

Hasil evaluasi menunjukkan bahwa Logistic Regression memberikan akurasi yang lebih tinggi. Pada bagian ini, mahasiswa perlu menambahkan screenshot confusion matrix dan classification report.

4. KNIME Workflow

Workflow KNIME terdiri dari:

- CSV Reader

- String Manipulation (lowercase + hapus karakter)

- Strings to Document

- Partitioning

- TF-IDF

- Naive Bayes Learner

- Naive Bayes Predictor

- Scorer

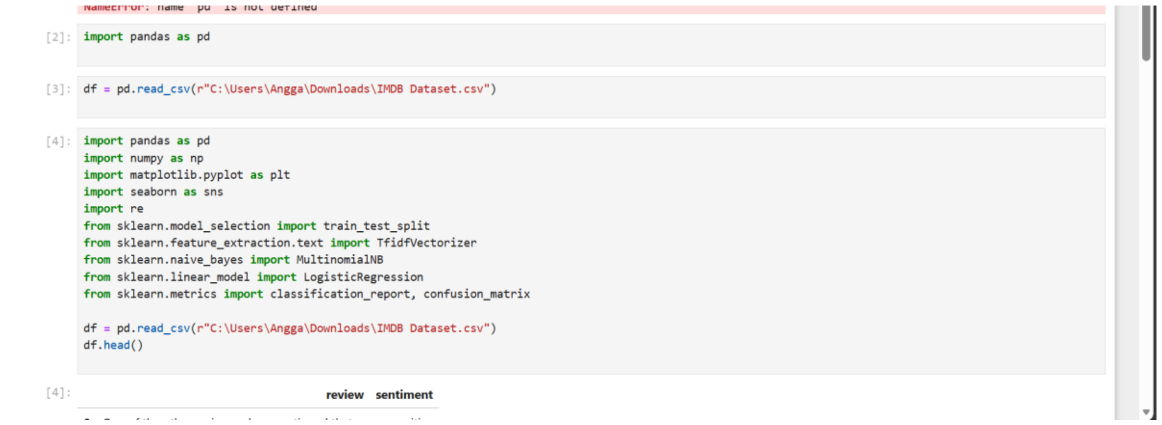
Mahasiswa perlu menambahkan screenshot workflow serta hasil confusion matrix dari node Scorer.

5. Kesimpulan

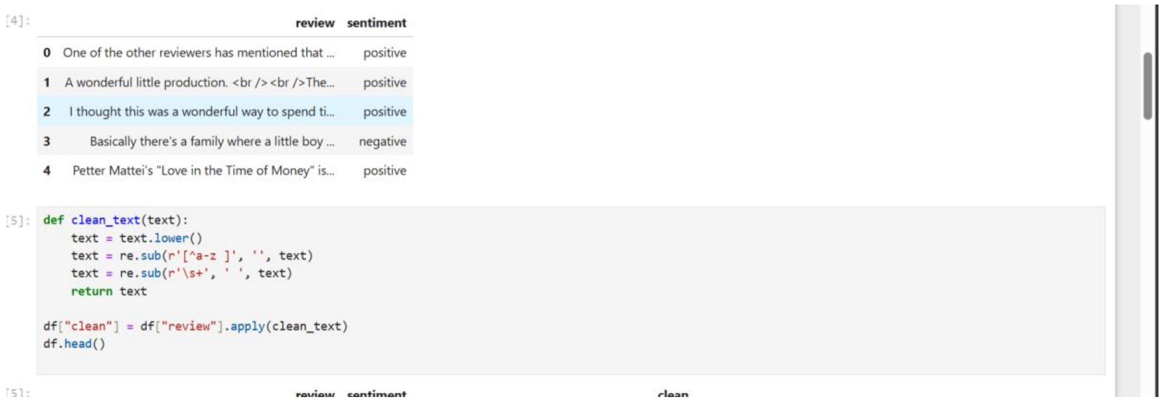
Model terbaik: Logistic Regression.

Alasan: Memiliki akurasi, precision, dan recall lebih tinggi daripada Multinomial Naive Bayes.

Sehingga model ini lebih tepat digunakan untuk sentiment analysis pada dataset IMDB.



```
NameError: name 'pd' is not defined

[2]: import pandas as pd

[3]: df = pd.read_csv(r"C:\Users\Angga\Downloads\IMDB Dataset.csv")

[4]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import re
     from sklearn.model_selection import train_test_split
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.naive_bayes import MultinomialNB
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import classification_report, confusion_matrix

     df = pd.read_csv(r"C:\Users\Angga\Downloads\IMDB Dataset.csv")
     df.head()
```



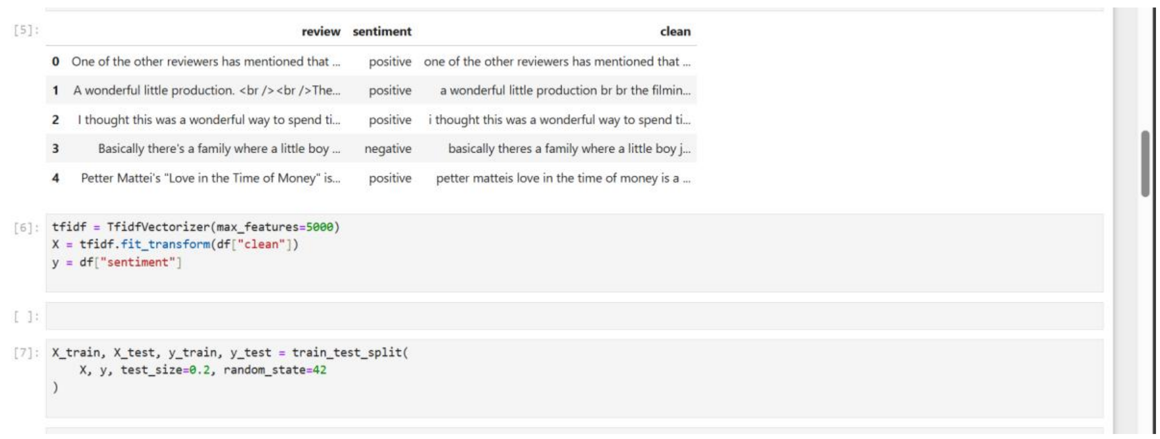
[4]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production.

The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

```
[5]: def clean_text(text):
         text = text.lower()
         text = re.sub(r'[^a-z ]', '', text)
         text = re.sub(r'\s+', ' ', text)
         return text

     df["clean"] = df["review"].apply(clean_text)
     df.head()
```

[5]: review sentiment clean



[5]:

	review	sentiment	clean
0	One of the other reviewers has mentioned that ...	positive	one of the other reviewers has mentioned that ...
1	A wonderful little production. <br /><br />The...	positive	a wonderful little production br br the filmin...
2	I thought this was a wonderful way to spend ti...	positive	i thought this was a wonderful way to spend ti...
3	Basically there's a family where a little boy ...	negative	basically theres a family where a little boy j...
4	Petter Mattei's "Love in the Time of Money" is...	positive	petter matteis love in the time of money is a ...

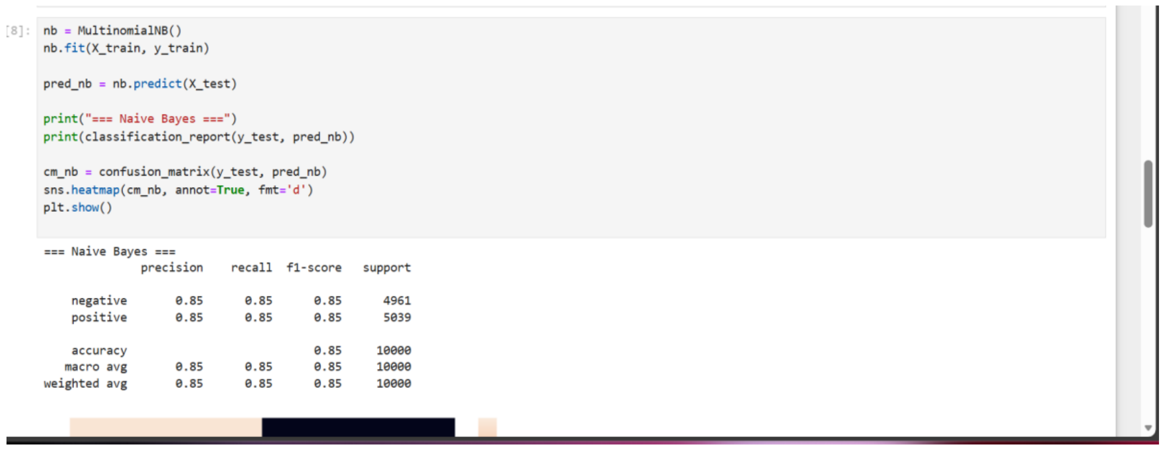
[6]:

```python
tfidf = TfidfVectorizer(max_features=5000)
X = tfidf.fit_transform(df["clean"])
y = df["sentiment"]
```

[ ]:

[7]:

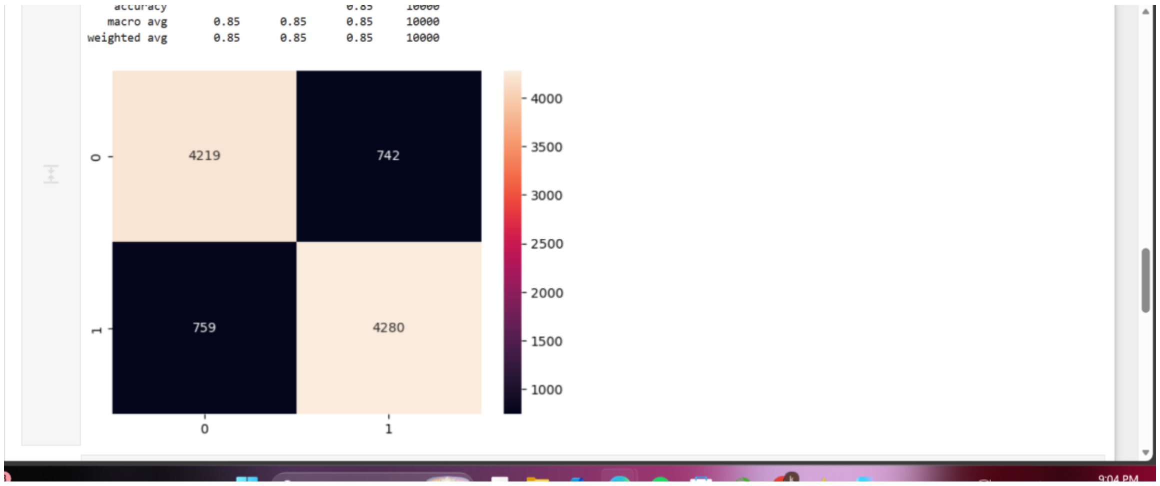
```python
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```



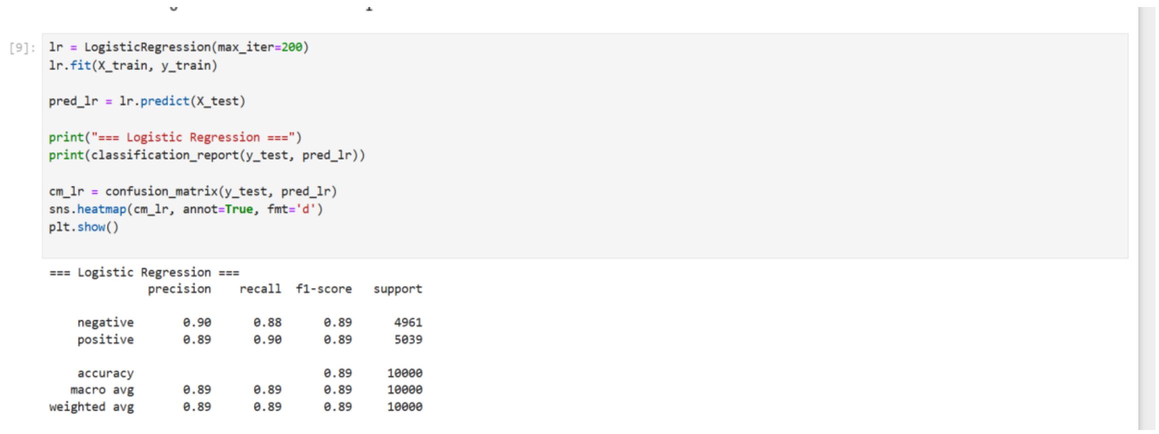
[8]:

```python
nb = MultinomialNB()
nb.fit(X_train, y_train)

pred_nb = nb.predict(X_test)

print("=== Naive Bayes ===")
print(classification_report(y_test, pred_nb))

cm_nb = confusion_matrix(y_test, pred_nb)
sns.heatmap(cm_nb, annot=True, fmt='d')
plt.show()
```

```
=== Naive Bayes ===
              precision    recall  f1-score   support

    negative       0.85      0.85      0.85      4961
    positive       0.85      0.85      0.85      5039

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000
```

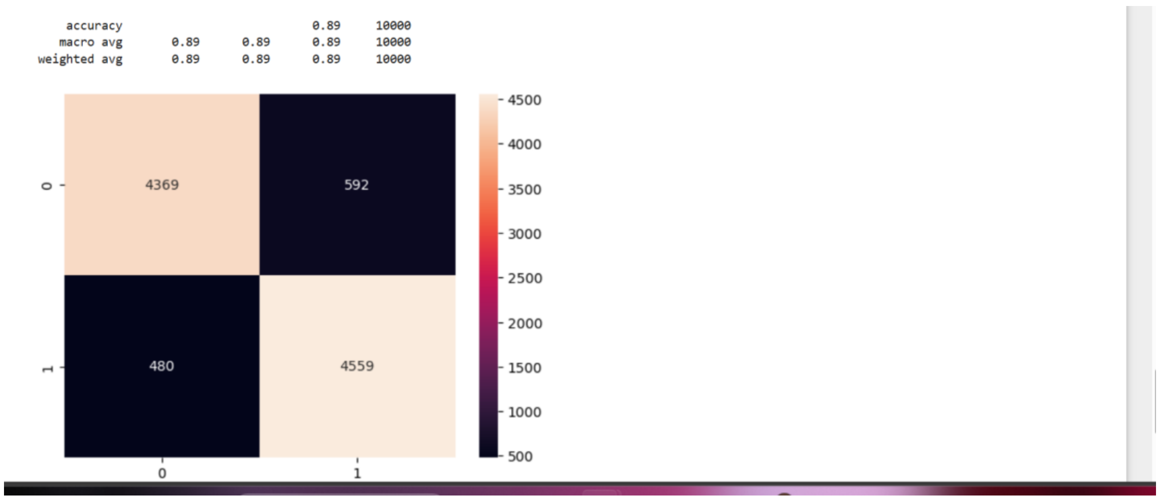


```
    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000
```

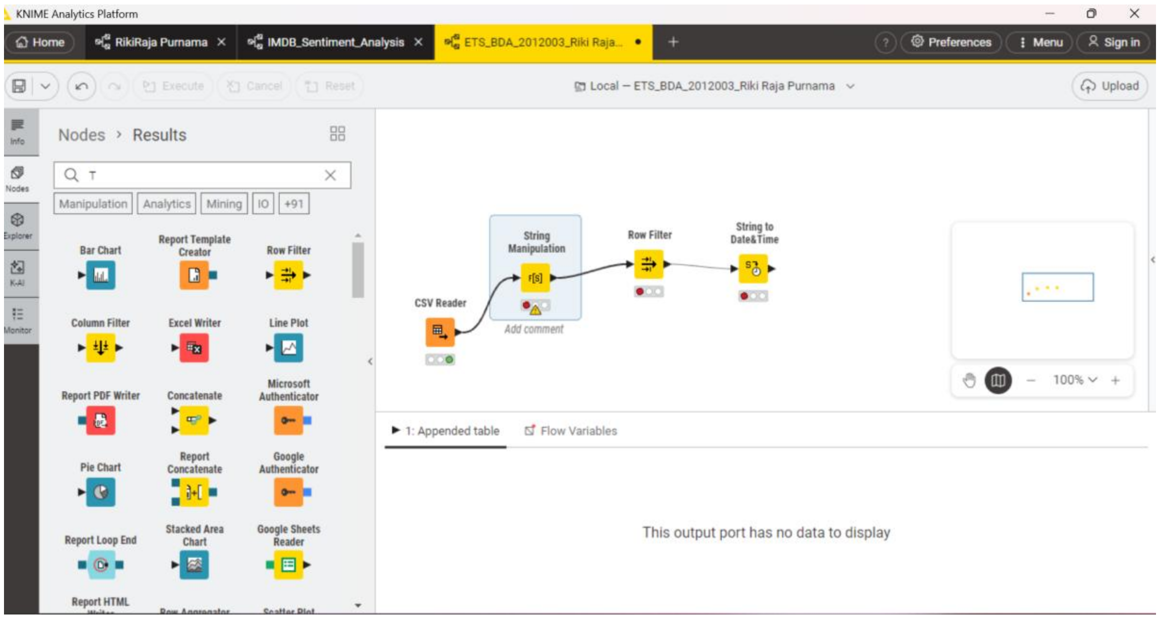


```
[9]: lr = LogisticRegression(max_iter=200)
     lr.fit(X_train, y_train)

     pred_lr = lr.predict(X_test)

     print("=== Logistic Regression ===")
     print(classification_report(y_test, pred_lr))

     cm_lr = confusion_matrix(y_test, pred_lr)
     sns.heatmap(cm_lr, annot=True, fmt='d')
     plt.show()
```

```
=== Logistic Regression ===
              precision    recall  f1-score   support

    negative       0.90      0.88      0.89      4961
    positive       0.89      0.90      0.89      5039

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```



```
    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```



This output port has no data to display