Laporan Analisis *Text Classification* and *Sentiment Analaysis* Menggunakan Naïve Bayes

Text Classification adalah proses mengkategorikan teks ke dalam kelompok terorganisir. Dengan menggunakan Natural Language Processing (NLP), pengklasifikasi teks dapat secara otomatis menganalisis teks dan kemudian menetapkan sekumpulan. Dataset yang digunakan bersumber dari https://github.com/rizalespe/Dataset-Sentimen-Analisis-Bahasa Indonesia/blob/master. Yang terdiri dari 200 sentiment opini film dengan 100 tweet positive dan 100 tweet negative.

1. **Lakukan evaluasi dan analisis hasil klasifikasi berdasarkan Precision, Recall, dan F-Measure.**

```
Accuracy Score : 0.88
Report :
              precision    recall  f1-score   support

    negative       0.84      0.91      0.87        23
    positive       0.92      0.85      0.88        27

    accuracy                           0.88        50
   macro avg       0.88      0.88      0.88        50
weighted avg       0.88      0.88      0.88        50
```

Dengan menggunakan algoritma Naïve Bayes untuk melakukan klasifikasi berhasil mendapatkan *Accuracy* mencapai 88%, *accuracy* adalah rasio prediksi Benar (positif dan negatif) dengan keseluruhan data. *Accuracy yang tinggi* artinya rasio prediksi benarnya sangat baik. *Precision* mencapai 84% untuk tweet negative dan 92% untuk tweet positive yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model, R*ecall* mencapai 91% untuk tweet negative dan 85% untuk tweet positive yang menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. F1-measure atau *F-1 score* yaitu perbandingan rata-rata presisi dan recall yang dibobotkan berhasil mencapai 87% untuk tweet negative dan 88% untuk tweet positive, jika F1-Score punya skor yang baik mengindikasikan bahwa model klasifikasi kita punya precision dan recall yang baik.Berdasarkan hasil F1- Score, *Recall,* dan juga *Precision* yang tinggi mengambarkan semakin efektif sebuah sistem tersebut.

2. Lakukan analisis terhadap kinerja algoritma klasifikasi berdasarkan:

a. Pengaruh persentase data latih dan data uji

1. Split Dataset 70% data latih dan 30% data uji

```
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)
```

\

```
Accuracy Score : 0.8833333333333333
Report :
              precision    recall  f1-score   support

    negative       0.87      0.90      0.88        29
    positive       0.90      0.87      0.89        31

    accuracy                           0.88        60
   macro avg       0.88      0.88      0.88        60
weighted avg       0.88      0.88      0.88        60
```

2.Split Dataset 80% data latih dan 20% data uji

```
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)
```

```
Accuracy Score : 0.925
Report :
              precision    recall  f1-score   support

    negative       0.90      0.95      0.93        20
    positive       0.95      0.90      0.92        20

    accuracy                           0.93        40
   macro avg       0.93      0.93      0.92        40
weighted avg       0.93      0.93      0.92        40
```

Jika dataset yang digunakan 70% data latih dan 30% data uji menghasilkan akurasi sebesar 88,3% sedangkan jika dataset 80% data latih dan 20% data uji menghasilkan akurasi sebesar 92,5% artinya pembagian persentase pada data latih dan data uji berpengaruh terhadap akurasi yang dihasilkan

2. Waktu pelatihan yang diperlukan

a. Logistic Regression

```
%%time
# Create and train a logistic regression
logreg = LogisticRegression(C=1e9, solver='lbfgs', max_iter=1000)
logreg.fit(X, y)
```

```
CPU times: user 27.2 ms, sys: 13 ms, total: 40.2 ms
Wall time: 25.7 ms
```

b. Random Forest

```
%%time
# Create and train a random forest classifier
forest = RandomForestClassifier(n_estimators=50)
forest.fit(X, y)
```

```
CPU times: user 147 ms, sys: 78.9 ms, total: 226 ms
Wall time: 132 ms
```

c. **Linear SVC**

```
%%time
# Create and train a linear support vector classifier (LinearSVC)
svc = LinearSVC()
svc.fit(X, y)
```

```
CPU times: user 5.87 ms, sys: 0 ns, total: 5.87 ms
Wall time: 6.1 ms
```

d. **Naïve Bayes**

```
%%time
# Create and train a multinomial naive bayes classifier (MultinomialNB)
bayes = MultinomialNB()
bayes.fit(X, y)
```

```
CPU times: user 7.57 ms, sys: 124 µs, total: 7.7 ms
Wall time: 7.23 ms
```

Waktu pelatihan yang diperlukan dengan metode Naïve Bayes mencapai 7.7ms sangat baik dibandingkan dengan algoritma Random Forest yang mencapai 40.2ms dan Logistic Regression yang mencapai 226ms. Artinya metode Naïve Bayes memerlukan waktu pelatihan yang lebih cepat dibandingkan algoritma Random Forest dan juga Logistic Regression, Tetapi metode SVC lebih unggul dibanding dengan metode Naïve Bayes karena waktu yang diperlukan relative lebih kecil yaitu 5.87ms

3. Data yang salah diklasifikasikan, jelaskan mengapa data tersebut salah klasifikasi! Lampirkan juga data yang salah diklasifikasikan

	content	pred_bayes	pred_bayes_proba
0	Jelek banget filmnya	negative	0.343761
1	Ini jadi film kesukaan	negative	0.343915
2	filmnya bagus banget	positive	0.646660
3	kecewa sama filmnya	negative	0.240218
4	bosen nonton filmnya	positive	0.585866
5	filmnya keren	positive	0.768228
6	menarik sih buat ditonton	positive	0.742361
7	Seru banget filmnya	positive	0.705564
8	Recommended buat ditonton	positive	0.743204
9	Film ini wajib ditonton	positive	0.726947

Pada tweet data testing dengan content “ini jadi film kesukaan” dan “bosen nonton filmnya” salah klasifikasi, ini terjadi karena adanya kelangkaan data untuk nilai yang mungkin dari suatu fitur perlu memperkirakan nilai kemungkinan dengan pendekatan frequentist. Hal ini

dapat mengakibatkan probabilitas menuju 0 atau 1, yang pada gilirannya menyebabkan ketidakstabilan numerik dan hasil yang lebih buruk. Naive Bayes mengasumsikan bahwa semua prediktor (atau fitur) independen, jarang terjadi dalam kehidupan nyata. Ini membatasi penerapan algoritme ini dalam kasus penggunaan dunia nyata.