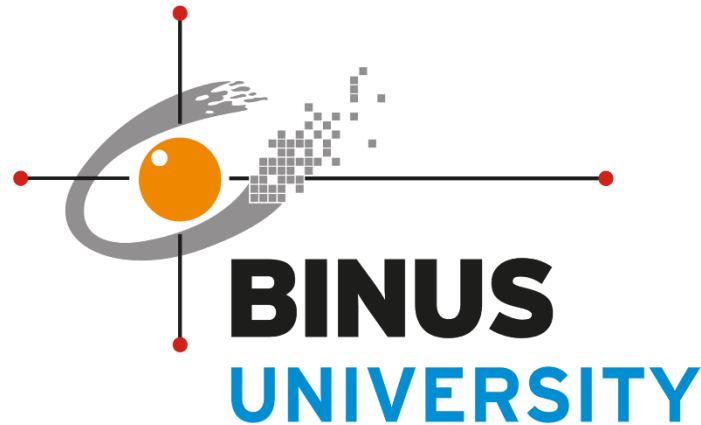


Laporan Analisis Project “Detecting Fake News with Support Vector Machine (SVM), Multinomial Naïve Bayes and Passive Aggressive Classifier”



Anggota Kelompok
2301931251 - Daffa Rizki Rizaly
2301937293 - Edbert Khovey
2301869840 - William Yulio

**School of Computer Science Program
Studi Teknik Informatika
Universitas Bina Nusantara
2020/2021**

Idea Summary

Kami memilih tema project ini, karena ingin berfokus untuk menghilangkan berita-berita palsu(hoax) yang beredar dimasyarakat. Untuk pembuatan model project ini, kami menggunakan algoritma TF-IDF untuk melakukan proses *vectorize* dokumen dataset. Kemudian kami berfokus untuk menggunakan tiga jenis algoritma yaitu Support Vector Machine(SVM), Multinomial Naïve Bayes dan Passive Aggressive Classifier. Dimana ketiga algoritma ini akan kami bandingkan untuk melihat algoritma mana yang paling unggul dalam membedakan berita palsu dan berita asli.

Dataset

Untuk dataset pada project kali ini berasal dari Kaggle yaitu `fake_or_real_news.csv` (29.27 Mb) yang dibuat oleh Hassan Amin dan dataset ini terdiri dari 6060 data. Dataset ini memuat berbagai macam berita yang berasal dari berbagai sumber, didalam dataset ini berita-berita tersebut akan dilabel dengan “fake” apabila berita tersebut merupakan berita palsu dan “real” apabila berita tersebut merupakan berita asli(sesuai dengan faktanya). Dataset ini terdiri dari 4 jenis attribute, number, title, text dan label. Dataset berita ini sudah dibagi secara rata untuk masing-masing labelnya yaitu “fake” dan “real” sebesar 50%.

Berikut adalah link untuk datasetnya : <https://www.kaggle.com/hassanamin/textdb3>

Literature Review

1. TF-IDF Vectorizer

TF-IDF merupakan sebuah metode yang digunakan untuk menghitung jumlah kemunculan kata dalam sebuah dokumen, kemudian memberikan bobot pada setiap kata berdasarkan jumlah kemunculan kata tersebut dalam sebuah dokumen. Perhitungan ini dilakukan untuk melihat seberapa penting kata tersebut didalam sebuah dokumen. TF-IDF terdiri dari dua bagian yaitu *term frequency* (TF) dan *inverse document frequency* (IDF). *Term frequency*(TF) digunakan untuk menghitung seberapa banyak kata tersebut muncul dalam sebuah dokumen. Sedangkan *inverse document frequency* (IDF) memiliki tugas yang bertolak belakang dari TF, dimana IDF akan memberikan bobot yang besar terhadap kata-kata yang memiliki jumlah kemunculan yang sedikit. Sehingga kata-kata yang memiliki makna penting seperti kata kerja, bisa memiliki bobot yang besar meskipun tidak sering muncul didalam dokumen. Rumus *term frequency* dihitung sebagai berikut :

$$TF(t, d) = \frac{n_t}{n}$$

dimana n_t merupakan jumlah berapa kali *term t* muncul dalam sebuah dokumen dan n adalah total jumlah dari *term* di dalam dokumen.

Disisi lain, rumus *inverse document frequency* adalah sebagai berikut :

$$IDF(t) = \frac{N}{df(t)}$$

dimana N merupakan jumlah dokumen yang terdapat term t dan $df(t)$ merupakan Frekuensi dokumen dari sebuah *term* t . Semakin tinggi skor, semakin relevan bahwa kata dalam dokumen tertentu. Berat tiap kata w dalam dokumen d dihitung sebagai berikut :

$$TF\ IDF(t, d) = TF(t, d) * IDF(t)$$

TF-IDF Vectorizer akan mengubah teks dokumen yang semula berbentuk string kedalam bentuk angka yang dapat dimengerti oleh mesin. Proses ini akan menggunakan algoritma yang sudah dijelaskan diatas.

2. Passive Aggressive Classifier

Passive Aggressive Classifier adalah sebuah algoritma yang termasuk kedalam online algorithm, algoritma ini cocok untuk memproses data yang besar karena input sebelumnya tidak akan berdampak terhadap data yang paling baru. Algoritma ini akan tetap *passive* sampai adanya terjadi *miscalculation*, ketika terjadi *miscalculation* maka algoritma ini akan dengan *aggressive* mengubah perhitungannya, tidak seperti algoritma lain yang pelan-pelan berubah. Berikut adalah rumus dasar Passive Aggressive Classifier :

$$\tilde{y}_t = \text{sign}(\bar{w}^T \cdot \bar{x}_t)$$

Algoritma passive aggressive ini berpatokan terhadap *Hinge loss function*, sebagai berikut:

$$L(\bar{\theta}) = \max(0, 1 - y \cdot f(\bar{x}_t; \bar{\theta}))$$

Kemudian ini adalah rumus bagaimana cara Passive Aggressive Classifier melakukan proses update *rule* :

$$\begin{cases} \bar{w}_{t+1} = \underset{\bar{w}}{\operatorname{argmin}} \frac{1}{2} \|\bar{w} - \bar{w}_t\|^2 + C\xi^2 \\ L(\bar{w}; \bar{x}_t, y_t) \leq \xi \end{cases}$$

Karena algoritma ini selalu melakukan proses update *rule*nya maka data input yang paling baru dimasukkan itu akan selalu benar diklasifikasi, tetapi pendekatan yang *aggressive* seperti ini juga memiliki kelemahan dimana algoritma ini mungkin akan membuat keputusan yang tidak optimal untuk data-data sebelumnya sehingga meningkatkan kemungkinan terjadinya salah klasifikasi data.

3. Multinomial Naïve Bayes

Multinomial Naive Bayes algorithm adalah metode pembelajaran yang digunakan dalam natural language processing (NLP). Algoritma ini didasarkan pada Teorema Bayes yang akan memprediksi tag teks seperti sepotong email atau artikel surat kabar. Menghitung probabilitas dari setiap tag untuk contoh yang diberikan dan kemudian memberikan tag dengan probabilitas tertinggi sebagai *output*. Bayes theorem yang dibuat

oleh Thomas Bayes mengkalkulasi probabilitas dari sebuah kejadian terjadi berdasarkan informasi sebelumnya dari kondisi yang berhubungan dari kejadian tersebut. Berikut merupakan formulanya :

$$P(A|B) = P(A) * P(B|A) / P(B)$$

Dimana kita mengkalkulasi probabilitas dari kelas A saat prediktor B sudah disediakan.

$P(B)$ = prior probability dari kelas B

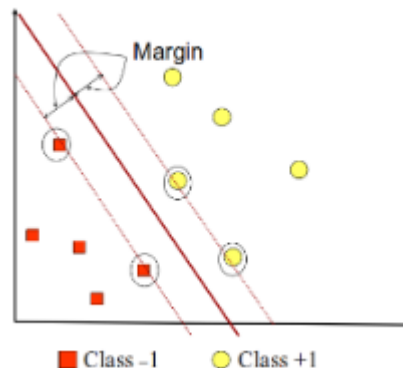
$P(A)$ = prior probability dari kelas A

$P(B|A)$ = terjadinya predictor B saat diberi probabilitas kelas A

Dengan rumus diatas, kita dapat menghitung probabilitas dari sebuah tag di dalam text.

4. Support Vector Machine

Support Vector Machine (SVM) adalah sistem pembelajaran yang menggunakan hipotesis fungsi linear dalam ruang berdimensi tinggi dan dilatih dengan algoritma berdasarkan teori optimasi dengan menerapkan learning bias yang berasal dari teori statistik. Tujuan utama dari metode ini adalah untuk membangun OSH (Optimal Separating Hyperplane), yang membuat fungsi pemisahan optimal yang dapat digunakan untuk klasifikasi.



Gambar 1.Konsep Hyperplane pada SVM

SVM merupakan *binary classifier* yang membagi data menjadi dua class dengan sebuah *hyperplane*. Dapat dilihat dalam gambar 1, *hyperplane* ini tepat berada di tengah-tengah kedua class dengan jarak d ke titik data terdekat untuk masing-masing class. d disebut *margin*, dan titik-titik data yang berada tepat pada jarak d dari *hyperplane* disebut *support vector*. *Hyperplane* SVM dinyatakan dengan persamaan sebagai berikut:

$$w \cdot x + b = 0$$

dimana w merupakan *normal* dari *hyperplane*, dan $\frac{b}{\|w\|}$ adalah jarak *hyperplane* ke titik *origin*. Titik-titik data yang masuk ke class 1 adalah titik-titik data yang memenuhi persamaan $w \cdot x + b \leq -1$ dan titik-titik data yang masuk ke class 2 adalah

titik-titik data yang memenuhi persamaan $w \cdot x + b \geq 1$. Untuk mendapatkan *hyperplane* terbaik, kita harus mencari nilai *hyperplane* yang terletak di tengah-tengah antara dua bidang pembatas kelas. Kemudian untuk mendapatkan *hyperplane* terbaik, sama saja dengan memaksimalkan *margin* atau jarak antara dua set objek dari kelas yang berbeda.

Methodology

Link Code :

<https://colab.research.google.com/drive/17uRtcrrNR6h2pxmVm4gqWMgxk30MTzsH?usp=sharing>

1. Import Library

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.naive_bayes import MultinomialNB
import re
from sklearn import svm
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import timeit
import array
```

- Import numpy = digunakan untuk seluruh kalkulasi array.
- Import pandas = digunakan untuk mengimport dataset csv menjadi dataframe.
- Import train_test_split = digunakan untuk membagi dataset untuk testing dan training model.
- Import TfidfVectorizer = digunakan untuk mengubah dataset yang berisikan kalimat menjadi sekumpulan integer agar dimengerti oleh mesin.
- Import Passive Aggressive Classifier = digunakan untuk menjadi premade model dari sklearn linear model.
- Import MultinomialNB = digunakan untuk menjadi premade model dari sklearn naive bayes.
- Import re = digunakan untuk preprocessing dokumen teks.
- Import svm = digunakan untuk menjadi premade model.

- Import accuracy score dan confusion matrix = digunakan sebagai metrics penilaian untuk menilai model yang kami buat.
- Import sns = digunakan untuk melakukan proses plotting heatmap dari confusion matrix.
- Import timeit = digunakan untuk menghitung waktu yang dibutuhkan sebuah model untuk dilatih.
- Import array = digunakan untuk melakukan proses sorting array waktu yang dibutuhkan ketiga model.

2. Data exploration

```
[2] df = pd.read_csv('/content/fake_or_real_news.csv')
df.fillna(' ')
df.shape
df.head()
```

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg LinkedIn Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Dataset kami ambil dari kaggle.com yang terdiri 6060 dataset dengan 3030 real news dan 3030 fake news.

```
df.isnull().sum()
```

```

↳ Unnamed: 0      0
   title          0
   text           0
   label          0
   dtype: int64

```

Dalam dataset yang kami gunakan tidak terdapat value null didalamnya sehingga kami tidak perlu melakukan proses drop dataset.

3. Preprocessing

```
df["title"]=df["title"].str.lower()
df["text"]=df["text"].str.lower()
df["label"]=df["label"].str.lower()

# remove punctuation
df['text'] = df['text'].apply(lambda x: re.sub('[^\w\s]', ' ', x))
df['title'] = df['title'].apply(lambda x: re.sub('[^\w\s]', ' ', x))

# remove one and two character words
df['text'] = df['text'].apply(lambda x: re.sub(r'\b\w{1,3}\b', '', x))
df['title'] = df['title'].apply(lambda x: re.sub(r'\b\w{1,3}\b', '', x))

# remove numerical values|
df['text'] = df['text'].apply(lambda x: re.sub(r'[0-9]+', '', x))
df['title'] = df['title'].apply(lambda x: re.sub(r'[0-9]+', '', x))

# \s+ means all empty space (\n, \r, \t)
df['text'] = df['text'].apply(lambda x: re.sub('\s+', ' ', x))
df['title'] = df['title'].apply(lambda x: re.sub('\s+', ' ', x))
```

Pada tahapan ini, kami akan melakukan *preprocessing* terhadap dataset yang akan digunakan. Kami akan melakukan *data cleaning* untuk menghasilkan dataset yang optimal sehingga ketika masuk kedalam proses *training* dan *testing*, model dapat menghasilkan output yang sesuai dengan akurasi yang optimal.

Tahapan preprocessing ini dibagi menjadi beberapa bagian, yaitu :

- Mengubah dataset ke dalam bentuk lowercase.
- Menghilangkan punctuation yang ada di dalam dataset.
- Menghilangkan kata-kata yang terdiri dari 1 atau 2 huruf.
- Menghilangkan nilai angka di dalam dataset.
- Menghilangkan empty space seperti 'tab', 'newline', dan 'return'.

4. Split Dataset

```
x_train, x_test, y_train, y_test = train_test_split(
    df['text'], labels, test_size=0.2, random_state=5)
```

Kemudian untuk splitting dataset kita akan melakukan proses *splitting* sebesar 20% untuk data testing dan 80% untuk data training dengan random state 5. Untuk proses split dataset kami menggunakan ukuran seperti ini, agar model dapat belajar dengan baik dengan menggunakan jumlah dataset yang cukup banyak. Sehingga saat melakukan proses klasifikasi akan didapatkan hasil yang optimal.

5. Build Model

a. TF-IDF Vectorizer

```
tfidf_vectorizer=TfidfVectorizer(stop_words='english')

# Fit, and transform train set, transform test set
tfidf_train=tfidf_vectorizer.fit_transform(x_train)
tfidf_test=tfidf_vectorizer.transform(x_test)|
```

TF-IDF Vectorizer digunakan untuk mengubah dataset yang berbentuk tulisan menjadi ke dalam bentuk angka agar dapat dimengerti oleh mesin.

b. Passive Aggressive Classifier

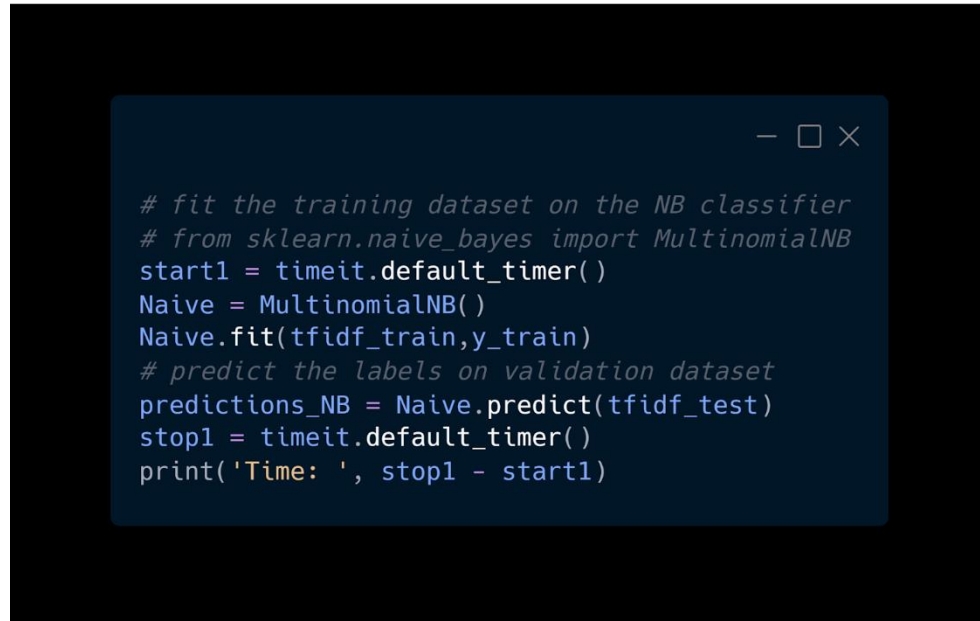
```
# Initialize a PassiveAggressiveClassifier
start = timeit.default_timer()
pac=PassiveAggressiveClassifier(early_stopping= True, verbose=1)
pac.fit(tfidf_train,y_train)

# Predict on the test set
y_pred = pac.predict(tfidf_test)
stop = timeit.default_timer()
print('Time: ', stop - start)
```

Kami membuat model Passive Aggressive Classifier menggunakan library dari sklearn. Setelah proses pembuatan model selesai, kami memasukkan dataset yang sudah di preprocessing sebelumnya dan sudah masuk kedalam TF-IDF Vectorizer. Selanjutnya kami melakukan proses fitting menggunakan model yang sudah dibuat dan melakukan predict dengan menggunakan dataset testing dari TF-IDF Vectorizer. Lalu kami juga menggunakan library timeit untuk mendapatkan waktu fitting dan prediksi dari Passive Aggressive Classifier. Konfigurasi early

stopping juga digunakan untuk mempercepat proses training karena early stopping akan menghentikan proses iterasi jika tidak ada peningkatan dari proses iterasi sebelumnya.

c. Multinomial Naïve Bayes



```
# fit the training dataset on the NB classifier
# from sklearn.naive_bayes import MultinomialNB
start1 = timeit.default_timer()
Naive = MultinomialNB()
Naive.fit(tfidf_train,y_train)
# predict the labels on validation dataset
predictions_NB = Naive.predict(tfidf_test)
stop1 = timeit.default_timer()
print('Time: ', stop1 - start1)
```

Kami membuat model Multinomial Naive Bayes menggunakan library dari sklearn. Setelah proses pembuatan model selesai, kami memasukkan dataset yang sudah di preprocessing sebelumnya dan sudah masuk kedalam TF-IDF Vectorizer. Selanjutnya kami melakukan proses fitting menggunakan model yang sudah dibuat dan melakukan predict dengan menggunakan dataset testing dari TF-IDF Vectorizer. Lalu kami juga menggunakan library timeit untuk mendapatkan waktu fitting dan prediksi dari Multinomial Naive Bayes.

d. Support Vector Machine

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
start2 = timeit.default_timer()
SVM = svm.SVC()
SVM.fit(tfidf_train,y_train)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(tfidf_test)
stop2 = timeit.default_timer()
print('Time: ', stop2 - start2)
```

Kami membuat model Support Vector Machine menggunakan library dari sklearn. Setelah proses pembuatan model selesai, kami memasukkan dataset yang sudah di preprocessing sebelumnya dan sudah masuk kedalam TF-IDF Vectorizer. Selanjutnya kami melakukan proses fitting menggunakan model yang sudah dibuat dan melakukan predict dengan menggunakan dataset testing dari TF-IDF Vectorizer. Lalu kami juga menggunakan library timeit untuk mendapatkan waktu fitting dan prediksi dari Support Vector Machine.

6. Evaluation

No.	Evaluation Type	Passive Aggressive Classifier	Multinomial Naive Bayes	Support Vector Machine
1.	Accuracy	94.24 %	82.64 %	93.21 %
2.	Time	0.127 sec	0.035 sec	61.561 sec

Kami melakukan perbandingan antara ketiga algoritma ini dari segi akurasi dan waktu. Dari hasil evaluasi ini dapat dilihat bahwa algoritma Passive Aggressive unggul dalam kedua jenis evaluasi, kemudian pada posisi kedua diduduki oleh algoritma SVM dengan nilai akurasi yang cukup tinggi namun membutuhkan waktu komputasi yang relatif lama dibandingkan dengan algoritma lainnya. Selanjutnya posisi terakhir diduduki oleh Multinomial Naive Bayes, karena memiliki hasil akurasi model yang cukup rendah dibandingkan dengan model lainnya walaupun memakan waktu komputasi yang cukup singkat. Kami juga akan menampilkan confusion matrix dari setiap algoritma dan didapatkan hasil sebagai berikut :

Keterangan :

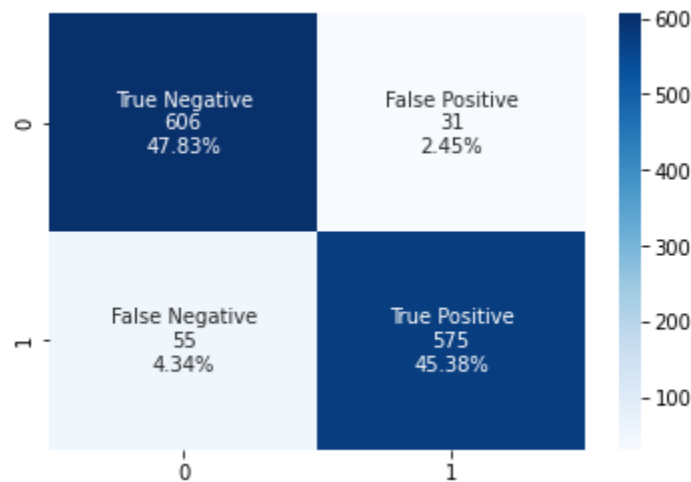
True Negative = Prediksi yang model tebak sebagai kelas negative(label = fake) dan dataset asli menunjukkan bahwa data tersebut termasuk kedalam kelas negative (label = fake)

True Positive = Prediksi yang model tebak sebagai kelas positive(label = real) dan dataset asli menunjukkan bahwa data tersebut termasuk kedalam kelas positive (label = real)

False Negative = Prediksi yang model tebak sebagai kelas negative(label = fake) namun dataset asli menunjukkan bahwa data tersebut termasuk kedalam kelas positive (label = real)

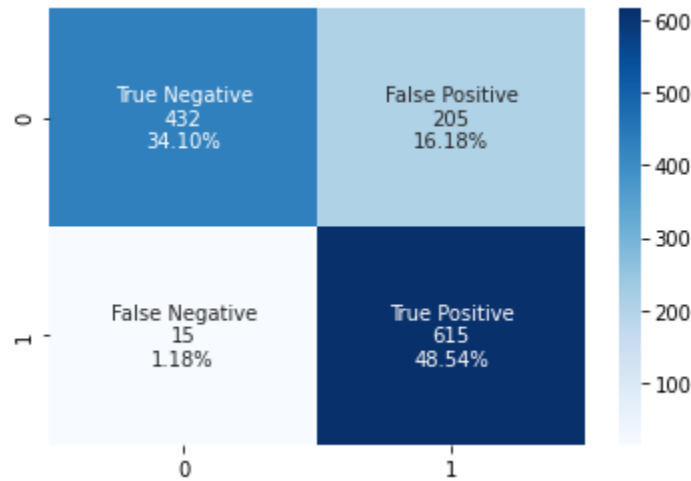
False Positive = Prediksi yang model tebak sebagai kelas positive(label = real) namun dataset asli menunjukkan bahwa data tersebut termasuk kedalam kelas negative (label = fake)

Support Vector Machine Confusion Matrix



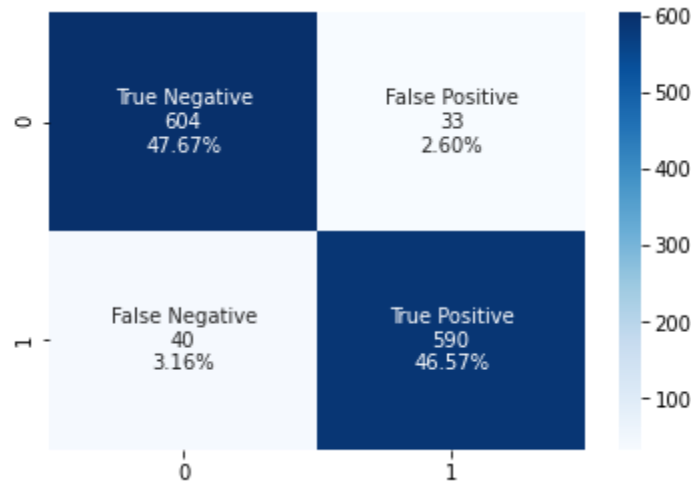
Dari 100% test data(1,267), 47.83% True Negative, 45.38 True Positive, 2.45% False Positive, dan 4.34% False Negative.

Multinomial Naive Bayes Confusion Matrix



Dari 100% test data(1,267), 34.10% itu True Negative, 48.54 True Positive, 16.18% False Positive, dan 1.18% False Negative.

Passive Aggressive algorithm



Dari 100% test data(1,267), 47.67% itu True Negative, 46.57 True Positive, 2.60% False Positive, dan 3.16% False Negative.

7. Conclusion

Berdasarkan percobaan kami, Passive Aggressive Classifier merupakan model yang paling optimal untuk digunakan dalam melakukan proses text classification untuk project ini dibandingkan dengan algoritma Support Vector Machine dan Multinomial Naïve Bayes. Karena Passive Aggressive Classifier ini mempunyai kecepatan komputasi yang baik meskipun tidak secepat Naive Bayes, tapi runtime passive aggressive masih dibawah 1 detik yang menurut kami masih termasuk kedalam runtime yang cukup cepat. Kemudian akurasi yang dihasilkan Passive Aggressive Classifier juga mengeluarkan persentasi yang sangat bagus yaitu 94.24%

dibandingkan dengan algoritma Support Vector Machine yang memakan waktu sangat lama dan memiliki akurasi yang sedikit lebih rendah dibanding Passive Aggressive Classifier.

Daftar Pustaka

- [1] Xu, S., Li, Y., & Wang, Z. (2017). Bayesian Multinomial Naïve Bayes Classifier to Text Classification. *Lecture Notes in Electrical Engineering*, 347–352.
https://doi.org/10.1007/978-981-10-5041-1_57
- [2] Horne, B., & Sibel Adali. (2017). This Just In: Fake News Packs A Lot In Title, Uses Simpler, Repetitive Content in Text Body, More Similar To Satire Than Real News. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1). Retrieved from <https://ojs.aaai.org/index.php/ICWSM/article/view/14976>
- [3] Gupta, S., & Meel, P. (2020). Fake News Detection Using Passive-Aggressive Classifier. *Lecture Notes in Networks and Systems*, 155–164.
https://doi.org/10.1007/978-981-15-7345-3_13
- [4] Octaviani, P., Wilandari, Y., Ispriyanti, D., Statistika, M., Undip, F., Pengajar, S., & Statistika, J. (2014). *PENERAPAN METODE KLASIFIKASI SUPPORT VECTOR MACHINE (SVM) PADA DATA AKREDITASI SEKOLAH DASAR (SD) DI KABUPATEN MAGELANG*. 3(4), 811–820. Retrieved from <https://media.neliti.com/media/publications/137948-ID-penerapan-metode-klasifikasi-support-vec.pdf>