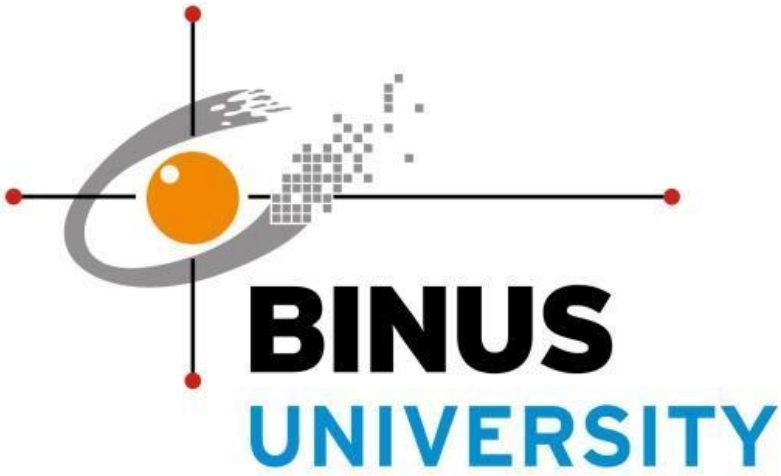
# Laporan Project Natural Language Processing




# Hate Speech Sentiment Analysis Model

Oleh:

| | | |
|---|---|---|
| Louis | 2602147276 | Computer Science |
| Kelvin Andreas | 2602121794 | Computer Science |
| Hendrik Nicolas Carlo | 2602124171 | Computer Science |
| Janssen Mitchellano Hamaziah | 2602117525 | Computer Science |

**UNIVERSITAS BINA NUSANTARA JAKARTA**

**2024**

## A. Deskripsi Dataset

Dataset yang digunakan dalam proyek Natural Language Processing kali ini bersumber dari Kaggle. Terdapat dua dataset, yaitu:

1. Hatespeech Dataset: berisi tweet dan label biner yang menandakan apakah sebuah tweet mengandung ujaran kebencian atau tidak.
2. Abbreviation Dataset: berisi kata-kata yang disingkat, seperti: ASAP (As Soon As Possible)

## B. Exploratory Data Analysis

1. **Example of Hate Speech Tweet**

| | Content | Label | Content_int |
|---|---|---|---|
| 0 | denial of normal the con be asked to comment o... | 1 | [146715, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,... |
| 1 | just by being able to tweet this insufferable ... | 1 | [146715, 14, 15, 16, 17, 7, 18, 19, 20, 21, 22... |
| 2 | that is retarded you too cute to be single tha... | 1 | [146715, 28, 29, 30, 26, 31, 32, 7, 5, 33, 28,... |
| 3 | thought of a real badass mongol style declarat... | 1 | [146715, 35, 1, 24, 36, 37, 38, 39, 40, 1, 41,... |
| 4 | afro american basho | 1 | [146715, 46, 47, 48, 146714] |

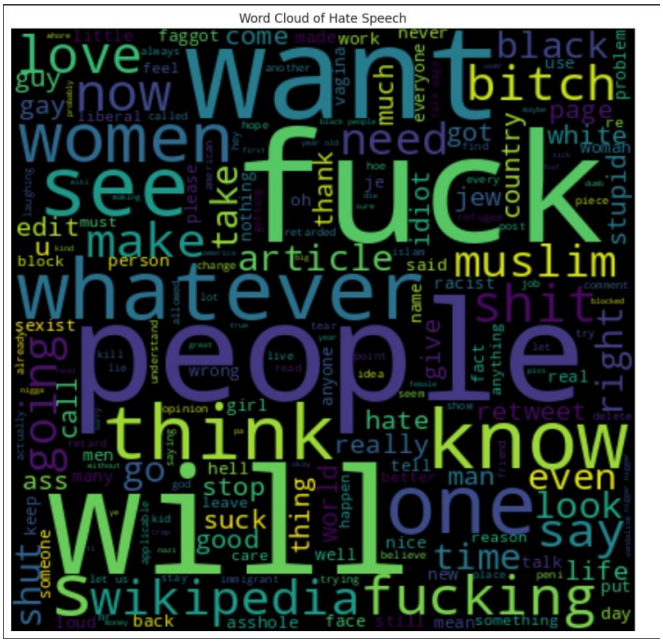
2. **Example of Non-Hate Speech Tweet**

| | Content | Label | Content_int |
|---|---|---|---|
| 502 | simply copy and paste the following text into ... | 0 | [146715, 1805, 1806, 111, 1807, 3, 1808, 1809,... |
| 503 | in order to help increase the booklets downloa... | 0 | [146715, 94, 1814, 7, 342, 1815, 3, 1816, 1817... |
| 504 | as of the booklet had been downloaded over tim... | 0 | [146715, 273, 1, 3, 1822, 998, 284, 1823, 706,... |
| 506 | click on the download my bad green banner link | 0 | [146715, 1825, 9, 3, 1827, 129, 1551, 1838, 18... |
| 507 | booklet updated on | 0 | [146715, 1822, 1840, 9, 146714] |

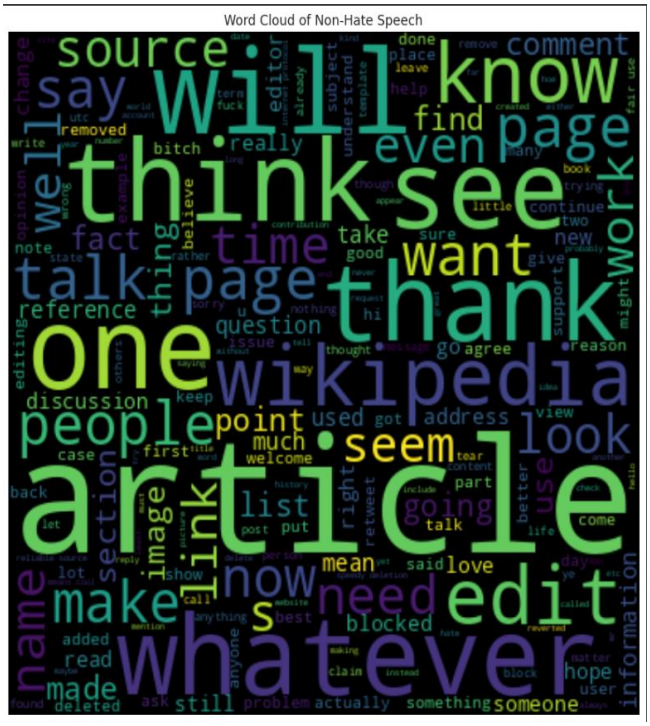
3. **Word Cloud**

Word cloud adalah salah satu cara untuk menggambarkan kata-kata yang sering muncul dalam sebuah dataset.

- Word cloud dari hate speech




- Word cloud dari non hate speech




## 4. Number of Rows

```
print("Jumlah baris:", tweet_df.shape[0])
print("Jenis Label:", tweet_df['Label'].unique())
```

```
Jumlah baris: 384098
Jenis Label: ['1' '0' 'Label']
```

Terlihat bahwa data tersebut terdiri atas 384098 baris dan untuk labelnya sendiri terdapat 3 jenis, yaitu;

- 0 : Non Hate Speech
- 1: Hate Speech
- Label: Label yang anomali

**5. Dataset Info and Describe**

```
tweet_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 384098 entries, 0 to 384097
Data columns (total 3 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Content      384098 non-null  object
 1   Label        384098 non-null  object
 2   Content_int  384098 non-null  object
dtypes: object(3)
memory usage: 8.8+ MB
```

```
tweet_df.describe()
```

| | Content | Label | Content_int |
|---|---|---|---|
| count | 384098 | 384098 | 384098 |
| unique | 364412 | 3 | 364412 |
| top | content | 0 | [146715, 7139, 146714] |
| freq | 7 | 309759 | 7 |

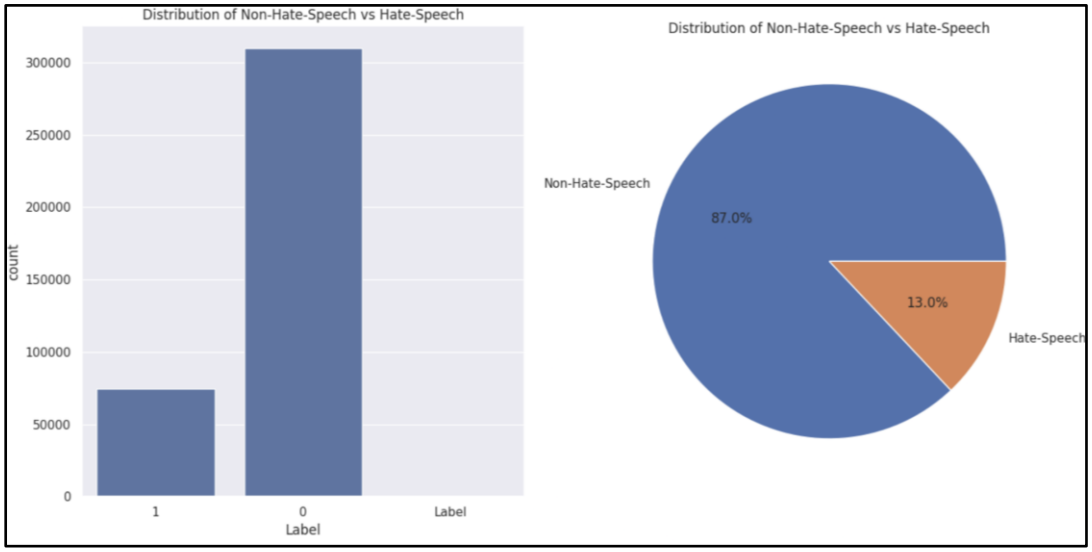
Dari informasi dan describe tersebut, kita dapat mengetahui bahwa:

- Terdapat 3 kolom, yaitu Content (berisi tweet), Label(menandakan hate speech atau bukan), dan Content_int
- Tidak terdapat baris data yang memiliki null value
- Terdapat data unik untuk content sebanyak 364412 baris, sedangkan total data untuk content sebanyak 384098 baris. Dimana hal ini berarti bahwa terdapat data yang duplikat.

**6. Distribution of Dataset**

Mengetahui distribusi dari dataset akan membantu kita dalam memahami apakah dataset tersebut seimbang atau tidak. Pada step ini, kita memperoleh informasi bahwa distribusinya adalah

| Label | Jumlah data |
|---|---|
| “0” | 309759 |
| “1” | 74332 |
| “Label” | 7 |




**7. Check Duplicate and Conflict Data**

Pada step ini kita akan memeriksa jumlah data yang duplikat dan konflik. Konflik yang dimaksud disini adalah data yang memiliki content sama, tetapi memiliki 2 buah label yang berbeda. Pengecekan ini bertujuan agar kita bisa melakukan pembersihan data sebelum modeling.

```
print("Jumlah baris duplikat:", tweet_df.duplicated().sum())
```

```
Jumlah baris duplikat: 18835
```

```
Contoh yang konflik: Content
a kitten for you because heavy is clearly a pussy that just tries to because trouble
a message to you shaded hey i think you are a follow n gg
a question for elf why is it you persist in deleting my delete on the page about nothing for a page to be about nothing it must
a reminder you are late on the rent put thanks for trading the salem page f you to stalker page la rent put made
a spade is a spade do not get your panties in a twisted bunch or bees in your bonnet over it

your psychological problems some advice dude i am looking over your extensive edit history and i want to share some guidance tha
your recent misguided deletion i created an article for the first fucking time with many references and you deleted the article
your sock puppet s account is blocked forever has been blocked indefinitely i do not understand how a muslim like you can lie in
your types you did not like my comments on your type do not be a cry baby ok you break in my hard hat
zionist terrorism why are you following every time and are aligned with your jewish friends i do not know how they let you becom
Name: Label, Length: 851, dtype: int64
Total 'Content' dengan label yang berbeda: 851
```

| Content | Label |
|---|---|
| a message to you shaded hey i think you are a follow n gg | 1 |
| a message to you shaded hey i think you are a follow n gg | 0 |

Diketahui bahwa terdapat 18835 baris data yang duplikat dan 851 baris data yang konflik. Hal ini akan kita handle dipreprocessing nanti untuk hilangkan agar bisa memperoleh dataset yang lebih bersih dan konsisten.

## C. Preprocessing

**1. Remove “Content_int” Column**

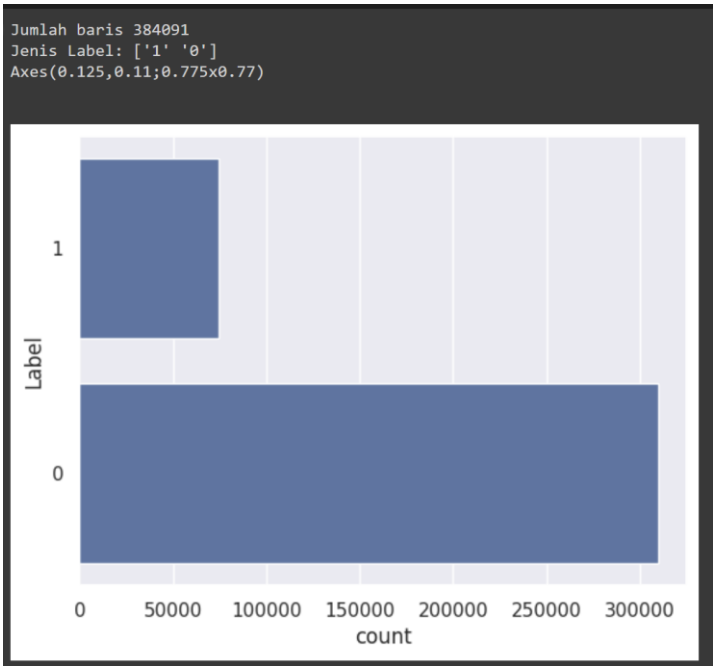
Menghapus kolom Content_int agar dataset yang digunakan untuk modeling nanti hanya berisi kolom-kolom yang diperlukan saja.

| | Content | Label |
|---|---|---|
| 0 | denial of normal the con be asked to comment o... | 1 |
| 1 | just by being able to tweet this insufferable ... | 1 |
| 2 | that is retarded you too cute to be single tha... | 1 |
| 3 | thought of a real badass mongol style declarat... | 1 |
| 4 | afro american basho | 1 |

**2. Remove Label Other Than 1 or 0**

Ketika EDA tadi kita memperoleh informasi bahwa terdapat label selain 1 atau 0. Oleh karena itu, akan kita hilangkan label tersebut sehingga memperoleh hasil sebagai berikut.

```
Jumlah baris 384091
Jenis Label: ['1' '0']
Axes(0.125,0.11;0.775x0.77)
```




**3. Drop Duplicate Row**

Menghapus data yang duplikat sehingga memperoleh hasil sebagai berikut.

```
tweet_df = tweet_df.drop_duplicates()
print("Jumlah baris duplikat:", tweet_df.duplicated().sum())
```

```
Jumlah baris duplikat: 0
```

4. **Drop Conflict Row**

Menghapus data yang konflik sehingga memperoleh hasil sebagai berikut.

```
conflicted_content_list = conflicted_contents.index.tolist()
tweet_df = tweet_df[~tweet_df['Content'].isin(conflicted_content_list)]

grouped = tweet_df.groupby('Content')['Label'].nunique()
conflicted_contents = grouped[grouped > 1]
print("Total 'Content' dengan label yang berbeda:", conflicted_contents.count())
```

```
Total 'Content' dengan label yang berbeda: 0
```

5. **Handle Imbalance Class**

Pada EDA tadi kita memperoleh informasi bahwa dataset tersebut imbalance, dimana dataset dengan label “0” lebih banyak dibanding dataset dengan label “1”. Tujuan dari menghandle imblance class adalah agar model yang dihasilkan tidak bisa terhadap kelas mayoritas.

Untuk menghandle imbalance class, kita akan menggunakan Under Sampling yang merupakan salah satu cara dalam menghandle imbalance class dengan mengurangi jumlah sampel dari kelas yang mayoritas agar memiliki jumlah yang setara dengan jumlah sampel dari kelas minoritas.

Alasan kami lebih memilih Under Sampling dibanding Over Sampling adalah untuk menghindari overfitting, karena seperti yang kita ketahui bahwa Over Sampling akan menghasilkan sampel baru dengan cara menduplikasi. Hal ini beresiko menyebabkan overfitting apalagi untuk case ini, kelas minoritas memiliki jumlah yang sangat kecil hanya sekitar 13% dari total keseluruhan. Alasan lainnya adalah agar komputasi yang dilakukan lebih ringan karena jumlah dataset yang dihasilkan akan lebih sedikit dibanding menggunakan Over Sampling.

```
print("Distribusi label sebelum undersampling:")
print(tweet_df['Label'].value_counts())

rus = RandomUnderSampler(random_state=42)
content_resampled, labels_resampled = rus.fit_resample(tweet_df[['Content']], tweet_df['Label'])

print("\nDistribusi label setelah undersampling:")
print(pd.Series(labels_resampled).value_counts())

tweet_df_resampled = pd.DataFrame({'Content': content_resampled.squeeze(), 'Label': labels_resampled})
```

```
Distribusi label sebelum undersampling:
Label
0    292633
1     70927
Name: count, dtype: int64

Distribusi label setelah undersampling:
Label
0    70927
1    70927
Name: count, dtype: int64
```

6. **Fix Slang Word**

Slang word merupakan kata-kata informal/tidak baku. Pada step ini kita akan melakukan proses mengubah kata-kata tidak baku tersebut menjadi bentuk baku. Hal ini bertujuan untuk meningkatkan kualitas data teks yang akan digunakan untuk melatih model, dimana akan berdampak pada kualitas dan akurasi dari model.

| | Content | Label | Fix_Slang |
|---|---|---|---|
| 0 | you currently appear to be engaged in an edit ... | 0 | you currently appear to be engaged in an edit ... |
| 1 | i do not know why you people think america wil... | 0 | i do not know why you people think america wil... |
| 2 | full protection of article excessive i have be... | 0 | full protection of article excessive i have be... |
| 3 | i added rows to the popular vote table to refl... | 0 | i added rows to the popular vote table to refl... |
| 4 | arbor notification as you participated in the ... | 0 | arbor notification as you participated in the ... |

7. **Fix Typo**

Pada tahap ini kita akan memperbaiki kata-kata yang typo agar menjadi kata yang benar. Hal ini bertujuan untuk kualitas data teks yang akan digunakan untuk melatih model, dimana akan berdampak pada kualitas dan akurasi dari model.

| | Content | Label | Fix_Slang | Fix_Typo |
|---|---|---|---|---|
| 0 | you currently appear to be engaged in an edit ... | 0 | you currently appear to be engaged in an edit ... | you currently appear to be engaged in an edit ... |
| 1 | i do not know why you people think america wil... | 0 | i do not know why you people think america wil... | i do not know why you people think america wil... |
| 2 | full protection of article excessive i have be... | 0 | full protection of article excessive i have be... | full protection of article excessive i have be... |
| 3 | i added rows to the popular vote table to refl... | 0 | i added rows to the popular vote table to refl... | i added rows to the popular vote table to refl... |
| 4 | arbor notification as you participated in the ... | 0 | arbor notification as you participated in the ... | arbor notification as you participated in the ... |

8. **Cleaning Text**

Pada tahap ini kita akan membersihkan teks agar menjadi lebih bersih. Hal ini bertujuan untuk meningkatkan kualitas data teks yang akan digunakan untuk melatih model, dimana akan berdampak pada kualitas dan akurasi model.

| Fix_Typo | Case_folded |
|---|---|
| you currently appear to be engaged in an edit ... | you currently appear to be engaged in an edit ... |
| i do not know why you people think america wil... | i do not know why you people think america wil... |
| full protection of article excessive i have be... | full protection of article excessive i have be... |
| i added rows to the popular vote table to refl... | i added rows to the popular vote table to refl... |
| arbor notification as you participated in the ... | arbor notification as you participated in the ... |

9. **Text Tokenize**

Pada tahap ini kita akan memecah teks tersebut menjadi token-token kecil. Hal ini bertujuan untuk mempersiapkan data agar bisa diproses dan diolah secara individu di step selanjutnya.

| Case_folded | Tokenized |
|---|---|
| you currently appear to be engaged in an edit ... | [you, currently, appear, to, be, engaged, in, ... |
| i do not know why you people think america wil... | [i, do, not, know, why, you, people, think, am... |
| full protection of article excessive i have be... | [full, protection, of, article, excessive, i, ... |
| i added rows to the popular vote table to refl... | [i, added, rows, to, the, popular, vote, table... |
| arbor notification as you participated in the ... | [arbor, notification, as, you, participated, i... |

**10. Lemmatizing**

Lemmatizing merupakan proses dimana kita mengubah sebuah token menjadi bentuk dasarnya dengan mempertimbangkan konteks dan bagian dari tata bahasa.

| Tokenized | Lemma |
|---|---|
| [you, currently, appear, to, be, engaged, in, ... | [you, currently, appear, to, be, engaged, in, ... |
| [i, do, not, know, why, you, people, think, am... | [i, do, not, know, why, you, people, think, am... |
| [full, protection, of, article, excessive, i, ... | [full, protection, of, article, excessive, i, ... |
| [i, added, rows, to, the, popular, vote, table... | [i, added, row, to, the, popular, vote, table,... |
| [arbor, notification, as, you, participated, i... | [arbor, notification, a, you, participated, in... |

**11. Remove Stopwords**

Merupakan tahap dimana kita akan menghapus stopwords/kata-kata umum yang sering muncul dan tidak memiliki kontribusi yang signifikan dalam pemrosesan teks.

| Lemma | No_StopWord |
|---|---|
| [you, currently, appear, to, be, engaged, in, ... | [currently, appear, engaged, edit, war, accord... |
| [i, do, not, know, why, you, people, think, am... | [know, people, think, america, change, always,... |
| [full, protection, of, article, excessive, i, ... | [full, protection, article, excessive, wikiped... |
| [i, added, row, to, the, popular, vote, table,... | [added, row, popular, vote, table, reflect, es... |
| [arbor, notification, a, you, participated, in... | [arbor, notification, participated, eye, threa... |

**12. Combining to One String**

Setelah melakukan preprocessing, kita akan menggabungkan token-token tadi kembali menjadi sebuah string yang utuh agar dapat digunakan untuk analisis lebih lanjut.

| No_StopWord | Clean |
|---|---|
| [currently, appear, engaged, edit, war, accord... | currently appear engaged edit war according re... |
| [know, people, think, america, change, always,... | know people think america change always racist... |
| [full, protection, article, excessive, wikiped... | full protection article excessive wikipedia tw... |
| [added, row, popular, vote, table, reflect, es... | added row popular vote table reflect estimate ... |
| [arbor, notification, participated, eye, threa... | arbor notification participated eye thread led... |

**13. Combining Cleaned Text and Labels to New DataFrame**

Setelah digabung kita akan membuat Data Frame baru yang berisi teks yang telah dibersihkan dan labelnya. Hal ini dilakukan untuk menjaga keteraturan dan menghindari tercampurnya data yang telah diproses dengan data yang masih mentah.

| | Clean | Label |
|---|---|---|
| 0 | current appear engag edit war accord revert ma... | 0 |
| 1 | know whi peopl think america chang becaus alwa... | 0 |
| 2 | full protect articl excess wikipedia two year ... | 0 |
| 3 | ad row popular vote tabl reflect estim michiga... | 0 |
| 4 | arbor notif particip eye thread led request se... | 0 |

## D. Feature Extraction and Modeling

### 1. Machine Learning

#### a. Splitting Data

```
train, test = train_test_split(tweet_df, test_size=0.3, stratify=tweet_df['Label'], random_state=42)

X_train = train['Clean'].values.astype('U')
y_train = train['Label']

X_test = test['Clean'].values.astype('U')
y_test = test['Label']

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(99297,)
(42557,)
(99297,)
(42557,)
```

Pada tahap ini kita melakukan splitting untuk train set dan test set dari tweet_df yang sudah dipreprocessing tadi. Dari hasil code diatas dapat disimpulkan bahwa training set 0.7 dan test set 0.3 dengan menggunakan data random.

**b. Feature Extraction**

```
tfidf_vectorizer = TfidfVectorizer()
X_train_vect = tfidf_vectorizer.fit_transform(X_train)

X_test_vect = tfidf_vectorizer.transform(X_test)
```

Pada tahap feature extraction kita menggunakan TF-IDF sebelum melakukan pemodelan menggunakan machine learning. TF-IDF (Term Frequency-Inverse Document Frequency) merupakan sebuah metode yang mengubah teks menjadi sebuah vektor dimana setiap elemen merepresentasikan seberapa sering suatu kata muncul dalam suatu dokumen. Selain itu TF-IDF dapat membedakan pada kata yang sering muncul namun kurang informatif dan kata yang jarang muncul namun sangat informatif. Karena data pada Sentiment Analysis banyak dan cenderung unik maka, menurut kami sangat cocok apabila dilakukan TF-IDF untuk bagian feature extraction.

Pentingnya melakukan feature extraction karena model machine learning membutuhkan data yang bersifat numerik untuk membuat suatu prediksi apakah text tersebut positif atau negatif. Karena pada dasarnya model machine learning tidak bisa memahami raw text atau text secara langsung. Maka dari itu perlu melakukan TF-IDF sebelum ke tahap train model menggunakan metode-metode machine learning

$$tf(t,d) = \frac{Jumlah\ kemunculan\ kata\ t\ dalam\ dokumen\ d}{Jumlah\ total\ kata\ dalam\ dokumen\ d}$$

$$idf(t) = log\frac{Total\ jumlah\ dokumen\ dalam\ d}{Jumlah\ dokumen\ di\ mana\ kata\ t\ muncul}$$

TF-IDF juga sangat membantu dalam mengurangi dimensi data serta relevansi antar feature. TF-IDF membantu menentukan seberapa umum sebuah kata. Jadi jika sebuah kata sering muncul, maka kata tersebut memiliki IDF yang rendah. Dengan demikian kata tersebut bisa dikatakan kata yang cukup umum. Sebagai contoh di dataset kita seperti “the”, “is”, “he”, dll. Kata kata tersebut umum dan cenderung kurang penting untuk sebuah prediksi sentiment. Jadi dengan demikian akan membantu model untuk lebih fokus pada kata kata yang unik untuk meningkatkan akurasi prediksi yang dihasilkan oleh model. Kata kata yang kurang berguna akan memiliki bobot yang rendah sedangkan kata kata yang penting akan memiliki bobot yang lebih tinggi. Sehingga ketika train model akan sedikit menerima noise dan banyak gain informasi informasi yang penting untuk mengatasi sebuah overfitting pada model.

Selanjutnya kita melakukan fit_transform pada train data dan transform pada test data. Secara singkat fit digunakan untuk train model untuk menemukan kata yang unik dalam data dan transform digunakan untuk data dalam bentuk text menjadi data berbentuk numerik. Yang pada akhirnya nanti digunakan untuk melakukan training oleh model machine learning. Sama seperti yang disebutkan diatas dimana model machine learning tidak dapat menangkap data text, maka dari itu perlu di convert ke dalam data numerik untuk menghasilkan kalkulasi prediksi.

**c. Modeling**

1) **Support Vector Machine (SVM)**

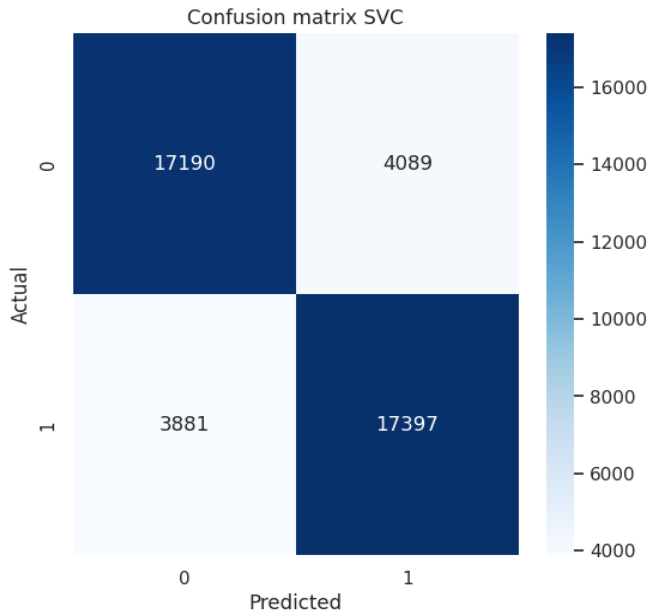
```
svc = SVC(kernel='linear')
svc.fit(X_train_vect, y_train)
y_pred_svc = svc.predict(X_test_vect)
```

SVM model merupakan salah satu model klasifikasi yang cukup bagus. SVM bekerja dengan menemukan hyperplane yang paling optimal untuk memisahkan dua kelas. Pada kasus ini SVM jarang mengalami overfitting

dibandingkan dengan model-model machine learning yang lain yang kompleks. Dalam kasus project ini kami menggunakan kernel “linear”, hal ini dikarenakan prediksi yang akan kita hasilkan biner. SVM juga cukup bagus dalam menangani data outliers, karena SVM berfokus pada margin dan support vectors. Maka dari itu untuk data yang kami miliki, bisa dikatakan model SVM lumayan cocok jika diimplementasikan dalam dataset yang kita miliki.

Evaluasi dari model SVM akan menggunakan confusion metrics dengan menampilkan ROC curve(0.5 berarti model menebak secara random dan nilai terbaik adalah 1), f1-score, recall, dan precision.

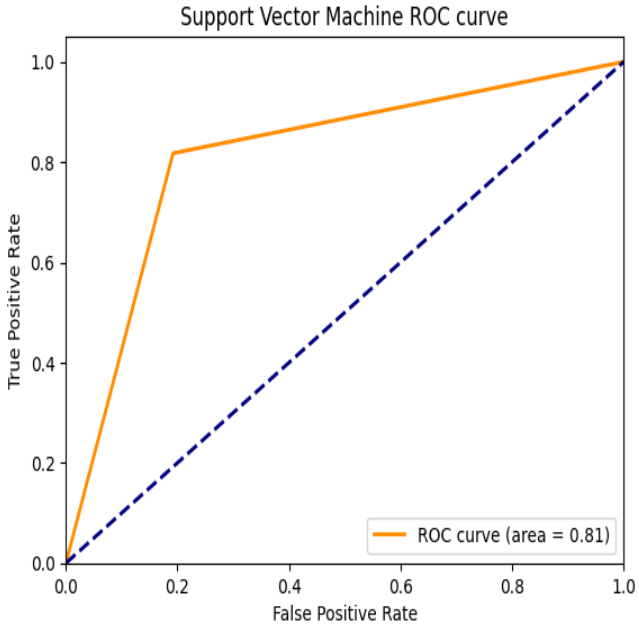
Dari confusion matrix berikut, dapat diperoleh informasi bahwa model ini memiliki performa yang cukup seimbang dalam memprediksi kedua kelas. Terbukti dengan keseimbangan antara prediksi benar dan salah untuk kedua kelas.




Dari hasil classification report berikut, dapat diketahui bahwa terdapat keseimbangan antar precision dan recall terbukti dengan adanya f1-score yang mencapai 0.81. Serta akurasi dari model adalah 0.81.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.82 | 0.81 | 0.81 | 21279 |
| 1 | 0.81 | 0.82 | 0.81 | 21278 |
| accuracy | | | 0.81 | 42557 |
| macro avg | 0.81 | 0.81 | 0.81 | 42557 |
| weighted avg | 0.81 | 0.81 | 0.81 | 42557 |

Dari hasil ROC curve berikut menunjukkan kemampuan diagnostik model dengan AUC sebesar 0,81. Hal tersebut menunjukkan model memiliki kemampuan yang baik dan juga efektif dalam membedakan kedua kelas.




**2) Logistic Regression**

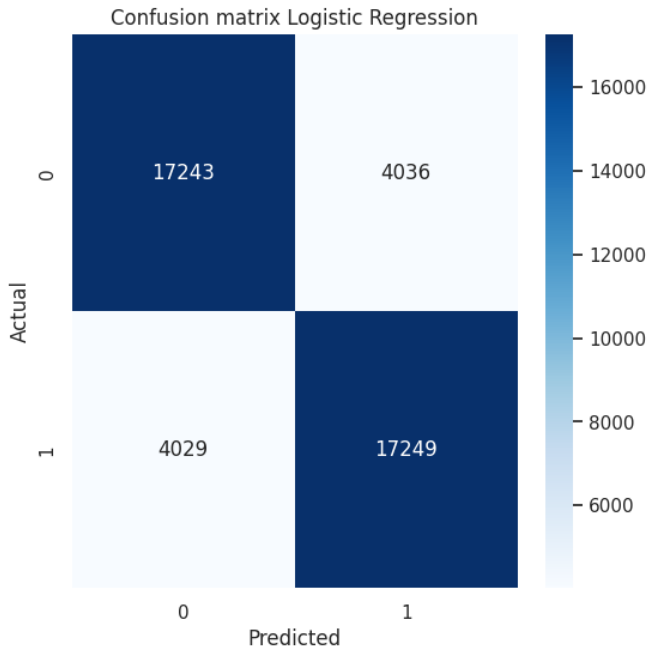
```
logistic_regression = LogisticRegression()
logistic_regression.fit(X_train_vect, y_train)
y_pred_logistic_regression = logistic_regression.predict(X_test_vect)
```

Model Logistic Regression merupakan salah satu model klasifikasi yang cocok untuk topik text sentiment analysis. Hal itu karena logistic regression bisa dikatakan cukup efisien dalam hal waktu dan sumber daya ketika menghadapi sumber data yang besar. Model ini relatif sederhana dalam training model. Hampir sama seperti Naive Bayes Logistic Regression menganalisis text based on probability. Cara menghitung probability nya dengan sigmoid. Jadi, kata lebih dekat dengan nilai 1 atau 0. Jika lebih dekat dengan nilai 1 maka dikategorikan sebagai hate speech dan sebaliknya ketika 0 dikategorikan sebagai non hate speech. Karena bersifat klasifikasi Logistic

regression hanya memiliki rentang nilai 0 hingga 1 saja. Kelebihan lain dari model logistic regression adalah model yang bisa menyesuaikan dengan pola data yang non linear, sehingga model ini bisa dikatakan akan menghasilkan performa yang cukup bagus.

Evaluasi dari model logistic regression akan menggunakan confusion metrics dengan menampilkan ROC curve(0.5 berarti model menebak secara random dan nilai terbaik adalah 1),  f1-score, recall, dan precision.

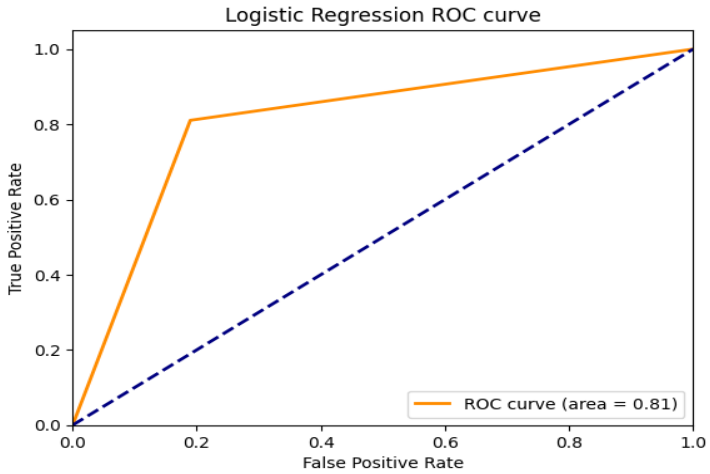
Dari confusion matrix berikut, dapat diperoleh informasi bahwa model ini memiliki performa yang cukup seimbang dalam memprediksi kedua kelas. Terbukti dengan keseimbangan antara prediksi benar dan salah untuk kedua kelas.




Dari hasil classification report berikut, dapat diketahui bahwa terdapat keseimbangan antar precision dan recall terbukti dengan adanya f1-score yang mencapai 0.81. Serta akurasi dari model adalah 0.81.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.81 | 0.81 | 0.81 | 21279 |
| 1 | 0.81 | 0.81 | 0.81 | 21278 |
| accuracy | | | 0.81 | 42557 |
| macro avg | 0.81 | 0.81 | 0.81 | 42557 |
| weighted avg | 0.81 | 0.81 | 0.81 | 42557 |

Dari hasil ROC curve berikut menunjukkan kemampuan diagnostik model dengan AUC sebesar 0,81. Hal tersebut menunjukkan model memiliki kemampuan yang baik dan juga efektif dalam membedakan kedua kelas.




3) **Naive Bayes**

```
mnb = MultinomialNB()
mnb.fit(X_train, y_train)
y_pred_mnb = mnb.predict(X_test)
```

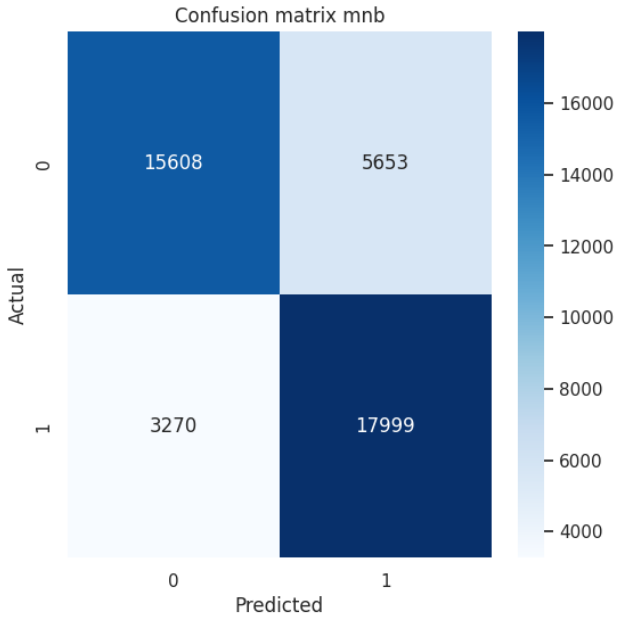
Model ketiga yang kita gunakan untuk melakukan perbandingan machine learning model adalah Naive Bayes. Naive Bayes juga merupakan salah satu model klasifikasi machine learning. Salah satu keunggulan dari machine learning adalah model yang efektif karena proses train yang dilakukan cepat. Naive Bayes hanya menghitung probabilitas dan menentukan berdasarkan probabilitas yang paling besar. Jika dilihat secara implementasi, Naive Bayes bisa dikatakan model yang cukup sederhana. Naive bayes menghitung probabilitas per kata, jadi bisa dikatakan kata tersebut bersifat independen.

Salah satu kekurangan dari model Naive Bayes adalah tidak memahami konteks secara keseluruhan. Sehingga terkadang kurang bisa memahami sarkas atau kata atau makna kiasan. Namun dalam banyak kasus Naive Bayes bisa dikatakan cukup konsisten dan cukup akurat, walaupun dengan perhitungan yang sederhana. Model Naive Bayes juga bersifat adil dimana memiliki probabilistic prior. Dimana jika terjadi ketidakseimbangan dataset, sebagai contoh data text positif lebih banyak dari negatif dan juga sebaliknya.

Naive bayes membagi sesuai dengan jumlah labelnya masing-masing. Maka dari itu tidak ada ketidak seimbangan probabilitas.

Pada kasus ini kami menggunakan Multinomial Naive Bayes karena model ini cocok untuk masalah klasifikasi teks dan juga karena inputnya berupa sparse matrix setelah proses transformasi TF-IDF. Evaluasi dari model ini akan menggunakan confusion matrix dengan menampilkan ROC curve(0.5 berarti model menebak secara random dan nilai terbaik adalah 1), f1-score, recall, dan precision.

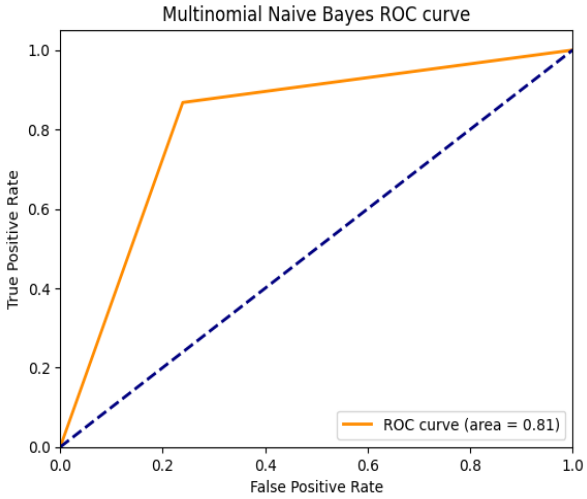
Dari confusion matrix berikut, dapat diperoleh informasi bahwa model ini memiliki performa yang cukup seimbang dalam memprediksi kedua kelas. Terbukti dengan keseimbangan antara prediksi benar dan salah untuk kedua kelas, meskipun tidak lebih baik daripada SVM dan juga Logistic Regression.




Dari hasil classification report berikut, dapat diketahui bahwa akurasi dari model adalah 0.79. Hal ini tentu menunjukkan bahwa model MNB ini tidak lebih baik daripada SVM dan Logistic Regression.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.73 | 0.78 | 21261 |
| 1 | 0.76 | 0.85 | 0.80 | 21269 |
| accuracy | | | 0.79 | 42530 |
| macro avg | 0.79 | 0.79 | 0.79 | 42530 |
| weighted avg | 0.79 | 0.79 | 0.79 | 42530 |

Dari hasil ROC curve berikut menunjukkan kemampuan diagnostik model dengan AUC sebesar 0,81. Hal tersebut menunjukkan model memiliki kemampuan yang baik dan juga efektif dalam membedakan kedua kelas.




## 2. Deep Learning

### a. Feature Extraction

```
vocab_size = 500
max_length = 50
trunc_type = 'post'
padding_type = 'post'
oov_tok = "<OOV>"
labels = tweet_df['Label']

tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(tweet_df['Clean'].astype(str))

sequences = tokenizer.texts_to_sequences(tweet_df['Clean'].astype(str))

padded_sequences = pad_sequences(sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type)
```

Pada tahap ini kami melakukan feature extraction pada data sebelum pada akhirnya dipakai untuk training model menggunakan deep learning model. Pada tahap feature extraction untuk deep learning method yang kami pakai adalah Tokenizer.

Tokenizer merupakan sebuah class yang terdapat dalam library tensorflow yang bekerja untuk mengubah sebuah text menjadi sebuah urutan angka (token). Hal ini dikarenakan model deep learning tidak bisa di train dengan data dalam bentuk text. Maka dari itu diperlukan untuk merubah text menjadi sebuah urutan angka yang nantinya akan di proses dalam training model deep learning.

Function tokenizer memiliki dua parameter yaitu num_words dan oov_token. num_words menunjukan seberapa banyak vocab yang menjadi angka maksimum yang dikatakan paling sering muncul dalam dataset. Dalam konteks

code diatas, kita menentukan 500 kata teratas yang sering muncul dalam dataset. Sedangkan untuk bagian oov_token adalah untuk menentukan token apa yang digunakan untuk kata yang tidak ada dalam database atau bisa disebut out of vocabullary. Data text yang diubah kedalam bentuk urutan angka merupakan text dari kolom “clean” yang dimana berisi text yang sudah bersih atau sudah di data cleaning. Pada tahapan sequence bertujuan untuk mengurutkan angka berdasarkan vocab yang telah dibuat oleh tokenizer dan kata kata yang umum atau sering muncul di letakan di index terdepan.

**b. Modeling**

**1) LSTM (Long Short-Term Memory)**

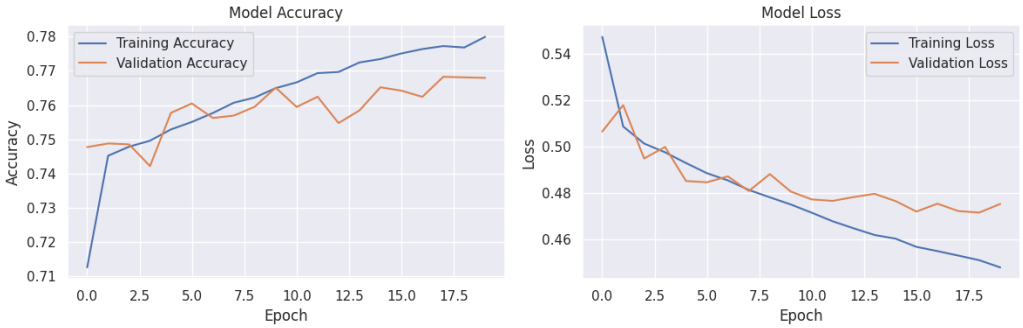
LSTM merupakan salah satu model deep learning yang berasal dari model RNN (recurrent neural network). LSTM merupakan salah satu model yang sangat cocok untuk melakukan pemrosesan sentiment analysis. Hal itu karena LSTM dapat memahami konteks jangka panjang yang terdapat dalam teks. Salah satu tantangan yang ada di dalam dunia NLP yaitu dimana kesulitan AI untuk memahami konteks dari sebuah kalimat atau teks.

Karena LSTM adalah model yang bersifat recurrent maka dia melihat node sebelumnya untuk memahami konteks yang ada. Namun perbedaan dengan RNN adalah RNN tidak dapat memahami konteks secara jangka panjang, yang dimana mengalami semacam vanishing gradient. Maka dari itu tentunya akan di solve oleh model LSTM. LSTM disebut memahami text secara kontekstual juga berkaitan dengan urutan dari sebuah teks. terkadang urutan dari sebuah teks menentukan makna dari teks tersebut. dari hal ini tentunya bisa dikatakan

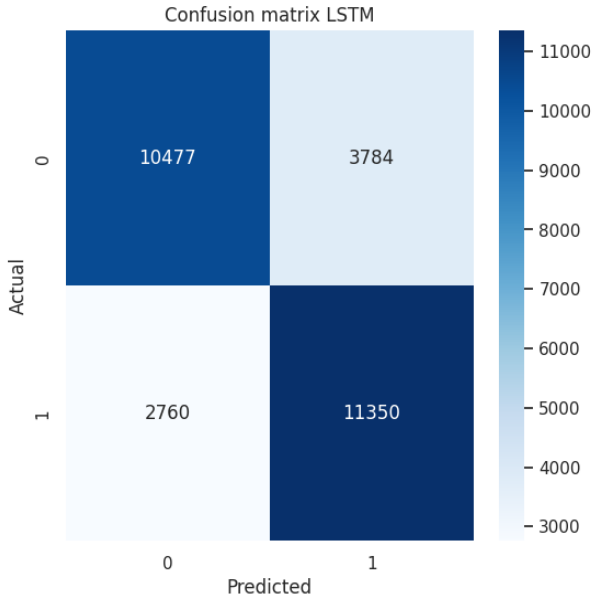
LSTM seharusnya bisa lebih unggul dari model Machine learning tradisional yang tidak memahami urutan teks secara kontekstual. Untuk model LSTM, activation function yang digunakan antar hidden layer adalah relu. Tetapi outputnya menggunakan activation function 'sigmoid' (output sekitar 0 hingga 1) dikarenakan klasifikasi data yang berupa biner sehingga dapat menghitung loss dengan 'binary_crossentropy' dan metrics 'accuracy'.

Kita melakukan training dengan batch_size = 128, dan epoch sebanyak 20. Dan berikut adalah hasil plotting dari accuracy dan loss model per epoch, terlihat bahwa accuracy dari training meningkat secara stabil hingga mencapai 0.78, sementara accuracy dari validation cenderung fluktuatif.

Terlihat juga bahwa loss dari training menurun secara konsisten yang menandakan bahwa model belajar dengan baik, sementara loss dari validation cenderung fluktuatif menandakan bahwa model tidak mampu menangkap pola secara konsisten pada data baru yang tidak ada di training.



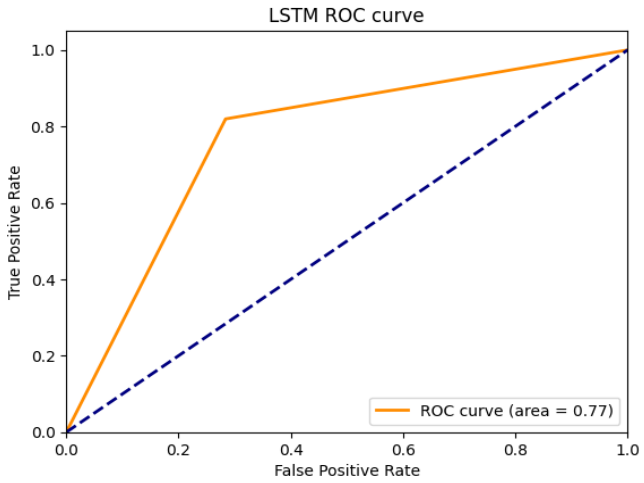

Dari confusion matrix berikut, dapat diperoleh informasi bahwa model ini memiliki performa yang cukup seimbang dalam memprediksi kedua kelas. Terbukti dengan keseimbangan antara prediksi benar dan salah untuk kedua kelas, meskipun hasilnya tidak lebih baik dari model-model sebelumnya.




Dari hasil classification report berikut, dapat diketahui bahwa akurasi dari model adalah 0.77. Hal ini tentu menunjukkan bahwa model LSTM ini tidak lebih baik daripada SVM, Logistic Regression, dan Multinomial Naive Bayes.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.80 | 0.72 | 0.76 | 14261 |
| 1 | 0.74 | 0.82 | 0.78 | 14110 |
| accuracy | | | 0.77 | 28371 |
| macro avg | 0.77 | 0.77 | 0.77 | 28371 |
| weighted avg | 0.77 | 0.77 | 0.77 | 28371 |

Dari hasil ROC curve berikut menunjukkan kemampuan diagnostik model dengan AUC sebesar 0,77. Hal tersebut menunjukkan model memiliki kemampuan yang masih kalah dibandingkan dengan model-model sebelumnya yang mendapatkan AUC sebesar 0.81.




**2) BiLSTM**

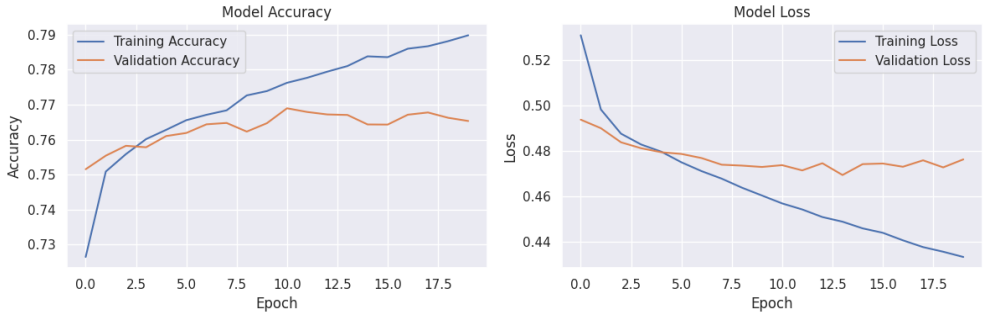
BiLSTM (Bidirectional Long Short-Term Memory) pada dasarnya merupakan model deep learning yang strukturnya mengambil referensi dari LSTM. Namun yang jadi pembedanya, BiLSTM memproses data dua arah. Hal ini sangat cocok dengan topik sentiment analysis karena BiLSTM dapat menangkap konteks dari sebiuah teks dari awal sampai akhir. Dengan demikian model BiLSTM dapat memahami konteks text tersebut secara lengkap. Model BiLSTM juga dapat mengatasi text text yang berisi sarkasme dan kiasan, karena model BiLSTM dapat menangkap konteks dari dua arah.

Karena BiLSTM memproses data dua arah, maka model ini akan lebih banyak menangkap detail detail yang terdapat dalam text yang tidak mampu untuk dilakukan dengan model machine learning yang tradisional. Hal ini sangat penting karena terkadang hal hal kecil dalam teks bisa mempengaruhi makna dari teks tersebut. Contoh kalimat “Sungguh gila, kamu mengerjakan semua itu” Kalimat tersebut terkesan negatif namun sebenarnya merupakan

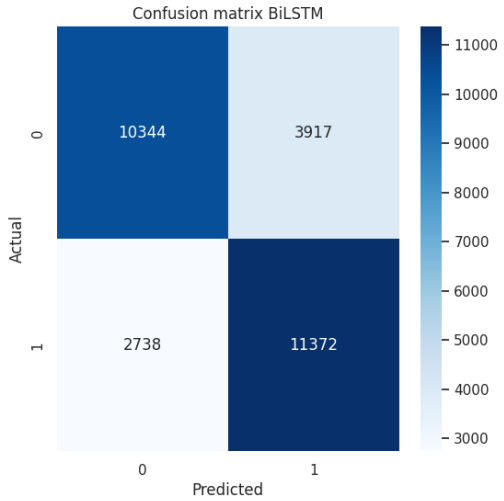
kalimat yang positif. Hanya karena ada kata “gila” tidak berarti menjadi kata2 yang negatif. contoh kalimat lain “Wow sangat luar biasa, selama 10 jam kamu hanya selesai 1 tugas.” sebaliknya dari kalimat sebelumnya, kalimat ini terkesan positif namun aslinya negatif. kata kata ini hanya terlihat positif karena kata “Wow” dan “luar biasa” padahal sebenarnya hanya kata sarkasme.

Maka dari itu BiLSTM seharusnya lebih unggul dari model machine learning tradisonal dalam hal menangkap konteks yang ada, sehingga tidak salah dalam memberikan prediksi untuk teks yang berisi sarkasme.

Kita melakukan training dengan batch_size = 128, dan epoch sebanyak 20. Dan berikut adalah hasil plotting dari accuracy dan loss model per epoch, terlihat bahwa accuracy dari training meningkat secara stabil hingga mencapai 0.79, sementara accuracy dari validation cenderung fluktuatif. Terlihat juga bahwa loss dari training menurun secara konsisten yang menandakan bahwa model belajar dengan baik, sementara loss dari validation cenderung fluktuatif menandakan bahwa model tidak mampu menangkap pola secara konsisten pada data baru yang tidak ada di training.




Dari confusion matrix berikut, dapat diperoleh informasi bahwa model ini memiliki performa yang cukup seimbang dalam memprediksi kedua kelas. Terbukti dengan keseimbangan antara prediksi benar dan salah untuk kedua kelas, meskipun hasilnya tidak lebih baik dari model-model sebelumnya.

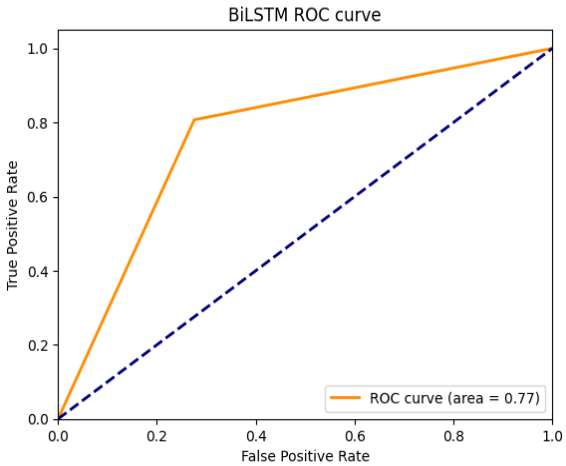



Dari hasil classification report berikut, dapat diketahui bahwa akurasi dari model adalah 0.77. Hal ini tentu menunjukkan bahwa model BiLSTM ini memiliki akurasi yang sama dengan model LSTM, namun tidak lebih baik daripada SVM, Logistic Regression, dan Multinomial Naive Bayes.

```
print(metrics.classification_report(y_test, y_pred_blstm))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.73 | 0.76 | 14261 |
| 1 | 0.74 | 0.81 | 0.77 | 14110 |
| accuracy | | | 0.77 | 28371 |
| macro avg | 0.77 | 0.77 | 0.77 | 28371 |
| weighted avg | 0.77 | 0.77 | 0.77 | 28371 |

Dari hasil ROC curve berikut menunjukkan kemampuan diagnostik model dengan AUC sebesar 0,77. Hal tersebut menunjukkan model memiliki kemampuan yang masih kalah dibandingkan dengan model-model sebelumnya yang mendapatkan AUC sebesar 0.81.



## E. Model Comparison and Analysis

| Model | Accuracy |
|---|---|
| SVM | 0.81 |
| Logistic Regression | 0.81 |
| Multinomial Naive Bayes | 0.79 |
| LSTM | 0.77 |
| BiLSTM | 0.77 |

Melakukan perbandingan antara model Machine Learning dengan Deep Learning bisa dianggap kurang adil. Pada kasus ini, memiliki performa yang lebih baik dibandingkan LSTM dan BiLSTM. Hal ini dikarenakan LSTM dan BiLSTM, sebagai model sequential yang mempelajari pola data, memerlukan dataset yang besar untuk dapat membaca pola data dengan baik dan akurat, sedangkan pada kasus ini dataset yang digunakan relatif kecil.

Metode tradisional yang menggunakan feature extraction seperti TF-IDF bisa bekerja lebih baik dibandingkan dengan model deep neural network yang mampu menangkan pola dalam teks mentah. Hal tersebut juga dikarenakan jumlah dari dataset pada kasus ini yang relatif kecil. LSTM dan BiLSTM juga rentan terhadap masalah vanishing loss dan bisa terjebak di local minima selama proses training, sehingga memerlukan tuning yang lebih teliti.

Training LSTM dan BILSTM juga membutuhkan usaha besar, termasuk dataset yang besar dan teks panjang untuk menangkap pola dengan baik. Oleh karena itu, implementasi LSTM dan BILSTM untuk kasus ini bisa menjadi terlalu kompleks.

Model Logistic Regression dianggap sebagai yang terbaik untuk kasus ini karena memiliki akurasi yang baik dengan waktu training yang lebih cepat dibandingkan SVM. Meskipun SVM juga menunjukkan performa yang baik dan sama dengan Logistic Regression, namun waktu training yang dibutuhkan oleh SVM lebih lama dibandingkan Logistic Regression. Hal ini disebabkan oleh kompleksitas algoritma SVM dalam mencari hyperplane yang optimal untuk memisahkan kelas-kelas data, sehingga membutuhkan waktu komputasi yang lebih lama. Oleh karena itu, Logistic Regression adalah model yang paling tepat untuk kasus ini meskipun akurasi yang diperoleh tidak mencapai angka diatas 0.9 dikarenakan isi daripada datasetnya yang kurang bagus.

## F. Predict New Data With Logistic Regression

```
Input Text: Black people are losers
Fix Slang: Black people are losers
Correct Typo: Black people are losers
Clean Text: black people are losers
Tokenize: ['black', 'people', 'are', 'losers']
Lemmatize Tokens: ['black', 'people', 'are', 'loser']
Remove Stopwords: ['black', 'people', 'loser']


Input Text: You're a good person.
Fix Slang: you are a good person.
Correct Typo: you are a good person
Clean Text: you are a good person
Tokenize: ['you', 'are', 'a', 'good', 'person']
Lemmatize Tokens: ['you', 'are', 'a', 'good', 'person']
Remove Stopwords: ['good', 'person']


Input Text: I h8 u so much, you're a worthless piece of shit with a vagina!
Fix Slang: I hate you so much, you are a worthless piece of shit with a vagina!
Correct Typo: I hate you so much you are a worthless piece of shit with a vagina
Clean Text: i hate you so much you are a worthless piece of shit with a vagina
Tokenize: ['i', 'hate', 'you', 'so', 'much', 'you', 'are', 'a', 'worthless', 'piece', 'of', 'shit', 'with', 'a', 'vagina']
Lemmatize Tokens: ['i', 'hate', 'you', 'so', 'much', 'you', 'are', 'a', 'worthless', 'piece', 'of', 'shit', 'with', 'a', 'vagina']
Remove Stopwords: ['hate', 'much', 'worthless', 'piece', 'shit', 'vagina']


Input Text: Great job on the project, keep it up!
Fix Slang: Great job on the project, keep it up!
Correct Typo: Great job on the project keep it up
Clean Text: great job on the project keep it up
Tokenize: ['great', 'job', 'on', 'the', 'project', 'keep', 'it', 'up']
Lemmatize Tokens: ['great', 'job', 'on', 'the', 'project', 'keep', 'it', 'up']
Remove Stopwords: ['great', 'job', 'project', 'keep']
```

```
Input Text: Black people are losers
Processed Text: black people loser
Prediction: Hate Speech!


Input Text: You're a good person.
Processed Text: good person
Prediction: Not a Hate Speech.


Input Text: I h8 u so much, you're a worthless piece of shit with a vagina!
Processed Text: hate much worthless piece shit vagina
Prediction: Hate Speech!


Input Text: Great job on the project, keep it up!
Processed Text: great job project keep
Prediction: Not a Hate Speech.
```

## G. Link Colab (code)

- Preprocessing: https://colab.research.google.com/drive/1UgnBuwi6KpL5FD7uYpt8683DYpoTogf9
- SVM**:** https://colab.research.google.com/drive/1QPg4sLvd8RtWXi9wXa8m-bWrjSdGruPo#scrollTo=PSJCw09AnIDD
- Logistic Regression: https://colab.research.google.com/drive/1aFRtXjbIPoNDRyxHxJ8C0Usr3EVV0atU#scrollTo=2OTI7bpbD-kH
- Naive Bayes:

https://colab.research.google.com/drive/1Zj-g6e0TY2kXpvjU1C6FfW2C9-ABmT_P

- LSTM: https://colab.research.google.com/drive/1-re7peMA_sdpy6u7zUSNTA0yHxOXTdZw
- BiLSTM: https://colab.research.google.com/drive/1b0Luke1oGJVaNZt6yKVOmk-p9KK37Ug1

## H. Conclusion

Secara keseluruhan model machine learning yang sederhana seperti SVM dan Logistic Regression memiliki accuracy yang lebih bagus jika dibandingkan dengan model deep learning seperti LSTM dan BiLSTM. Hal ini disebabkan oleh keterbatasan dataset yang digunakan dan sulit menemukan dataset yang sesuai. Model Deep Learning membutuhkan dataset yang besar untuk memahami konteks dari sebuah kalimat agar dapat menghasilkan akurasi yang baik. Oleh karena itu, model Deep Learning memiliki performa yang kurang optimal untuk kasus ini.

Namun, dalam praktik di dunia nyata, jika dataset yang dimiliki cukup besar dan sesuai, model deep learning dapat sangat efektif. Untuk proyek ini, karena dataset yang digunakan tidak cukup besar, model machine learning sederhana seperti SVM dan Logistic Regression menunjukkan performa terbaik. Jika harus memilih satu model terbaik, Logistic Regression menjadi pilihan utama karena meskipun akurasinya sama dengan SVM, waktu training yang dibutuhkan lebih sedikit dibandingkan SVM.