Twitter Sentiment Analysis

Gregorius Reynaldi Pratama

10 October 2025

Meet the Team

Gregorius Reynaldi Pratama

https://github.com/GregReynaldi

Background

Fenomena Media Sosial:

- Platform seperti Twitter, Instagram, dan Facebook telah menjadi ruang utama masyarakat menyampaikan pendapat tentang politik, kebijakan, produk, atau layanan.

Tantangan Utama:

- Jutaan komentar dan posting setiap hari mustahil dianalisis secara manual yang dimana prosesnya lambat, mahal, dan rentan misinterpretasi.

Kebutuhan Solusi Teknologi:

- Diperlukan alat otomatis untuk mengolah data besar secara real-time, agar organisasi dapat memahami sentimen publik dengan cepat dan akurat.

Peran NLP (Natural Language Processing):

- Teknologi ini mampu mengenali dan mengklasifikasikan emosi dalam teks (positif, negatif, netral), memberikan wawasan berbasis data untuk mendukung keputusan strategis di bidang politik, bisnis, atau komunikasi.

📌 TUJUAN PENELITIAN :

- **Analisis Sentimen**

Mengukur opini publik terhadap kandidat politik melalui tweet, diklasifikasikan sebagai positif, negatif, atau netral.

- **Pengembangan Model**

Membandingkan algoritma tradisional (Logistic Regression, Naive Bayes) dan deep learning (IndoBERT) untuk analisis teks bahasa Indonesia.

- **Aplikasi Praktis**

Memberikan wawasan berbasis data untuk strategi komunikasi politik dan kampanye.

📌 CAKUPAN PENELITIAN :

- **Sumber Data:**

  Tweet berbahasa Indonesia terkait kandidat politik dan isu pemilu.

- **Praproses Data:**

  Menghapus URL, mention, hashtag, dan stopwords yang tidak perlu.
  Normalisasi ejaan, slang, dan emotikon.
  Tokenisasi teks untuk analisis.

- **Pemodelan**:

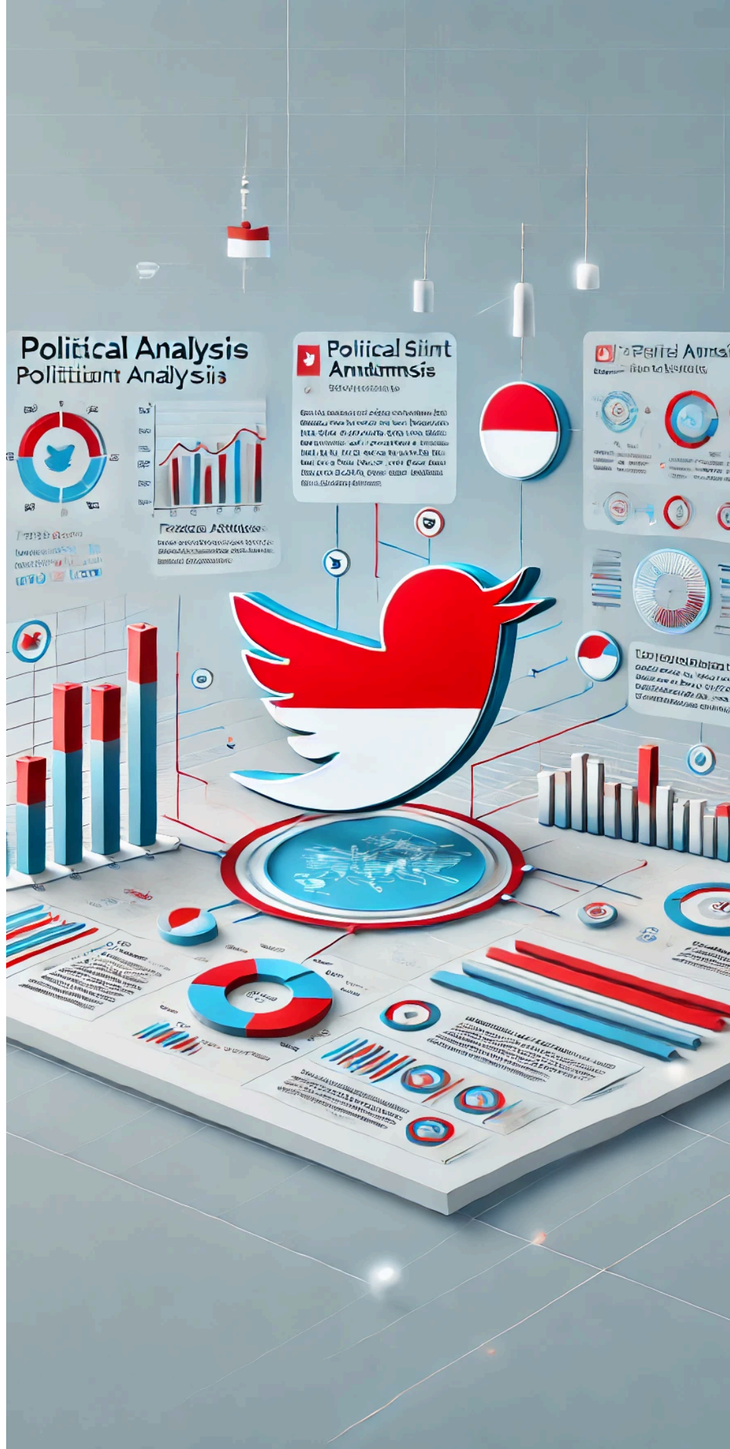
  Eksperimen dengan model machine learning tradisional dan Transformer
  Optimasi model melalui fine-tuning dan hyperparameter tuning.

- **Evaluasi:**

  Validasi dengan cross-validation dan metrik performa komprehensif accuracy, precision, recall, F1-Score.

- **Hasil:**

  Visualisasi tren sentimen dan laporan wawasan sebagai pendukung data



Objectives & Requirements

OBJECTIVES:

- Membangun model analisis sentimen untuk teks politik berbahasa Indonesia, khususnya terkait dengan kampanye politik dan kandidat presiden.
- Membandingkan model machine learning tradisional (seperti LogRegression, RandomForest, Naive Bayes, Categorical Boosting,XGBoost) dengan model deep learning (seperti LSTM, CNN, RNN, dan Transformer pretrained model) untuk memilih metode yang paling optimal.
- Mengevaluasi performa model menggunakan metrik akurasi, precision, recall, dan F1-score, untuk memastikan model dapat mengklasifikasikan sentimen dengan akurat.
- Menentukan algoritma terbaik untuk analisis sentimen pada teks politik Indonesia, yang dapat digunakan dalam berbagai aplikasi politik dan komunikasi publik.
- Mempublikasikan hasil proyek dan kode sumber di GitHub untuk referensi dan kolaborasi lebih lanjut.

Tanggal Proyek:

- Tanggal Mulai: 26 September 2025
- Tanggal Selesai: 15 Oktober 2025

Project Step

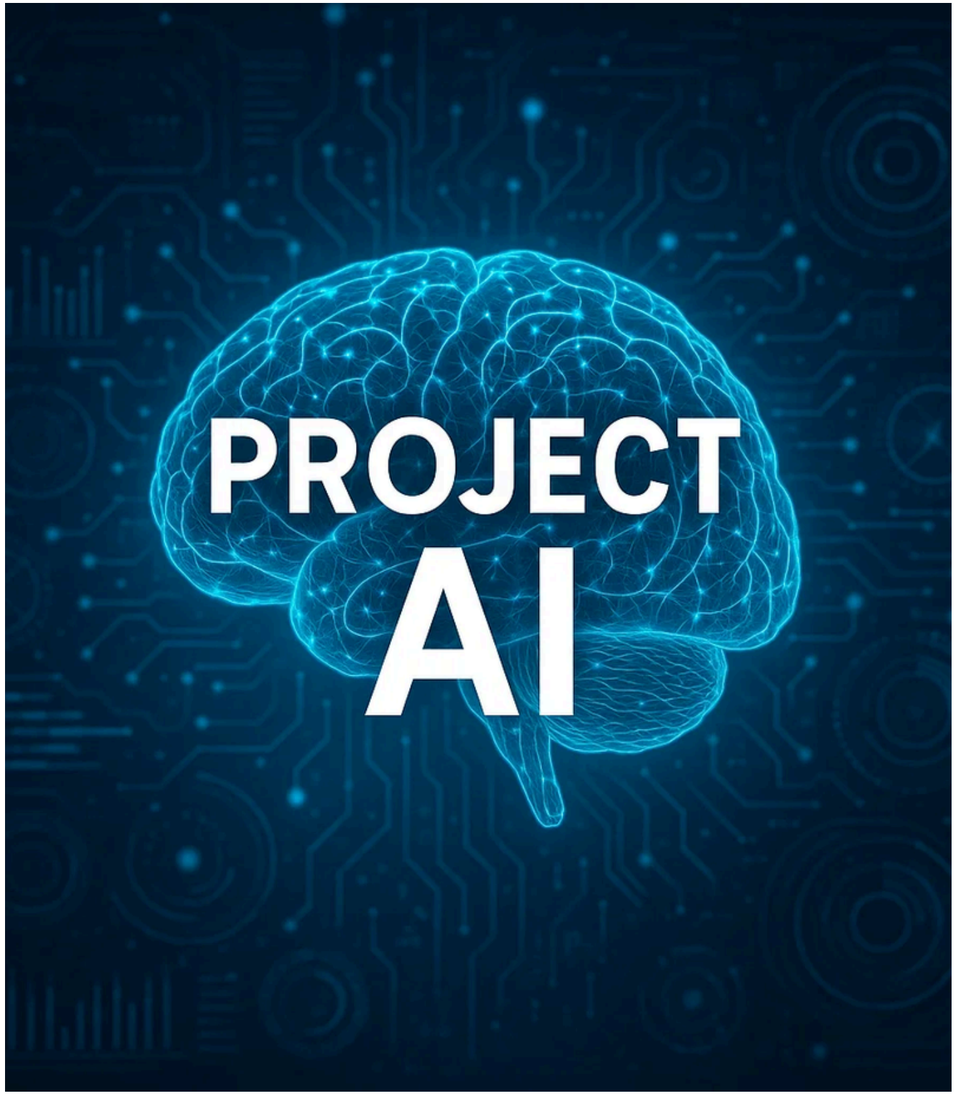
Step 1
Data Collection & Preparation

Step 2
Exploratory Data Analysis (EDA)

Step 3
Data Preprocessing

Step 4
Modeling

Step 5
Evaluation



Data Collection & Preparation

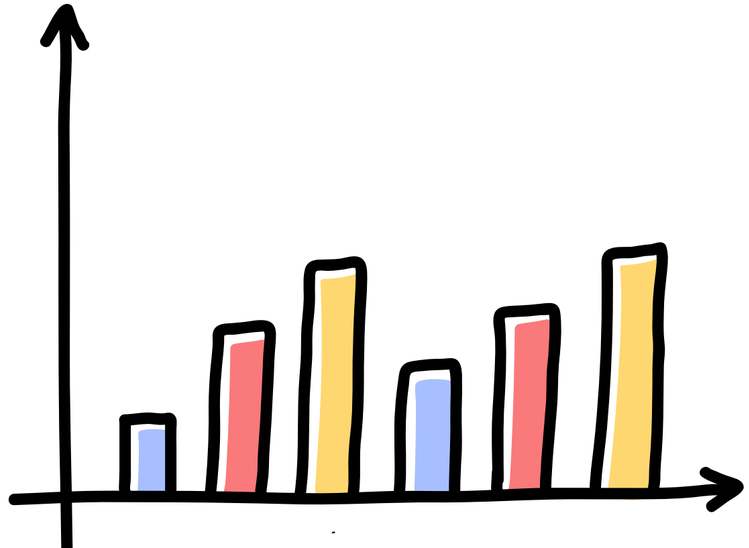
Sumber Data:
- Dataset "tweet.csv" yang disediakan oleh Indonesia AI
- Data diperoleh melalui web scraping dari platform Twitter
- Topic of Data : Tweet Mengenai Pemilihan Presiden Indonesia 2019

Fokus Penelitian:
- Membangun model klasifikasi sentimen untuk mendeteksi sebuat tweet apakah tergolong positif, negatif, atau netral.
- Memprediksi hasil sentimen dari tweet yang dipublikasi pengguna tanpa menggunakan cara manual nantinya setelah dilakukan training.

Cakupan Data:
- Data tweet berbahasa Indonesia terkait dinamika Pilpres 2019
- Merepresentasikan opini publik dalam periode politik tinggi
- Data kotor dan belum di preprocess dan cleaning oleh Indonesia AI

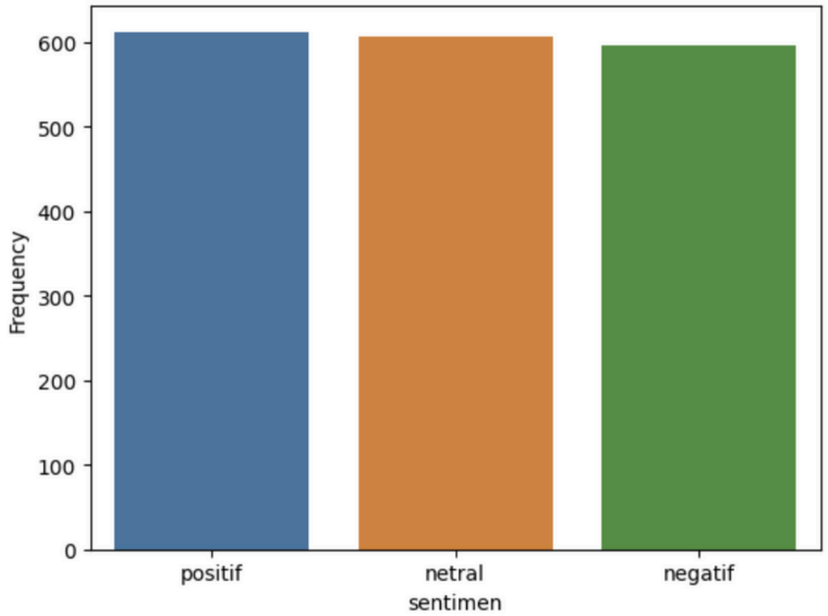
Untuk project kali ini, kita hanya menggunakan data csv yang telah diberikan, lalu convert data dari csv ini menjadi DataFrame menggunakan pandas module

Variabel	Tipe Data	Deskripsi
Tweet	Text	Teks lengkap tweet asli yang dipublikasi pengguna
Sentimen	Kategorikal	Label sentimen yang telah diannotasi (Positif, Negatif, Netral)



Exploratory Data Analysis (EDA)

- #PopulerB1 4: Jokowi: Naikkan Tax Ratio Drastis Bisa Guncang Ekonomi http://brt.st/6b4KÂ
- Monggo silahkan..itu hak anda...hargai juga kami yg inginkan perubahan lebih baik...
- "mencontoh ekonomi china, bukannya sujud seperti sekarang iniðŸ¤£"



	sentimen	tweet
0	negatif	Kata @prabowo Indonesia tidak dihargai bangsa ...
1	netral	Batuan Langka, Tasbih Jokowi Hadiah dari Habib...
2	netral	Di era Jokowi, ekonomi Indonesia semakin baik....
3	positif	Bagi Sumatera Selatan, Asian Games berdampak p...
4	negatif	Negara kita ngutang buat bngun infrastruktur y...
...	...	...
1810	netral	Negarawan sejati sll bangga dan mengedepankan ...
1811	netral	1. HRS ceramah di Damai Indonesiaku 2. Perekon...
1812	netral	Mari bangun bangsa dgn mendukung perekonomian ...
1813	netral	Bantu majukan perekonomian bangsa bersama Pak ...
1814	netral	Pak @jokowi mengubah cara pandang ekonomi. Kin...

1815 rows × 2 columns

```
data.loc[data.duplicated()]
```

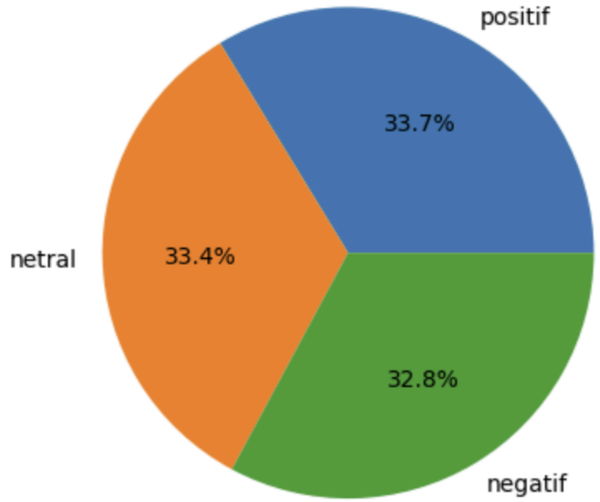
✓ 0.0s

sentimen	tweet

```
data.isna().sum()
```

✓ 0.0s

```
sentimen    0
tweet       0
dtype: int64
```



Exploratory Data Analysis (EDA)

Distribusi Panjang Tweet Kategori Positif:
Distribusi panjang tweet untuk kategori positif memiliki puncak di sekitar 250 karakter, tetapi dengan distribusi yang sedikit lebih sempit dibandingkan dengan kategori negatif.

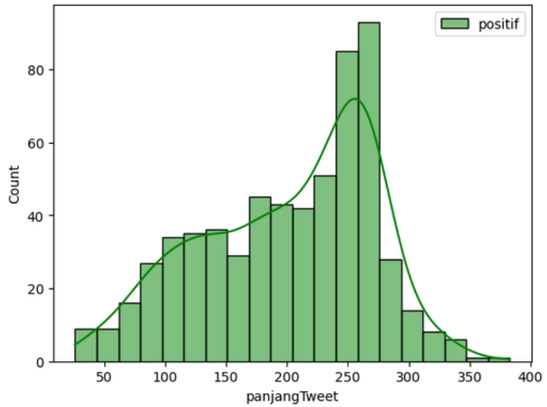
Tweet positif cenderung memiliki panjang yang lebih seragam dengan lebih banyak tweet berfokus di sekitar panjang 250 karakter dan lebih sedikit tweet dengan panjang ekstrem.

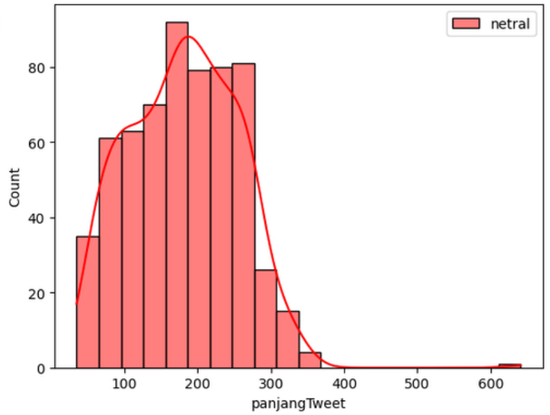
Distribusi Panjang Tweet Kategori Netral:
Distribusi panjang tweet untuk kategori netral menunjukkan puncak yang signifikan pada sekitar 200 karakter.

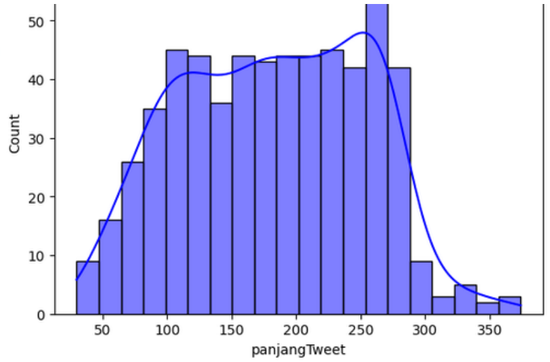
Tweet dengan panjang sangat tinggi terlihat lebih jarang pada kategori netral, dan distribusinya lebih konsisten di sepanjang rentang panjang, meskipun ada beberapa tweet panjang yang sangat ekstrem (600).

Distribusi Panjang Tweet Kategori Negatif:

Distribusi panjang tweet untuk kategori negatif terlihat memiliki rentang panjang yang lebih beragam, tetapi dengan puncak yang jelas pada panjang tweet sekitar 250 karakter.





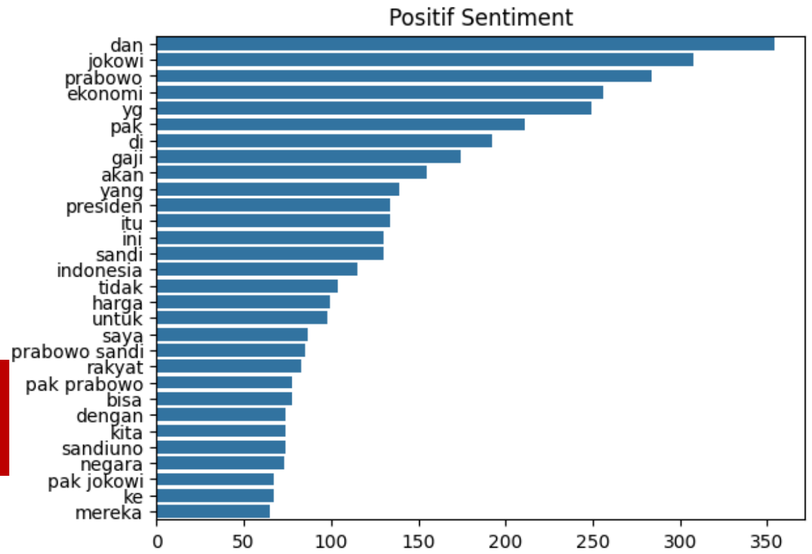


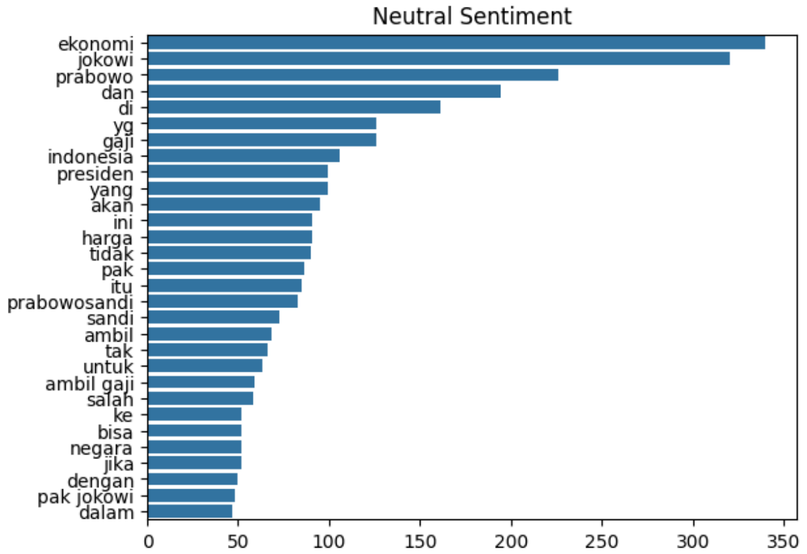
Exploratory Data Analysis (EDA)

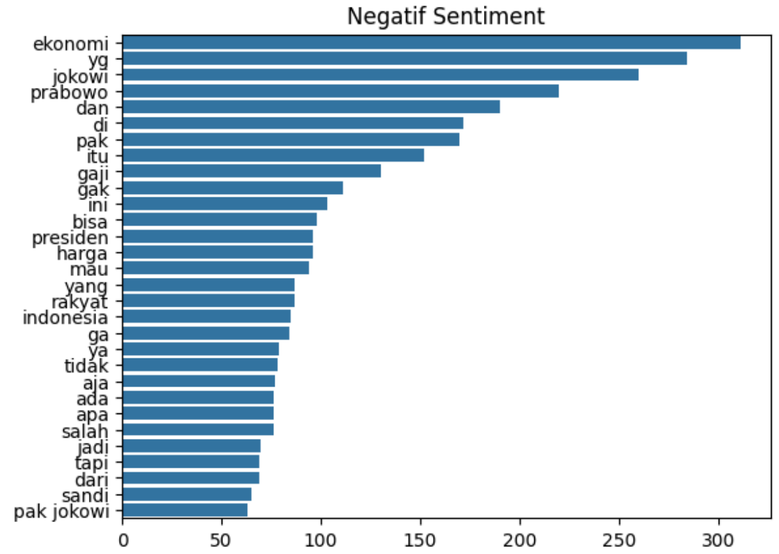
SENTIMEN POSITIF
- **Kata Kunci Utama:** "jokowi", "prabowo", "ekonomi", "gaji"
- **Karakteristik:** Banyak stopwords tidak penting “dan”,”yg”,”di”,”yang”,”itu” → akan dihilangkan saat preprocessing
- **Interpretasi:** Cenderung mengangkat capaian positif dan harapan terhadap pemimpin

SENTIMEN NEGATIF
- **Kata Kunci Utama:** "ekonomi", "gak", "pajak", "kerja", "negara"
- **Karakteristik:** Banyak stopwords tidak penting “dan”,”di”,”yg”,”yang”,”akan” → akan dihilangkan saat preprocessing
- **Interpretasi:** Merefleksikan kritik dan protes terhadap kondisi sosial-ekonomi

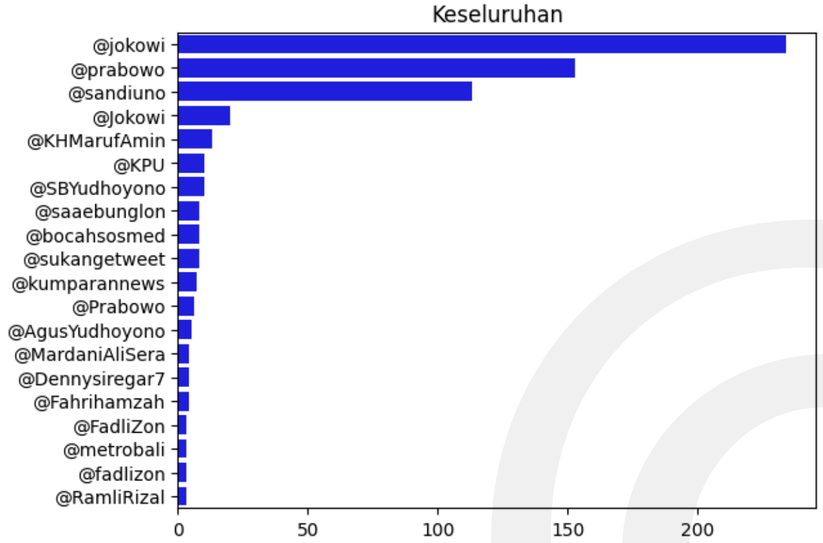
SENTIMEN NETRAL
- **Kata Kunci Utama:** "ekonomi", "pertumbuhan", "debat", "pilih", "janji"
- **Karakteristik:** Banyak stopwords tidak penting “dan”,”di”,”yg”,”yang”,”akan” → akan dihilangkan saat preprocessing
- **Interpretasi:** Berisi diskusi berimbang tanpa muatan emosional kuat

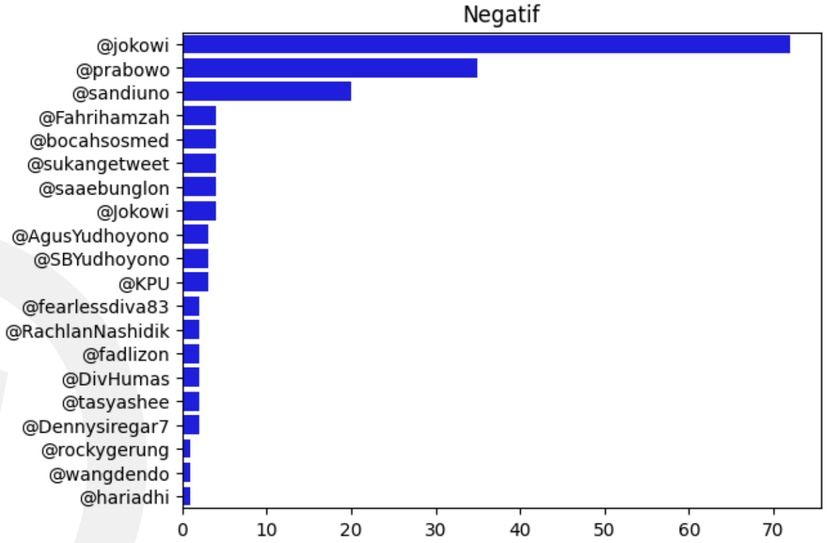


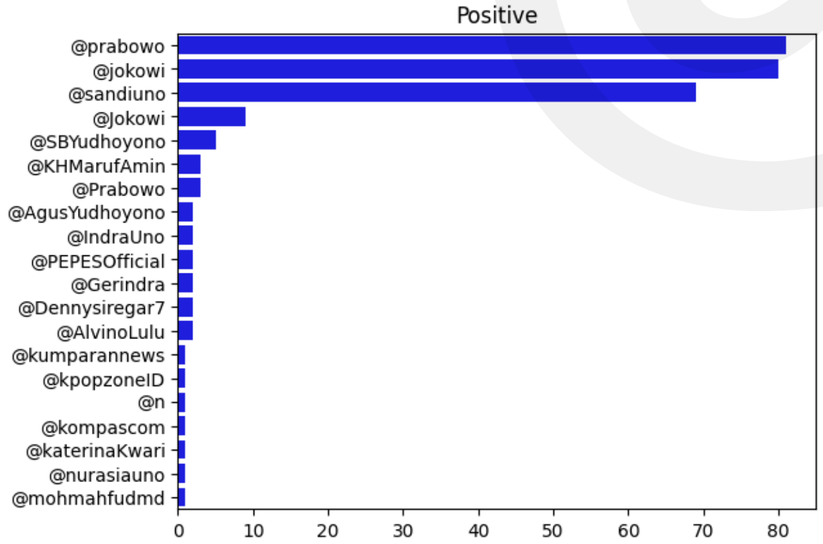


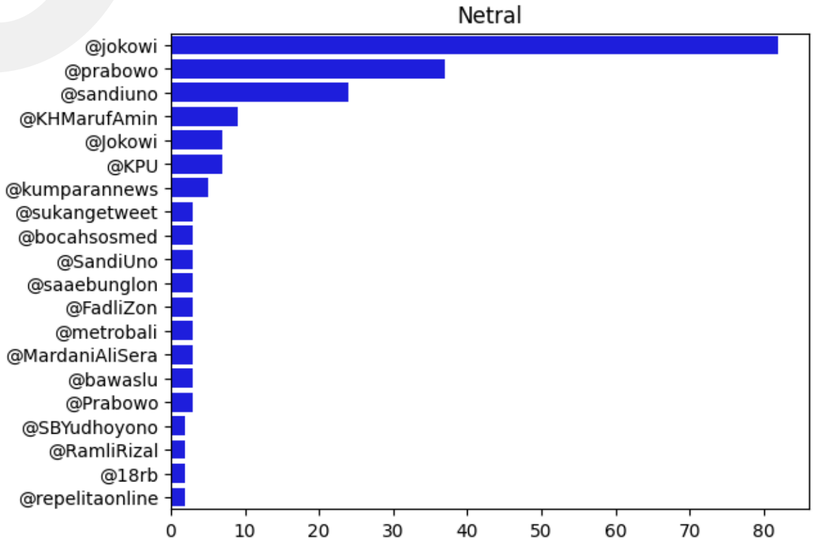


Exploratory Data Analysis (EDA)

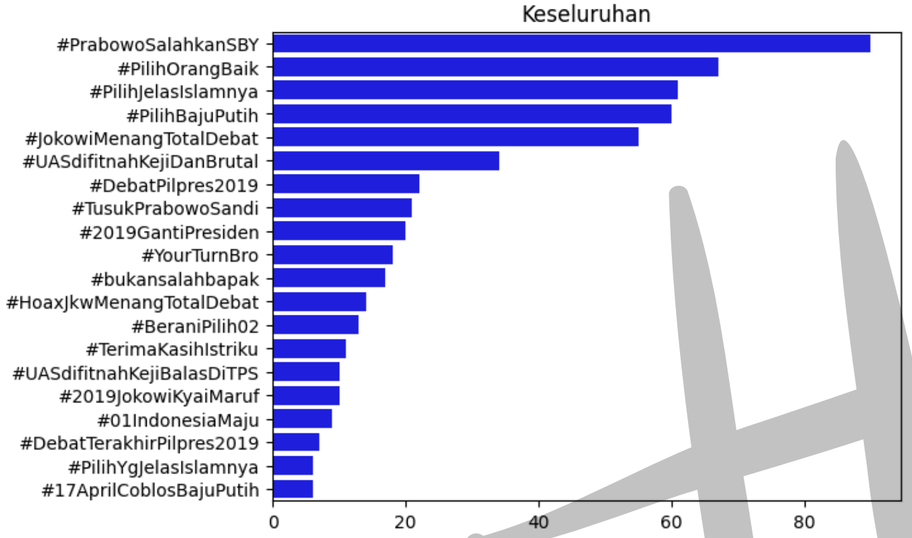


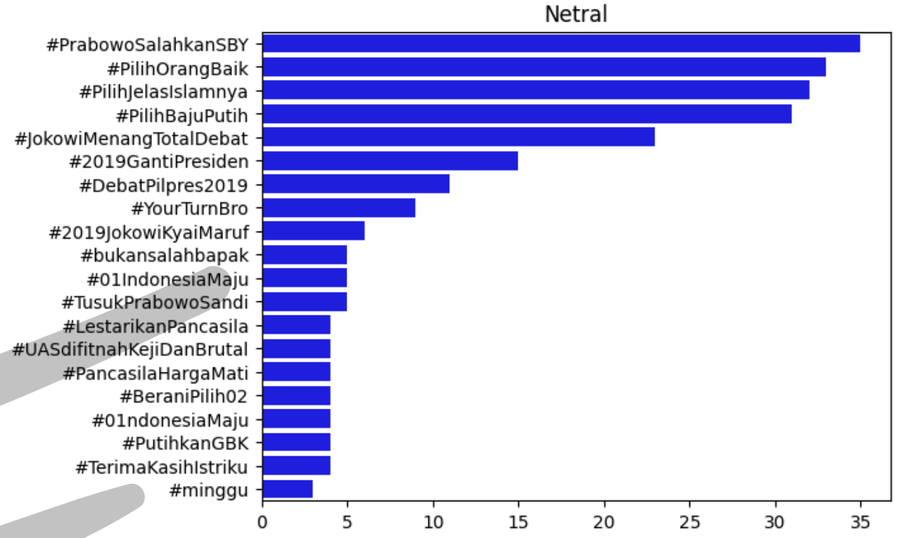


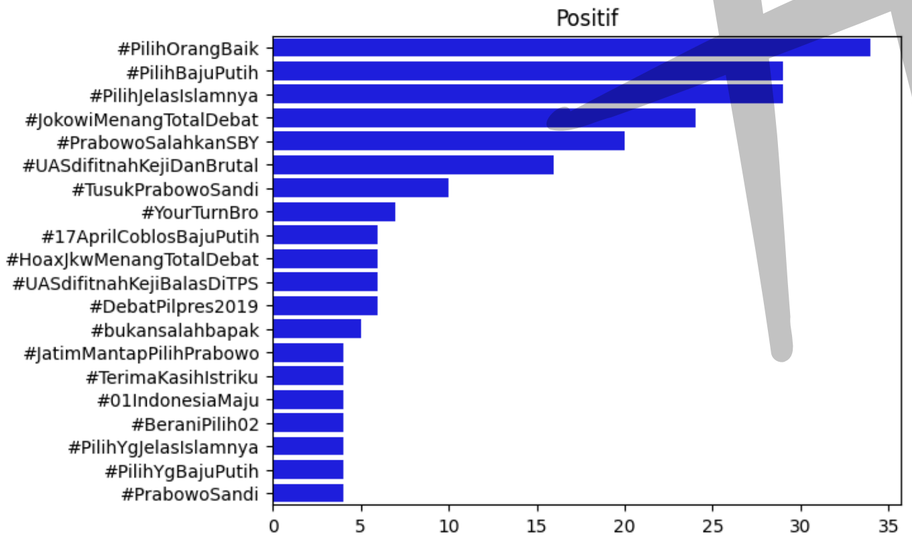


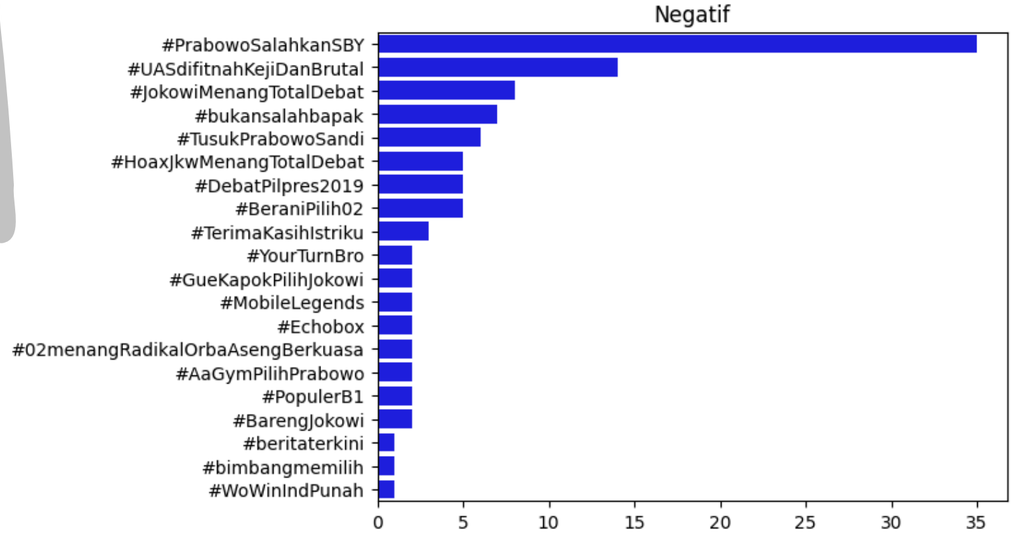


Exploratory Data Analysis (EDA)









Data Preprocessing

```python
def preProcessText(text) :
    global kata_tidak_baku
    hasil = []
    stopWord_List = [
        "di", "ke", "dari", "pada",
        "yang", "untuk", "dengan",
        "dan", "atau", "tetapi", "namun", "serta",
        "jika", "kalau", "agar", "supaya",
        "karena", "sebab", "sehingga", "maka",
        "dalam", "atas", "antara", "oleh",
        "secara", "terhadap", "tentang",
        "sampai", "hingga", "sejak",
        "ini", "itu", "sini", "situ", "sana",
        "ia", "dia", "mereka", "kami", "kita", "kamu", "anda",
        "nya",
        "si", "sang",
        "sangat", "amat", "terlalu",
        "juga", "pula",
        "hanya", "cuma", "sekadar",
        "sudah", "telah", "selesai",
        "akan", "hendak", "mau",
        "sedang", "lagi",
        "pernah",
        "sekali", "saja",
        "nanti", "tadi",
        "baru",
        "wah", "aduh", "ah",
        "adalah", "ialah", "merupakan",
        "ada",
        "lah", "kah", "pun"
    ]
    for sentence in text :
        sentence = sentence.lower()
        sentence = re.sub("(https?://[^\s]+|pic.[^\s]+)", repl = " ",string = sentence)
        sentence = re.sub(pattern = "@[^\s]+", repl = " ", string = sentence)
        sentence = re.sub(pattern = "#[^\s]+", repl = " ", string = sentence)
        sentence = re.sub(pattern = "[^a-zA-Z\s]", repl = " ", string = sentence)
        sentence = re.sub(pattern = "\s+",repl = " ", string = sentence)
        sentence = pipe(sentence)
        sementara = []
        for word in sentence.split(" ") :
            if word in kata_tidak_baku.keys() :
                if kata_tidak_baku[word] not in stopWord_List:
                    sementara.append(kata_tidak_baku[word])
            else :
                if word not in stopWord_List and len(word)>2:
                    sementara.append(word)

        sentence = " ".join(sementara).strip()
        hasil.append(sentence)
    return hasil
```

Data Preprocessing:

- Menghapus stopwords custom yang tidak diperlukan yang berada di dataset.
- Mengubah kata slang menjadi kata lebih lengkap dengan menggunakan Dictionary (kata_tidak_baku variable) misal “gak” menjadi “tidak”, “yg” menjadi “yang”, dll *https://github.com/louisowen6/NLP_bahasa_resources/blob/master/combined_slang_words.txt*
- Lowercasing dan penghapusan URL, Mention, dan Hashtag.
- Pengubahan slang dan converting emoji ke text dengan menggunakan module indoNLP
- Splitting train and test data menjadi ratio 85:15

```python
pd.DataFrame({"Real Text":hehehe,"Processed Text":preProcessText([hehehe])})
```

✓ 0.0s

	Real Text	Processed Text
0	Lha itu artinya ekonomi anda baik"" saja ,,mal...	lah itu artinya ekonomi anda baik saja bahkan ...

Modelling

Traditional Machine Learning Model:

- Random Forest
- Categorical Boosting
- Logistic Regression
- Multinomial Naive Bayes
- Bernoulli Naive Bayes
- XGBoost Classifier
- Voting Classifier Model (Kombinasi dari model diatas)

Metrics Evaluation:

- Accuracy
- Precision
- Recall
- F1-Score

Train Test Split:

- 85 : 15 Ratio for Train and Test Data Splitting

Neural Network Model:

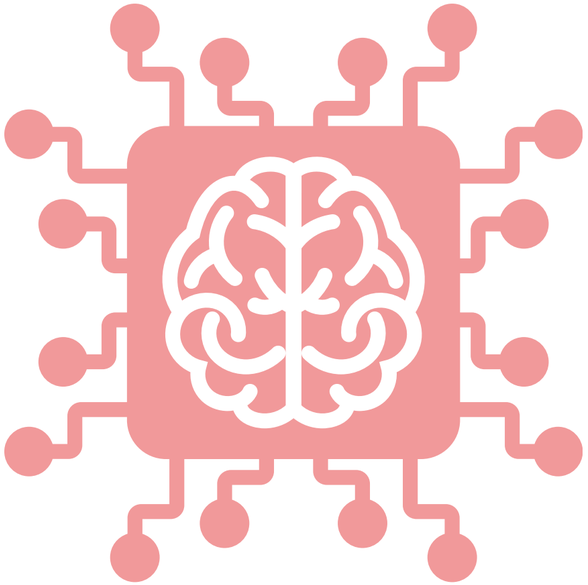
- SimpleRNN
- Long Short Term Memory (LSTM)
- Convolutional Neural Network (CNN)
- Transformer Hidden State Model Pretrained (IndoBERT)+ Simple Keras Model

NEXT STEP → Later:

- Fine Tune the IndoBERT Model Pretrained (not use Hidden State)

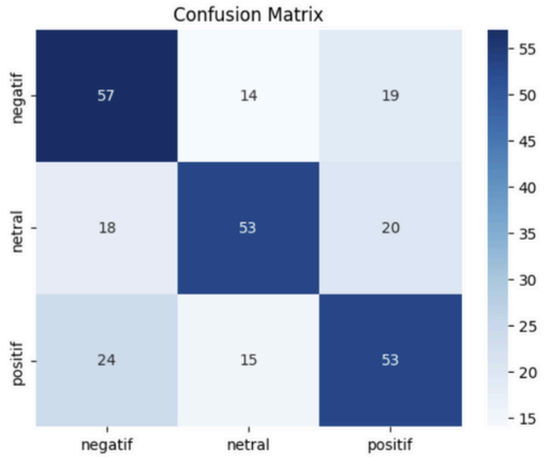
Tokenization Model:

- CountVectorizer from sklearn.feature_extraction import CountVectorizer
- Tokenizer from keras.preprocessing.text import Tokenizer

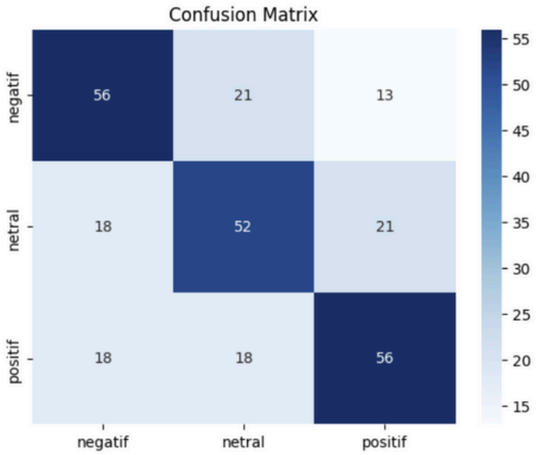


Evaluation

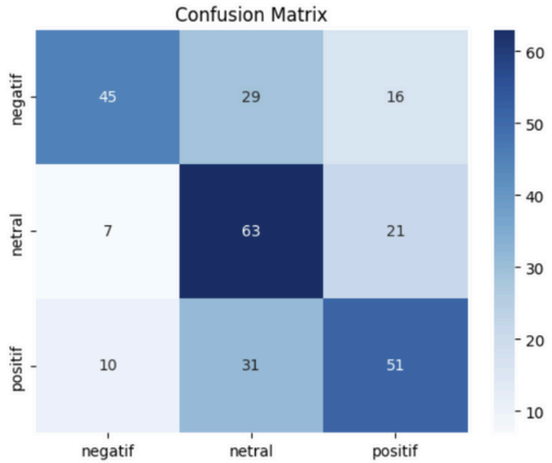
Random Forest



Log Regression



Bernoulli NB

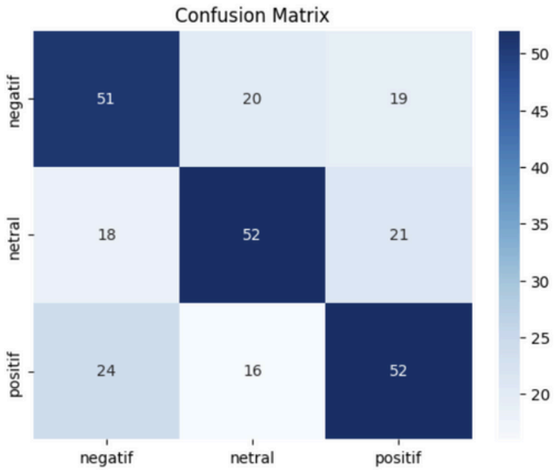


Accuracy:

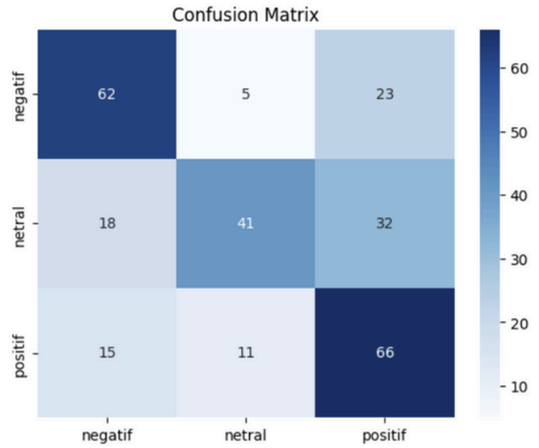
RandomForest	0.597070
LogRegr	0.600733
Multinomial NB	0.619048
CatBoost	0.567766
Bernoulli NB	0.582418
XGBoost	0.538462

Best Model : MultiNomial NB

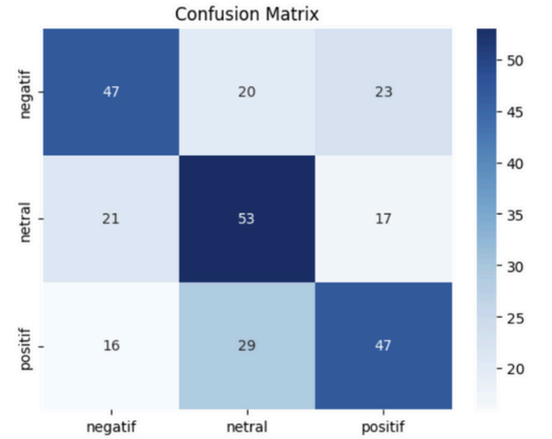
CatBoost



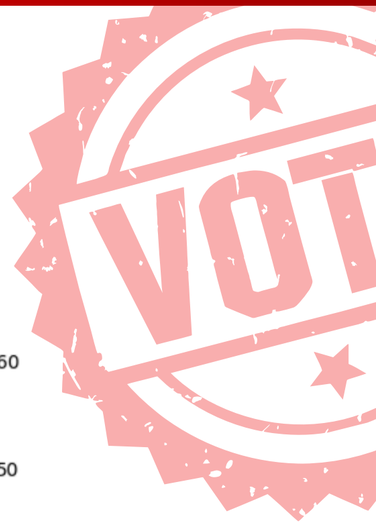
Multinomial NB



XGBoost



Machine Learning Improvement



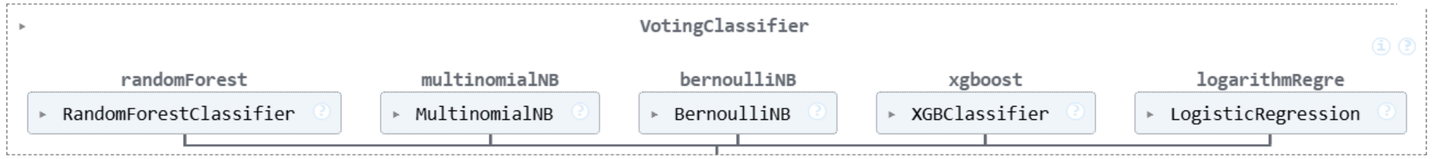
Idea:
Untuk membuat model yang lebih bagus dan memiliki tingkat akurasi yang lebih tinggi, menggunakan model ensemble yaitu VotingClassifier dengan menggabungkan model RandomForest, MultiNomialNB, BernoulliNB, XGBoost, dan LogarithmRegression

```
model_vote = VotingClassifier(estimators = [
    ("randomForest",modelRandomForest),
    ("multinomialNB",modelNBMultinomial),
    ("bernoulliNB",modelNBBernoulli),
    ("xgboost",modelXGBoost),
    ("logarithmRegre",model_log)
], weights = [4,5,3,2,4], n_jobs = -1, voting="hard")
```

✓ 0.0s

```
model_vote.fit(train_data_vectorizer.toarray(), train_label)
```

✓ 1m 9.1s

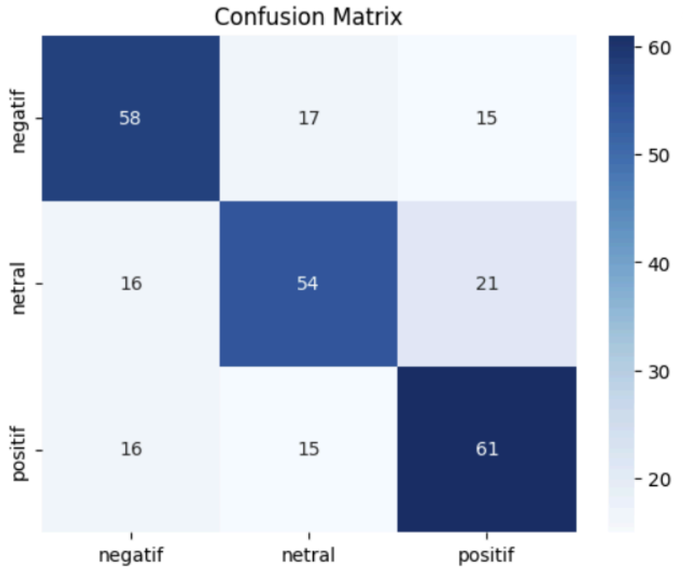



```
accuracy = accuracy_score(y_true = test_label, y_pred = prediksi)
print(f"Accuracy Voting Classifier : {accuracy}")
```

✓ 0.0s

```
Accuracy Voting Classifier : 0.6336996336996337
```

Voting Classifier



	precision	recall	f1-score	support
0	0.64	0.64	0.64	90
1	0.63	0.59	0.61	91
2	0.63	0.66	0.65	92
accuracy			0.63	273
macro avg	0.63	0.63	0.63	273
weighted avg	0.63	0.63	0.63	273

Neural Network Model - Started

```python
def preProcessText(text) :
    global kata_tidak_baku
    # lemma_model = Lemmatizer()
    hasil = []
    # stopWord_List = StopWord().get_stopword()
    # for janganHapus in ["tidak","bukan","gak","tdk","bkn","belum","jangan"] :
    #     if janganHapus in stopWord_List :
    #         stopWord_List.remove(janganHapus)
    # stopWord_List.extend(["yg", "aja", "ya", "kalo", "nya", "utk", "untuk"])
    for sentence in text :
        sentence = sentence.lower()
        sentence = re.sub("(https?://[^\s]+|pic.[^\s]+)", repl = " ",string = sentence)
        sentence = re.sub(pattern = "@[^\s]+", repl = " ", string = sentence)
        sentence = re.sub(pattern = "#[^\s]+", repl = " ", string = sentence)
        sentence = re.sub(pattern = "[^a-zA-Z\s]", repl = " ", string = sentence)
        sentence = re.sub(pattern = "\s+",repl = " ", string = sentence)
        sentence = pipe(sentence)
        sementara = []
        for word in sentence.split(" ") :
            if word in kata_tidak_baku.keys() :
                sementara.append(kata_tidak_baku[word])
            else :
                sementara.append(word)

        sentence = " ".join(sementara).strip()
        hasil.append(sentence)
    return hasil
```

Data Preprocessing di Neural Network:

- **Tidak** menghapus stopwords seperti pada model Machine Learning sebelumnya, karena mungkin stopwords akan memiliki makna berharga jika di process menggunakan prediction based vectorizer (from keras.preprocessing.text)
- Mengubah kata slang menjadi kata lebih lengkap dengan menggunakan Dictionary (kata_tidak_baku variable) misal “gak” menjadi “tidak”, “yg” menjadi “yang”, dll *https://github.com/louisowen6/NLP_bahasa_resources/blob/master/combined_slang_words.txt*
- Lowercasing dan penghapusan URL, Mention, dan Hashtag.
- Pengubahan slang dan converting emoji ke text dengan menggunakan module indoNLP
- Splitting train and test data menjadi ratio 80:20

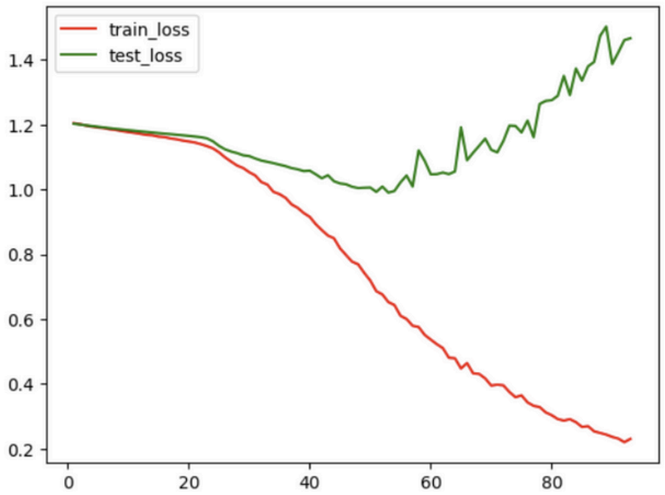
Simple RNN

```
model_simpleRNN = Sequential()
model_simpleRNN.add(Input([70]))
model_simpleRNN.add(Embedding(input_dim = 4001, output_dim = 32))

model_simpleRNN.add(SimpleRNN(units = 64, activation = "relu", kernel_regularizer = l2(l2 = 0.001)))

model_simpleRNN.add(Dense(units = 64, activation = "relu",kernel_regularizer = l2(l2 =0.001)))
model_simpleRNN.add(Dropout(rate =0.4))

model_simpleRNN.add(Dense(units = 3, activation = "softmax"))

model_simpleRNN.compile(loss = "sparse_categorical_crossentropy", optimizer = RMSprop(learning_rate=5e-5),metrics = ["accuracy"])

history_simpleRNN = model_simpleRNN.fit(train_data_vectorizer, train_label, validation_data = (test_data_vectorizer,test_label), epochs = 100,


prediksi = model_simpleRNN.predict(test_data_vectorizer)
sns.heatmap(confusion_matrix(y_pred = np.argmax(prediksi,-1),y_true = test_label),annot = True,fmt = "d",cmap="Blues")
acc_simpleRNN = accuracy_score(y_pred = np.argmax(prediksi,-1),y_true = test_label)
print(f"Accuracy Score : {acc_simpleRNN}")
```

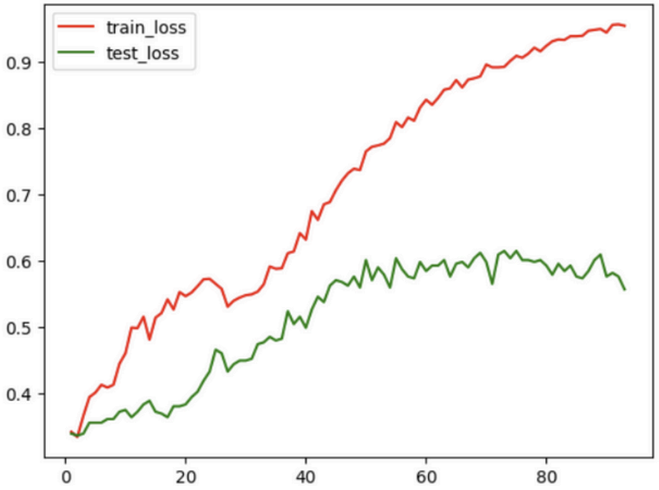
```
Epoch 73/100
91/91 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - a
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
Accuracy Score : 0.6115702479338843
```

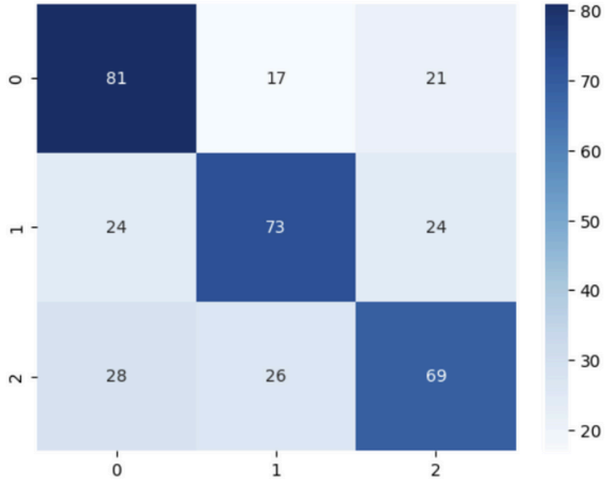
- Batch size : 16 (reduce overfit)
- Optimizer : RMSprop lr = 5e-5 (kecil)
- SimpleRNN units = 64 dengan L2
- Dense dengan unit kecil 64
- Menggunakan 1 layer Hidden saja (belum termasuk SimpleRNN)

Loss



Accuracy





Long Short Term Memory (LSTM)

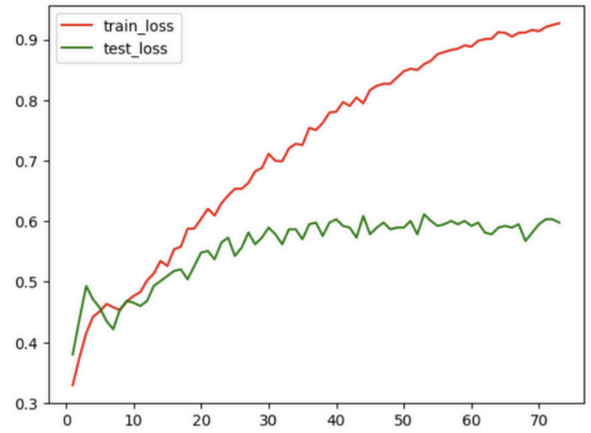
```
model_LSTM = Sequential()
model_LSTM.add(Input([70]))
model_LSTM.add(Embedding(input_dim = 4001, output_dim = 32))

model_LSTM.add(LSTM(units = 32))

model_LSTM.add(Dense(units = 32, activation = "relu",kernel_regularizer = l2(l2 =0.001)))
model_LSTM.add(Dropout(rate =0.4))

model_LSTM.add(Dense(units = 3, activation = "softmax"))

model_LSTM.compile(loss = "sparse_categorical_crossentropy", optimizer = RMSprop(learning_rate=0.0001),metrics = ["accuracy"])

history_LSTM = model_LSTM.fit(train_data_vectorizer, train_label, validation_data = (test_data_vectorizer,test_label), epochs = 100, batch_size = 16, call


prediksi = model_LSTM.predict(test_data_vectorizer)
sns.heatmap(confusion_matrix(y_pred = np.argmax(prediksi,-1),y_true = test_label),annot = True,fmt = "d",cmap="Blues")
acc_LSTM = accuracy_score(y_pred = np.argmax(prediksi,-1),y_true = test_label)
print(f"Accuracy Score : {acc_LSTM}")
```

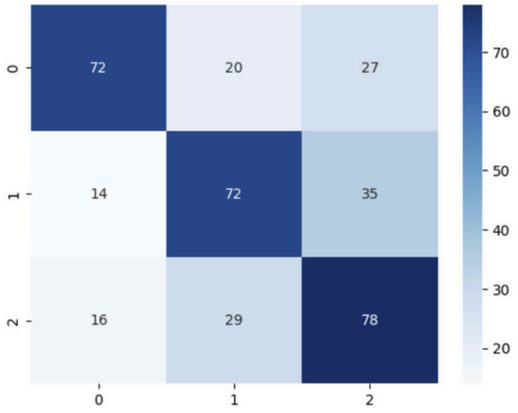
```
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/st
Accuracy Score : 0.6115702479338843
```

- Batch size : 16 (reduce overfit)
- Optimizer : RMSprop lr = 5e-5 (kecil)
- LSTM units = 32 tanpa dropout recurrent
- Dense dengan unit kecil 64
- Menggunakan 1 layer Hidden saja (belum termasuk LSTM)

Loss

Accuracy





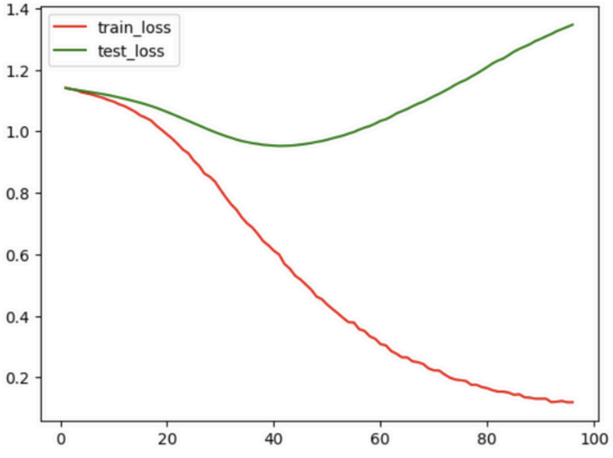
Convolutional Neural Network (CNN)

```
model_CNN = Sequential()
model_CNN.add(Input([70]))
model_CNN.add(Embedding(input_dim = 4001, output_dim = 32))

model_CNN.add(Conv1D(64, kernel_size = 2, strides=2))
model_CNN.add(GlobalMaxPooling1D())

model_CNN.add(Dense(units = 32, activation = "relu",kernel_regularizer = l2(l2 =0.001)))
model_CNN.add(Dropout(rate =0.3))

model_CNN.add(Dense(units = 3, activation = "softmax"))

model_CNN.compile(loss = "sparse_categorical_crossentropy", optimizer = RMSprop(learning_rate=0.0001),metrics = ["accuracy"])

history_CNN = model_CNN.fit(train_data_vectorizer, train_label, validation_data = (test_data_vectorizer,test_label), epochs = 100, batch_size = 16, callb


prediksi = model_CNN.predict(test_data_vectorizer)
sns.heatmap(confusion_matrix(y_pred = np.argmax(prediksi,-1),y_true = test_label),annot = True,fmt = "d",cmap="Blues")
acc_CNN = accuracy_score(y_pred = np.argmax(prediksi,-1),y_true = test_label)
print(f"Accuracy Score : {acc_CNN}")
```

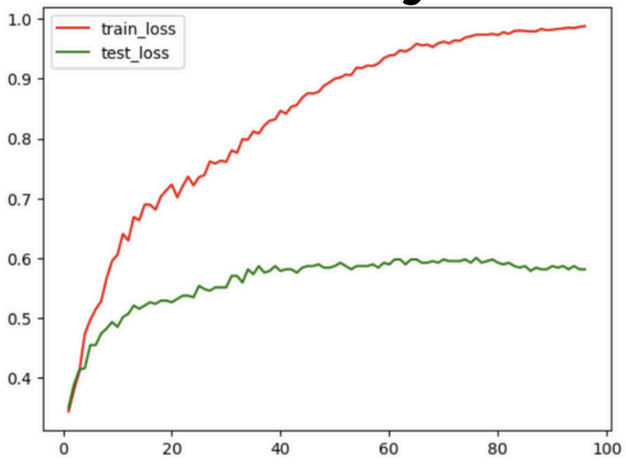
```
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
Accuracy Score : 0.6005509641873278
```

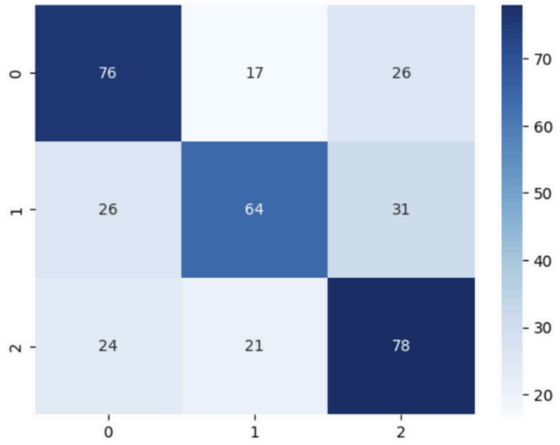
- Batch size : 16 (reduce overfit)
- Optimizer : RMSprop lr = 5e-5 (kecil)
- CNN units = 64 dengan kernel = 3 dan strides = 2 tanpa padding
- Dense dengan unit kecil 64
- Menggunakan 1 layer Hidden saja (belum termasuk Conv1D dan MaxPool Layer nya)

Loss



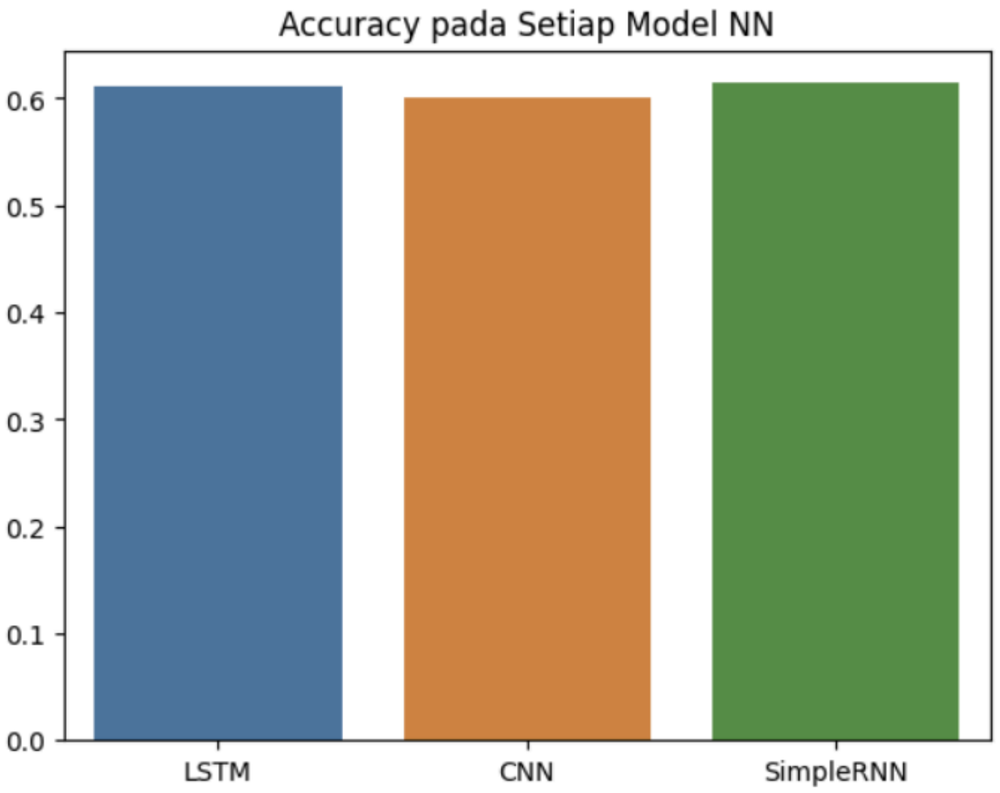
Accuracy





Neural Network Models (Conclussion)

	LSTM	CNN	SimpleRNN
0	0.61157	0.600551	0.614325

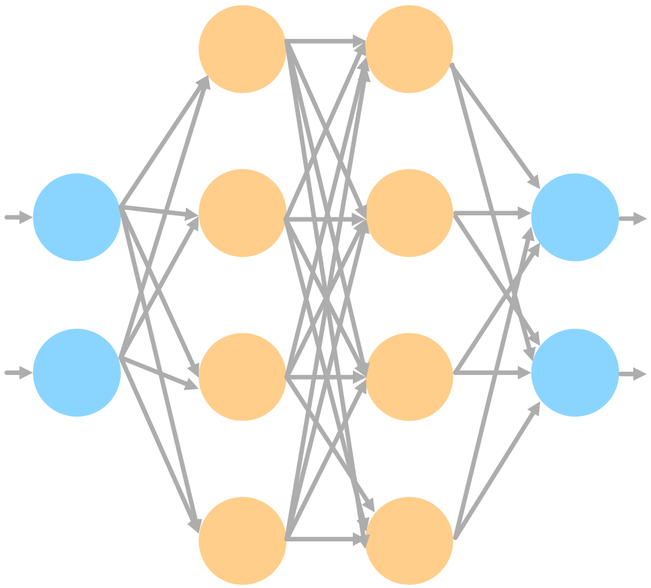


Accuracy pada Setiap Model NN:

Grafik ini menunjukkan akurasi dari tiga model Neural Network yang berbeda: LSTM, CNN, dan SimpleRNN.

LSTM: Akurasi sebesar 0.61157
CNN: Akurasi sebesar 0.60055
SimpleRNN: Akurasi sebesar 0.61433

SimpleRNN memiliki akurasi tertinggi di antara ketiga model, diikuti oleh LSTM, dan CNN yang sedikit lebih rendah.



Neural Network + IndoBERT

```python
def preProcessText(text) :
    global kata_tidak_baku
    hasil = []
    for sentence in text :
        sentence = sentence.lower()
        sentence = re.sub("(https?://[^\s]+|pic.[^\s]+)", repl = " ",string = sentence)
        sentence = re.sub(pattern = "@[^\s]+", repl = " ", string = sentence)
        sentence = re.sub(pattern = "#[^\s]+", repl = " ", string = sentence)
        sentence = re.sub(pattern = "[^a-zA-Z\s]", repl = " ", string = sentence)
        sentence = re.sub(pattern = "\s+",repl = " ", string = sentence)
        sentence = pipe(sentence)
        sementara = []
        for word in sentence.split(" ") :
            if word in kata_tidak_baku.keys() :
                sementara.append(kata_tidak_baku[word])
            else :
                sementara.append(word)

        sentence = " ".join(sementara).strip()
        hasil.append(sentence)
    return hasil
```

✓ 0.0s

```python
model_tokenizer_transformer = AutoTokenizer.from_pretrained("indobenchmark/indobert-base-p1")
model_transformer = TFAutoModel.from_pretrained("indobenchmark/indobert-base-p1")
```

✓ 9.7s

```
WARNING:tensorflow:From d:\Conda\envs\ReynaldiENV\lib\site-packages\tf_keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.c

WARNING:tensorflow:From d:\Conda\envs\ReynaldiENV\lib\site-packages\tf_keras\src\backend.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_defau

TensorFlow and JAX classes are deprecated and will be removed in Transformers v5. We recommend migrating to PyTorch classes or pinning your version of Transformers.
Some layers from the model checkpoint at indobenchmark/indobert-base-p1 were not used when initializing TFBertModel: ['mlm___cls', 'nsp___cls']
- This IS expected if you are initializing TFBertModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequence
- This IS NOT expected if you are initializing TFBertModel from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification
All the layers of TFBertModel were initialized from the model checkpoint at indobenchmark/indobert-base-p1.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFBertModel for predictions without further training.
```

Data Preprocessing di Neural Network + Transformers :

- **Tidak** menghapus stopwords seperti pada model Machine Learning sebelumnya, karena mungkin stopwords akan memiliki makna berharga jika di process menggunakan prediction based vectorizer (from keras.preprocessing.text)
- Mengubah kata slang menjadi kata lebih lengkap dengan menggunakan Dictionary (kata_tidak_baku variable) misal “gak” menjadi “tidak”, “yg” menjadi “yang”, dll *https://github.com/louisowen6/NLP_bahasa_resources/blob/master/combined_slang_words.txt*
- Lowercasing dan penghapusan URL, Mention, dan Hashtag.
- Pengubahan slang dan converting emoji ke text dengan menggunakan module indoNLP
- Splitting train and test data menjadi ratio 80:20

Neural Network + IndoBERT

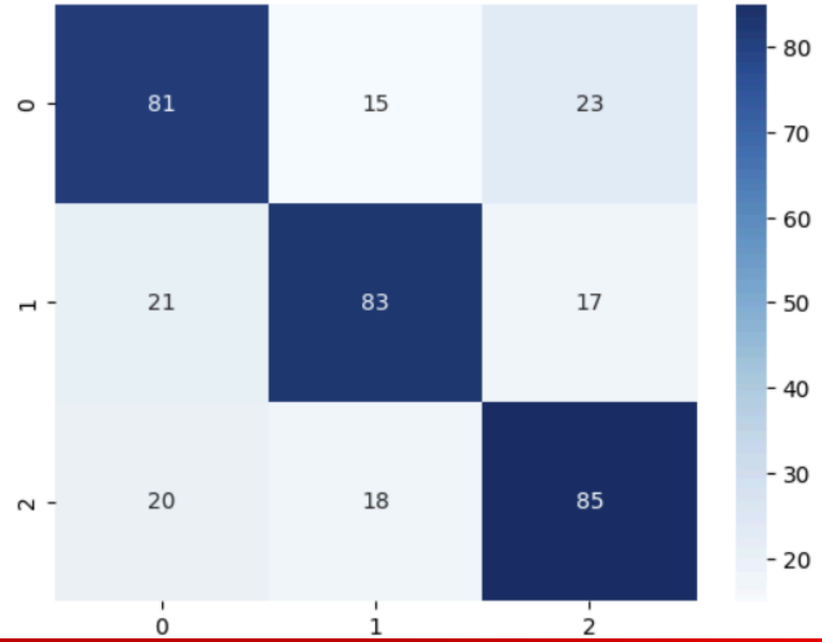
```
model = Sequential()
model.add(Input(shape = [768]))

model.add(Dense(units = 64, activation = "relu", kernel_regularizer = l2(l2 = 0.0001)))
model.add(Dropout(0.5))

model.add(Dense(units = 3, activation="softmax"))

model.compile(loss = "sparse_categorical_crossentropy", optimizer=RMSprop(learning_rate=5e-5))
```

```
def tokenization(batch) :
    tokenizationResult = model_tokenizer_transformer(batch["text"], padding = True, truncation = True, max_length=100)
    return tokenizationResult
```

✓ 0.0s

```
def input_to_model(batch) :
    input_to_model = {"input_ids":batch["input_ids"], "token_type_ids":batch["token_type_ids"], "attention_mask":batch["attention_mask"]}
    with torch.no_grad() :
        hasil = model_transformer(**input_to_model)["last_hidden_state"][:,0,:]
    return {"hidden_state":hasil}
```

Accuracy Score : 0.6859504132231405



	precision	recall	f1-score	support
0	0.66	0.68	0.67	119
1	0.72	0.69	0.70	121
2	0.68	0.69	0.69	123
accuracy			0.69	363
macro avg	0.69	0.69	0.69	363
weighted avg	0.69	0.69	0.69	363

Thank You

Twitter Sentiment Analysis using NLP