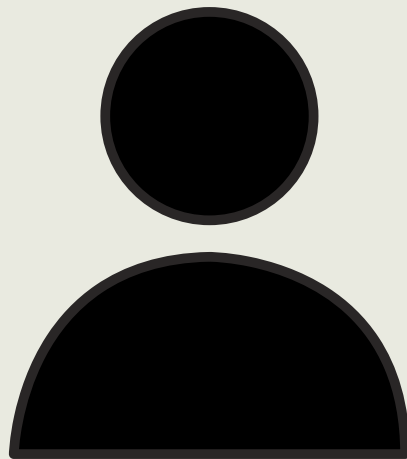


ANALYZING YOUTUBE GAMING COMMENTS: LIKE COUNT REGRESSION AND SENTIMENT- EMOTION CLASSIFICATION

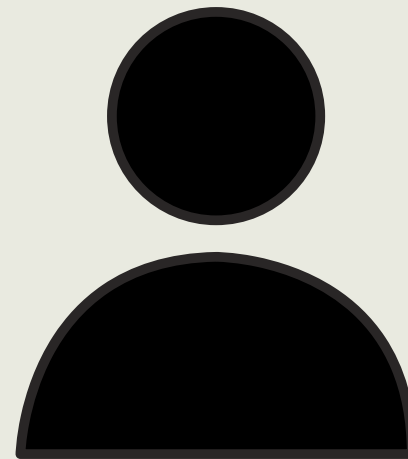
Kelompok 7

MEET THE TEAM



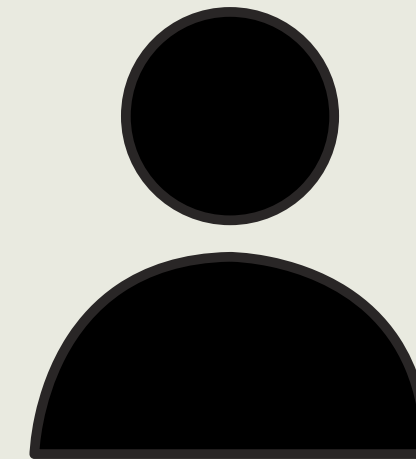
Alfito Faiz Rizqi

2702316724



Bryan Cristhoper Chandra
Putra

2702272306



Reuben Anselmus Adel

2702314901

Latar Belakang

Perkembangan teknologi digital telah mengubah cara manusia berinteraksi, termasuk melalui media sosial seperti YouTube. Konten bertema gaming menjadi salah satu yang paling populer, dengan kolom komentar yang aktif dan mencerminkan opini serta emosi pengguna. Meskipun komentar-komentar ini mengandung informasi yang bernilai, data teks tersebut masih jarang dimanfaatkan secara optimal. Dengan bantuan Natural Language Processing (NLP) dan machine learning, komentar dapat dianalisis untuk memprediksi jumlah like serta mengklasifikasikan sentimen dan emosi. Penelitian ini bertujuan untuk membangun model regresi yang memprediksi jumlah like berdasarkan isi komentar, serta melakukan klasifikasi sentimen dan emosi guna memahami pola interaksi pengguna dalam komunitas YouTube gaming.

DATASET

DATASET

- Data dikumpulkan otomatis menggunakan YouTube API, mengambil komentar dari berbagai YouTuber.
- Komentar dikategorikan ke dalam tiga bin popularitas berdasarkan jumlah like:
- Low (≤ 100 like), Medium (101–500), High (> 500)
- Untuk mencegah ketimpangan data, dilakukan proses trimming:
- Setiap YouTuber hanya menyumbang jumlah komentar yang sama dari tiap bin (mengikuti jumlah bin terkecil).
- Jika ada bin yang kosong, komentar dari YouTuber tersebut tidak digunakan.

VISUALISASI DATA

LIST NAMA YOUTUBE CHANEL (Gambar 2.1)

youtuber_name	
1.	Dream
2.	DanTDM
3.	Technoblade
4.	PewDiePie
5.	Ludwig
6.	MrBeast
7.	jacksepticeye
8.	Markiplier
9.	TommyInnit
10.	Asmongold TV
11.	penguinz0

1 - 21 / 21 < >

Total Like per Channel (Gambar 2.2)

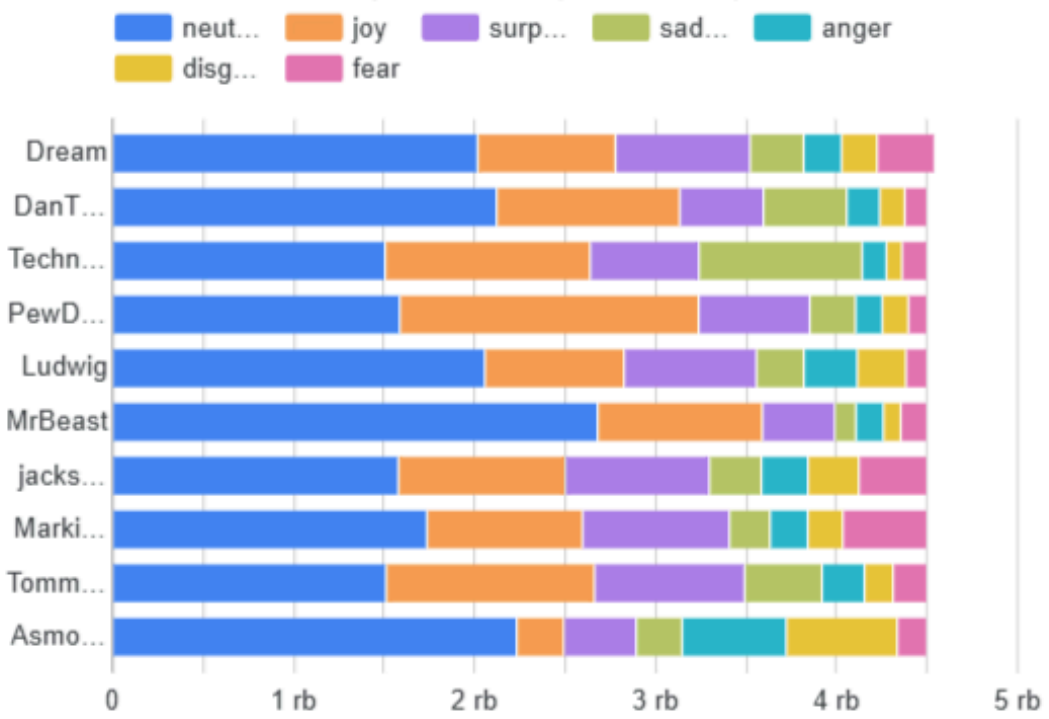
youtuber_name ▾		like_co...
1.	stampylonghead	1.001.947
2.	penguinz0	5.648.944
3.	ohnepixel	1.229.194
4.	jacksepticeye	6.149.007
5.	VanossGaming	3.142.506
6.	TommyInnit	6.946.752
7.	TenZ	12.189
8.	Technoblade	6.899.792
9.	Shroud	2.993.258

1 - 21 / 21 < >

Sentiment per Channel (Gambar 2.3)

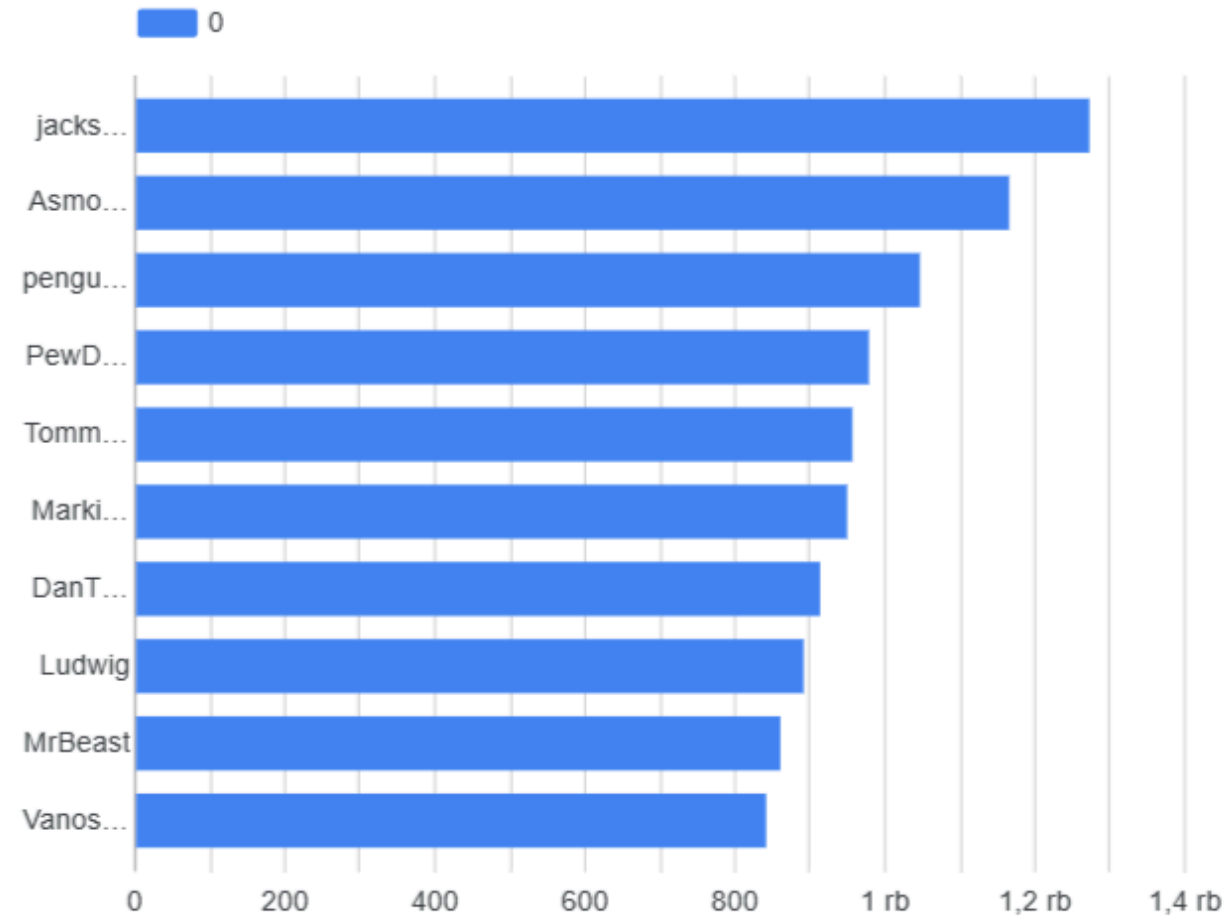
youtuber_name	sentiment	comment
Dream	positive	2.782
	negative	1.623
	neutral	137
PewDiePie	positive	3.037
	negative	1.357
	neutral	106
TommyInnit	positive	2.774
	negative	1.586
	neutral	140
...

Emotion per Channel (Gambar 2.4)

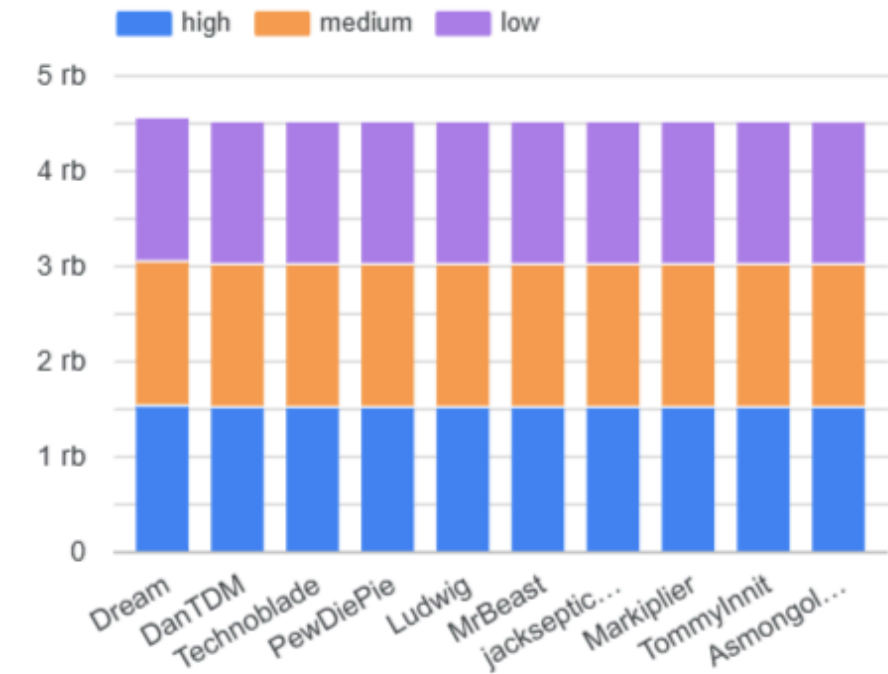


VISUALISASI DATA

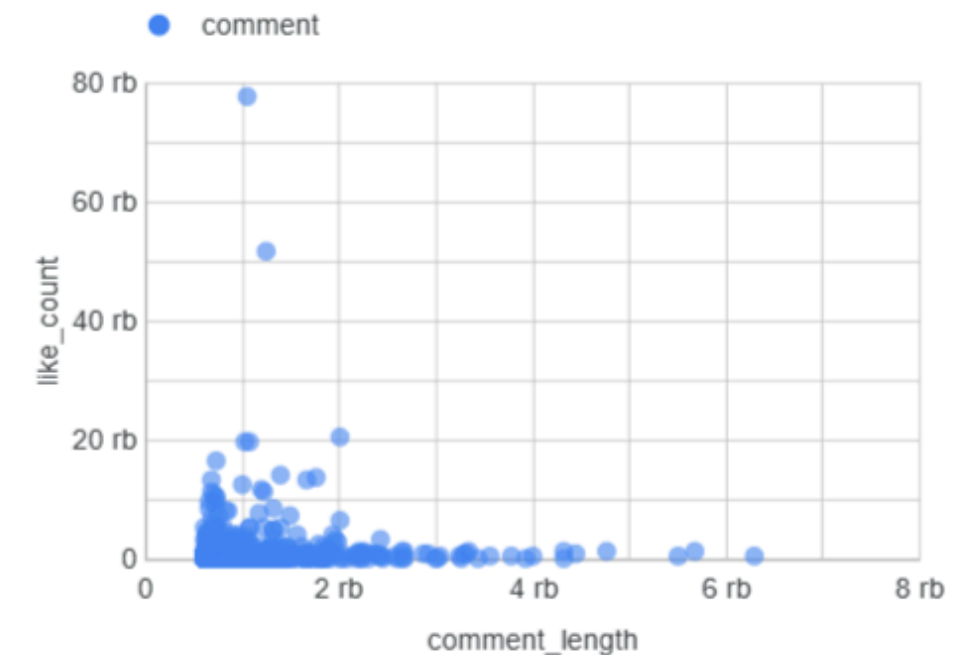
Jumlah Data dengan like_count = 0 (Gambar 2.5)



Like Count (Low, Medium, High) per Channel (Gambar 2.6)



Panjang Komentar vs Jumlah Like (Gambar 2.7)



DATASET

```
def get_bin_category(like_count):  
    """Categorizes like_count into bins."""  
    if like_count <= 100:  
        return "low"  
    elif like_count <= 500:  
        return "medium"  
    else:  
        return "high"
```

```
def trim_comments_for_youtuber(df_youtuber):  
    """  
    Trims the comment bins for a single YouTuber's DataFrame.  
    All bins ('low', 'medium', 'high') will be reduced to the size of the smallest bin.  
    """  
    if df_youtuber.empty:  
        return pd.DataFrame()  
  
    # Count comments in each bin for this YouTuber  
    bin_counts = df_youtuber['bin_category'].value_counts()  
  
    # Initialize counts for all bins, in case some are missing for a YouTuber  
    low_count = bin_counts.get('low', 0)  
    medium_count = bin_counts.get('medium', 0)  
    high_count = bin_counts.get('high', 0)  
  
    min_bin_size = min(low_count, medium_count, high_count)  
  
    if min_bin_size == 0:  
        print(f"Youtuber {df_youtuber['youtuber_id'].iloc[0]} has a bin with 0 comments. "  
              f"Effective target size is 0, so this YouTuber will have no comments in the trimmed set.")  
        return pd.DataFrame(columns=df_youtuber.columns)
```


TRAINING

TRAINING CODE

Pada proyek ini, kita akan menggunakan dua model, yaitu RoBERTa untuk klasifikasi sentiment dan emotion, serta XGBoostRegressor untuk melakukan regresi dalam memprediksi jumlah like dari komentar.

RoBERTa adalah model transformer yang mengubah teks menjadi representasi numerik (embedding) yang kaya konteks, sehingga memudahkan komputer memahami makna kalimat secara mendalam. Model ini sangat efektif untuk berbagai tugas pemrosesan bahasa alami.

XGBoost adalah algoritma boosting berbasis pohon keputusan yang kuat dan cepat, digunakan untuk regresi dan klasifikasi. Dalam proyek ini, XGBoost memprediksi jumlah like komentar menggunakan embedding dari RoBERTa, dengan tuning hyperparameter untuk hasil optimal.

ROBERTA

```
if CURRENT_TASK_TYPE == 'sentiment':
    TEXT_COLUMN = 'comment'
    LABEL_COLUMN = 'sentiment'
    SENTIMENT_LABELS = ['positive', 'negative', 'neutral']
    NUM_LABELS = len(SENTIMENT_LABELS)
    MODEL_OUTPUT_DIR = os.path.join(GOOGLE_DRIVE_BASE_PATH, 'results_sentiment')
    SAVED_MODEL_DIR = os.path.join(GOOGLE_DRIVE_BASE_PATH, 'saved_model_sentiment')
elif CURRENT_TASK_TYPE == 'emotion':
    TEXT_COLUMN = 'comment'
    LABEL_COLUMN = 'emotion'

    EMOTION_LABELS_FROM_USER = [
        "joy", "sadness", "anger", "fear", "disgust", "surprise", "neutral"
    ]
    NUM_LABELS = len(EMOTION_LABELS_FROM_USER)
    MODEL_OUTPUT_DIR = os.path.join(GOOGLE_DRIVE_BASE_PATH, 'results_emotion')
    SAVED_MODEL_DIR = os.path.join(GOOGLE_DRIVE_BASE_PATH, 'saved_model_emotion')
```

Bagian ini menentukan jenis tugas yang akan dijalankan, apakah klasifikasi sentiment atau emotion. Berdasarkan tugas tersebut, variabel penting seperti nama kolom data, label, jumlah kelas, dan folder penyimpanan model diatur secara otomatis.

Bagian ini melakukan pembersihan dan pemrosesan label tergantung dari jenis tugas klasifikasi yang sedang dikerjakan (sentiment atau emotion), agar label bisa dipetakan ke dalam bentuk numerik (label_id) yang bisa digunakan oleh model.

```
if CURRENT_TASK_TYPE == 'sentiment':
    df[LABEL_COLUMN] = df[LABEL_COLUMN].str.lower()
    df = df[df[LABEL_COLUMN].isin([l.lower() for l in SENTIMENT_LABELS])]
    label_map = {label: i for i, label in enumerate(SENTIMENT_LABELS)}
    df['label_id'] = df[LABEL_COLUMN].map(label_map)

elif CURRENT_TASK_TYPE == 'emotion':
    df[LABEL_COLUMN] = df[LABEL_COLUMN].apply(clean_emotion_label)
    df.dropna(subset=[LABEL_COLUMN], inplace=True)
    df = df[df[LABEL_COLUMN].isin([l.lower() for l in EMOTION_LABELS_FROM_USER])]
    label_map = {label: i for i, label in enumerate(EMOTION_LABELS_FROM_USER)}
    df['label_id'] = df[LABEL_COLUMN].map(label_map)
```


ROBERTA

```
print("Tokenizing data...")
tokenizer = RobertaTokenizerFast.from_pretrained(MODEL_NAME)

def tokenize_function(examples):
    return tokenizer(examples[TEXT_COLUMN], padding="max_length", truncation=True, max_length=128)

hg_dataset = Dataset.from_pandas(df[[TEXT_COLUMN, 'label_id']].rename(columns={'label_id': 'labels'}))

# Tokenize the dataset
tokenized_dataset = hg_dataset.map(tokenize_function, batched=True)
```

Teks dari dataset diubah menjadi token numerik menggunakan tokenizer dari model RoBERTa. Tokenisasi ini menstandarkan panjang input agar sesuai dengan format yang bisa diproses model.

Kode ini membagi dataset token menjadi data pelatihan dan evaluasi. Jika jumlah data sangat kecil (di bawah 20), sistem menyesuaikan: semua data dipakai jika ≤ 2 , atau dibagi sekitar 90:10 jika lebih dari itu. Jika data ≥ 20 , pembagian standar 80:20 digunakan. Tujuannya agar proses tetap berjalan meski datanya sedikit.

```
if len(tokenized_dataset) < 20:
    print("Warning: Dataset is very small. Using a larger portion for training or all data if less than a few samples for eval.")
    if len(tokenized_dataset) <= 2:
        train_dataset = tokenized_dataset
        eval_dataset = tokenized_dataset
        print("Dataset too small to split meaningfully. Using all data for training and evaluation.")
    else:
        split_datasets = tokenized_dataset.train_test_split(test_size=max(1, int(len(tokenized_dataset)*0.1)), shuffle=True, seed=42)
        train_dataset = split_datasets['train']
        eval_dataset = split_datasets['test']
else:
    split_datasets = tokenized_dataset.train_test_split(test_size=0.2, shuffle=True, seed=42)
    train_dataset = split_datasets['train']
    eval_dataset = split_datasets['test']
```

ROBERTA

```
print("Loading pre-trained model...")
model = RobertaForSequenceClassification.from_pretrained(
    MODEL_NAME,
    num_labels=NUM_LABELS,
    id2label={i: label for i, label in enumerate(label_map.keys())}, # For inference label names
    label2id=label_map
)

training_args = TrainingArguments(
    output_dir=MODEL_OUTPUT_DIR,
    num_train_epochs=2,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=64,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir=os.path.join(GOOGLE_DRIVE_BASE_PATH, 'logs', CURRENT_TASK_TYPE.lower()),
    logging_steps=200,
    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    metric_for_best_model="f1",
    fp16=torch.cuda.is_available(),
    learning_rate=2e-5,
)
```

Bagian ini membuat objek Trainer dari Hugging Face untuk melatih model RoBERTa dengan dataset, tokenizer, dan metrik evaluasi yang telah disiapkan. Proses pelatihan dijalankan, dan jika terjadi error akan ditangani serta ditampilkan.

Kode ini memuat model RoBERTa yang telah dilatih sebelumnya dan mengatur parameter pelatihan seperti jumlah epoch, batch size, learning rate, dan strategi evaluasi. Model dikonfigurasi untuk tugas klasifikasi teks dengan label yang sudah dipetakan, serta menyimpan model terbaik berdasarkan skor F1.

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset_dict['train'],
    eval_dataset=dataset_dict['eval'],
    compute_metrics=compute_metrics,
    tokenizer=tokenizer,
)

print(f"Starting training for {CURRENT_TASK_TYPE} task...")
try:
    trainer.train()
except Exception as e:
    print(f"An error occurred during training: {e}")
    exit()

print("Training complete.")
```

XGBOOST

```
df.dropna(subset=['comment', 'emotion', 'sentiment', 'like_count'], inplace=True)

df['text_input'] = df['comment'].astype(str) + ' [SEP] ' + df['emotion'].astype(str) + ' [SEP] ' + df['sentiment'].astype(str)

df['like_count'] = pd.to_numeric(df['like_count'], errors='coerce')
df.dropna(subset=['like_count'], inplace=True)
df['like_count'] = df['like_count'].astype(int)
```

Menggabungkan komentar, emosi, dan sentimen menjadi satu string input yang akan diolah oleh BERT untuk menangkap konteks yang lebih kaya.

Fungsi ini menghasilkan embedding vektor untuk tiap input teks dengan mengambil representasi [CLS] dari output BERT yang digunakan sebagai representasi kalimat.

```
def get_bert_embeddings(texts, batch_size=64):
    model.eval()
    all_embeddings = []
    for i in tqdm(range(0, len(texts), batch_size), desc="Generating Embeddings"):
        batch_texts = texts[i:i+batch_size].tolist()
        inputs = tokenizer(batch_texts, return_tensors='pt', truncation=True, padding=True, max_length=128)
        inputs = {key: val.to(device) for key, val in inputs.items()}
        with torch.no_grad():
            outputs = model(**inputs)
        cls_embeddings = outputs.last_hidden_state[:, 0, :].cpu().numpy()
        all_embeddings.extend(cls_embeddings)
    return np.array(all_embeddings)

x_embeddings = get_bert_embeddings(df['text_input'])
y_labels = df['like_count'].values
```


XGBOOST

```
x_train, x_test, y_train, y_test = train_test_split(
    x_embeddings,
    y_labels,
    test_size=0.2,
    random_state=42
)
print(f"Data split into {len(x_train)} training samples and {len(x_test)} testing samples.")

print("\n--- Starting Hyperparameter Tuning ---")
param_dist = {
    'objective': ['reg:squarederror', 'reg:absoluteerror'],
    'n_estimators': [100, 200, 300, 500, 800],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 5, 7, 10, 15],
    'reg_alpha': [0.0, 0.1, 0.5, 1.0],
    'reg_lambda': [0.0, 0.1, 0.5, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'subsample': [0.6, 0.8, 1.0],
}
```

Kode ini menjalankan pencarian hyperparameter terbaik untuk model XGBoost menggunakan RandomizedSearchCV dengan 10 percobaan dan 5-fold cross-validation, mengoptimalkan berdasarkan mean absolute error. Model terbaik disimpan di best_model untuk evaluasi selanjutnya.

Kode ini membagi dataset embedding dan label menjadi data latih (80%) dan data uji (20%) untuk evaluasi model yang adil. Selanjutnya, disiapkan beberapa kombinasi hyperparameter penting seperti jumlah pohon, laju pembelajaran, kedalaman pohon, dan regularisasi yang akan diuji melalui proses tuning untuk menemukan konfigurasi terbaik guna meningkatkan performa model XGBoost dalam memprediksi jumlah like secara akurat.

```
xgbr = xgb.XGBRegressor(random_state=42, tree_method='gpu_hist')
random_search = RandomizedSearchCV(
    estimator=xgbr,
    param_distributions=param_dist,    n_iter=10,
    cv=5,
    scoring='neg_mean_absolute_error',
    n_jobs=-1,
    random_state=42,
    verbose=1
)
random_search.fit(x_train, y_train)

print("\nBest hyperparameters found:")
print(random_search.best_params_)
best_model = random_search.best_estimator_
```

EVALUASI

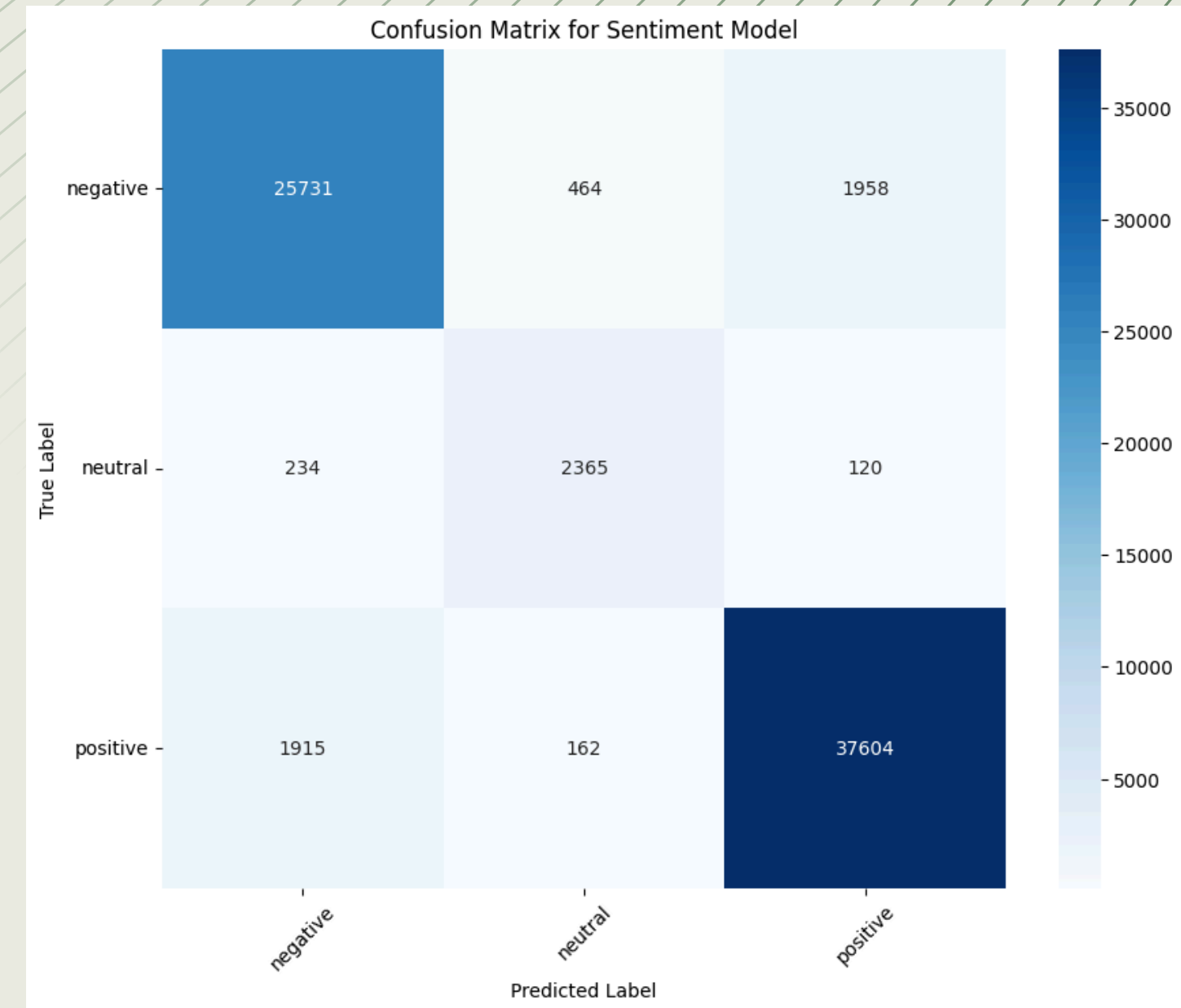
ROBERTA

--- Evaluation Report for Sentiment Model ---

Classification Report:

	precision	recall	f1-score	support
negative	0.92	0.91	0.92	28153
neutral	0.79	0.87	0.83	2719
positive	0.95	0.95	0.95	39681
accuracy			0.93	70553
macro avg	0.89	0.91	0.90	70553
weighted avg	0.93	0.93	0.93	70553

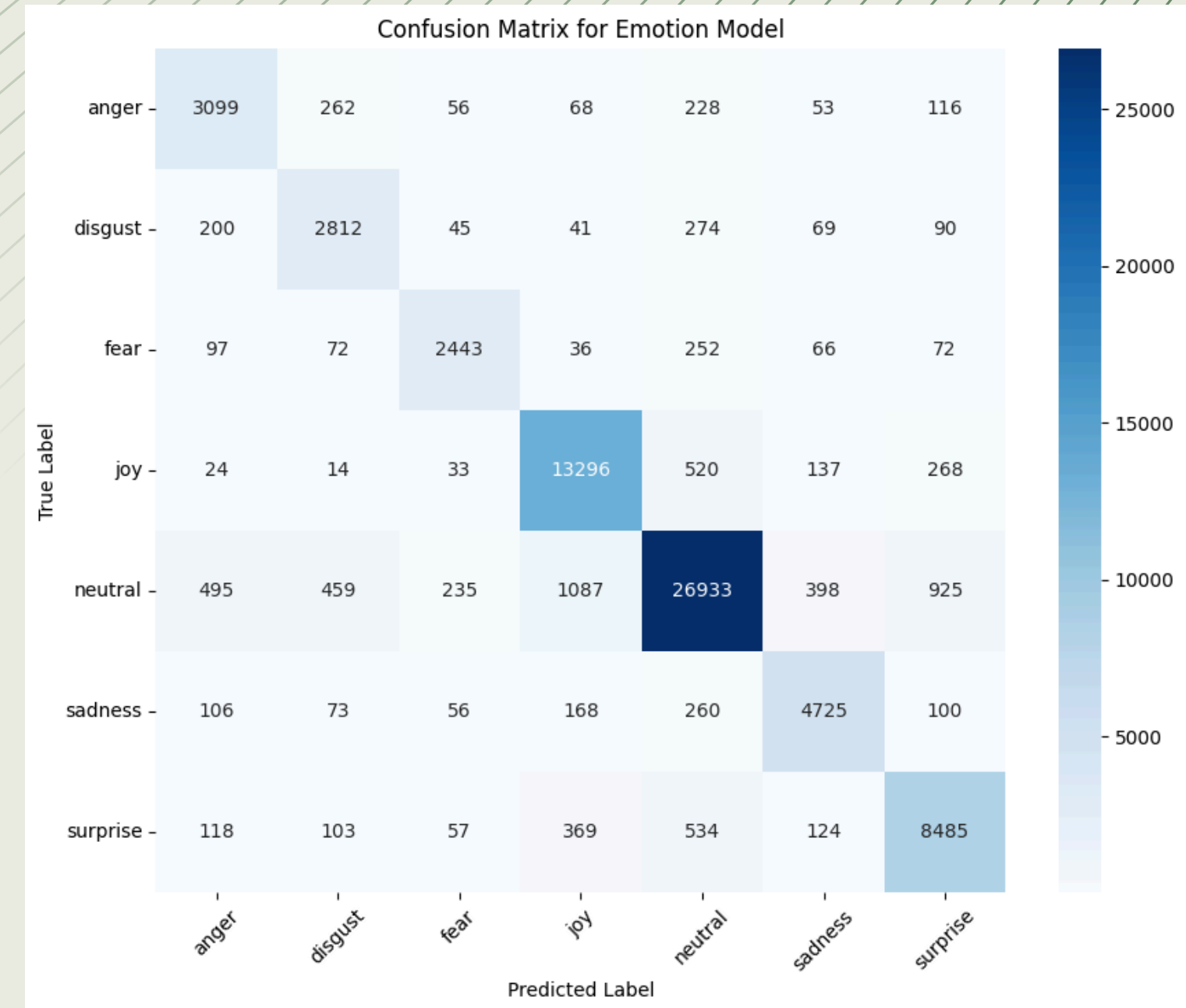
Overall Accuracy: 0.9312



ROBERTA

Classification Report:

	precision	recall	f1-score	support
anger	0.75	0.80	0.77	3882
disgust	0.74	0.80	0.77	3531
fear	0.84	0.80	0.82	3038
joy	0.88	0.93	0.91	14292
neutral	0.93	0.88	0.90	30532
sadness	0.85	0.86	0.85	5488
surprise	0.84	0.87	0.86	9790
accuracy			0.88	70553
macro avg	0.83	0.85	0.84	70553
weighted avg	0.88	0.88	0.88	70553
Overall Accuracy: 0.8758				



HASIL

The background features a series of thin, dark green lines that originate from the right edge and fan out towards the left, creating a sense of motion and depth. The lines are closely spaced and vary in length, giving the impression of a stylized, abstract landscape or a dynamic graphic element.

HASIL

 **Sentiment Classification Accuracy: 93.12%**

 **Emotion Classification Accuracy: 87.58%**

HASIL

Comment Input


This video is absolutely amazing and entertaining! Thank you for sharing

72/500

Analyze Comment

Reset


Emotion



Joy

Detected emotion


Sentiment



Positive

Detected sentiment

Like Count



Medium

100-500 likes
Predicted engagement level

Thank you.

