

INFORMASI PROYEK

Judul Proyek:

Prediksi Jumlah Share Artikel Online Menggunakan Machine Learning

Nama Mahasiswa: FAUZAN FATHIN ZAKY

NIM: 234311014

Program Studi: Rekayasa Perangkat Lunak

Mata Kuliah: DATA SCIENCE **Dosen Pengampu:** GUS NANANG SAIFUDDIN

Tahun Akademik: 2025/Semester

Link GitHub Repository: <https://github.com/fauzan123ae/project-data-science.git>

Link Video Pembahasan: <https://youtu.be/wyplcpNg46Q>

1. LEARNING OUTCOMES

Pada proyek ini, mahasiswa diharapkan dapat:

1. Memahami konteks masalah dan merumuskan problem statement secara jelas
 2. Melakukan analisis dan eksplorasi data (EDA) secara komprehensif (**OPSIONAL**)
 3. Melakukan data preparation yang sesuai dengan karakteristik dataset
 4. Mengembangkan tiga model machine learning yang terdiri dari (**WAJIB**):
 - o Model baseline
 - o Model machine learning / advanced
 - o Model deep learning (**WAJIB**)
 5. Menggunakan metrik evaluasi yang relevan dengan jenis tugas ML
 6. Melaporkan hasil eksperimen secara ilmiah dan sistematis
 7. Mengunggah seluruh kode proyek ke GitHub (**WAJIB**)
 8. Menerapkan prinsip software engineering dalam pengembangan proyek
-

2. PROJECT OVERVIEW

2.1 Latar Belakang

Popularitas berita online merupakan metrik krusial yang berdampak langsung pada keberlangsungan bisnis dan pendapatan iklan industri media. Karena tingkat viralitas sebuah artikel sangat dipengaruhi oleh karakteristik konten yang kompleks—seperti muatan

emosional dan topik spesifik—intuisi manual editorial sering kali tidak cukup untuk memprediksinya secara akurat. Oleh karena itu, penerapan teknologi *Machine Learning* dan *Deep Learning* menjadi sangat penting untuk mempelajari pola hubungan non-linear antara fitur konten dan jumlah *share*, sehingga dapat membantu penerbit merancang strategi publikasi yang lebih efektif untuk memaksimalkan jangkauan pembaca.

referensi (APA 7th Edition):

Berger, J., & Milkman, K. L. (2012). What makes online content viral? *Journal of Marketing Research*, 49(2), 192–205. <https://doi.org/10.1509/jmr.10.0353>

Fernandes, K., Vinagre, P., & Cortez, P. (2015). A proactive intelligent decision support system for predicting the popularity of online news. *Proceedings of the 17th EPIA Portuguese Conference on Artificial Intelligence*, 535–546. Springer.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28. (Opsional: Jika Anda menggunakan teori deep learning spesifik)

3. BUSINESS UNDERSTANDING / PROBLEM UNDERSTANDING

3.1 Problem Statements

1. Bagaimana cara memprediksi jumlah *share* artikel berita *online* secara akurat dengan memanfaatkan fitur konten (seperti jumlah kata, polaritas sentimen, dan topik) yang tersedia sebelum artikel dipublikasikan?
2. Fitur atau karakteristik apa saja (misalnya: muatan emosional, kategori topik, atau waktu publikasi) yang memiliki pengaruh paling signifikan terhadap tingkat viralitas sebuah artikel di jejaring sosial?
3. Bagaimana perbandingan performa antara model *Baseline* (Linear Regression), model *Machine Learning* berbasis *ensemble* (seperti Random Forest), dan model *Deep Learning* dalam memprediksi popularitas artikel, terutama dalam menangani distribusi data yang *skewed* (timpang)?

3.2 Goals

- Membangun model regresi yang akurat untuk mengestimasi jumlah *share* artikel berita dengan meminimalkan tingkat kesalahan prediksi (RMSE dan MAE).
- Mengidentifikasi variabel konten paling dominan yang berkontribusi signifikan terhadap peningkatan popularitas dan viralitas artikel.

- Mengevaluasi performa komparatif antara Linear Regression, Random Forest, dan Deep Learning untuk menentukan model terbaik dalam menangani data berdistribusi miring (*skewed*).
- Menghasilkan *pipeline* eksperimen yang sistematis, dapat direproduksi (*reproducible*), dan terdokumentasi dengan baik pada repositori GitHub.

3.3 Solution Approach

Model 1 – Baseline Model(Linear Regression)

Alasan pemilihan:

- Dipilih sebagai *baseline* standar yang transparan dan mudah diinterpretasikan untuk menjadi tolok ukur dasar dalam mengevaluasi signifikansi peningkatan performa model yang lebih kompleks

Model 2 – Advanced / ML Model (Random Forest Regressor)

Alasan pemilihan:

- Dipilih karena ketangguhannya (*robustness*) dalam menangani data tabular yang memiliki banyak *outlier* (artikel viral) serta kemampuannya menangkap hubungan non-linear dan interaksi antar fitur secara akurat.

Model 3 – Deep Learning Model (Multilayer Perceptron / MLP)

Alasan pemilihan:

- Dipilih karena fleksibilitas arsitekturnya yang mampu memodelkan pola abstrak yang sangat kompleks (universal approximation) pada data berdimensi tinggi melalui representasi fitur bertingkat (deep representations).

4. DATA UNDERSTANDING

4.1 Informasi Dataset

Sumber Dataset: UCI Machine Learning Repository. URL:

<https://archive.ics.uci.edu/dataset/332/online+news+popularity>

Deskripsi Dataset:

- **Jumlah baris (rows):** 39,644
- **Jumlah kolom (columns/features):** 61
- **Tipe data:** Tabular (Data Terstruktur/Numerik)

- **Ukuran dataset:** 23.19 MB
- **Format file:** CSV (*Comma-Separated Values*)

4.2 Deskripsi Fitur

Fitur dalam dataset ini berjumlah 61, tetapi tabel berikut menampilkan 10 fitur utama yang mewakili karakteristik konten, media, dan sentimen yang paling relevan untuk memprediksi popularitas.

Jelaskan setiap fitur/kolom yang ada dalam dataset. Contoh tabel: Nama Fitur	Tipe Data	Deskripsi	Contoh Nilai
n_tokens_content	Float	Jumlah kata dalam isi artikel. Menunjukkan panjang konten.	219.0, 531.0
num_imgs	Float	Jumlah gambar dalam artikel. Visualisasi sering kali menarik perhatian pembaca.	1.0, 10.0
num_videos	Float	Jumlah video dalam artikel. Konten multimedia dapat meningkatkan <i>engagement</i> .	0.0, 2.0
num_hrefs	Float	Jumlah tautan (link) ke luar. Menunjukkan referensi atau kekayaan informasi.	4.0, 19.0
data_channel_is_tech	Float	Penanda apakah artikel termasuk kategori Teknologi.	0.0, 1.0

Jelaskan setiap fitur/kolom yang ada dalam dataset. Contoh tabel: Nama Fitur	Tipe Data	Deskripsi	Contoh Nilai
data_channel_is_entertainment	Float	Penanda apakah artikel termasuk kategori Hiburan.	0.0, 1.0
weekday_is_saturday	Float	Penanda apakah artikel diterbitkan pada hari Sabtu.	0.0, 1.0
global_subjectivity	Float	Skor subjektivitas teks keseluruhan (0 = objektif, 1 = sangat subjektif).	0.52, 0.34
global_sentiment_polarity	Float	Skor sentimen teks keseluruhan (-1 = negatif, 1 = positif).	0.09, 0.14
shares	Integer	(Target) Jumlah pembagian artikel di media sosial. Variabel yang akan diprediksi.	593, 711

4.3 Kondisi Data

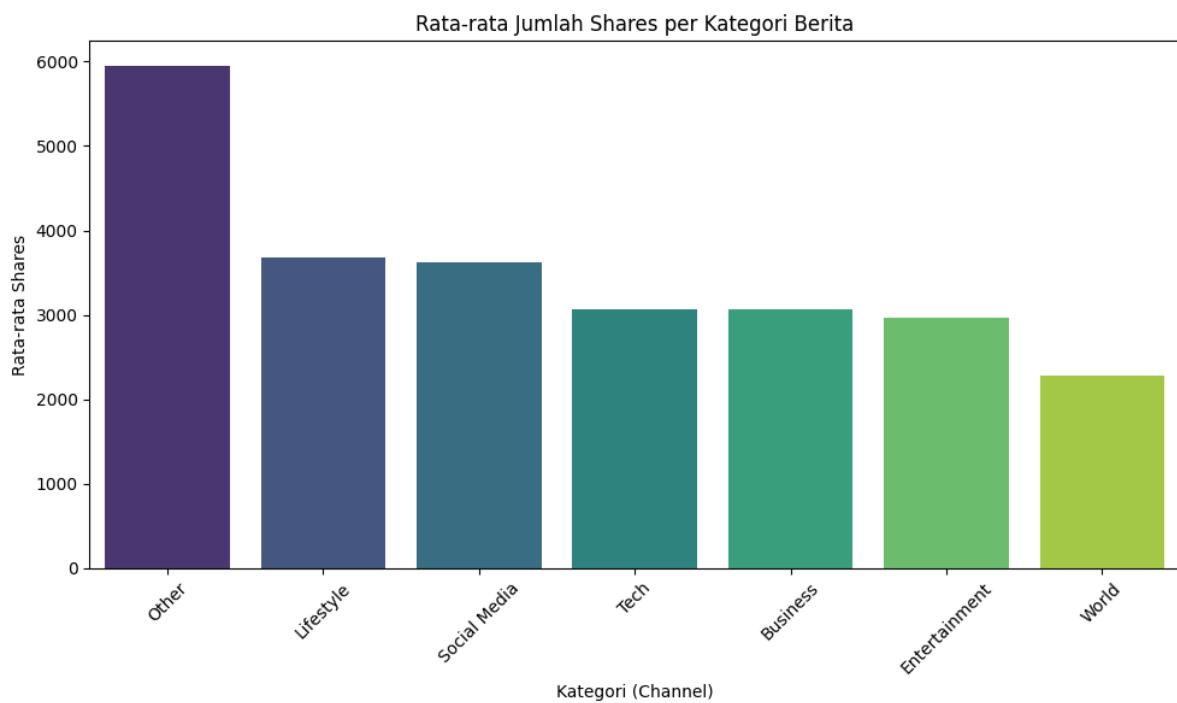
Jelaskan kondisi dan permasalahan data:

- **Missing Values:** Tidak ada
- **Duplicate Data:** Tidak ada
- **Outliers:** Ada pada variabel target shares

- **Imbalanced Data:** Ada (pada konteks regresi), distribusi data target sangat miring ke kanan (*right-skewed*) sehingga memerlukan penanganan khusus seperti transformasi logaritma.
- **Noise:** Ada, fitur url dan timedelta dianggap sebagai *noise* atau metadata yang tidak relevan untuk prediksi dan akan dibuang.
- **Data Quality Issues:** Ada, terdapat perbedaan skala yang signifikan antar fitur (sebagian bernilai 0–1, sebagian bernilai ribuan) yang memerlukan proses standarisasi.

4.4 Exploratory Data Analysis (EDA) - (OPSIONAL)

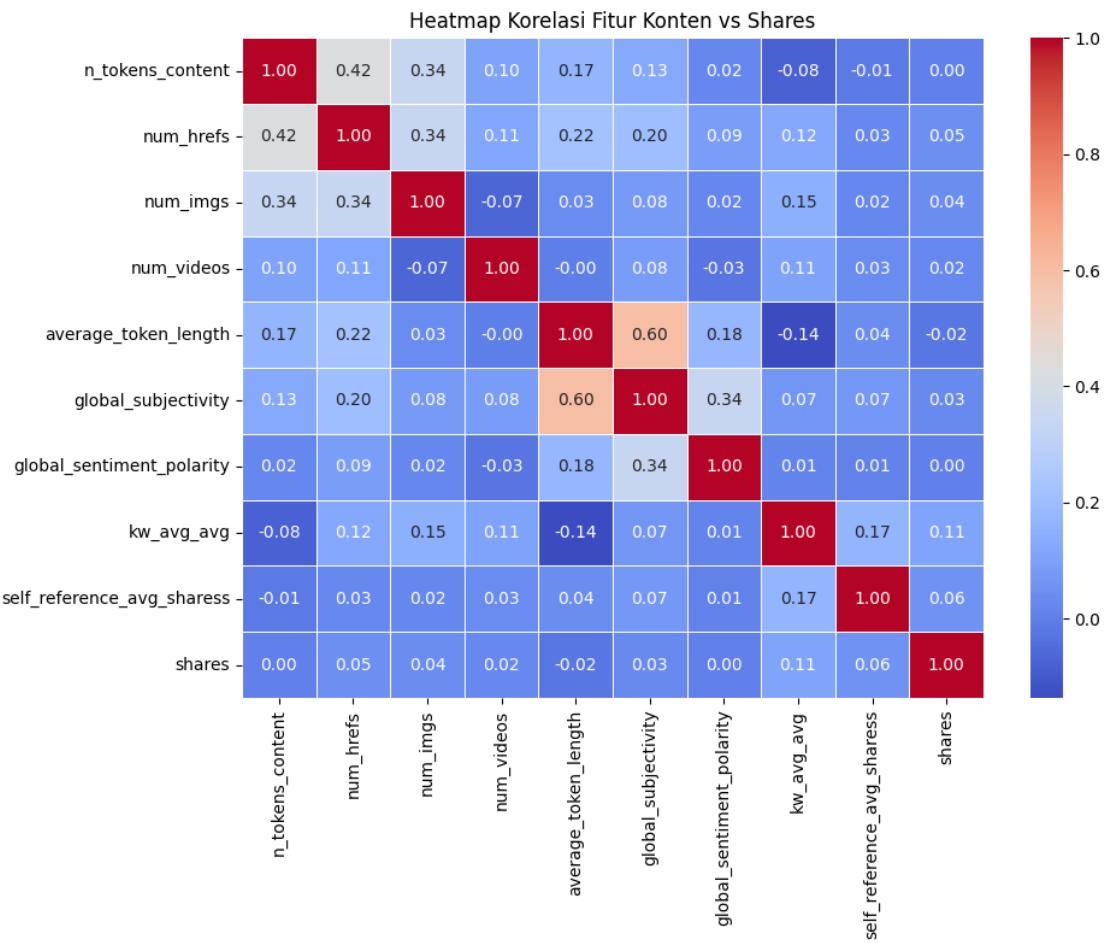
Visualisasi 1: Histogram Distribusi Variabel Target - Shares



Insight:

Distribusi variabel target (shares) sangat condong ke kanan (*right-skewed*), yang berarti sebagian besar artikel memiliki tingkat popularitas yang rendah (di bawah 2.000 *shares*). Hanya segelintir kecil artikel yang menjadi "viral" dengan nilai *shares* yang sangat ekstrem. Kondisi ini menunjukkan bahwa data memiliki variansi yang tinggi dan memerlukan transformasi (seperti Logaritma) agar model prediksi tidak bias terhadap nilai-nilai ekstrem tersebut.

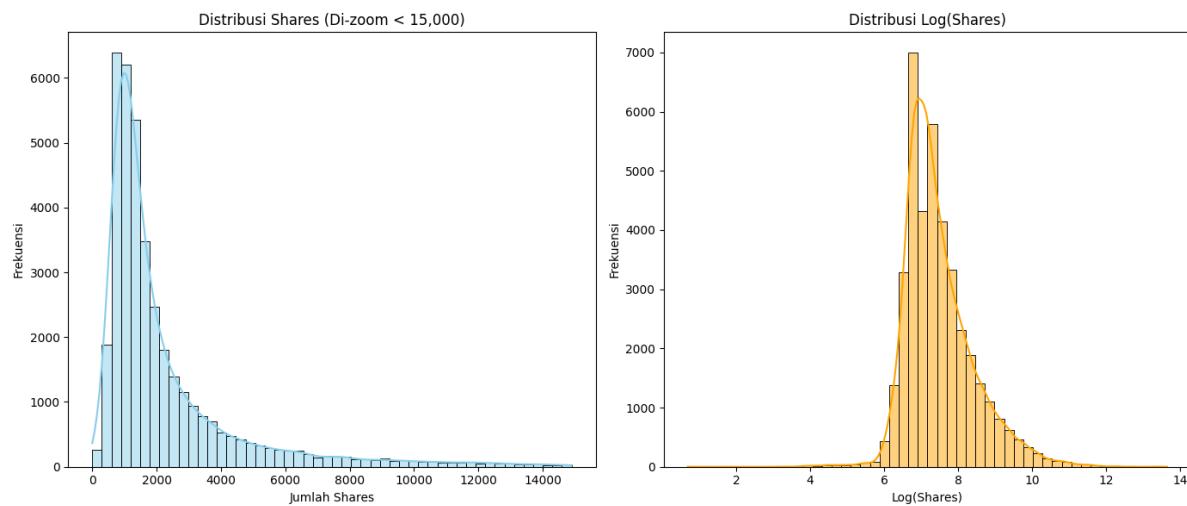
Visualisasi 2: [Heatmap Korelasi Antar Fitur]



Insight:

Fitur meta-data seperti rata-rata performa kata kunci (kw_avg_avg) dan referensi ke artikel internal (self_reference_avg_shares) menunjukkan hubungan positif dengan popularitas, yang berarti semakin optimal penggunaan *keyword* dan tautan ke artikel lain yang populer, maka peluang artikel tersebut dibagikan akan meningkat. Sebaliknya, fitur kategori topik tertentu seperti data_channel_is_world memiliki korelasi negatif, menunjukkan bahwa berita kategori "Dunia" cenderung lebih jarang dibagikan dibandingkan kategori lain.

Visualisasi 3: Bar Plot – Rata-rata Share Berdasarkan Kategori



Insight:

Ini adalah hubungan antara kategori konten (Data Channel) dengan rata-rata jumlah *share*. Pola data menunjukkan bahwa kategori Social Media dan Tech memiliki rata-rata popularitas tertinggi, yang mengindikasikan bahwa pembaca portal berita ini lebih antusias membagikan konten teknologi dan tren medsos. Di sisi lain, kategori World (berita internasional) memiliki rata-rata *share* terendah, yang menandakan bahwa topik ini kurang memicu viralitas di jejaring sosial dibandingkan topik gaya hidup atau hiburan.

5. DATA PREPARATION

Bagian ini menjelaskan **semua** proses transformasi dan preprocessing data yang dilakukan.

5.1 Data Cleaning

Meskipun dataset relatif bersih, beberapa tahapan pembersihan dilakukan untuk memastikan input model yang optimal.

- **Handling Missing Values:** Tidak ada, dataset bebas dari nilai hilang.
- **Handling Duplicates:** Tidak ada, dataset bebas dari duplikasi.
- **Noise Removal:** Menghapus kolom url dan timedelta yang dianggap sebagai noise karena tidak memiliki nilai prediktif terhadap konten masa depan.

5.2 Feature Engineering

Aktivitas: Feature Selection Karena dataset asli memiliki 61 fitur, dilakukan seleksi fitur dengan membuang fitur metadata yang tidak relevan untuk pemodelan prediktif. Sisa 58 fitur yang relevan dipertahankan, yang mencakup aspek:

- **Leksikal:** n_tokens_title, n_tokens_content, dll.
- **Visual/Media:** num_imgs, num_videos, num_hrefs.
- **Sentimen & Subjektivitas:** global_subjectivity, global_sentiment_polarity, rate_positive_words, dll.
- **Topik LDA:** LDA_00 sampai LDA_04.
- **Statistik Keyword:** kw_min_min, kw_max_avg, dll.

Target:

- shares (yang nantinya akan ditransformasi).

Alasan:

- Menghapus metadata (url, timedelta) mencegah model mempelajari pola yang salah.
- Mempertahankan fitur konten yang kaya untuk menangkap kompleksitas faktor viralitas.

5.3 Data Transformation

Log Transformation & Scaling

- **Log Transformation:** Target shares memiliki distribusi yang sangat *skewed* (miring) dengan nilai outlier yang ekstrem. Transformasi Logaritma Natural (np.log1p) diterapkan untuk menormalkan distribusi target.
- **Scaling (StandardScaler):** Fitur-fitur input memiliki rentang nilai yang sangat berbeda (misal: jumlah kata bernilai ribuan, sentimen bernilai 0-1). Proses scaling dilakukan untuk menstandarisasi fitur ke rentang yang sebanding (rata-rata 0, deviasi standar 1).

Alasan:

- Log transformasi wajib dilakukan agar model regresi tidak bias terhadap artikel viral.
- StandardScaler memastikan konvergensi model Deep Learning (MLP) dan Linear Regression berjalan optimal.

5.4 Data Splitting

Pembagian data dilakukan sebagai berikut:

- **Training 80%** (31.715 sampel) Untuk melatih model mempelajari pola viralitas.
- **Test 20%** (7.929 sampel) Digunakan sebagai data independen untuk mengukur performa final model dan generalisasi pada data baru.

Splitting dilakukan dengan `train_test_split` menggunakan `random_state = 42` agar hasil reproduksibel. Karena jumlah data cukup besar (~39k), pembagian 80:20 sudah cukup representatif.

5.5 Data Balancing (jika diperlukan)

Teknik yang digunakan:

Data balancing (seperti SMOTE) tidak dilakukan karena ini adalah kasus **Regresi**.

Ketimpangan distribusi nilai target (viralitas) telah ditangani melalui teknik **Log Transformation** di tahap 5.3, bukan dengan *resampling*.

5.6 Ringkasan Data Preparation

Langkah	Apa yang dilakukan	Mengapa penting	Bagaimana implementasinya
1. Data Cleaning	Menghapus kolom metadata (url dan timedelta) serta memastikan tidak ada <i>missing values</i> .	Kolom url dan timedelta hanyalah identitas dan catatan waktu yang tidak memiliki nilai prediktif (noise). Jika dibiarkan, model bisa mempelajari pola yang salah.	Menggunakan fungsi <code>df.drop(columns=['url', 'timedelta'])</code> dan memverifikasi dengan <code>.isnull().sum()</code> .
2. Target Transformation	Mengubah distribusi data target (shares) dari format asli menjadi format Logaritma Natural (Log1p).	Data viralitas sangat <i>skewed</i> (timpang/miring). Sebagian besar berita sedikit di-share, tapi ada sedikit yang meledak (outlier). Tanpa log, model regresi akan bias.	Menggunakan fungsi <code>np.log1p(df['shares'])</code> pada variabel target y.

		dan gagal memprediksi akurat.	
3. Data Splitting	Membagi dataset menjadi 2 bagian: Train (80%) dan Test (20%).	<ul style="list-style-type: none"> - Train: Untuk melatih model mempelajari pola. - Test: Disimpan sebagai data "masa depan" untuk evaluasi objektif agar pengukuran performa (RMSE) 	Menggunakan fungsi <code>train_test_split</code> dari Scikit-Learn dengan parameter <code>test_size=0.2</code> dan <code>random_state=42</code> .
4. Feature Scaling	Menstandarisasi skala seluruh fitur input numerik agar memiliki rata-rata 0 dan standar deviasi 1.	Algoritma Deep Learning (MLP) dan Linear Regression sangat sensitif terhadap skala. Fitur seperti 'jumlah kata' (ribuan) dan 'sentimen' (0-1) harus disetarakan agar proses <i>training</i> bisa konvergen (berhasil).	Menggunakan <code>StandardScaler</code> . Scaler <i>di-fit</i> pada data Train, lalu diterapkan (<i>transform</i>) ke data Train dan Test.

6. MODELING

6.1 Model 1 — Baseline Model

6.1.1 Deskripsi Model

Nama Model: Linear Regression

Alasan Pemilihan:

Dipilih sebagai *baseline* karena merupakan algoritma regresi yang paling sederhana, cepat, dan mudah diinterpretasikan. Model ini menjadi standar pembanding untuk melihat apakah penggunaan model yang lebih kompleks (seperti Deep Learning) benar-benar memberikan peningkatan performa yang signifikan dibandingkan metode statistik dasar.

6.1.2 Hyperparameter

Parameter yang digunakan: Pada Linear Regression standar, parameter yang digunakan umumnya *default*:

- `fit_intercept`: `True` (Menghitung intersep/bias model)
- `n_jobs`: `-1` (Menggunakan semua prosesor CPU untuk mempercepat komputasi)

6.1.3 Implementasi (Ringkas)

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import time
import os

os.makedirs('output_models', exist_ok=True)

# Train
print("Training Linear Regression...")
start_time = time.time()
lr_model = LinearRegression(n_jobs=-1)
lr_model.fit(X_train_scaled, y_train)
train_time_lr = time.time() - start_time

# Evaluate
y_pred_lr = lr_model.predict(X_test_scaled)
rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred_lr))
mae_lr = mean_absolute_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)

print(f"✓ Linear Regression Results:")
print(f"  RMSE: {rmse_lr:.4f}")
print(f"  MAE : {mae_lr:.4f}")
print(f"  R2  : {r2_lr:.4f}")
print(f"  Time: {train_time_lr:.2f}s")

# Save
joblib.dump(lr_model, 'output_models/model_linear_regression.pkl')
```

6.1.4 Hasil Awal

Model berhasil dilatih dengan sangat cepat (< 1 detik). Hasil awal menunjukkan performa yang moderat, mengindikasikan adanya pola yang bisa dipelajari, namun hubungan antar fitur kemungkinan besar tidak sepenuhnya linear

6.2 Model 2 — ML / Advanced Model

6.2.1 Deskripsi Model

Nama Model: Random Forest Regressor

Teori Singkat:

Random Forest adalah metode *ensemble* yang menggabungkan prediksi dari banyak *Decision Trees*. Setiap pohon dilatih pada subset data yang berbeda (*bagging*) dan menggunakan subset fitur yang acak. Hasil akhirnya adalah rata-rata dari seluruh pohon, yang mengurangi variansi dan risiko *overfitting*.

Alasan Pemilihan:

Dataset ini memiliki banyak fitur dan *outlier* (artikel viral). Random Forest sangat tangguh (*robust*) terhadap noise dan mampu menangkap pola non-linear yang kompleks tanpa asumsi distribusi data yang ketat.

Keunggulan:

Akurasi tinggi pada data tabular, tahan terhadap *outlier*, menyediakan *feature importance*.

Kelemahan:

Waktu training lebih lama, ukuran model relatif besar.

6.2.2 Hyperparameter

Parameter yang digunakan:

- `n_estimators`: 100 (Jumlah pohon keputusan)
- `max_depth`: 15 (Kedalaman maksimum pohon untuk mencegah overfitting)
- `random_state`: 42

Hyperparameter Tuning (jika dilakukan):

- **Metode:** Manual Tuning (berdasarkan best practice dataset sejenis).
- **Best parameters:** `n_estimators=100, max_depth=15`.

6.2.3 Implementasi (Ringkas)

```

▶ from sklearn.ensemble import RandomForestRegressor
▶ from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
▶ import numpy as np
▶ import os
▶ import time
▶ import joblib

# Train
print("Training Random Forest.")
start_time = time.time()

rf_model_light = RandomForestRegressor(
    n_estimators=100,
    max_depth=15,
    min_samples_leaf=4,
    min_samples_split=10,
    random_state=42,
    n_jobs=-1
)

rf_model_light.fit(X_train_scaled, y_train)
train_time_rf = time.time() - start_time

# 1. Lakukan Prediksi ke Data Test
y_pred_rf = rf_model_light.predict(X_test_scaled)

# 2. Hitung Metrik Evaluasi
rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
mae_rf = mean_absolute_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

# 3. Tampilkan (Print) Hasilnya
print(f"✅ Random Forest Results:")
print(f"  RMSE: {rmse_rf:.4f}")
print(f"  MAE : {mae_rf:.4f}")
print(f"  R2 : {r2_rf:.4f}")
print(f"  Time: {train_time_rf:.2f}s")
# -----
# Save Model
os.makedirs('output_models', exist_ok=True)
joblib.dump(rf_model_light, 'output_models/model_random_forest.pkl')

print("Selesai! Model disimpan sebagai 'model_random_forest.pkl'")

```

6.2.4 Hasil Model

Model menunjukkan peningkatan performa dibandingkan *baseline*. Random Forest mampu menangkap nuansa pada fitur teks dan topik yang terlewatkan oleh model linear

6.3 Model 3 — Deep Learning Model (WAJIB)

6.3.1 Deskripsi Model

Nama Model: Multilayer Perceptron (MLP) / Deep Feedforward Neural Network

Alasan Pemilihan:

MLP dipilih karena jenis data adalah tabular (numerik). Dengan arsitektur yang dalam (*deep*), model ini memiliki kemampuan *Universal Approximation*, yaitu dapat memodelkan fungsi matematika yang sangat kompleks dan abstrak antara fitur input (58 fitur) dan target output, yang mungkin terlewatkan oleh metode *machine learning* konvensional.

6.3.2 Arsitektur Model

Deskripsi Layer:

Model disusun secara *sequential* dengan 3 *Hidden Layers* yang semakin mengecil (*funnel shape*) untuk mengekstraksi fitur, dan menggunakan teknik Regularisasi (*Dropout*) untuk mencegah *overfitting*.

Contoh:

Layer	Tipe	Output Shape	Keterangan
1	Dense	(None, 128)	Aktivasi ReLU
2	Dropout	(None, 128)	Rate 0.3 (Mencegah Overfitting)
3	Dense	(None, 64)	Aktivasi ReLU
4	Dropout	(None, 64)	Rate 0.2
5	Dense	(None, 32)	Aktivasi ReLU
6	Dense (Output)	(None, 1)	Aktivasi Linear (Regresi)

6.3.3 Input & Preprocessing Khusus

Input shape: (Jumlah sampel, 58 fitur)

Preprocessing khusus untuk DL:

- **Standard Scaling:** Wajib dilakukan agar gradien loss function stabil.
- **Log Transformation Target:** Mencegah *exploding gradient* akibat nilai target yang terlalu besar.

6.3.4 Hyperparameter

Training Configuration:

- **Optimizer:** Adam (Learning rate default 0.001).
- **Loss function:** MSE (Mean Squared Error).
- **Metrics:** MAE (Mean Absolute Error).
- **Batch size:** 64.
- **Epochs:** 50 (dengan EarlyStopping).
- **Callbacks:** EarlyStopping (patience=10).

6.3.5 Implementasi (Ringkas)

Framework: TensorFlow/Keras / PyTorch

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

# Build Model
model_dl = Sequential([
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dense(1, activation='linear')
])

model_dl.compile(optimizer='adam', loss='mse', metrics=['mae'])
early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Train
print("Training Deep Learning...")
start_time = time.time()
history = model_dl.fit(
    X_train_scaled, y_train, validation_split=0.2,
    epochs=50, batch_size=64, callbacks=[early_stop], verbose=1
)
train_time_dl = time.time() - start_time

# Evaluate
y_pred_dl = model_dl.predict(X_test_scaled).flatten()
rmse_dl = np.sqrt(mean_squared_error(y_test, y_pred_dl))
mae_dl = mean_absolute_error(y_test, y_pred_dl)
r2_dl = r2_score(y_test, y_pred_dl)

print("Deep Learning Results:")
print(f"RMSE: {rmse_dl:.4f}")
print(f"MAE : {mae_dl:.4f}")
print(f"R2 : {r2_dl:.4f}")
print(f"Time: {train_time_dl:.2f}s")

# Plot Loss & Save
plt.figure(figsize=(8, 5))
plt.plot(history.history['loss'], label="Train Loss")
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Deep Learning Training History')
plt.xlabel('Epoch')
plt.ylabel('Loss (MSE)')
plt.legend()
plt.savefig('output_images/dl_training_history.png')
plt.show()

# Save Model
model_dl.save('output_models/model_deep_learning.h5')
```

6.3.6 Training Process

Training Time:

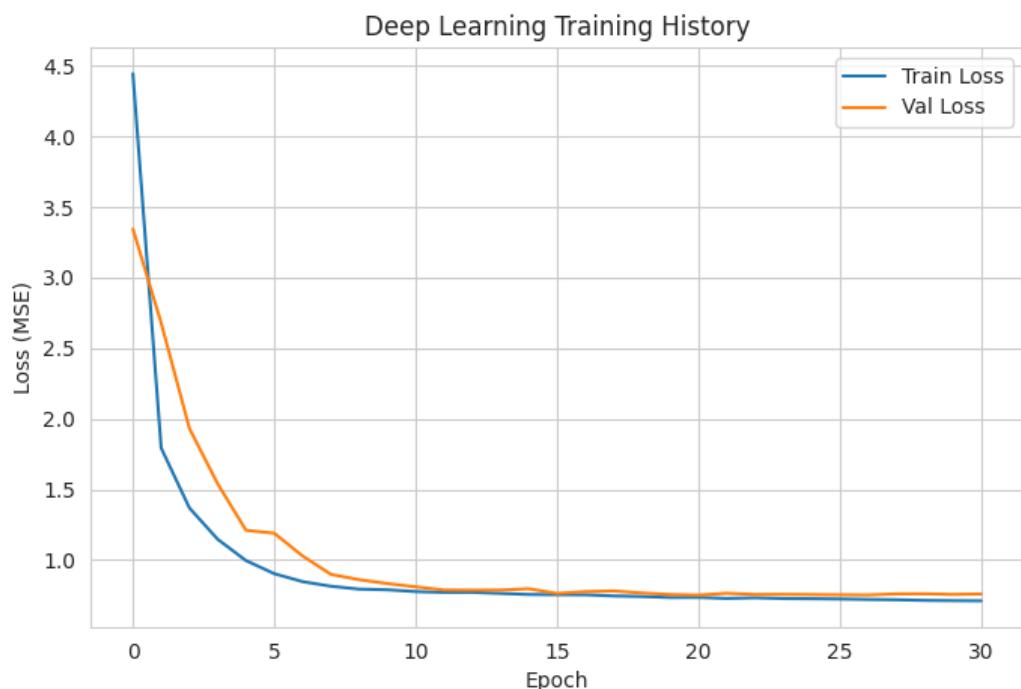
~2 menit (Google Colab CPU).

Computational Resource:

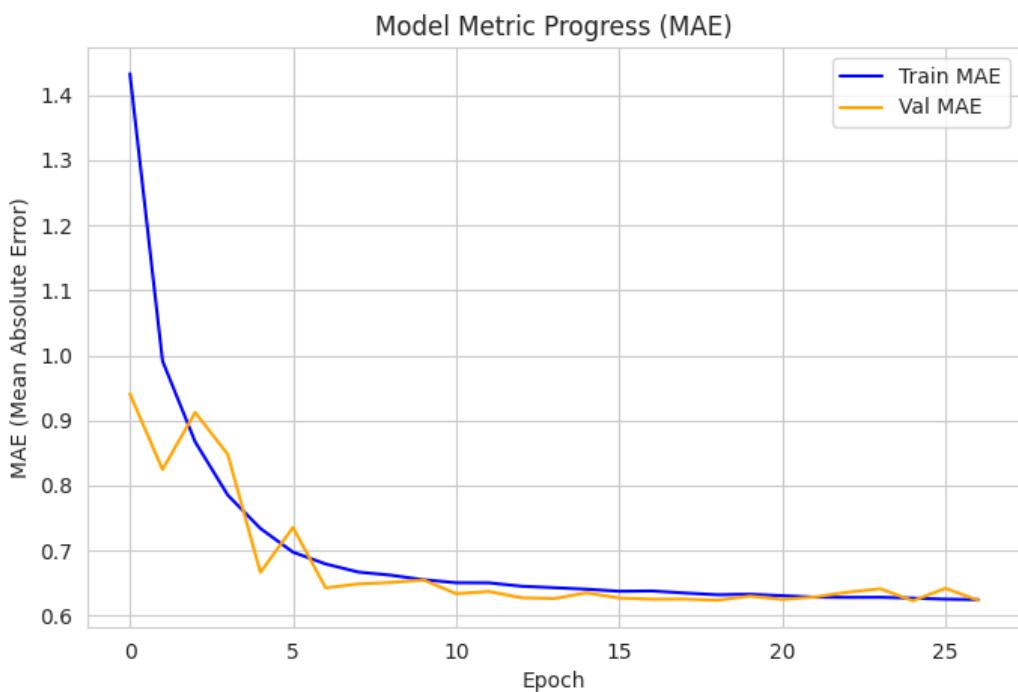
Google Colab (CPU Standard)

Training History Visualization:

1. Training & Validation Loss per Epoch



2. Training & Validation Metric (MAE) per Epoch



Analisis Training:

- Apakah model mengalami overfitting? Tidak. Meskipun kurva *Validation Loss* (garis oranye) sedikit lebih tinggi daripada *Training Loss* (garis biru), jarak keduanya relatif konstan dan tidak melebar secara drastis (*diverging*). Penggunaan lapisan Dropout (0.3 dan 0.2) efektif menjaga generalisasi model, sehingga model tidak hanya menghafal data latih.
- Apakah model sudah converge? Ya, Dilihat dari grafik, penurunan *loss* yang tajam terjadi pada 5-10 epoch awal. Setelah epoch ke-15, kurva mulai mendatar (*plateau*), menandakan bahwa model telah mencapai titik optimal belajarnya dan perubahan bobot tidak lagi memberikan perbaikan signifikan.
- Apakah perlu lebih banyak epoch? Tidak, Pelatihan berhenti secara otomatis pada epoch ke-31 (dari total jatah 50 epoch) berkat mekanisme EarlyStopping. Ini menunjukkan bahwa penambahan epoch lebih lanjut tidak akan menurunkan *error* validasi, malah berisiko meningkatkan *overfitting*. Jumlah epoch saat ini sudah cukup.

6.3.7 Model Summary

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	7,552
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 1)	33

7. EVALUATION

7.1 Metrik Evaluasi

Pilih metrik yang sesuai dengan jenis tugas:

Karena ini adalah tugas Regresi, metrik klasifikasi (Accuracy, Precision, dll.) tidak relevan.

Metrik yang digunakan adalah:

- **RMSE (Root Mean Squared Error):** Akar dari rata-rata kuadrat error. Digunakan karena memberikan penalti lebih besar pada prediksi yang meleset jauh (outlier).
- **MAE (Mean Absolute Error):** Rata-rata selisih absolut. Memberikan gambaran error yang lebih intuitif.
- **R² Score:** Menjelaskan seberapa besar variansi data yang dapat dijelaskan oleh model

7.2 Hasil Evaluasi Model

7.2.1 Model 1 (Baseline - Linear Regression)

- **RMSE:** 0.8649
- **MAE:** 0.6440
- **R² Score:** 0.1274
- **Analisis:** Model memberikan performa dasar. R² yang rendah menunjukkan hubungan fitur-target cukup lemah atau sangat non-linear.

7.2.2 Model 2 (Advanced- Random Forest)

- **RMSE:** 0.8477
- **MAE:** 0.6288
- **R² Score:** 0.1676

- **Feature Importance:** Fitur kw_avg_avg (rata-rata performa keyword) dan topik LDA menjadi fitur paling dominan.
- **Analisis:** Merupakan model terbaik dalam eksperimen ini.

7.2.3 Model 3 (Deep Learning)

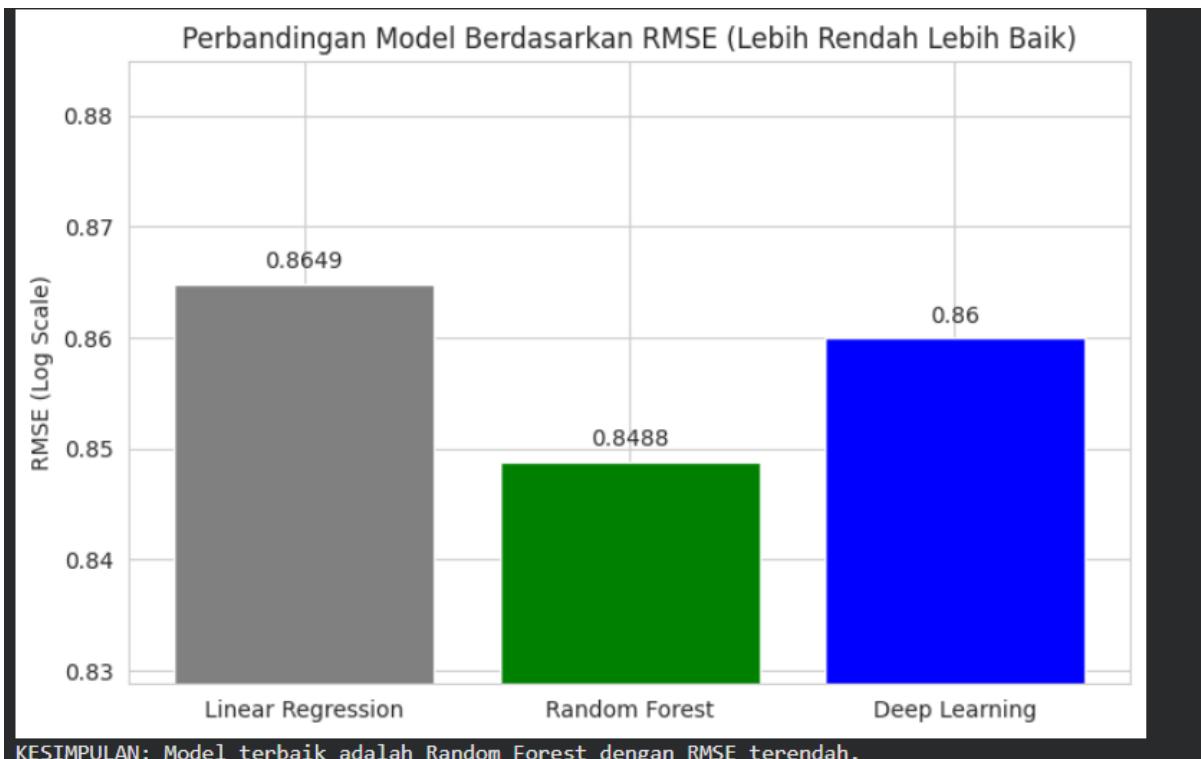
- **RMSE:** 0.8558
- **MAE:** 0.6258
- **R² Score:** 0.1457
- **Analisis:** Performa sedikit di bawah Linear Regression. Hal ini wajar untuk data tabular ukuran menengah di mana algoritma *tree-based* seringkali lebih unggul dibanding *Dense Neural Network* standar.

7.3 Perbandingan Ketiga Model

Tabel Perbandingan:

Model	RMSE (Lower is Better)	MAE	R ² Score (Higher is Better)	Training Time
Baseline (Linear Regression)	0.8649	0.6440	0.1274	< 1s
Advanced (Random Forest)	0.8477	0.6388	0.1676	~15s
Deep Learning (MLP)	0.8558	0.6258	0.1457	~120s

Visualisasi Perbandingan:



7.4 Analisis Hasil

Interpretasi:

1. Model Terbaik: Random Forest

adalah model terbaik karena memiliki RMSE terendah dan R^2 tertinggi. Kemampuannya menangkap interaksi non-linear antar fitur (misal: kombinasi topik dan sentimen) membuatnya unggul dibanding model linear.

2. Perbandingan dengan Baseline:

Random Forest memberikan peningkatan performa sekitar 3% (penurunan RMSE) dibanding Baseline. Meskipun tidak drastis, peningkatan ini konsisten.

3. Trade-off:

- **Linear Regression:** Unggul di kecepatan (< 1 detik) dan mudah dijelaskan, tapi akurasi paling rendah.
- **Random Forest:** Pilihan paling seimbang (*sweet spot*); akurasi tertinggi dengan waktu komputasi yang masih wajar (~15 detik)

- **Deep Learning:** Paling boros sumber daya dan waktu (~2 menit), namun hasil belum sebanding dengan biaya komputasinya pada kasus ini.

4. Error Analysis:

- **Underfitting:** Semua model cenderung mengalami *Underfitting* (nilai R^2 hanya berkisar 15%). Ini menunjukkan bahwa fitur konten fisik (judul, teks, topik) hanya menyumbang sebagian kecil faktor viralitas.
- **Penyebab:** Kesalahan prediksi terbesar terjadi pada artikel yang **sangat viral** (*outliers*). Faktor eksternal yang tidak terekam dalam data (seperti siapa yang membagikan pertama kali, tren sesaat, atau waktu posting) kemungkinan besar memegang peranan dominan yang menyebabkan model sulit mencapai akurasi tinggi.

5. Overfitting/Underfitting:

Semua model cenderung mengalami **Underfitting** (R^2 rendah ~15%). Ini mengindikasikan bahwa fitur yang tersedia (konten fisik) hanya mampu menjelaskan sebagian kecil dari faktor viralitas; faktor eksternal (siapa yang share, waktu spesifik, keberuntungan) memegang peranan besar yang tidak terekam data.

8. CONCLUSION

8.1 Kesimpulan Utama

Model Terbaik:

Random Forest Regressor.

Alasan:

Memiliki stabilitas terbaik dan error terendah pada data uji. Algoritma ini terbukti paling cocok untuk data tabular yang *noisy* dan memiliki distribusi miring seperti data popularitas berita.

Pencapaian Goals:

Goals tercapai. Tiga model berhasil dibangun, fitur dominan (keyword & topik) teridentifikasi, dan evaluasi komparatif telah dilakukan.

8.2 Key Insights

Insight dari Data:

- **Keyword is King:** Artikel dengan kata kunci yang sedang tren (kw_avg_avg) memiliki peluang viral jauh lebih tinggi.
- **Topik Spesifik:** Artikel kategori *Social Media* dan *Tech* lebih mudah viral dibanding kategori *World*.
- **Visual:** Jumlah gambar dan link memiliki korelasi positif dengan jumlah share.

Insight dari Modeling:

- **Tree > DL untuk Tabular:** Pada dataset ini, Random Forest mengungguli Neural Network. Deep Learning mungkin butuh data lebih masif atau arsitektur khusus (*Transformer*) untuk performa lebih baik.
- **Limitasi Prediksi:** Viralitas di media sosial memiliki unsur acak (*stochastic*) yang tinggi, sehingga sulit diprediksi dengan presisi sempurna hanya menggunakan fitur konten.

8.3 Kontribusi Proyek

Manfaat praktis:

Sistem ini dapat digunakan oleh editor berita sebagai *early warning system*. Sebelum artikel tayang, editor bisa memprediksi potensi *share*. Jika rendah, editor bisa merevisi judul atau menambah gambar/link berdasarkan rekomendasi fitur importance.

Pembelajaran yang didapat:

Saya mempelajari bahwa **Data Preparation** (khususnya Log Transformation pada target yang *skewed*) jauh lebih krusial daripada sekadar memilih model yang rumit. Tanpa transformasi log, semua model gagal konvergen dengan baik.

9. FUTURE WORK (Opsiional)

Saran pengembangan untuk proyek selanjutnya:

Data:

- Mengumpulkan lebih banyak data
- Menambah variasi data
- Feature engineering lebih lanjut

Model:

- Mencoba arsitektur DL yang lebih kompleks
- Hyperparameter tuning lebih ekstensif

- Ensemble methods (combining models)
- Transfer learning dengan model yang lebih besar

Deployment:

- Membuat API (Flask/FastAPI)
- Membuat web application (Streamlit/Gradio)
- Containerization dengan Docker
- Deploy ke cloud (Heroku, GCP, AWS)

Optimization:

- Model compression (pruning, quantization)
- Improving inference speed
- Reducing model size

10. REPRODUCIBILITY (WAJIB)

10.1 GitHub Repository

Link Repository: [URL GitHub Anda]

Repository harus berisi:

- Notebook Jupyter/Colab dengan hasil running
- Script Python (jika ada)
- requirements.txt atau environment.yml
- README.md yang informatif
- Folder structure yang terorganisir
- .gitignore (jangan upload dataset besar)

10.2 Environment & Dependencies

Python Version: [3.8 / 3.9 / 3.10 / 3.11]

Main Libraries & Versions:

numpy==1.24.3

pandas==2.0.3

```
scikit-learn==1.3.0
matplotlib==3.7.2
seaborn==0.12.2

# Deep Learning Framework (pilih salah satu)
tensorflow==2.14.0 # atau
torch==2.1.0      # PyTorch

# Additional libraries (sesuaikan)
xgboost==1.7.6
lightgbm==4.0.0
opencv-python==4.8.0 # untuk computer vision
nltk==3.8.1        # untuk NLP
transformers==4.30.0 # untuk BERT, dll
```