Supervised_Infolvators

Team:

1. M Rizqi Fadhilah
2. M Arvin Fadriansyah
3. Melliza Nastasia Izazi
4. Thufael Bintang Alfattah
5. Zulfikar fauzi
6. Annisa Sulistyaningsih
7. Niken Mustikaweni
8. Galih refa

Submission:

1. Report: (Link menuju google docs agar dapat diberikan feedback, tolong tambahkan akses comment)
2. Notebook: (Link menuju google colab agar dapat diberikan feedback, tolong tambahkan akses comment)

Simple Exploratory Data Analysis

```
[ ]  # Melakukan Importing Library yang akan dibutuhkan nanti
     import warnings
     import pandas as pd
     import numpy as np
     from sklearn.preprocessing import LabelEncoder
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.linear_model import LinearRegression
     from sklearn.metrics import mean_absolute_error, mean_squared_error
     from lazypredict.Supervised import LazyRegressor
```

Pada tahap EDA kami Mengimpor berbagai library dan modul dari **sklearn** (scikit-learn) dan **lazypredict** yang akan digunakan untuk melakukan preprocessing data, pemodelan regresi, dan evaluasi model.

```
[ ]  # Mengabaikan semua peringatan
     warnings.filterwarnings('ignore')

     # Proses ekstraksi data
     raw_data = pd.read_excel('youtube_statistics.xlsx')

     # Tampilkan data
     display(raw_data)
```

Disini kami Menggunakan warnings. Filterwarning untuk mengabaikan semua peringatan yang mungkin muncul selama eksekusi kode.

```
[ ]  # Periksa variabel kategorik
     raw_data.select_dtypes(include = 'object')
```

	title	channel_title	tags	description
0	Sharry Mann: Cute Munda (Song Teaser) | Parmi...	Lokdhun Punjabi	sharry mann|"sharry mann new song"|"sharry man...	Presenting Sharry Mann latest Punjabi Song Cu...
1	पीरियड्स के समय, पेट पर पति करता ऐसा, देखकर दं...	HJ NEWS	पीरियड्स के समय|"पेट पर पति करता ऐसा"|"देखकर द...	पीरियड्स के समय, पेट पर पति करता ऐसा, देखकर दं...
2	Stylish Star Allu Arjun @ ChaySam Wedding Rece...	TFPC	Stylish Star Allu Arjun @ ChaySam Wedding Rece...	Watch Stylish Star Allu Arjun @ ChaySam Weddin...
3	Eruma Saani | Tamil vs English	Eruma Saani	Eruma Saani|"Tamil Comedy Videos"|"Films"|"Mov...	This video showcases the difference between pe...
4	why Samantha became EMOTIONAL @ Samantha naga ...	Filmylooks	Filmylooks|"latest news"|"telugu movies"|"telu...	why Samantha became EMOTIONAL @ Samantha naga ...
...	...	...	...	...
36786	फेकू आशिक़ - राजस्थान की सबसे शानदार कॉमेडी | ...	RDC Rajasthani	twinkle vaishnav comedy|"twinkle vaishnav"|"tw...	PRG Music & RDC Rajasthani presents फेकू आशिक़...
36787	Seetha | Flowers | Ep# 364	Flowers TV	flowers serials|"actress"|"malayalam serials"|...	Flowers - A R Rahman Show,Book your Tickets He...
36788	Bhramanam I Episode 87 - 12 June 2018 I Mazhav...	Mazhavil Manorama	mazhavil manorama|"bhramanam full episode"|"gt...	Subscribe to Mazhavil Manorama now for your da...
36789	Nua Bohu | Full Ep 285 | 13th June 2018 | Odia...	Tarang TV	tarang|"tarang tv"|"tarang tv online"|"tarang ...	Nuabohu : Story of a rustic village girl who w...
36790	Ee Nagaraniki Emaindi Trailer | Tharun Bhascke...	Suresh Productions	Ee Nagaraniki Emaindi|"Ee Nagaraniki Emaindi T...	Check out Ee Nagaraniki Emaindi Trailer #EeNag...

36791 rows × 4 columns

Menampilkan semua kolom dalam DataFrame raw_data yang memiliki tipe data 'object'. yang berguna untuk mengidentifikasi variabel-variabel kategorik dalam dataset

```
[ ]  # Hapus kolom free text
     raw_data = raw_data.drop(columns = ['title', 'channel_title', 'tags', 'description'])
```

```
     raw_data.head()
```

rending_date	category_id	publish_time	views	likes	dislikes	comment_count	comments_disabled	ratings_disabled	video_error_or_removed	No_tags	desc_len	len_title	publish_date
2017-11-14	1	0 days 12:20:39	1096327	33966	798	882	False	False	False	15	920	81	2017-11-12
2017-11-14	25	0 days 05:43:56	590101	735	904	0	True	False	False	19	2232	58	2017-11-13
2017-11-14	24	0 days 15:48:08	473988	2011	243	149	False	False	False	14	482	58	2017-11-12
2017-11-14	23	0 days 07:08:48	1242680	70353	1624	2684	False	False	False	20	263	30	2017-11-12
2017-11-14	24	0 days 01:14:16	464015	492	293	66	False	False	False	11	753	88	2017-11-13

Data Preprocessing

```
[ ]  col_dates = ['trending_date', 'publish_date']

     for coldate in col_dates:
         raw_data[coldate] = pd.to_datetime(raw_data[coldate])

     raw_data['publish_time'] = pd.to_timedelta(raw_data['publish_time'])
```

```
[ ]  raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36791 entries, 0 to 36790
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   trending_date          36791 non-null  datetime64[ns]
 1   title                  36791 non-null  object
 2   channel_title          36791 non-null  object
 3   category_id            36791 non-null  int64
 4   publish_time           36791 non-null  timedelta64[ns]
 5   tags                   36791 non-null  object
 6   views                  36791 non-null  int64
 7   likes                  36791 non-null  int64
 8   dislikes               36791 non-null  int64
 9   comment_count          36791 non-null  int64
 10  comments_disabled      36791 non-null  bool
 11  ratings_disabled       36791 non-null  bool
 12  video_error_or_removed 36791 non-null  bool
 13  description            36746 non-null  object
 14  No_tags                36791 non-null  int64
 15  desc_len               36791 non-null  int64
 16  len_title              36791 non-null  int64
 17  publish_date           36791 non-null  datetime64[ns]
dtypes: bool(3), datetime64[ns](2), int64(8), object(4), timedelta64[ns](1)
memory usage: 4.3+ MB
```

Mengubah tipe data kolom 'publish_time' menjadi tipe data **timedelta** sedangkan data kolom 'trending_date' menjadi tipe data **datetime** Tujuannya adalah untuk menyiapkan dataset dengan format tanggal dan waktu yang sesuai, agar dapat diproses dan dianalisis dengan lebih mudah.

Feature Engineering

```
[ ]  def categorize_time(td):
         hours = td.total_seconds() / 3600
         if (5 <= hours < 12):
             return 'morning'
         elif (12 <= hours < 17):
             return 'afternoon'
         elif (17 <= hours < 21):
             return 'evening'
         else:
             return 'night'
```

Untuk Keterangan Feature Engineering sebagai Berikut :

- hours = td.total_seconds() / 3600 menghitung jumlah jam dari parameter td (yang adalah tipe data timedelta).
- Fungsi ini menggunakan kondisi if-elif-else untuk mengkategorikan waktu berdasarkan jumlah jam:
- Jika jam antara 5 dan kurang dari 12, fungsi akan mengembalikan string 'morning'.
- Jika jam antara 12 dan kurang dari 17, fungsi akan mengembalikan string 'afternoon'.
- Jika jam antara 17 dan kurang dari 21, fungsi akan mengembalikan string 'evening'.
- Jika tidak masuk ke kategori di atas, fungsi akan mengembalikan string 'night'.

Fungsi ini berguna untuk mengkategorikan waktu publikasi suatu konten menjadi "pagi", "siang", "sore", atau "malam" berdasarkan jam publikasinya. Hal ini dapat membantu dalam analisis tren konten yang dipublikasikan pada waktu-waktu tertentu dalam sehari.

After

```
[ ]  raw_data['time_of_day'] = raw_data['publish_time'].apply(categorize_time)

     raw_data.head()
```

	trending_date	category_id	publish_time	views	likes	dislikes	comment_count	comments_disabled	ratings_disabled	video_error_or_removed	No_tags	desc_len	len_title	publish_date	time_of_day
0	2017-11-14	1	0 days 12:20:39	1096327	33966	798	882	False	False	False	15	920	81	2017-11-12	afternoon
1	2017-11-14	25	0 days 05:43:56	590101	735	904	0	True	False	False	19	2232	58	2017-11-13	morning
2	2017-11-14	24	0 days 15:48:08	473988	2011	243	149	False	False	False	14	482	58	2017-11-12	afternoon
3	2017-11-14	23	0 days 07:08:48	1242680	70353	1624	2684	False	False	False	20	263	30	2017-11-12	morning
4	2017-11-14	24	0 days 01:14:16	464015	492	293	66	False	False	False	11	753	88	2017-11-13	night

Kemudian kami menambahkan kolom baru 'time_of_day' ke dataset raw_data, yang mengkategorikan waktu publikasi video menjadi "morning", "afternoon, "evening", atau "night" berdasarkan jam publikasinya.

Feature Selection

Mengidentifikasi fitur-fitur paling relevan yang dapat membantu dalam memahami karakteristik dan tren video pada dataset ini. Fitur-fitur ini dapat mewakili karakteristik video, seperti popularitas (views, likes, dislikes), interaksi pengguna (comment_count, comments_disabled, ratings_disabled), dan informasi teknis (video_error_or_removed, No_tags, desc_len, len_title).

```
[ ] raw_data = raw_data.drop(columns = ['trending_date', 'category_id', 'publish_time', 'publish_date'])

    raw_data.head()
```

	views	likes	dislikes	comment_count	comments_disabled	ratings_disabled	video_error_or_removed	No_tags	desc_len	len_title	time_of_day
0	1096327	33966	798	882	False	False	False	15	920	81	afternoon
1	590101	735	904	0	True	False	False	19	2232	58	morning
2	473988	2011	243	149	False	False	False	14	482	58	afternoon
3	1242680	70353	1624	2684	False	False	False	20	263	30	morning
4	464015	492	293	66	False	False	False	11	753	88	night

Disini kami menyederhanakan dataset dengan hanya mempertahankan kolom-kolom yang relevan untuk analisis lebih lanjut, seperti metrik terkait video **(views, likes, dislikes, dll.)**, informasi tentang video **(No_tags, desc_len, len_title)**, dan kategori waktu publikasi** (time_of_day)**.

Encoding

```
[ ] col_encode = ['comments_disabled', 'ratings_disabled', 'video_error_or_removed']

    map_boolean = {
        False : 0,
        True : 1
    }

    for col_enc in col_encode:
        raw_data[col_enc] = raw_data[col_enc].map(map_boolean)
```

1. **col_encode = ['comments_disabled', 'ratings_disabled','video_error_or_removed']:**
    - menentukan kolom-kolom yang akan di-encode menggunakan One-Hot Encoding.
2. **map_boolean = { 'False': 0, 'True': 1 }**:

- mendefinisikan sebuah pemetaan (dictionary) untuk mengubah nilai boolean 'False' menjadi 0 dan 'True' menjadi 1.

3. **for col_enc in col_encode:**
   - merupakan loop yang akan berjalan untuk setiap kolom yang terdapat dalam col_encode.
4. raw_data[col_enc] = raw_data[col_enc].map(map_boolean):
   - pemetakan menggunakan map_boolean dictionary pada setiap kolom yang terdapat dalam col_encode. yang akan mengubah nilai boolean pada kolom-kolom tersebut menjadi nilai numerik 0 dan 1.

Secara singkat, kode ini melakukan One-Hot Encoding pada kolom-kolom tertentu untuk mengubah data kategorikal menjadi format numerikal.

```
[ ] from sklearn.preprocessing import LabelEncoder

    # Inisialisasi LabelEncoder
    le = LabelEncoder()

    # Fit dan transform kolom kategorik
    raw_data['time_of_day'] = le.fit_transform(raw_data['time_of_day'])

    # Tampilkan
    raw_data.head()
```

	views	likes	dislikes	comment_count	comments_disabled	ratings_disabled	video_error_or_removed	No_tags	desc_len	len_title	time_of_day
0	1096327	33966	798	882	0	0	0	15	920	81	0
1	590101	735	904	0	1	0	0	19	2232	58	2
2	473988	2011	243	149	0	0	0	14	482	58	0
3	1242680	70353	1624	2684	0	0	0	20	263	30	2
4	464015	492	293	66	0	0	0	11	753	88	3

1. From sklearn.preprocessing import **LabelEncoder**:
   - mengimport kelas **LabelEncoder** dari modul **sklearn.preprocessing**. LabelEncoder digunakan untuk mengkodekan fitur kategorikal menjadi nilai numerik.
2. Inisialisasi LabelEncoder:
   - menginisialisasi objek LabelEncoder dengan perintah **le = LabelEncoder()**.
3. Fit dan transform kolom kategorik:
   - Baris **raw_data['time_of_day'] = le.fit_transform(raw_data['time_of_day'])** melakukan transformasi pada kolom 'time_of_day' menggunakan LabelEncoder. Transformasi ini akan mengubah nilai kategorikal pada kolom tersebut menjadi nilai numerik.
4. Tampilkan: Perintah **raw_data.head()** akan menampilkan beberapa baris awal dari dataset raw_data.

Secara singkat, kode ini melakukan Label Encoding pada kolom 'time_of_day' dalam dataset raw_data. Proses ini bertujuan untuk mengubah nilai kategorikal menjadi nilai numerik agar dapat diproses oleh model pembelajaran mesin.

Splitting Data

```
[ ]  X = raw_data.drop(columns = ['views'])
     y = raw_data[['views']]
```

X: DataFrame yang berisi fitur-fitur **(variabel independen)** yang akan digunakan sebagai input ke dalam model. **Kolom 'views' telah dihapus dari X**.

Y: DataFrame yang hanya berisi **kolom target 'views'**, yang merupakan nilai yang ingin diprediksi oleh model.

```
X
```

	likes	dislikes	comment_count	comments_disabled	ratings_disabled	video_error_or_removed	No_tags	desc_len	len_title	time_of_day
0	33966	798	882	0	0	0	15	920	81	0
1	735	904	0	1	0	0	19	2232	58	2
2	2011	243	149	0	0	0	14	482	58	0
3	70353	1624	2684	0	0	0	20	263	30	2
4	492	293	66	0	0	0	11	753	88	3
...	...	...	...	...	...	...	...	...	...	...
36786	3291	404	196	0	0	0	19	818	100	2
36787	1726	478	1428	0	0	0	8	468	26	2
36788	1216	453	697	0	0	0	8	2187	57	2
36789	698	115	65	0	0	0	15	432	64	0
36790	22466	1609	1205	0	0	0	14	1030	76	3

36791 rows × 10 columns

Untuk menampilkan Nilai X yang tidak memiliki kolom views

Setiap kolom dalam DataFrame X mewakili fitur yang digunakan untuk memprediksi views:

1. likes: Jumlah likes yang diterima oleh video.
2. dislikes: Jumlah dislikes yang diterima oleh video.
3. comment_count: Jumlah komentar yang diterima oleh video.
4. comments_disabled: Status apakah komentar pada video dinonaktifkan (0 berarti tidak dinonaktifkan, 1 berarti dinonaktifkan).

5. ratings_disabled: Status apakah rating pada video dinonaktifkan (0 berarti tidak dinonaktifkan, 1 berarti dinonaktifkan).
6. video_error_or_removed: Status apakah video memiliki error atau dihapus (0 berarti tidak, 1 berarti ya).
7. No_tags: Jumlah tag yang digunakan dalam video.
8. desc_len: Panjang deskripsi video dalam jumlah karakter.
9. len_title: Panjang judul video dalam jumlah karakter.
10. time_of_day: Waktu dalam hari saat video diunggah (dinyatakan dalam bilangan bulat yang mewakili jam).

```
[ ] y
```

	views
0	1096327
1	590101
2	473988
3	1242680
4	464015
...	...
36786	214378
36787	406828
36788	386319
36789	130263
36790	1278249

36791 rows × 1 columns

Menampilkan kolom y yang berisi kolom views yang sudah dipisah sebelumnya.

```
[ ] from sklearn.model_selection import train_test_split

    # Bagi data menjadi training dan testing set
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Melakukan pembagian Data Training dan Testing set yaitu :

1. Test_size = 0.3 yang berarti dari data tersebut akan diambil sekitar 30% dari total data pada `youtube_statistic`
2. Training = Maka sisa atau 70% dari data tersebut akan menjadi data training.

```
[ ] print(f'Total data awal {raw_data.shape}')
    print(f'Jumlah data latih : {X_train.shape} dan jumlah data test : {X_test.shape}')

    Total data awal (36791, 11)
    Jumlah data latih : (25753, 10) dan jumlah data test : (11038, 10)
```

Dari data yang didapatkan Total data awal dari Dataset `youtube_statistic` terdapat **36791 data dan 11 kolom,** yang kita **bagi menjadi menjadi 7 banding 3**

dimana 7 merupakan data training : **Jumlah data latih : (25753, 10)** , dan 3 pada data testing : **jumlah data test : (11038, 10)**

Scaling

```
[ ] from sklearn.preprocessing import MinMaxScaler

    # Rentang default adalah [0, 1]
    scaler = MinMaxScaler()

    # Fit dan transform data
    scaled_train = scaler.fit_transform(X_train)

    # Buat DataFrame dari data yang telah diskalakan
    X_train_scaled = pd.DataFrame(scaled_train, columns=X_train.columns)

    X_train_scaled
```

	likes	dislikes	comment_count	comments_disabled	ratings_disabled	video_error_or_removed	No_tags	desc_len	len_title	time_of_day
0	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.02	0.49	1.00
1	0.00	0.00	0.00	0.00	0.00	0.00	0.31	0.07	0.87	0.67
2	0.01	0.00	0.00	0.00	0.00	0.00	0.24	0.16	0.75	0.33
3	0.01	0.00	0.01	0.00	0.00	0.00	0.42	0.06	0.48	0.33
4	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.11	0.77	0.67
...	...	...	...	...	...	...	...	...	...	...
25748	0.00	0.00	0.00	0.00	0.00	0.00	0.31	0.06	0.25	0.67
25749	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.06	0.31	0.33
25750	0.00	0.00	0.00	0.00	0.00	0.00	0.24	0.19	0.43	0.00
25751	0.00	0.00	0.00	0.00	0.00	0.00	0.28	0.22	0.98	0.00
25752	0.00	0.00	0.00	0.00	1.00	0.00	0.08	0.10	0.85	0.67

25753 rows × 10 columns

Kode di atas menunjukkan proses normalisasi data menggunakan MinMaxScaler pada set data pelatihan (X_train). Data dalam X_train_scaled menunjukkan bahwa semua fitur telah dinormalisasi ke dalam rentang [0, 1]. Contoh data yang ditampilkan menunjukkan nilai-nilai yang telah dinormalisasi untuk beberapa fitur seperti likes, dislikes, comment_count, dan lainnya.

Normalisasi ini penting untuk memastikan bahwa semua fitur berkontribusi secara proporsional dalam model pembelajaran mesin dan tidak ada fitur yang mendominasi karena rentang nilainya yang lebih besar.

```
[ ] # Fit dan transform data
    scaled_y_train = scaler.fit_transform(y_train)

    # Buat DataFrame dari data yang telah diskalakan
    y_train_scaled = pd.DataFrame(scaled_y_train, columns=y_train.columns)

    y_train_scaled
```

	views
0	0.00
1	0.00
2	0.03
3	0.04
4	0.00
...	...
25748	0.00
25749	0.00
25750	0.00
25751	0.00
25752	0.03

25753 rows × 1 columns

1.Fit dan Transform Data:

scaled_y_train = scaler.fit_transform(y_train):

Melakukan fitting scaler pada data target (y_train) dan kemudian mentransformasi data tersebut. Proses fitting menghitung nilai minimum dan maksimum dari y_train, dan transformasi mengubah nilai-nilai tersebut ke dalam rentang [0, 1].

2.Membuat DataFrame Baru:

y_train_scaled = pd.DataFrame(scaled_y_train, columns=y_train.columns):

Membuat DataFrame y_train_scaled dari data yang telah diskalakan (scaled_y_train), dengan mempertahankan nama kolom yang sama seperti pada y_train.

3.Hasil Akhir:

Data dalam y_train_scaled menunjukkan bahwa semua nilai target (views) telah dinormalisasi ke dalam rentang [0, 1]. Ini berarti nilai terkecil diubah menjadi 0 dan nilai terbesar diubah menjadi 1, dengan nilai lainnya disesuaikan secara proporsional di antara 0 dan 1.

```python
from sklearn.preprocessing import MinMaxScaler

# Rentang default adalah [0, 1]
scaler = MinMaxScaler()

# Fit dan transform data
scaled_test = scaler.fit_transform(X_test)
scaled_y_test = scaler.fit_transform(y_test)

# Buat DataFrame dari data yang telah diskalakan
X_test_scaled = pd.DataFrame(scaled_test, columns=X_test.columns)
y_test_scaled = pd.DataFrame(scaled_y_test, columns=y_test.columns)

X_test_scaled
```

	likes	dislikes	comment_count	comments_disabled	ratings_disabled	video_error_or_removed	No_tags	desc_len	len_title	time_of_day
0	0.00	0.00	0.00	0.00	0.00	0.00	0.30	0.20	0.94	0.67
1	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.02	0.83	0.33
2	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.06	0.67	0.00
3	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.33	0.94	1.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.27	0.21	1.00	0.67
...	...	...	...	...	...	...	...	...	...	...
11033	0.00	0.00	0.00	0.00	0.00	0.00	0.24	0.31	0.53	0.00
11034	0.01	0.00	0.00	0.00	0.00	0.00	0.27	0.59	0.12	0.33
11035	0.00	0.00	0.00	0.00	0.00	0.00	0.30	0.49	0.98	1.00
11036	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.07	0.27	0.33
11037	0.00	0.00	0.00	1.00	0.00	0.00	0.13	0.09	0.69	0.33

11038 rows × 10 columns

- **from sklearn.preprocessing import MinMaxScaler** untuk mengubah skala data numerik ke rentang yang ditentukan
- **scaler = MinMaxScaler()** MinMaxScaler dibuat dengan rentang default [0, 1].
- **scaled_test = scaler.fit_transform(X_test) scaled_y_test =** scaler.fit_transform(y_test) digunakan untuk menyesuaikan skala dengan data dan kemudian mentransformasi data tersebut. Dalam hal ini, X_test dan y_test adalah dataset yang akan diskalakan.
- **X_test_scaled** = pd.DataFrame(scaled_test, columns=X_test.columns)
- **y_test_scaled** = pd.DataFrame(scaled_y_test, columns=y_test.columns)
- Setelah data diskalakan, hasilnya dikonversi kembali menjadi DataFrame Pandas dengan kolom yang sama seperti dataset asli.
- **X_test_scaled** Menampilkan DataFrame X_test_scaled yang berisi data X_test yang sudah diskalakan.

```
[ ] # Ubah y value menjadi 1 dimensi
    y_train_scaled = y_train_scaled.to_numpy().ravel()
    y_test_scaled = y_test_scaled.to_numpy().ravel()
```

```
[ ] print(X_train_scaled.shape)
    print(X_test_scaled.shape)
    print(y_train_scaled.shape)
    print(y_test_scaled.shape)
```

```
(25753, 10)
(11038, 10)
(25753,)
(11038,)
```

- Variabel target diubah menjadi 1 dimensi agar lebih mudah digunakan untuk perhitungan selanjutnya.

Modeling

```
# Cari Algoritma regresi terbaik
reg = LazyRegressor(verbose=0, custom_metric=None, predictions=True)
models, predictions = reg.fit(X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled)
models
```

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
ExtraTreesRegressor	0.97	0.97	0.00	10.81
RandomForestRegressor	0.95	0.95	0.01	21.94
XGBRegressor	0.95	0.95	0.01	1.14
BaggingRegressor	0.94	0.94	0.01	2.62
DecisionTreeRegressor	0.90	0.91	0.01	0.39
GradientBoostingRegressor	0.89	0.89	0.01	5.70
KNeighborsRegressor	0.89	0.89	0.01	2.32
NuSVR	0.89	0.89	0.01	276.28
LGBMRegressor	0.88	0.88	0.01	0.20
ExtraTreeRegressor	0.87	0.87	0.01	0.18
HistGradientBoostingRegressor	0.86	0.86	0.01	1.28
ElasticNetCV	0.76	0.76	0.01	0.51
LassoCV	0.76	0.76	0.01	0.45
RidgeCV	0.76	0.76	0.01	0.05
BayesianRidge	0.76	0.76	0.01	0.07
Ridge	0.76	0.76	0.01	0.03
LassoLarsCV	0.76	0.76	0.01	0.17
LassoLarsIC	0.76	0.76	0.01	0.12
LarsCV	0.76	0.76	0.01	0.18
Lars	0.76	0.76	0.01	0.35
TransformedTargetRegressor	0.76	0.76	0.01	0.03
LinearRegression	0.76	0.76	0.01	0.07
OrthogonalMatchingPursuitCV	0.75	0.76	0.01	0.04
AdaBoostRegressor	0.75	0.75	0.01	1.72
OrthogonalMatchingPursuit	0.71	0.71	0.01	0.03
TweedieRegressor	0.64	0.64	0.02	0.03
HuberRegressor	0.62	0.62	0.02	0.39
MLPRegressor	0.18	0.18	0.02	2.18
PoissonRegressor	0.07	0.07	0.03	0.04

Terbaik Secara Keseluruhan: ExtraTreesRegressor – Memiliki Adjusted R-Squared dan R-Squared tertinggi, serta RMSE terendah.
Peringkat Kedua: RandomForestRegressor – Performa mendekati ExtraTreesRegressor tetapi memerlukan waktu lebih lama untuk dijalankan.

Performa Cepat yang Baik: XGBRegressor – Metrik yang sebanding tetapi waktu eksekusi secara signifikan lebih cepat.
ExtraTreesRegressor umumnya dianggap sebagai model terbaik berdasarkan hasil ini, menyeimbangkan performa dan waktu komputasi.

```
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.model_selection import GridSearchCV, KFold

extraTreeRegressor = ExtraTreesRegressor()
cv = KFold(random_state=123, shuffle=True)
param_grid = {'criterion': ['squared_error', 'friedman_mse'],
              'min_samples_split': [2, 5, 10, 15, 20, 25],
              'max_depth' : [5, 10, 15, None]}
clf2 = GridSearchCV(estimator=extraTreeRegressor, param_grid=param_grid, cv=cv)

clf2.fit(X_train_scaled, y_train_scaled)
clf2.best_params_
```

```
{'criterion': 'squared_error', 'max_depth': None, 'min_samples_split': 2}
```

ExtraTreesRegressor: Sebuah model machine learning yang membangun beberapa decision tree dan merata-rata hasil prediksinya untuk meningkatkan akurasi dan mengurangi overfitting.
GridSearchCV: alat yang melakukan pencarian menyeluruh (grid search) untuk kombinasi hyperparameter terbaik berdasarkan kriteria evaluasi tertentu (misalnya, akurasi, MSE). GridSearchCV mencoba semua kombinasi yang mungkin dari hyperparameter yang diberikan.
KFold: Ini adalah metode cross-validation yang membagi data menjadi beberapa subset (folds). Model dilatih pada beberapa subset ini dan divalidasi pada subset lain, yang memastikan bahwa model diuji pada berbagai bagian data yang berbeda.

param_grid: dictionary yang menentukan hyperparameter yang akan diuji. Setiap hyperparameter memiliki beberapa opsi yang akan dicoba oleh GridSearchCV.
criterion: Menentukan fungsi yang digunakan untuk mengukur kualitas split di setiap node pohon.
squared_error: Mean Squared Error standar, yang menghitung kuadrat dari selisih antara nilai yang diprediksi dan nilai sebenarnya.
friedman_mse: Variasi dari MSE yang dioptimalkan untuk pohon regresi, dapat memberikan hasil yang lebih baik dalam situasi tertentu.
min_samples_split: Menentukan jumlah minimum sampel yang diperlukan untuk membagi node internal. Nilai yang lebih besar membuat pohon lebih konservatif, menghindari overfitting.
Contoh: min_samples_split=2 berarti node akan dibagi jika ada setidaknya 2 sampel.
max_depth: Menentukan kedalaman maksimum pohon. Batasan kedalaman pohon membantu mencegah pohon menjadi terlalu kompleks, yang bisa mengakibatkan overfitting.
Contoh: max_depth=10 berarti pohon akan dibatasi hingga 10 tingkat.

clf2.best_params_: Setelah model dilatih dengan berbagai kombinasi hyperparameter, kode ini akan mengembalikan kombinasi hyperparameter yang memberikan performa terbaik berdasarkan evaluasi cross-validation.

Prediksi

```
etr = ExtraTreesRegressor(**clf2.best_params_)
etr.fit(X_train_scaled, y_train_scaled)
y_pred = etr.predict(X_test_scaled)
y_pred
```

```
array([0.00261027, 0.00606368, 0.00253015, ..., 0.0045541 , 0.004891  ,
       0.00465161])
```

Kode dilatih dengan menggunakan hyperparameter clf2.best_param
kemudian model diuji ke x_test_scaled untuk mengetahui apakah model sudah akurat atau belum yang akan diuji di kose selanjutnya

Evaluasi

```
[ ] from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

    MAE = mean_absolute_error(y_test_scaled, y_pred)
    MSE = mean_squared_error(y_test_scaled, y_pred)
    RMSE = np.sqrt(mean_squared_error(y_test_scaled, y_pred))
    R_Squared = r2_score(y_test_scaled, y_pred)

    print(f'Error MAE = {MAE}')
    print(f'Error MSE = {MSE}')
    print(f'RMSE = {RMSE}')
    print(f'R-Squared = {R_Squared}')
```

```
Error MAE = 0.0018160574039167861
Error MSE = 2.091150892761421e-05
RMSE = 0.004572910334525947
R-Squared = 0.9689192049202318
```

MAE: Rata-rata dari perbedaan absolut antara nilai aktual (y_test_scaled) dan nilai prediksi (y_pred). MAE memberikan gambaran seberapa jauh prediksi berada dari nilai sebenarnya, tanpa mempertimbangkan arah kesalahan (apakah terlalu tinggi atau terlalu rendah).

MSE: Rata-rata dari kuadrat perbedaan antara nilai aktual dan nilai prediksi. MSE menekankan kesalahan yang lebih besar karena mengkuadratkan selisih, sehingga model yang menghasilkan prediksi dengan kesalahan besar akan memiliki nilai MSE yang lebih tinggi.
RMSE: Akar kuadrat dari MSE. RMSE berada dalam satuan yang sama dengan variabel target (dalam kasus ini, data yang telah di-scaled), sehingga lebih mudah diinterpretasikan daripada MSE. RMSE memberikan bobot lebih pada kesalahan besar, mirip dengan MSE, namun lebih mudah diinterpretasikan karena berada dalam skala yang sama dengan data.
R-Squared: Mengukur proporsi variasi dalam variabel target yang dapat dijelaskan oleh model. Nilai R^2 berkisar antara 0 dan 1, dengan 1 menunjukkan bahwa model sempurna dalam menjelaskan semua variasi dalam data target, dan 0 menunjukkan bahwa model tidak lebih baik daripada sekadar mengambil rata-rata dari target.

Summary

Dokumen ini merangkum proses EDA, mulai dari impor library hingga pemodelan dan evaluasi. Langkah-langkah utama meliputi:

1. Preprocessing: Konversi tipe data dan normalisasi.
2. Feature Engineering: Kategorisasi waktu publikasi menjadi "morning," "afternoon," "evening," atau "night."
3. Feature Selection: Identifikasi fitur penting seperti views, likes, dan comment_count.
4. Modeling & Evaluasi: Penggunaan berbagai algoritma regresi dengan penilaian RMSE dan R^2.

Rekomendasi Bisnis:
Analisis lebih dalam terhadap waktu publikasi dan fitur engagement dapat mengarahkan pada strategi konten yang optimal, misalnya, penjadwalan unggahan konten berdasarkan tren waktu yang menghasilkan interaksi tinggi.

Appendix

Tuliskan kontribusi pengerjaan masing-masing anggota tim di bagian ini

- M Rizqi Fadhilah (Ngedit Text, dan Menulis Kode awal, Menghimpun anggota)
- Mohamad Arvin Fadriansyah (Membantu menuliskan deskripsi dan revisi kode bersama, Menambahkan kode untuk modelling, Main Coder)
- Melliza Nastasia (Fokus Laporan, dan Membantu menuliskan deskripsi dan revisi kode bersama)
- Thufael Bintang Alfattah (Membantu menuliskan deskripsi dan revisi kode bersama

- Zulfikar fauzi
- Annisa Sulistyaningsih (Fokus Laporan, dan Membantu menuliskan deskripsi dan revisi kode bersama)
- Niken Mustikaweni (Ngedit Text, dan Membantu menuliskan deskripsi dan revisi kode bersama)
- Galih refa (Ngedit Text, dan Membantu menuliskan deskripsi dan revisi kode bersama

Tuliskan juga kesulitan-kesulitan dalam pengerjaan tugas ini di bagian ini (jika ada)

- Tingginya workload ketika modelling sehingga laptop lag
- Belum Menemukan Selain yang dipilih fitur selection yang tepat.