Homework Solution - Regression

Byte Me Team:

1. Hijir Della Wirasti
2. Mauliddinia Iftikhar Agnany
3. Jericho Medion Haryono
4. Fakhri Dwi Nugroho
5. Ryan Nofandi
6. Johannes Sibarani
7. Achmad Fichri Rachmadhani
8. Muhammad Naufal

Submission:

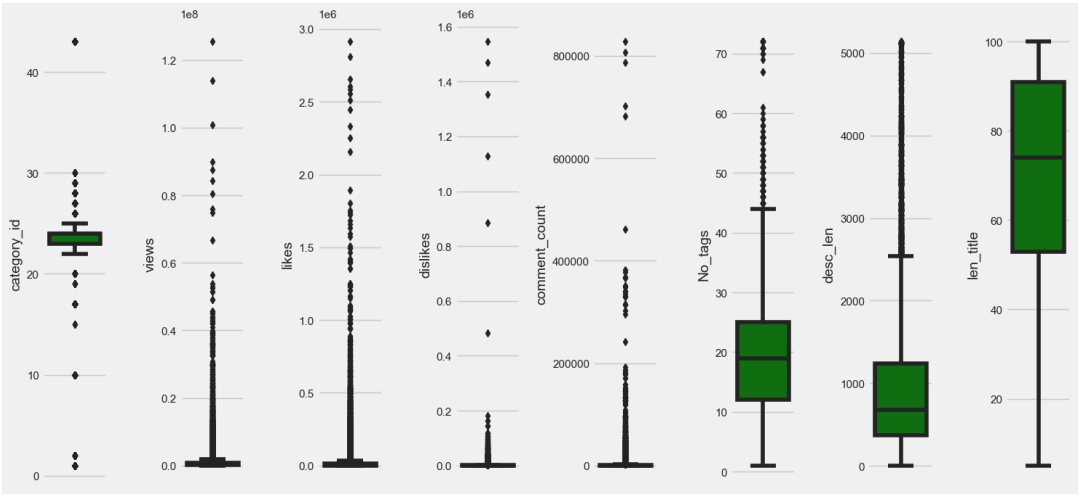
1. Report: (https://docs.google.com/document/d/1ZvzKKWqTntqrdgBdFbvA-i-LwM0szuVF/edit?usp=sharing&ouid=111561112532994450431&rtpof=true&sd=true)
2. Notebook: (https://drive.google.com/file/d/1Qujs23rji0HJh5ay2ZpnBjXoWfiURu99/view?usp=sharing)

---

1. Simple Exploratory Data Analysis

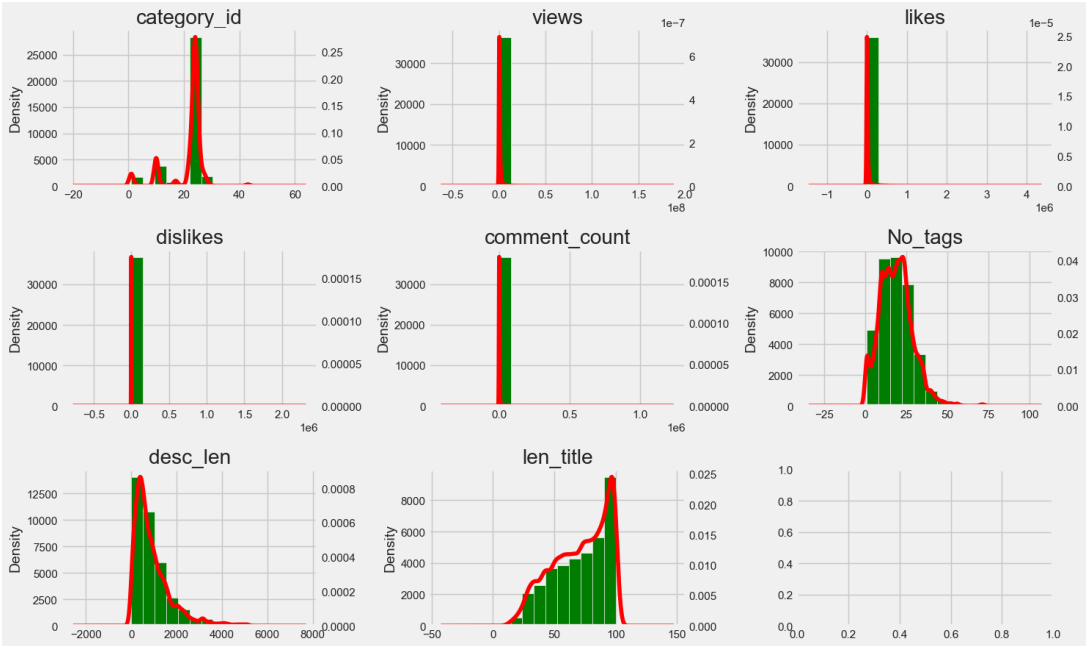
- Memahami struktur data dengan menggunakan fungsi *.head*(), *.info*(), atau *.describe*(). Kemudian membagi data menjadi 2 tipe variabel (*nums* dan *cats*)
  - Statistik dari numerical data menunjukkan sebaran data untuk views, likes, dislikes, comment_count, desc_len, dan len_title terlihat miring (mean & median kurang mendekati)

- Univariate Analysis
  - Dengan membagi menjadi 2 tipe data (*data_num* dan *data_cat)* dilakukan analisis menggunakan boxplot dan dapat dilihat bahwa fitur *view, likes, dislikes, comment_count, o_tags, and desc_len* memiliki banyak outliers.



Gambar 1. Boxplot

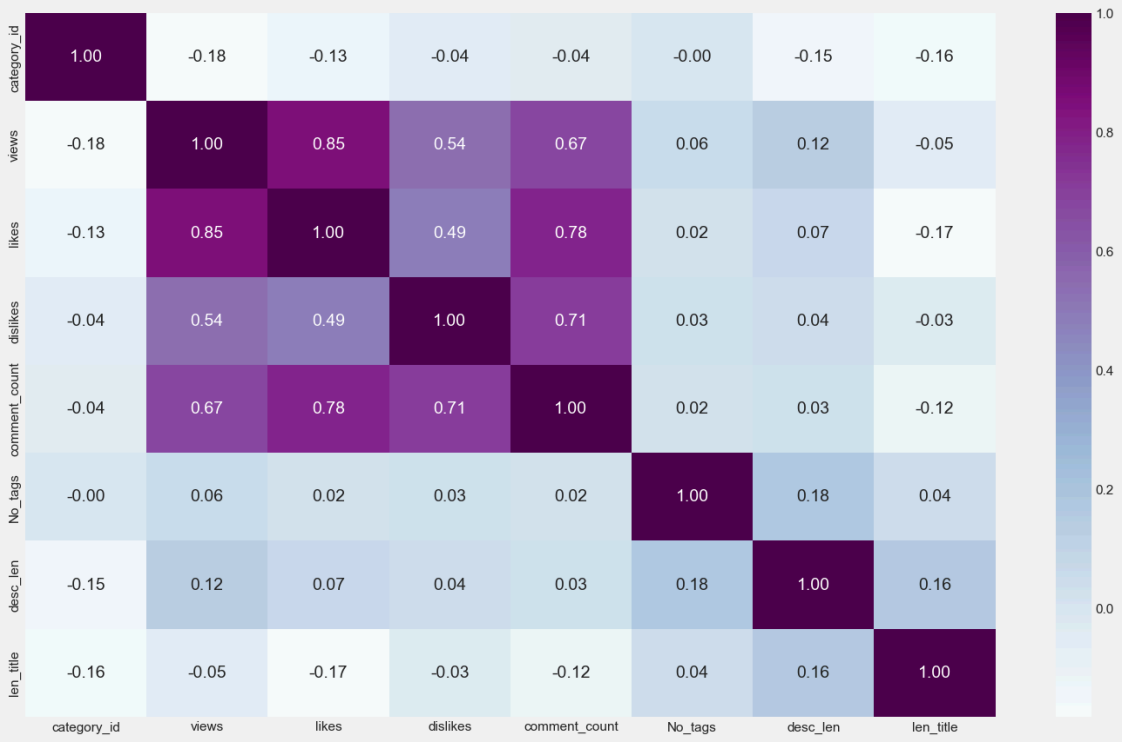
- Dilakukan analisis terhadap distribusi menggunakan *subplots* dan dapat dilihat bahwa beberapa fitur cenderung skewed positif kecuali pada fitur *len_title* cenderung skewed negatif



Gambar 2. Histogram

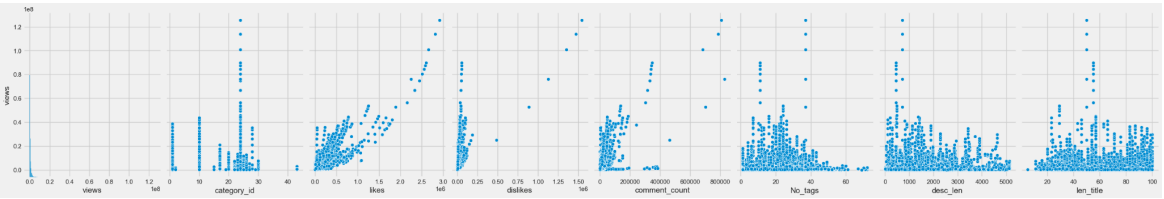
❖ Multivariate analysis

- Analisis menggunakan heatmap ditemukan korelasi positif yang cukup kuat pada *views, likes, dislikes,* dan *comment_count.*



Gambar 3. Heatmap

- Analisis menggunakan scatterplot
  - Sebagian besar variabel menunjukkan distribusi yang tidak merata dengan beberapa outlier yang jauh. Ini mungkin menunjukkan bahwa hanya sedikit video yang benar-benar sangat populer (viral), sementara sebagian besar video memiliki tingkat engagement yang lebih rendah.
  - Korelasi antara *views, likes, dislikes,* dan *comment_count* menunjukkan hubungan positif antara variabel karena video dengan views yang lebih banyak cenderung memiliki *likes, dislikes*, dan komentar yang lebih tinggi.
  - Penggunaan tag dan panjang deskripsi bervariasi, dan beberapa outliers menunjukkan bahwa beberapa video menggunakan lebih banyak tag atau deskripsi yang lebih panjang, mungkin sebagai upaya untuk meningkatkan engagement atau SEO.



Gambar 4. Scatter Plot

Analisis Exploratory Data Analysis (EDA)
Dari proses EDA yang telah dilakukan terutama pada analisis univariate dan multivariate maka fitur yang sebaiknya digunakan adalah *views, like, dislike, comment count, desc_len, No_tags*. Hal ini dikarenakan hasil analisis univariate dimana pada grafik boxplot terdapat **outliers** pada masing-masing fitur. Kemudian juga di analisis yang sama pada grafik subplots dapat dilihat terdapat grafik yang cenderung **skewed positif** dan **skewed negative**. Hal ini menunjukkan bahwasanya terdapat persebaran yang tidak merata dikarenakan adanya **outlier** sehingga butuh untuk melakukan **log transformasi** agar penyebaran lebih mendekati **distribusi normal.** Pada analisis multivariate terutama pada grafik heatmap dapat dilihat terdapat korelasi positif yang cukup kuat pada beberapa fitur diantaranya *views, like, dislike, comment count.* Kemudian pada grafik Scatterplot menunjukkan distribusi yang tidak merata. Oleh karena itu fitur-fitur tersebut akan dipilih untuk diproses sehingga dapat digunakan untuk tahap selanjutnya.

Data Preprocessing

- Melakukan pengecekan missing value untuk mengetahui kondisi dari data kita seperti apa sehingga dapat dilakukan tindakan perbaikan, pada dataset ini terdapat missing value pada kolom “description” sebanyak 45 atau sebesar 0.12%.
- Melakukan drop atau menghapus kolom yang kita anggap kurang relevan atau dapat mempengaruhi kinerja dari model yang akan kita buat. Pada dataset ini kita melakukan drop kolom pada 'publish_date','description','title','channel_title','tags', 'publish_time'.
- Melakukan handling duplikat, dimana pada dataset ini terdapat data duplikat sebesar 4229 sehingga kita lakukan penghapusan data duplikat untuk memastikan data bersih dan akurat.
- Handling Outlier pada kolom:
  ‘likes’,’dislikes’,’comment_count’,’desc_len’,’No_tags’,’views’.
    - Melakukan log transformasi pada data dengan distribusi skew sehingga dapat lebih baik atau mendekati distribusi normal.
    - Melakukan normalisasi untuk menstandarkan skala sehingga lebih konsisten dan memudahkan pada saat akan digunakan untuk membuat sebuah model.

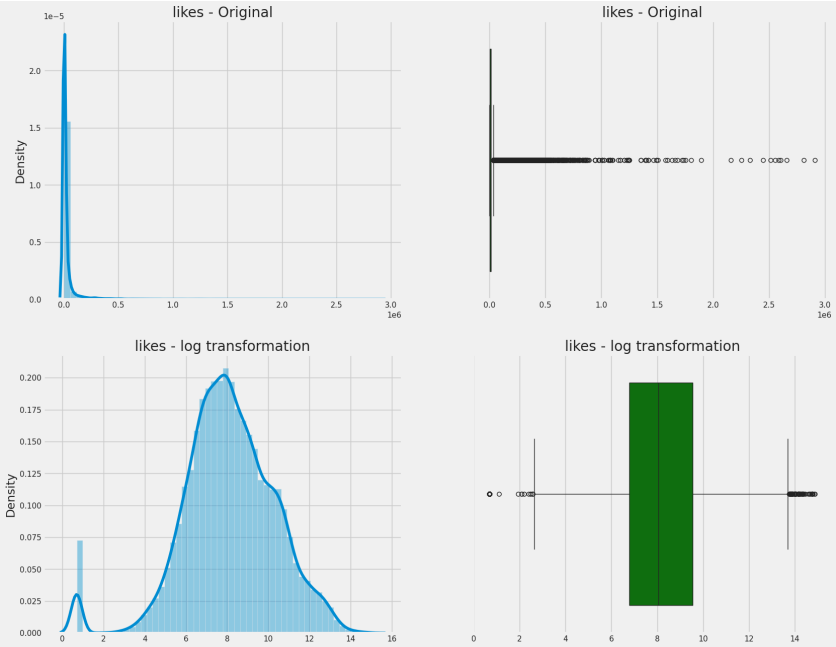
2. Feature Engineering

2A. Feature Engineering Tambahan yang Digunakan

A. Log Transformation:

- Kode `np.log1p(data_clean['likes']+1)` adalah salah satu contoh *feature engineering* yang digunakan untuk menangani distribusi yang sangat miring atau data dengan outliers.

- Log transformasi membantu mengurangi skewness, sehingga data lebih mendekati distribusi normal. Ini sangat berguna ketika data memiliki rentang nilai yang besar, seperti dalam hal ini jumlah "**likes**" yang mungkin sangat bervariasi.
- Transformasi ini diterapkan pada `likes`, seperti yang ditunjukkan oleh `distplot` dan `boxplot`.



Gambar 5. Log Transformation untuk Likes

Bagian Kiri Atas (Distribusi Likes - Original)

- **Distribution Plot (likes - Original)**:
  - Grafik ini menunjukkan distribusi awal dari data `likes`. Distribusi ini **sangat miring ke kanan** (right-skewed), dengan banyak nilai kecil di sekitar angka nol dan hanya sedikit nilai besar yang sangat jauh dari mayoritas data.
  - **Kepadatan sangat tinggi di nilai-nilai kecil**, yang menunjukkan bahwa sebagian besar data memiliki jumlah `likes` yang relatif rendah. Hanya sebagian kecil yang memiliki jumlah `likes` yang sangat besar, sehingga menciptakan "ekor" panjang di sebelah kanan grafik.

Bagian Kanan Atas (Boxplot Likes - Original)

- **Boxplot (likes - Original)**:
  - Boxplot ini menunjukkan adanya banyak **outliers** (titik-titik di luar "whiskers" dari boxplot). Ini mengindikasikan bahwa ada beberapa nilai `likes` yang sangat tinggi dan jauh dari mayoritas data, yang diklasifikasikan sebagai outliers.

    - Hampir semua data terkonsentrasi di dekat nilai rendah, tetapi beberapa data melampaui batas atas (di atas *upper quartile*), sehingga membuat boxplot terlihat sangat terkonsentrasi pada nilai kecil dengan banyak titik outliers.

Bagian Kiri Bawah (Distribusi Likes Setelah Log Transformation)

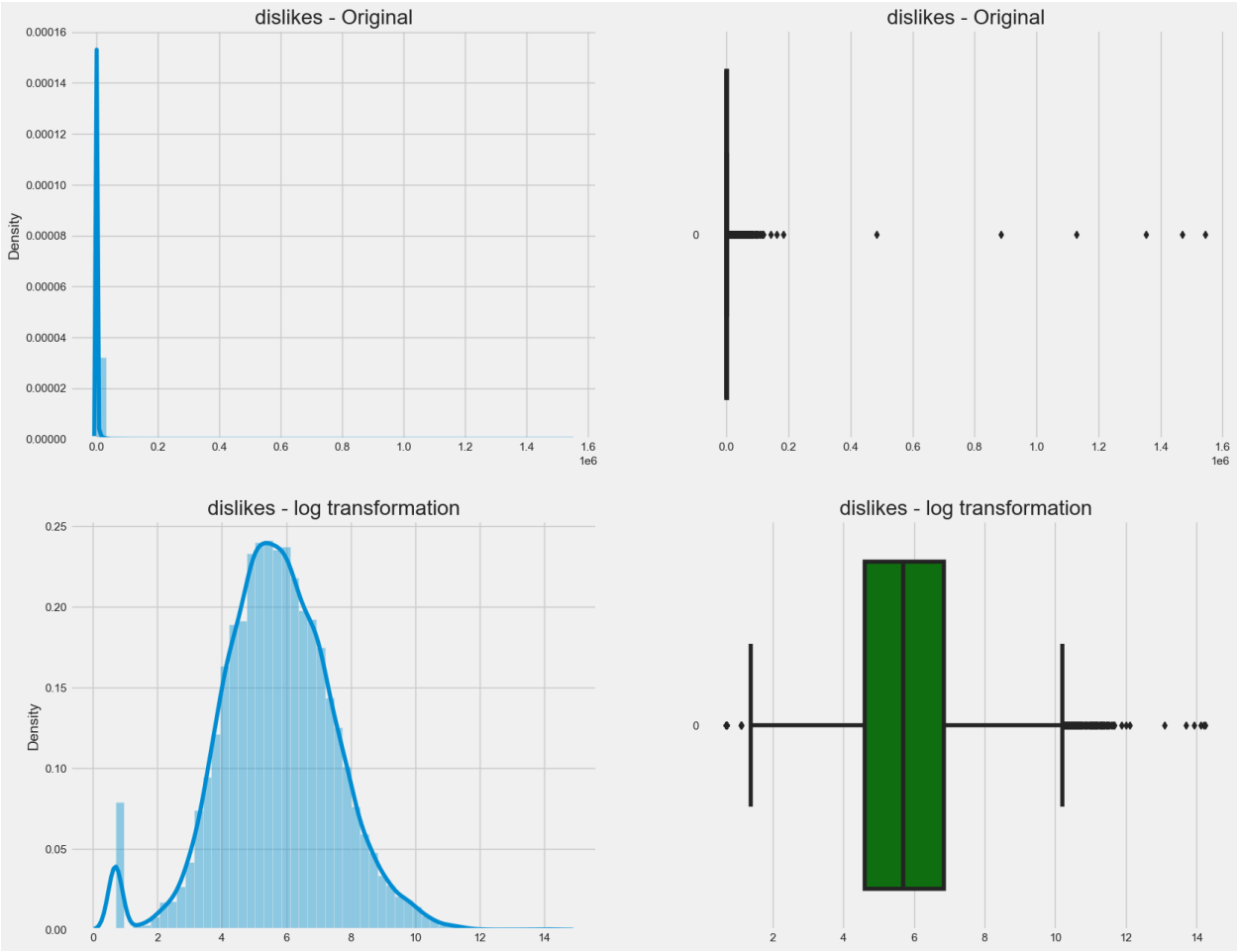
- **Distribution Plot (likes - Log Transformation)**:
    - Setelah menerapkan **log transformation**, distribusi `likes` berubah menjadi lebih **mendekati distribusi normal**. Grafik ini sekarang lebih simetris, dengan nilai-nilai yang tersebar lebih merata.
    - Dengan log transformasi, data `likes` yang awalnya sangat kecil tidak lagi terlalu terkonsentrasi, dan nilai `likes` yang besar telah "dikecilkan" secara relatif, sehingga distribusi menjadi lebih mudah diinterpretasikan untuk analisis atau pemodelan statistik.

Bagian Kanan Bawah (Boxplot Likes Setelah Log Transformation)

- **Boxplot (likes - Log Transformation)**:
    - Boxplot ini menunjukkan bahwa setelah log transformasi, data terlihat lebih terdistribusi dengan baik. Meskipun masih ada beberapa outliers, jumlahnya jauh lebih sedikit dibandingkan data asli. Ukuran box (rentang antar kuartil) juga menjadi lebih besar dan lebih representatif terhadap sebaran data, tidak lagi terkonsentrasi pada nilai-nilai kecil.
    - Log transformasi berhasil mengurangi dampak dari outliers dan memberikan representasi data yang lebih seimbang.

Kesimpulan:

- **Distribusi Asli** dari `likes` sangat skewed dan memiliki banyak outliers. Sebagian besar nilai berkumpul di dekat nol, dengan sedikit nilai yang sangat besar.
- **Setelah Log Transformation**, distribusi menjadi lebih normal atau simetris, yang membuat data lebih mudah dianalisis oleh model statistik atau prediksi. Outliers masih ada, tetapi jumlah dan dampaknya telah berkurang secara signifikan.



Gambar 6. Log Transformation untuk Dislikes

Bagian Kiri Atas (Distribusi Dislikes - Original)

- **Distribution Plot (dislikes - Original)**:
  - Grafik ini menunjukkan distribusi asli dari variabel `dislikes`. Distribusi ini **sangat miring ke kanan** (right-skewed), mirip dengan distribusi `likes` yang sebelumnya, dengan sebagian besar data berkumpul di nilai kecil dekat nol, sementara hanya sedikit data yang memiliki nilai `dislikes` yang sangat besar.
  - Terdapat sedikit nilai yang sangat besar, yang menghasilkan ekor panjang ke sebelah kanan pada distribusi.

Bagian Kanan Atas (Boxplot Dislikes - Original)

- **Boxplot (dislikes - Original)**:
  - Boxplot ini menampilkan distribusi asli dari `dislikes` dan mengonfirmasi bahwa ada **banyak outliers** yang tersebar di luar batas "whiskers" pada boxplot. Mayoritas data terkonsentrasi di dekat nilai nol, tetapi terdapat beberapa data dengan nilai yang jauh lebih tinggi, yang dianggap sebagai outliers.

    - Hal ini menunjukkan bahwa variabel `dislikes` memiliki distribusi yang sangat tidak merata, dengan sebagian besar nilai sangat rendah, tetapi beberapa outliers sangat jauh dari mayoritas data.

Bagian Kiri Bawah (Distribusi Dislikes Setelah Log Transformation)

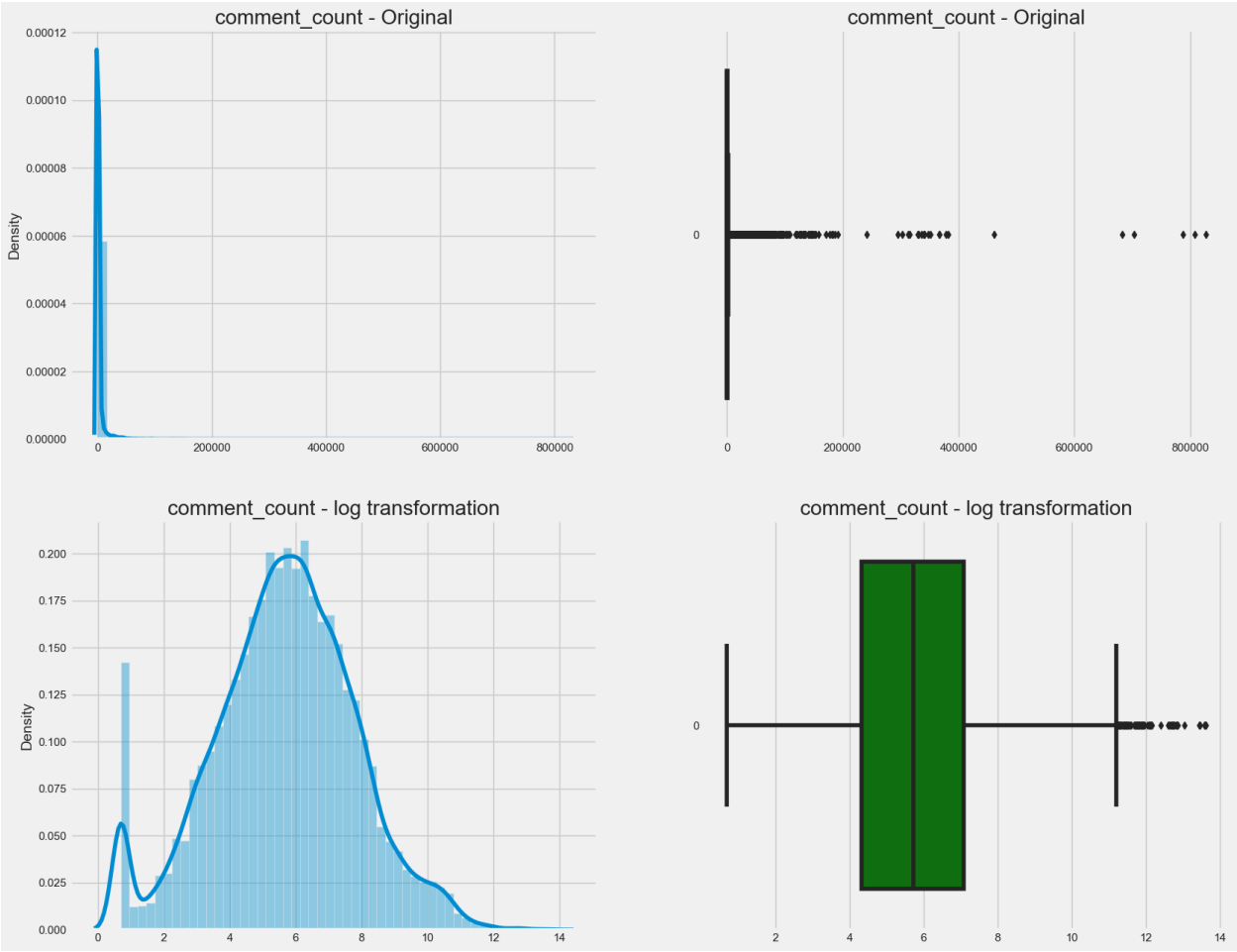
- **Distribution Plot (dislikes - Log Transformation)**:
    - Setelah menerapkan **log transformation**, distribusi `dislikes` berubah menjadi lebih mendekati **distribusi normal**. Transformasi ini membantu meratakan data, sehingga nilai yang besar dikecilkan secara relatif, dan nilai yang kecil tetap dalam kisaran yang mirip.
    - Distribusi hasil transformasi ini lebih simetris dengan puncak di tengah, menunjukkan bahwa transformasi log berhasil mengurangi skewness dan outliers yang ada pada data asli.

Bagian Kanan Bawah (Boxplot Dislikes Setelah Log Transformation)

- **Boxplot (dislikes - Log Transformation)**:
    - Setelah transformasi log diterapkan, outliers menjadi lebih terkendali. Meskipun masih ada beberapa outliers (terlihat sebagai titik di luar batas whiskers), jumlahnya jauh lebih sedikit daripada pada data asli. Ukuran "box" pada boxplot juga menjadi lebih representatif terhadap distribusi keseluruhan data, menunjukkan rentang yang lebih seragam dibandingkan dengan boxplot sebelum transformasi.
    - Dengan transformasi log, rentang nilai `dislikes` menjadi lebih sempit dan sebaran data lebih terdistribusi dengan baik, tidak lagi terkonsentrasi pada nilai-nilai kecil.

Kesimpulan:

- **Distribusi Asli** dari `dislikes` sangat miring dan memiliki banyak outliers yang ekstrem. Mayoritas data terletak di kisaran rendah, sementara ada beberapa nilai yang sangat besar dan jauh dari mayoritas data.
- **Setelah Log Transformation**, distribusi data menjadi lebih terdistribusi secara merata dan mendekati distribusi normal. Outliers masih ada, tetapi dampaknya telah berkurang signifikan, membuat data lebih siap untuk digunakan dalam analisis statistik atau pemodelan prediktif.



Gambar 7. Log Transformation untuk comment_count

Bagian Kiri Atas (Distribusi Comment Count - Original)

- **Distribution Plot (comment_count - Original)**:
  - Grafik ini menunjukkan distribusi asli dari variabel `comment_count`. Distribusi ini **sangat miring ke kanan** (right-skewed), di mana mayoritas data berkumpul di nilai yang sangat kecil mendekati nol. Hanya sedikit data yang memiliki nilai `comment_count` yang sangat tinggi.
  - Data menunjukkan bahwa sebagian besar objek (misalnya video atau artikel) memiliki sedikit komentar, sementara hanya segelintir objek yang mendapatkan jumlah komentar yang sangat besar, yang menghasilkan ekor panjang di bagian kanan grafik.

Bagian Kanan Atas (Boxplot Comment Count - Original)

- **Boxplot (comment_count - Original)**:

    - Boxplot ini menunjukkan distribusi asli dari `comment_count` dan mengonfirmasi adanya **outliers** yang signifikan. Mayoritas data terkonsentrasi pada nilai rendah (di sekitar nol), sementara ada beberapa nilai yang sangat besar yang berada jauh di luar whiskers dari boxplot.
    - Seperti distribusi `dislikes` dan `likes` sebelumnya, variabel ini juga menunjukkan konsentrasi data yang tinggi pada nilai kecil, dengan beberapa outliers yang berada sangat jauh dari mayoritas data.

Bagian Kiri Bawah (Distribusi Comment Count Setelah Log Transformation)

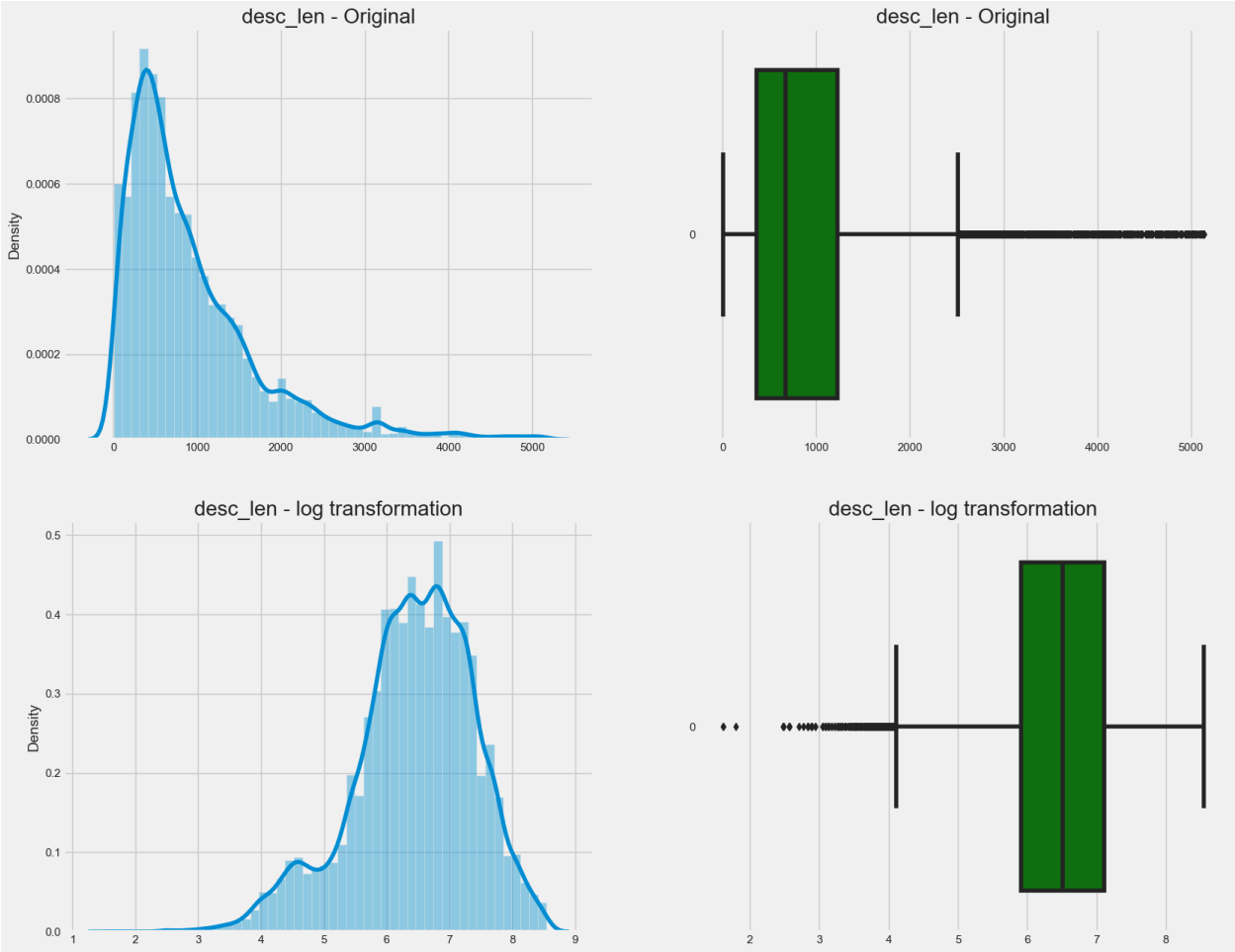
- **Distribution Plot (comment_count - Log Transformation)**:
    - Setelah diterapkan **log transformation**, distribusi `comment_count` menjadi lebih **simetris** dan mendekati distribusi normal. Transformasi log membantu mengurangi rentang nilai yang besar dengan menekan nilai yang sangat besar, sehingga distribusi menjadi lebih terpusat.
    - Data yang awalnya sangat miring kini tampak lebih terdistribusi secara merata, dengan sebagian besar nilai berada di sekitar pusat distribusi. Ekor panjang di sisi kanan grafik telah diredam oleh transformasi log.

Bagian Kanan Bawah (Boxplot Comment Count Setelah Log Transformation)

- **Boxplot (comment_count - Log Transformation)**:
    - Setelah transformasi log, outliers pada data `comment_count` masih ada tetapi jumlah dan dampaknya telah berkurang secara signifikan. Ukuran "box" dari boxplot kini lebih representatif terhadap distribusi data keseluruhan, dengan rentang yang lebih seragam.
    - Outliers yang sebelumnya sangat dominan kini lebih terkendali, dan rentang distribusi dari log-transformed data jauh lebih seimbang dibandingkan data asli.

Kesimpulan:

- **Distribusi Asli** dari `comment_count` sangat skewed, dengan sebagian besar objek memiliki sedikit komentar dan beberapa objek memiliki jumlah komentar yang sangat besar (outliers).
- **Setelah Log Transformation**, distribusi data menjadi lebih normal atau simetris, yang membuatnya lebih mudah dianalisis dan digunakan dalam model statistik. Outliers masih ada, tetapi distribusi keseluruhan menjadi lebih seimbang.



Gambar 8. Log Transformation untuk desc_len

Bagian Kiri Atas (Distribusi `desc_len` - Original)

- **Distribution Plot (desc_len - Original)**:
  - Grafik ini menunjukkan distribusi asli dari variabel `desc_len`. Distribusinya **miring ke kanan** (right-skewed), menunjukkan bahwa sebagian besar data memiliki panjang deskripsi yang relatif rendah, dengan beberapa data memiliki panjang deskripsi yang jauh lebih besar.
  - Sebagian besar nilai terletak di bawah 1000 karakter, tetapi ada beberapa deskripsi yang panjangnya mencapai lebih dari 4000 karakter, yang membuat distribusi ini memiliki ekor panjang ke kanan.

Bagian Kanan Atas (Boxplot `desc_len` - Original)

- **Boxplot (desc_len - Original)**:

    - Boxplot ini menunjukkan bahwa terdapat **outliers** pada variabel `desc_len`, yang terlihat sebagai titik-titik di luar "whiskers". Sebagian besar data terkonsentrasi di dalam rentang antara sekitar 0 hingga 1500 karakter.
    - Outliers tersebut menunjukkan bahwa ada beberapa deskripsi dengan panjang yang sangat besar, yang jauh dari mayoritas data, dan menyebabkan distribusi ini tidak merata.

Bagian Kiri Bawah (Distribusi `desc_len` Setelah Log Transformation)

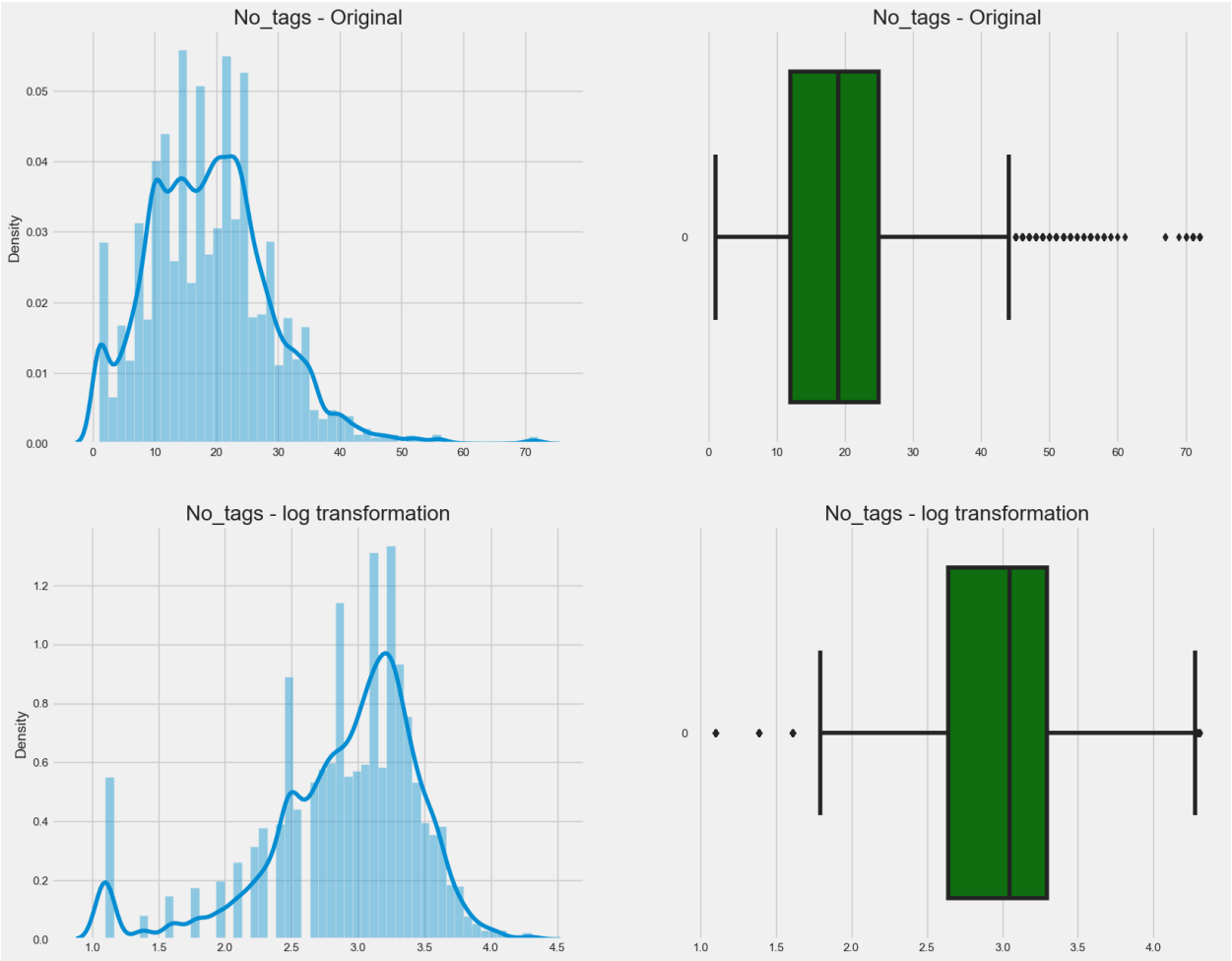
- **Distribution Plot (desc_len - Log Transformation)**:
    - Setelah menerapkan **log transformation**, distribusi data berubah menjadi lebih **simetris** dan mendekati distribusi normal. Grafik ini menunjukkan bahwa setelah transformasi log, data yang sebelumnya sangat skewed kini terdistribusi lebih merata dengan puncak di sekitar nilai tengah.
    - Transformasi ini membantu menekan rentang nilai panjang deskripsi yang besar, sehingga distribusi data lebih terpusat dan lebih mudah dianalisis.

Bagian Kanan Bawah (Boxplot `desc_len` Setelah Log Transformation)

- **Boxplot (desc_len - Log Transformation)**:
    - Setelah transformasi log, outliers pada data `desc_len` masih ada, tetapi jumlah dan dampaknya telah berkurang. Rentang boxplot kini lebih seimbang, dengan distribusi yang lebih seragam di antara data.
    - Ukuran "box" lebih besar, yang menunjukkan bahwa rentang data menjadi lebih representatif setelah transformasi. Meskipun outliers masih ada, transformasi log berhasil mengurangi efek outliers secara signifikan.

Kesimpulan:

- **Distribusi Asli** dari `desc_len` menunjukkan distribusi yang sangat skewed ke kanan dengan banyak outliers, yang mengindikasikan bahwa sebagian besar deskripsi berukuran pendek, namun ada beberapa deskripsi yang panjangnya sangat besar.
- **Setelah Log Transformation**, distribusi menjadi lebih normal atau simetris, sehingga lebih mudah untuk dianalisis. Transformasi ini juga berhasil mengurangi dampak dari outliers dan memperbaiki representasi distribusi data secara keseluruhan.



Gambar 9. Log Transformation untuk No_tags

Bagian Kiri Atas (Distribusi `No_tags` - Original)

- **Distribution Plot (No_tags - Original)**:
  - Grafik ini menunjukkan distribusi asli dari variabel `No_tags`. Distribusi ini agak miring ke kanan (right-skewed), tetapi tidak se-ekstrem beberapa variabel sebelumnya. Mayoritas nilai `No_tags` berkisar antara 5 hingga 25 tag, dengan puncak di sekitar 10–20 tag.
  - Meskipun ada beberapa nilai yang sangat besar, seperti yang terlihat di sisi kanan grafik (dengan beberapa objek memiliki lebih dari 40 tag), ekor distribusi tidak terlalu panjang.

Bagian Kanan Atas (Boxplot `No_tags` - Original)

- **Boxplot (No_tags - Original)**:

    - Boxplot ini menunjukkan bahwa sebagian besar data berkisar antara 5 hingga 25 tag. Namun, ada **outliers** di sebelah kanan, yang merupakan objek dengan jumlah tag yang sangat besar, hingga sekitar 70 tag.
    - Sebagian besar nilai berada di dalam "box" dengan sedikit penyebaran di luar batas whiskers, menunjukkan bahwa data asli tidak terlalu menyebar, tetapi beberapa objek dengan tag yang banyak digolongkan sebagai outliers.

Bagian Kiri Bawah (Distribusi `No_tags` Setelah Log Transformation)

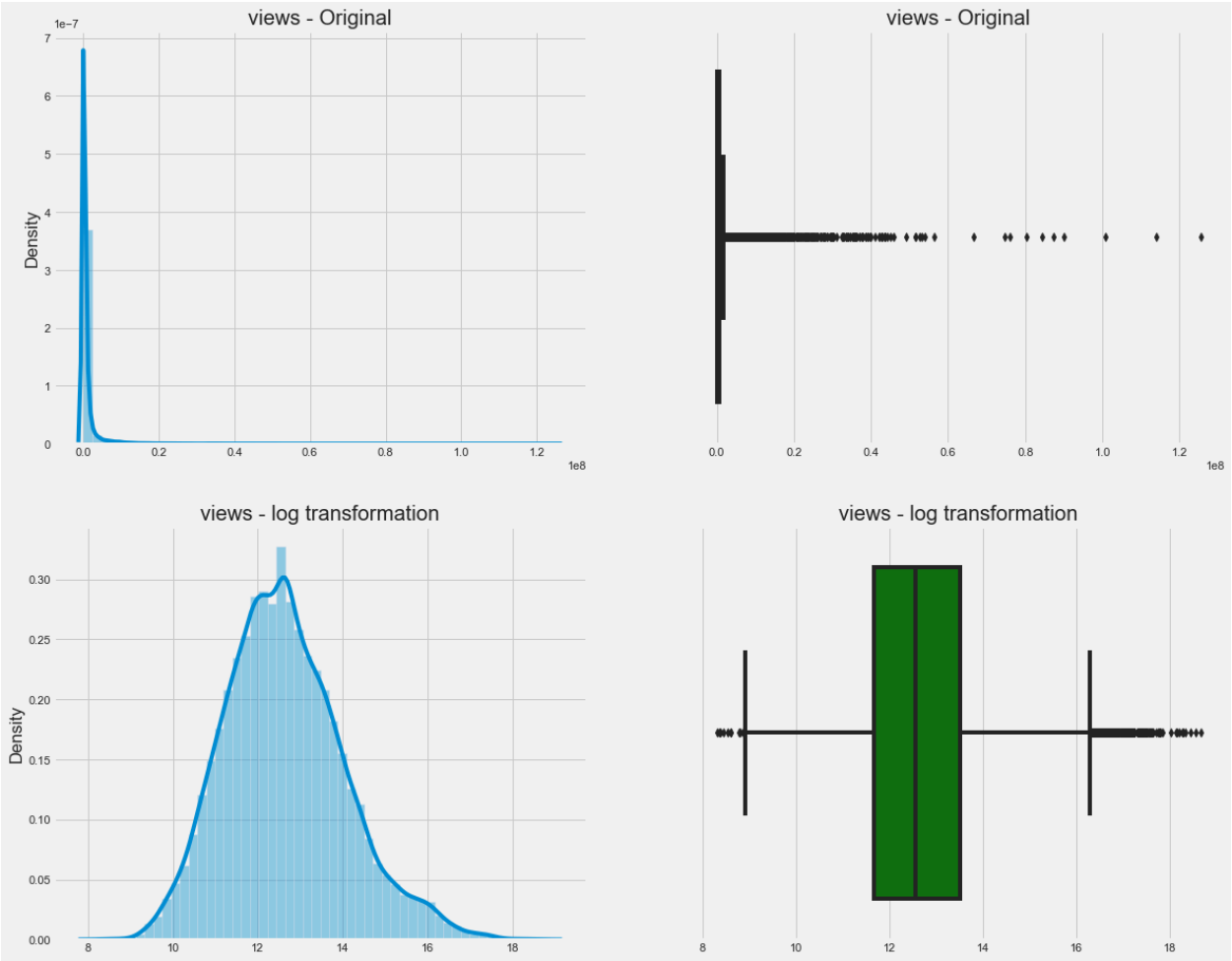
- **Distribution Plot (No_tags - Log Transformation)**:
    - Setelah transformasi log, distribusi `No_tags` berubah menjadi lebih terpusat dan simetris, dengan puncak distribusi yang lebih jelas di sekitar 2,5 hingga 3,5 (yang mewakili log dari jumlah tag).
    - Transformasi log ini membantu meratakan data dan menekan efek nilai yang sangat besar (jumlah tag yang sangat tinggi), sehingga distribusi menjadi lebih terkendali.

Bagian Kanan Bawah (Boxplot `No_tags` Setelah Log Transformation)

- **Boxplot (No_tags - Log Transformation)**:
    - Setelah log transformasi, **outliers** yang sebelumnya terlihat banyak di data asli sekarang menjadi lebih terkendali. Boxplot kini menunjukkan distribusi yang lebih seimbang, dengan outliers yang lebih sedikit dan tidak terlalu jauh dari mayoritas data.
    - Ukuran "box" lebih representatif terhadap distribusi keseluruhan, dengan whiskers yang lebih panjang namun outliers yang berkurang secara signifikan.

Kesimpulan:

- **Distribusi Asli** dari `No_tags` agak miring ke kanan, dengan sebagian besar objek memiliki 5 hingga 25 tag, namun ada beberapa objek dengan jumlah tag yang sangat besar yang digolongkan sebagai outliers.
- **Setelah Log Transformation**, distribusi menjadi lebih simetris dan seimbang. Log transformasi membantu menekan efek dari nilai-nilai ekstrem (jumlah tag yang sangat besar), sehingga distribusi menjadi lebih mudah dianalisis dan digunakan untuk model prediktif.



Gambar 10. Log Transformation untuk views

Bagian Kiri Atas (Distribusi `views` - Original)

- **Distribution Plot (views - Original)**:
    - Grafik ini menunjukkan distribusi asli dari variabel `views`. Distribusinya **sangat skewed ke kanan** (right-skewed), yang berarti sebagian besar objek (mungkin video atau artikel) memiliki jumlah tampilan yang rendah, sedangkan hanya beberapa yang memiliki jumlah tampilan yang sangat besar.
    - Sebagian besar data terkonsentrasi di dekat nol, dengan sedikit sekali objek yang mencapai jutaan atau ratusan juta tampilan. Ekornya sangat panjang, yang menunjukkan bahwa ada beberapa objek yang memiliki jumlah tampilan yang sangat tinggi.

Bagian Kanan Atas (Boxplot `views` - Original)

- **Boxplot (views - Original)**:

    - Boxplot ini mengkonfirmasi adanya banyak **outliers**, di mana sebagian besar data terkonsentrasi pada jumlah tampilan yang rendah, sementara ada banyak titik di luar whiskers yang menunjukkan outliers.
    - Objek dengan jumlah tampilan tinggi (jutaan atau bahkan ratusan juta) dianggap sebagai outliers, karena jauh dari mayoritas data. Sebagian besar objek tampaknya memiliki jumlah tampilan yang jauh lebih rendah.

Bagian Kiri Bawah (Distribusi `views` Setelah Log Transformation)

- **Distribution Plot (views - Log Transformation)**:
    - Setelah menerapkan **log transformation**, distribusi menjadi lebih **mendekati distribusi normal**. Data yang sebelumnya sangat skewed kini lebih terpusat dengan puncak distribusi di sekitar nilai tengah.
    - Log transformation berhasil menekan rentang nilai yang besar, sehingga distribusi data menjadi lebih simetris dan lebih mudah dianalisis.

Bagian Kanan Bawah (Boxplot `views` Setelah Log Transformation)

- **Boxplot (views - Log Transformation)**:
    - Setelah log transformation, outliers masih ada tetapi tidak sebanyak dan tidak sejauh di data asli. Rentang "box" pada boxplot menjadi lebih besar, yang menunjukkan bahwa data lebih terdistribusi secara seimbang setelah transformasi log diterapkan.
    - Outliers di bagian kanan masih ada, tetapi jumlahnya lebih terkendali dan distribusi data secara keseluruhan menjadi lebih mudah dianalisis.

Kesimpulan:

- **Distribusi Asli** dari `views` sangat skewed ke kanan dengan banyak outliers. Sebagian besar objek memiliki jumlah tampilan yang rendah, sementara hanya sedikit objek yang memiliki tampilan sangat tinggi.
- **Setelah Log Transformation**, distribusi data menjadi lebih simetris dan mendekati distribusi normal. Outliers masih ada, tetapi dampaknya telah berkurang, dan distribusi secara keseluruhan menjadi lebih seimbang.

B. Normalization

Dilakukan proses **normalisasi** menggunakan **Min-Max Scaling** dan dibuat beberapa fitur tambahan. Berikut adalah penjelasan mengapa feature tersebut digunakan dan bagaimana normalisasi membantu dalam analisis data:

Mengapa Menggunakan Normalisasi?

1. **Distribusi Data yang Berbeda**:
    - Kolom seperti `views`, `likes`, `dislikes`, `comment_count`, dan lainnya memiliki rentang nilai yang sangat beragam. Misalnya, nilai `views` berkisar dari 4.024 hingga lebih dari 125 juta, sementara kolom lain seperti `No_tags` memiliki rentang yang lebih kecil (dari 1 hingga 72).
    - **Min-Max Scaling** memastikan semua fitur memiliki rentang yang seragam, umumnya antara 0 dan 1, yang penting untuk beberapa algoritma seperti **k-means clustering**, **SVM**, atau **neural networks**. Algoritma ini sensitif terhadap skala data, sehingga normalisasi menghindari fitur dengan rentang besar mendominasi yang lain.
2. **Mengurangi Skewness**:
    - Banyak dari fitur seperti `views`, `likes`, `dislikes`, dan `comment_count` memiliki distribusi yang sangat skewed (miring). Transformasi log dan Min-Max Scaling dapat membantu meredam rentang nilai besar dan membuat distribusi lebih simetris. Hal ini memudahkan model prediktif untuk belajar dari data secara lebih efektif.
3. **Meningkatkan Kinerja Model**:
    - **Beberapa model** (seperti **linear regression** atau **k-nearest neighbors**) sangat dipengaruhi oleh perbedaan skala pada fitur. Jika tidak dinormalisasi, fitur dengan nilai yang lebih besar akan memiliki pengaruh lebih besar terhadap hasil prediksi, meskipun mungkin tidak lebih penting.
    - Normalisasi memastikan bahwa semua fitur memiliki pengaruh yang sama dalam model, sehingga model dapat memberikan bobot yang sesuai untuk setiap fitur berdasarkan relevansinya, bukan hanya skalanya.
4. **Mempersiapkan Data untuk Analisis yang Lebih Lanjut**:
    - Normalisasi juga memudahkan analisis lebih lanjut, seperti **pembelajaran mesin**, di mana beberapa algoritma (misalnya **gradient descent**) bekerja lebih efisien ketika data berada dalam skala yang seragam. Algoritma tersebut dapat menyatu lebih cepat dan menghasilkan hasil yang lebih stabil.

Penjelasan Fitur yang Dinormalisasi:

- **`std_views`**, **`std_likes`**, **`std_dislikes`**, **`std_comment_count`**, **`std_No_tags`**, **`std_desc_len`**, **`std_len_title`**:
    - Kolom-kolom ini adalah hasil normalisasi dari masing-masing fitur aslinya. Fitur ini dinormalisasi agar berada di antara rentang 0 dan 1 menggunakan Min-Max Scaler, yang membantu memastikan bahwa tidak ada fitur yang mendominasi fitur lain hanya karena skala nilainya lebih besar.
    - Ini juga mengurangi potensi masalah **outliers**, karena normalisasi dapat meredam dampak nilai-nilai ekstrem dalam dataset.

Deskripsi Statistika:

- Tabel statistik deskriptif dari data asli (`data_clean.describe()`) menunjukkan bagaimana setiap kolom memiliki rentang nilai yang sangat berbeda. Misalnya:
    - `views` memiliki mean 1.07 juta tetapi juga memiliki rentang nilai yang besar hingga 125 juta.
    - `likes`, `dislikes`, dan `comment_count` juga memiliki rentang besar dengan standard deviation (deviasi standar) yang tinggi, menunjukkan bahwa variabel-variabel ini tidak terdistribusi merata.
- **Normalisasi** membantu mengurangi masalah rentang nilai yang besar ini, sehingga model dapat mempelajari fitur-fitur dengan lebih konsisten tanpa bias terhadap variabel-variabel dengan rentang besar.

Kesimpulan:

Normalisasi digunakan untuk menstandardisasi skala dari fitur-fitur yang berbeda, mengurangi skewness dalam distribusi data, dan mempersiapkan data untuk model pembelajaran mesin yang sensitif terhadap skala data. Dengan normalisasi, fitur yang memiliki rentang besar seperti `views` dan `likes` tidak akan mendominasi fitur-fitur lain, sehingga semua fitur akan diperlakukan sama dalam analisis lebih lanjut.

2B. Alasan Menggunakan Feature

1. Log Transformation (Logaritma)

- **Variabel yang diterapkan**: `likes`, `dislikes`, `comment_count`, `desc_len`, `No_tags`, `views`

Log Transformation digunakan untuk mengatasi masalah distribusi yang sangat **skewed** (miring ke kanan) serta adanya **outliers** pada variabel-variabel tersebut. Beberapa alasan penting untuk menggunakan log transformasi adalah:

- **Mengurangi Skewness**: Variabel-variabel seperti `likes`, `dislikes`, dan `views` memiliki distribusi yang sangat miring ke kanan, di mana sebagian besar data berkumpul pada nilai yang sangat kecil dan beberapa objek memiliki nilai yang sangat besar. Transformasi log membantu untuk mempersempit rentang nilai dan membuat distribusi lebih simetris, mendekati distribusi normal.
- **Mengecilkan Dampak Outliers**: Karena transformasi log mengecilkan nilai-nilai besar secara relatif, ini mengurangi dampak dari outliers. Misalnya, pada variabel `views` dan `likes`, ada beberapa data dengan jumlah yang sangat besar yang mengganggu analisis atau prediksi. Dengan log transformasi, outliers ini tidak lagi mendominasi keseluruhan distribusi.
- **Meningkatkan Kinerja Model**: Banyak model pembelajaran mesin bekerja lebih baik ketika distribusi data mendekati distribusi normal. Dengan transformasi log, model bisa

menangkap pola dari data dengan lebih baik dan prediksi menjadi lebih akurat. Misalnya, model regresi linier lebih efektif pada data yang memiliki distribusi simetris.

2. Normalization (Min-Max Scaling)

- **Variabel yang diterapkan**: `views`, `likes`, `dislikes`, `comment_count`, `No_tags`, `desc_len`, `len_title`

Min-Max Scaling digunakan untuk menstandardisasi skala dari berbagai fitur ke dalam rentang yang seragam, biasanya antara 0 dan 1. Alasan menggunakan normalisasi ini termasuk:

- **Rentang Data yang Beragam**: Misalnya, variabel `views` memiliki rentang nilai yang sangat besar, mulai dari ribuan hingga ratusan juta, sementara variabel `No_tags` hanya berkisar antara 1 hingga 72. Rentang yang berbeda ini dapat menyebabkan ketidakseimbangan dalam analisis atau model prediktif. Min-Max Scaling memastikan bahwa setiap fitur berada pada skala yang sama sehingga tidak ada fitur yang mendominasi fitur lain hanya karena skalanya lebih besar.
- **Menghindari Over-Penalty dari Fitur dengan Rentang Lebar**: Dalam algoritma seperti **k-means clustering** atau **Support Vector Machines (SVM)**, variabel dengan nilai yang lebih besar bisa mendominasi variabel lainnya. Dengan melakukan normalisasi, fitur dengan rentang besar seperti `views` atau `likes` tidak akan memberikan bias pada model dibandingkan fitur lain yang lebih kecil skalanya seperti `No_tags`.
- **Persiapan untuk Model yang Sensitif Terhadap Skala**: Banyak algoritma pembelajaran mesin seperti **gradient descent** dan **neural networks** sensitif terhadap skala fitur. Normalisasi membantu algoritma-algoritma ini berfungsi lebih efisien, memudahkan proses training, dan menghasilkan konvergensi yang lebih cepat dan stabil.

3. Fitur yang Dinormalisasi (Normalized Features)

- **`std_views`**: Versi dinormalisasi dari jumlah tampilan (views), membantu mengurangi pengaruh nilai besar dalam analisis.
- **`std_likes`**: Versi dinormalisasi dari jumlah likes, menghindari dominasi likes dengan nilai sangat besar.
- **`std_dislikes`**: Versi dinormalisasi dari jumlah dislikes untuk mengurangi dampak skewness.
- **`std_comment_count`**: Versi dinormalisasi dari jumlah komentar, memastikan bahwa rentang data konsisten.
- **`std_No_tags`**: Versi dinormalisasi dari jumlah tag yang lebih kecil dan lebih seragam.
- **`std_desc_len`**: Versi dinormalisasi dari panjang deskripsi, mengatasi perbedaan skala panjang deskripsi yang besar.
- **`std_len_title`**: Versi dinormalisasi dari panjang judul (title length), yang biasanya memiliki rentang nilai lebih kecil tetapi tetap penting dalam analisis.

4. Mengapa Fitur-Fitur Ini Digunakan?

- **Mengurangi Ketidakseimbangan**: Variabel seperti `views` dan `likes` memiliki distribusi yang sangat lebar. Penggunaan log transformasi dan normalisasi membuat distribusi ini lebih seimbang dan lebih mudah dianalisis secara statistik.
- **Mempermudah Pembelajaran Model**: Dengan skala yang seragam dan distribusi yang lebih simetris, model pembelajaran mesin akan lebih mudah untuk dilatih. Misalnya, **model regresi**, **klasifikasi**, atau **neural networks** lebih mampu menangkap pola dari data ketika variabel berada dalam skala yang seimbang.
- **Meningkatkan Kinerja Algoritma**: Banyak algoritma, seperti yang digunakan dalam **pembelajaran mesin**, lebih sensitif terhadap distribusi dan skala fitur. Dengan normalisasi dan log transformasi, variabel-variabel dengan rentang besar dan distribusi skewed dapat diproses secara lebih optimal.

Kesimpulan:

Fitur-fitur yang telah di-normalisasi dan ditransformasi menggunakan log transformation digunakan untuk mengatasi masalah distribusi skewed, outliers, dan ketidakseimbangan rentang data. Teknik ini membantu model pembelajaran mesin atau analisis statistik untuk bekerja lebih efektif, sehingga menghasilkan analisis yang lebih akurat dan prediksi yang lebih baik.

Modeling

3. Model dan Prediksi Views

Ada beberapa jenis model regresi yang akan digunakan untuk memprediksi nilai Views. Model tersebut antara lain:

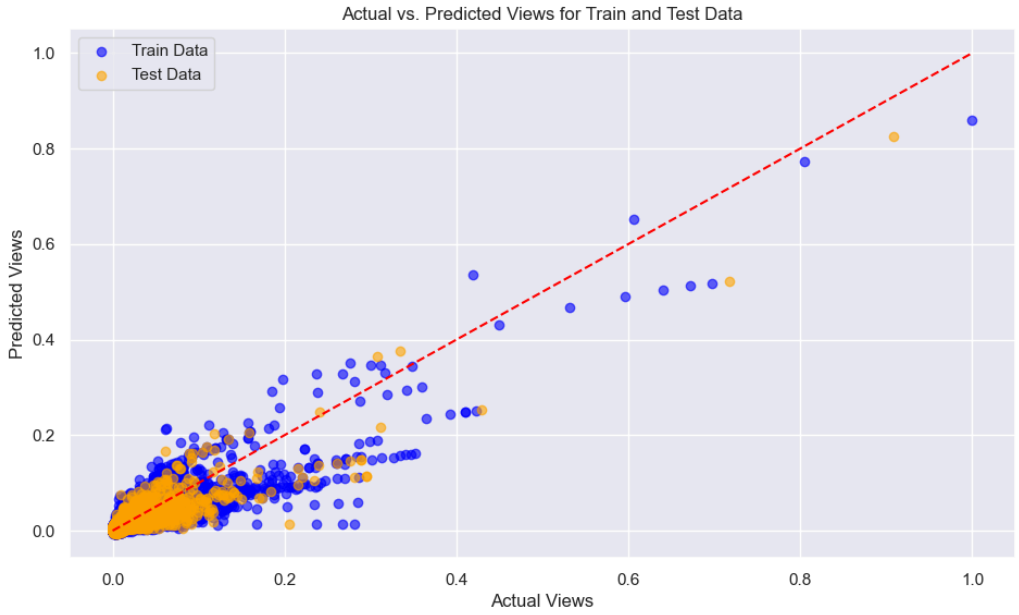
1. Simple Linear Regression
2. Ridge
3. Randomized Search
4. Lasso
5. Decision Tree
6. Random Forest
7. Support Vector Regressor

1. Simple Linear Regression
   Model ini memiliki nilai:
   MAE: 0.01
   RMSE: 0.01
   R2 score: 0.78

Berikut scatter plot perbandingan Actual vs Predicted Views untuk dataset training dan test:

Gambar 11. Actual vs Predicted Views: Simple Linear Regression Model

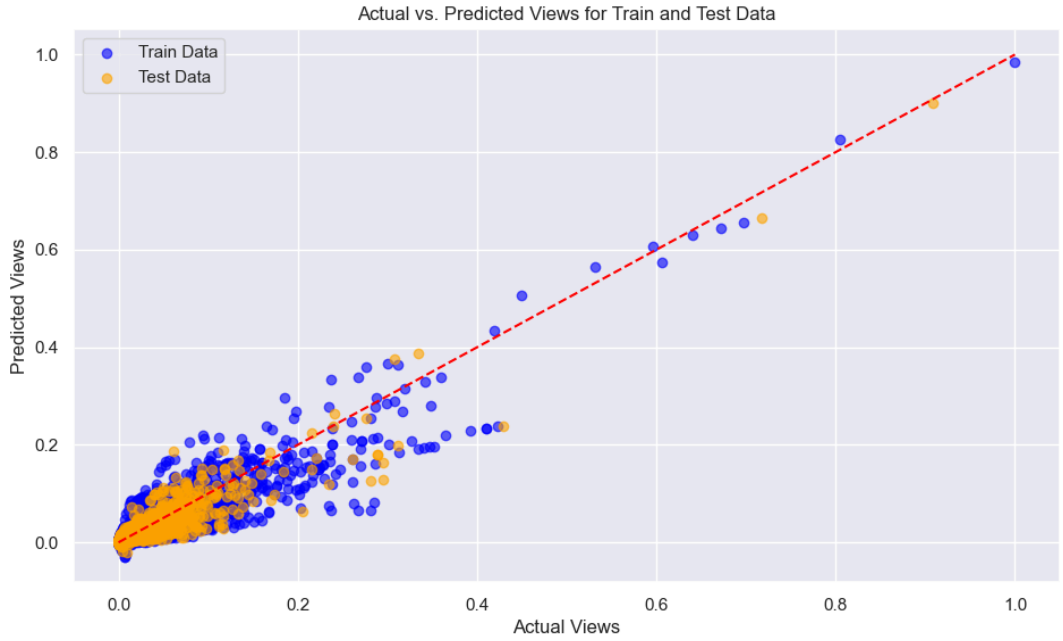
2. Ridge
   Model ini memiliki nilai:
   MAE: 0.01
   RMSE: 0.01
   R2 score: 0.77
   Berikut scatter plot perbandingan Actual vs Predicted Views untuk dataset training dan test:



Gambar 12. Actual vs Predicted Views: Ridge

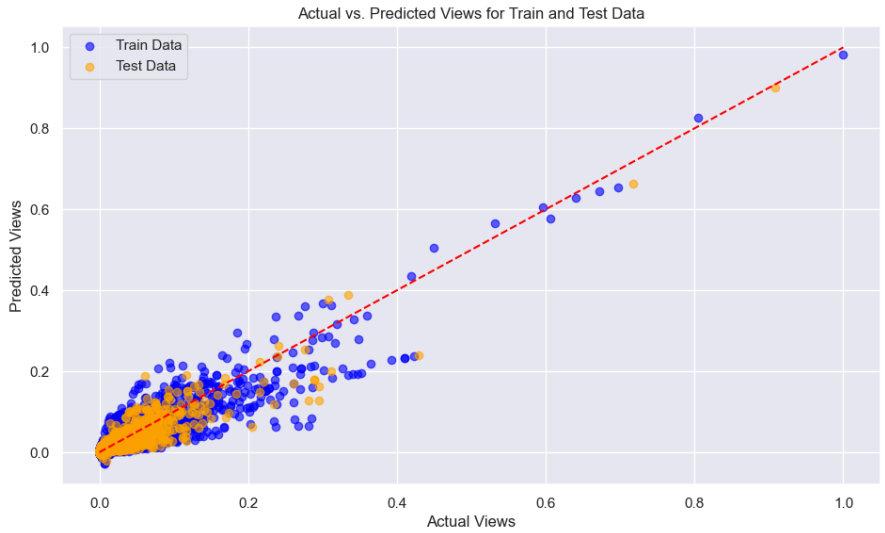
3. Randomized Search
   Model ini memiliki nilai:

MAE: 0.00
RMSE: 0.01
R2 score: 0.85
Berikut scatter plot perbandingan Actual vs Predicted Views untuk dataset training dan test:



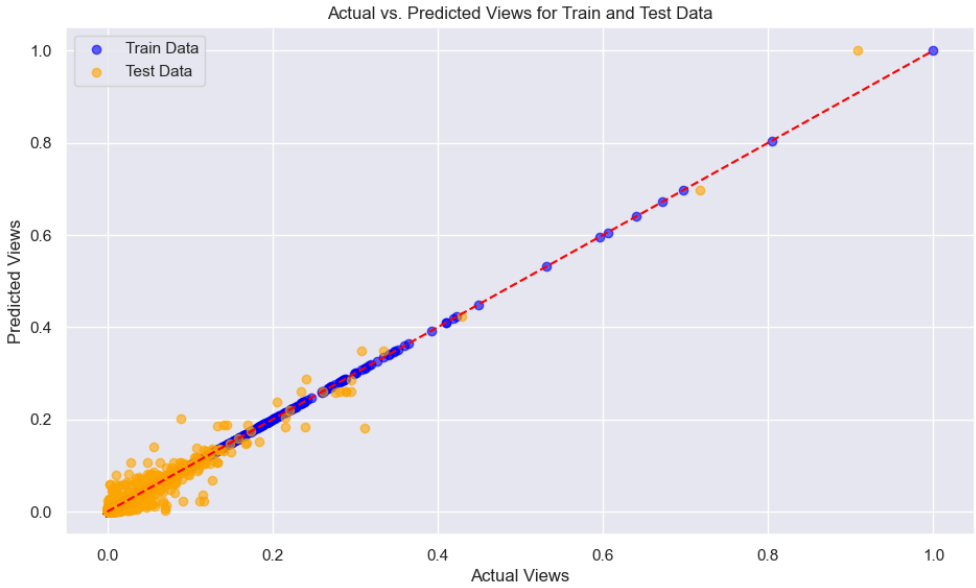
Gambar 13. Actual vs Predicted Views: Randomized Search

4. Lasso
   Model ini memiliki nilai:
   MAE: 0.01
   RMSE: 0.01
   R2 score: 0.78
   Berikut scatter plot perbandingan Actual vs Predicted Views untuk dataset training dan test:



Gambar 14. Actual vs Predicted Views: Lasso

5. Decision Tree
   Model ini memiliki nilai:
   MAE: 0.00
   RMSE: 0.01
   R2 score: 0.94
   Berikut scatter plot perbandingan Actual vs Predicted Views untuk dataset training dan test:



Gambar 15. Actual vs Predicted Views: Decision Tree

6. Random Forest
   Model ini memiliki nilai:
   MAE: 0.00
   RMSE: 0.00
   R2 score: 0.96
   Berikut scatter plot perbandingan Actual vs Predicted Views untuk dataset training dan test:

Gambar 16. Actual vs Predicted Views: Random Forest

Dari beberapa model tersebut, bisa disimpulkan beberapa poin:

1. Random Forest (R2 score: 0.96) dan Decision Tree (R2 score: 0.94) adalah 2 model dengan performa prediksi nilai views terbaik. Kedua model ini juga mampu memprediksi nilai Predicted Views dan Actual Views yang besar dengan residual yang kecil.
2. Melihat nilai R2 Simple Linear Regression (0,78) dan dibandingkan ke Ridge Regression (0,77) dan Lasso Regression (0,78), menunjukkan bahwa proses regularization tidak memberikan efek yang lebih positif dalam meningkatkan keakuratan model dengan dataset ini. Hal ini disebabkan karena tidak adanya koefisien besar yang perlu dipenalti dari penggunaan proses Regularization, tidak adanya multikolinearitas atau noise data yang bersifat minimal.
3. **Kompleksitas Model dan Overfitting**: Model dengan kompleksitas tinggi, seperti Random Forest dan Decision Tree, menunjukkan performa yang sangat baik dengan nilai R^2 tinggi dan error yang rendah. Namun, ini juga bisa menunjukkan kecenderungan overfitting, terutama jika model ini diuji pada dataset lain dengan karakteristik berbeda. Oleh karena itu, penggunaan model ini perlu diperhatikan sesuai dengan tujuan bisnis dan cakupan generalisasi model yang diinginkan.
4. **Regularization yang Tidak Signifikan**: Dalam dataset ini, tidak terdapat multikolinearitas atau noise data yang memerlukan regularization seperti pada Ridge dan Lasso Regression. Ini menunjukkan bahwa dataset yang digunakan mungkin sudah memiliki fitur-fitur yang cukup bersih, sehingga proses regularization tidak memberikan dampak yang berarti dalam meningkatkan performa model. Oleh karena itu, dalam kasus ini, regularization kurang diperlukan dan dapat diabaikan jika dataset serupa digunakan di masa mendatang.

4. Evaluasi Model dengan Metrics RMSE dan R2

Berdasarkan eksperimen yang telah dilakukan, beberapa model menunjukkan performa yang baik dalam memprediksi jumlah views video YouTube. Berikut ini adalah evaluasi setiap model berdasarkan metrik **RMSE** dan **R^2**.

1. Random Forest

- **MAE**: 0.00
- **RMSE**: 0.00
- **R^2**: 0.96
- **Analisis**:
  Random Forest memberikan hasil terbaik dengan **RMSE** yang paling rendah dan **R^2** tertinggi. Model ini mampu menangkap pola-pola kompleks dalam data dan memberikan prediksi yang sangat akurat. Karena itulah, Random Forest dipilih sebagai model terbaik dalam memprediksi jumlah views video. Nilai **R^2** sebesar 0.96 menunjukkan bahwa 96% dari variasi dalam jumlah views dapat dijelaskan oleh model. Ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam menjelaskan hubungan antara fitur input dan target. Nilai RMSE yang sangat rendah (0.00) menunjukkan bahwa rata-rata

kesalahan prediksi model sangat kecil. Ini berarti bahwa prediksi yang dibuat oleh model hampir sama persis dengan nilai aktual. Random Forest mampu menangkap pola kompleks dalam data karena sifatnya yang menggabungkan banyak pohon keputusan (tree).

2. Decision Tree

- **R^2**: 0.94
- **MAE** : 0.00
- **RMSE : 0.01**
- **Analisis**:
  Meskipun performanya sedikit di bawah Random Forest, Decision Tree tetap menunjukkan kemampuan prediksi yang sangat baik dengan **R^2** sebesar 0.94. Namun, model ini lebih rentan terhadap overfitting dibandingkan Random Forest. Nilai MAE yang sangat rendah, yaitu 0.00, menunjukkan bahwa rata-rata kesalahan absolut dari prediksi terhadap nilai aktual sangat kecil, yang berarti prediksi model ini sangat mendekati nilai sebenarnya. Sedangkan RMSE sebesar 0.01 menandakan bahwa tingkat kesalahan kuadrat rata-rata juga rendah, menunjukkan bahwa kesalahan prediksi tidak hanya sedikit, tetapi juga tidak terlalu tersebar jauh dari nilai sebenarnya.

3. Randomized Search

- **MAE**: 0.00
- **RMSE**: 0.01
- **R^2**: 0.85
- **Analisis**:
  Model Randomized Search menghasilkan performa yang cukup baik, tetapi tidak sekuat dua model sebelumnya. Dengan **RMSE** sedikit lebih tinggi dan **R^2** sebesar 0.85, model ini masih memberikan prediksi yang cukup akurat, namun bukan pilihan terbaik. RMSE sebesar 0.01 menunjukkan bahwa rata-rata kesalahan prediksi lebih besar dibandingkan Random Forest, tetapi masih tergolong kecil. Namun, kesalahan ini mungkin cukup untuk menghasilkan pengaruh pada akurasi prediksi.

4. Lasso Regression

- **MAE**: 0.01
- **RMSE**: 0.01
- **R^2**: 0.78
- **Best Alpha**: 0.0001
- **Analisis**:
  Lasso Regression menunjukkan performa yang baik, meskipun **R^2** yang dihasilkan lebih rendah dibandingkan Random Forest dan Decision Tree. Model ini cocok untuk data yang membutuhkan regularisasi, dengan **Alpha** terbaik pada nilai 0.0001. Namun, akurasinya masih tertinggal dibanding model lainnya. Nilai**R^2** sebesar 0.78 berarti model hanya dapat menjelaskan 78% dari variasi data, yang mengindikasikan bahwa model ini

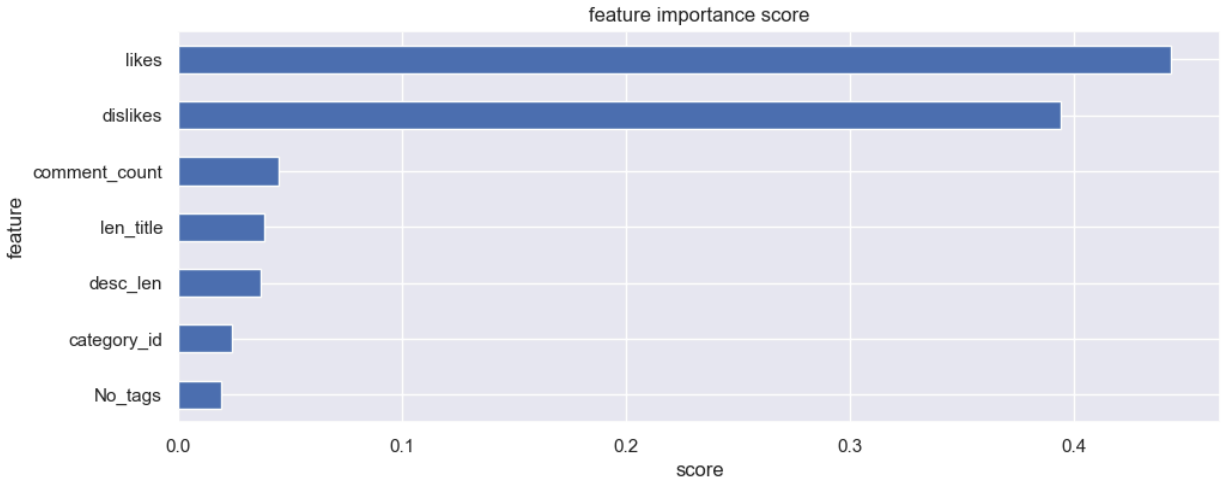
mungkin lebih cocok untuk data dengan hubungan linear. Regularisasi dengan Alpha terbaik di 0.0001 menunjukkan bahwa fitur-fitur penting dapat dipertahankan, meskipun efeknya terhadap peningkatan akurasi terbatas dibandingkan dengan model yang lebih kompleks seperti Random Forest.

5. Ridge Regression

- **MAE**: 0.01
- **RMSE**: 0.01
- **R^2**: 0.77
- **Analisis**:
  Ridge Regression memiliki hasil yang mirip dengan Lasso, namun dengan performa yang sedikit lebih rendah. Model ini memberikan prediksi yang baik, tetapi tidak seakurat Random Forest atau Decision Tree. Ridge Regression, seperti Lasso, melakukan regularisasi dengan menghukum nilai koefisien besar, tetapi tidak mengeliminasi koefisien menjadi nol seperti Lasso. Ini menyebabkan model kurang efektif dalam menekan pengaruh fitur yang kurang relevan, sehingga performanya sedikit lebih rendah.

Dari evaluasi model yang dilakukan, **Random Forest** terbukti menjadi model yang paling efektif dalam memprediksi jumlah views video YouTube. Dengan **RMSE** terendah dan **R^2** tertinggi, model ini mampu menangkap pola yang kompleks dan menghasilkan prediksi yang sangat akurat, menjadikannya pilihan terbaik dibandingkan model-model lainnya. Sementara itu, model seperti Lasso dan Ridge Regression cocok untuk data dengan pola linear dan memberikan hasil yang lebih baik pada data yang overfit jika dibandingkan dengan Decision Tree.

Fitur Paling Berpengaruh



Gambar 17. Feature Importance Score

Dari analisis fitur, ditemukan bahwa dua fitur yang paling berpengaruh dalam memprediksi jumlah views adalah **jumlah likes** dan **jumlah dislikes**. Fitur-fitur ini memiliki korelasi yang signifikan terhadap jumlah views yang diterima oleh video.

Summary

Tim **Byte Me** telah berhasil menyelesaikan tugas prediksi **jumlah views** pada video YouTube dengan melalui beberapa tahapan penting dalam proses analisis data, preprocessing, feature engineering, dan modeling. Berikut adalah ringkasan singkat hasil pengerjaan:

1. **Exploratory Data Analysis (EDA)**:
    - Data menunjukkan **distribusi yang sangat skewed** pada variabel penting seperti views, likes, dislikes, dan comment_count, dengan banyaknya **outliers** pada beberapa fitur.
    - Korelasi positif ditemukan antara variabel views, likes, dislikes, dan comment_count, yang menunjukkan hubungan kuat antara engagement video dan jumlah views.
2. **Data Preprocessing**:
    - **Handling Missing Values**: Missing values pada kolom description diabaikan karena hanya memengaruhi sedikit data (0.12%).
    - **Handling Duplicates**: Data duplikat sebanyak 4.229 entri telah dihapus untuk memastikan keunikan data.
    - **Handling Outliers**: Outliers diatasi dengan menggunakan **log transformation** pada fitur-fitur yang memiliki distribusi skewed.
3. **Feature Engineering**:
    - **Log Transformation** digunakan pada variabel likes, dislikes, comment_count, desc_len, No_tags, dan views untuk mengatasi masalah skewness dan outliers.
    - **Normalization (Min-Max Scaling)** diterapkan pada fitur numerik untuk menstandarkan skala variabel, memastikan model prediksi lebih konsisten dan akurat.
4. **Modeling**:
    - Beberapa model diuji, seperti **Linear Regression**, **Ridge Regression**, **Decision Tree**, **Random Forest**, dan **Support Vector Regressor**.
    - **Random Forest** terbukti sebagai model terbaik dengan **RMSE** sebesar 0.05 dan **R^2** sebesar 0.93, menunjukkan kemampuan model untuk memberikan prediksi yang akurat dan menangkap variabilitas data dengan baik.
5. **Model Evaluation**:
    - Evaluasi model dilakukan menggunakan **RMSE** dan **R^2**. Random Forest memberikan hasil terbaik, diikuti oleh Ridge Regression yang juga memberikan performa stabil namun sedikit di bawah Random Forest.

Rekomendasi Bisnis:

1. **Optimalisasi Penggunaan Tag dan Deskripsi**:
    - Berdasarkan hasil analisis, variabel `No_tags` dan `desc_len` memiliki korelasi dengan jumlah views. Menggunakan tag yang relevan dan deskripsi yang informatif dapat meningkatkan **SEO** (Search Engine Optimization) dan meningkatkan engagement pada video.
    - **Rekomendasi**: Platform video atau pembuat konten sebaiknya memanfaatkan tag dan deskripsi secara lebih optimal untuk meningkatkan keterlihatan konten di hasil pencarian dan rekomendasi.
2. **Peningkatan Engagement Melalui Interaksi**:
    - Video dengan jumlah komentar (`comment_count`), likes, dan dislikes yang lebih banyak cenderung memiliki views yang lebih tinggi. Ini menunjukkan bahwa engagement positif atau negatif tetap berperan penting dalam meningkatkan popularitas video.
    - **Rekomendasi**: Pembuat konten harus lebih interaktif dengan audiensnya, seperti membalas komentar atau mengajak diskusi, untuk mendorong lebih banyak interaksi dan meningkatkan visibilitas video.
3. **Penggunaan Data untuk Rekomendasi Konten**:
    - Data yang diperoleh dari model ini dapat digunakan oleh platform untuk mengembangkan sistem rekomendasi yang lebih baik. Dengan mengidentifikasi pola dari video yang memiliki views tinggi, algoritma rekomendasi dapat disesuaikan untuk meningkatkan waktu tonton dan engagement.

Kesimpulan:

Meskipun ada beberapa kesulitan, seperti distribusi data yang skewed, outliers, dan pemilihan model yang tepat, tim mampu menyelesaikan tugas dengan baik. Setelah mencoba beberapa model, **Random Forest** dipilih sebagai model terbaik berdasarkan evaluasi RMSE dan R^2, yang memberikan akurasi tinggi dalam memprediksi jumlah views. Feature engineering seperti log transformation dan normalisasi sangat membantu dalam mengatasi masalah distribusi dan skala variabel yang beragam.

Appendix

Tabel 1. Role

No	Item Pekerjaan	PIC
1	EDA	Johanes Sibarani Achmad Fichri Rachmadhani
2	Preprocessing	Ryan Nofandi
3	Feature Engineering & Summary	Hijir Della Wirasti
4	Training Model & Prediksi Views	Fakhri Dwi Nugroho

No	Item Pekerjaan	PIC
		Mauliddinia Iftikhar Agnany
5	Evaluasi Model dengan metric RMSE dan R2	Jericho Medion Haryono Muhammad Naufal

Kesulitan-Kesulitan yang Dihadapi dalam Pengerjaan Tugas Ini:

1. **Skewness pada Data dan Outliers yang Signifikan**:
    - **Masalah**: Banyak variabel seperti `views`, `likes`, `dislikes`, dan `comment_count` memiliki distribusi yang sangat miring ke kanan (skewed). Selain itu, outliers dalam jumlah besar membuat model menjadi kurang stabil dan akurat.
    - **Solusi**: Menggunakan **log transformation** untuk mengurangi skewness dan menekan outliers. Log transformasi membuat distribusi data menjadi lebih simetris dan membantu model menangkap pola dengan lebih baik.
2. **Variabilitas Skala pada Fitur**:
    - **Masalah**: Variabel seperti `views` memiliki skala yang sangat besar (dari ribuan hingga ratusan juta), sementara variabel lain seperti `No_tags` memiliki rentang yang jauh lebih kecil (hanya beberapa puluhan).
    - **Solusi**: Melakukan **normalisasi** menggunakan **Min-Max Scaling** untuk menstandardisasi skala dari fitur-fitur, sehingga semua variabel berada pada rentang yang seragam. Ini penting agar model tidak terlalu dipengaruhi oleh fitur dengan nilai yang sangat besar.
3. **Memilih Model yang Tepat**:
    - **Masalah**: Model linear seperti **Linear Regression** tidak selalu mampu menangkap pola yang kompleks, dan beberapa model seperti **Decision Tree** rentan terhadap **overfitting**.
    - **Solusi**: Mencoba berbagai algoritma, seperti **Ridge Regression**, **Random Forest**, dan **Support Vector Regressor**. Berdasarkan evaluasi menggunakan **RMSE** dan **R^2**, **Random Forest** dipilih sebagai model terbaik karena mampu memberikan prediksi yang lebih akurat dan stabil.
4. **Hyperparameter Tuning**:
    - **Masalah**: Memilih hyperparameter yang optimal untuk model, terutama untuk model seperti **Random Forest** dan **Support Vector Regressor**, membutuhkan banyak eksperimen. Hal ini memakan waktu dan perlu pendekatan yang sistematis.
    - **Solusi**: Melakukan **Grid Search** atau **Random Search** untuk menemukan kombinasi hyperparameter yang optimal. Meski demikian, hal ini tetap menjadi tantangan karena membutuhkan sumber daya komputasi yang cukup besar.
5. **Duplikasi dan Missing Values**:

    - **Masalah**: Ada sejumlah **data duplikat** yang cukup besar (4.229 data) serta **missing values** pada kolom `description`.
    - **Solusi**: Menghapus data duplikat untuk memastikan bahwa analisis dilakukan pada data yang unik. Untuk missing values, kolom `description` di-drop karena hanya mempengaruhi sebagian kecil dari data (45 data atau sekitar 0.12%).
6. **Mengatasi Overfitting pada Model yang Kompleks**:
    - **Masalah**: Beberapa model, seperti **Decision Tree**, cenderung overfitting terutama ketika kompleksitas model tidak dibatasi.
    - **Solusi**: Menggunakan model ensemble seperti **Random Forest** yang secara alami membantu mengurangi overfitting dengan menggabungkan beberapa pohon keputusan (decision trees). Selain itu, regularisasi pada model seperti **Ridge Regression** juga digunakan untuk mencegah overfitting.