

MODEL PEMBELAJARAN DAN LAPORAN AKHIR

PROJECT-BASED LEARNING

MATA KULIAH DATA MINING I

KELAS A



"Klasifikasi Genre Musik dengan Menggunakan Decision Tree, Random Forest dan XGBoost Berdasarkan Fitur Audio dari Spotify"

DISUSUN OLEH KELOMPOK III :

1. NAUVAL THEO JOVALDI (22083010096) - KETUA
2. TSABITA ROSYIDAH P (22083010012) - ANGGOTA
3. RIZKI AMANDA (22083010045) - ANGGOTA
4. RAFI IMAM DWIYANTO (22083010109) - ANGGOTA

DOSEN PENGAMPU:

KARTIKA MAULIDA HINDRAYANI, S.KOM, M.KOM (199209092022032009)

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

JAWA TIMUR

2024

DAFTAR ISI

DAFTAR GAMBAR.....	3
DAFTAR TABEL.....	4
BAB I.....	5
PENDAHULUAN.....	5
1.1 Latar Belakang.....	5
1.2 Permasalahan.....	5
1.3 Tujuan.....	6
1.4 Manfaat.....	6
BAB II.....	7
TINJAUAN PUSTAKA.....	7
2.1 Teori Penunjang.....	7
2.1.1 Spotify.....	7
2.1.2 Klasifikasi.....	7
2.1.3 Data Mining.....	7
2.1.4 Bahasa Pemrograman Python.....	8
2.1.5 Decision Tree.....	8
2.1.6 Random Forest.....	9
2.1.7 XGBoost.....	9
2.1.8 Akurasi.....	9
2.2 Penelitian Terkait.....	9
BAB III.....	11
METODOLOGI PENELITIAN.....	11
BAB IV.....	12
HASIL DAN PEMBAHASAN.....	12
BAB V.....	29
KESIMPULAN.....	29
DAFTAR PUSTAKA.....	30
LAMPIRAN.....	32

DAFTAR GAMBAR

Gambar 1. Logo Spotify.....	7
Gambar 2. Logo Bahasa Pemrograman Python.....	8
Gambar 3. Logo Decision Tree.....	8
Gambar 4. Logo Random Forest.....	9
Gambar 5. Logo XGBoost.....	9
Gambar 6. Desain Sistem Klasifikasi Genre Musik.....	11
Gambar 7. Dataset Spotify.....	12
Gambar 8. Frekuensi Genre Musik.....	13
Gambar 9. Spotify Audio Feature Density - by Genre.....	14
Gambar 10. Duration Outlier Boxplot.....	16
Gambar 11. Duration Outliers removed Boxplot.....	17
Gambar 12. Audio Feature Correlation Heatmap.....	18
Gambar 13. Correlation Between Median Genre Feature Values Heatmap.....	20
Gambar 14. Genre Decision Tree.....	23
Gambar 15. Random Forest Classifier.....	25
Gambar 16. XGBClassifier.....	25
Gambar 17. Grafik Feature Importance.....	27

DAFTAR TABEL

Tabel 1. Hasil Akurasi	28
------------------------------	----

BAB I

PENDAHULUAN

1.1 Latar Belakang

Musik adalah bahasa universal yang memungkinkan kita mengungkapkan perasaan dan emosi yang sulit diungkapkan dengan kata-kata[11]. Musik memiliki kemampuan untuk mempengaruhi suasana hati, emosi, bahkan perilaku seseorang. Dengan kemajuan perkembangan teknologi saat ini, semua orang dapat lebih mudah mengakses musik dengan bantuan layanan streaming musik, salah satu platform layanan streaming musik yang populer adalah Spotify. Dengan rekam jejak selama 15 tahun, pengguna aktif spotify telah mencapai lebih dari 500 juta[10]. Spotify merupakan platform streaming musik besar, sehingga data yang dihasilkan oleh spotify menjadi sangat besar dan beragam terkait dengan karakteristik audio dari jutaan lagu yang tersedia di Spotify. Data ini mencakup berbagai fitur audio yang dapat dimanfaatkan untuk analisis lebih lanjut.

Klasifikasi genre musik adalah salah satu tantangan yang menarik dalam bidang pengolahan sinyal audio dan *machine learning*. Genre musik adalah label musik berdasarkan karakteristik dan elemen musikalnya. Dengan tersedianya data fitur audio dari Spotify, kami dapat mengembangkan model yang dapat mengklasifikasikan fitur tersebut. Ini dapat membantu industri musik dalam analisis trend dan rekomendasi musik.

Decision Tree, *Random Forest* dan *XGBoost* merupakan metode *machine learning* yang populer dikenal efektif untuk melakukan tugas-tugas klasifikasi. *Decision Tree* adalah sebuah metode pengambilan keputusan dengan menyusun setiap opsi atau pilihan menjadi bentuk yang bercabang [9]. *Random Forest* adalah algoritma machine learning yang menggabungkan keluaran dari beberapa decision tree untuk mencapai satu hasil [12]. XGBoost, atau eXtreme Gradient Boosting, merupakan algoritma yang berbasis pada pohon keputusan (Chen dan Guestrin, 2016 [13]). XGBoost termasuk dalam keluarga algoritma berbasis pohon yang meliputi Decision Tree, Random Forest, bagging, boosting, dan gradient boosting.

Penelitian ini bertujuan untuk mengeksplorasi penggunaan *Decision Tree*, *Random Forest* dan *XGBoost* dalam mengklasifikasikan genre musik berdasarkan fitur audio dari Spotify. Dengan memanfaatkan data dari Spotify, penelitian ini akan mengkaji efektivitas kedua metode tersebut dalam mengenali pola dan karakteristik yang membedakan genre musik. Hasil dari penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan sistem klasifikasi genre musik yang lebih canggih dan akurat serta membuka peluang untuk pengembangan sistem rekomendasi musik dan analisis tren musik pada platform layanan streaming musik.

1.2 Permasalahan

1. Bagaimana mengklasifikasikan musik ke dalam genre yang tepat berdasarkan fitur audio?
2. Bagaimana membangun model klasifikasi yang akurat untuk mengklasifikasikan musik ke dalam genre yang tepat?

1.3 Tujuan

1. Mengklasifikasikan genre musik berdasarkan fitur audio yang tersedia di Spotify menggunakan algoritma *Decision Tree*, *Random Forest* dan *XGBoost*.
2. Menganalisis dan evaluasi kinerja model klasifikasi yang dibangun dengan menggunakan perhitungan akurasi.

1.4 Manfaat

1. Manfaat bagi Pengembangan Ilmu Pengetahuan
 - Proyek ini akan menambah wawasan tentang penerapan algoritma *Decision Tree*, *Random Forest* dan *XGBoost* dalam klasifikasi genre musik serta memberikan pemahaman lebih dalam mengenai fitur audio yang berpengaruh, sehingga dapat menjadi basis dan referensi untuk penelitian lebih lanjut dalam bidang machine learning, data mining, dan analisis musik.
2. Manfaat bagi Industri Musik
 - Hasil dari proyek ini dapat digunakan untuk meningkatkan sistem rekomendasi musik di platform streaming dan membantu industri musik dalam mengembangkan alat analisis untuk memahami tren musik dan preferensi pendengar.
3. Manfaat bagi Pengembang Aplikasi Musik
 - Proyek ini memungkinkan pengembang aplikasi musik untuk mengotomatisasi proses tagging genre dan memperkaya database musik dengan informasi genre yang lebih akurat, sehingga menghemat waktu dan usaha dalam pengelolaan dan pencarian lagu baru.
4. Manfaat bagi Edukasi dan Pelatihan
 - Proyek ini dapat digunakan sebagai studi kasus dalam kurikulum pendidikan terkait data science, machine learning, dan analisis musik, serta menjadi bahan latihan praktis bagi mahasiswa dan praktisi untuk memahami penerapan algoritma machine learning dalam kasus nyata.
5. Manfaat bagi Pengguna Umum
 - Proyek ini akan meningkatkan pengalaman mendengarkan musik pengguna layanan streaming dengan rekomendasi lagu yang lebih relevan, serta membantu mereka menemukan musik dari genre yang belum dieksplorasi, sehingga memperluas wawasan musik mereka.

BAB II

TINJAUAN PUSTAKA

2.1 Teori Penunjang

2.1.1 Spotify



Gambar 1. Logo Spotify

Spotify adalah salah satu layanan streaming musik yang sangat populer saat ini. Platform ini menyediakan akses ke jutaan lagu, podcast, dan video dari berbagai artis di seluruh dunia (Spotify.com, 2017). Dengan pesatnya perkembangan teknologi internet, audio streaming semakin menjadi pilihan favorit. Secara sederhana, media streaming adalah proses pengiriman konten audio, video, atau teks secara terus-menerus melalui internet (Illinois Online Network dalam Cox, 2005). Manfaat dari layanan music streaming, baik yang gratis maupun berbayar, adalah memberikan akses yang berkelanjutan untuk mendengarkan musik kapan saja dan di mana saja dengan mudah (American University, 2016).[1]

2.1.2 Klasifikasi

Klasifikasi adalah proses mengelompokkan bahan perpustakaan berdasarkan subjek atau isi pokok. Menurut Sulisty-Basuki (1991), klasifikasi berasal dari kata Latin "classis," yang berarti proses mengelompokkan benda yang sama dan memisahkan yang berbeda. Towa P. Hamakonda dan J.N.B. Tairas (1995) menyatakan bahwa klasifikasi adalah pengelompokan sistematis dari objek, ide, buku, atau benda lain ke dalam kelas tertentu berdasarkan kesamaan ciri. Hasil pengelompokan ini diberi simbol atau notasi, yang disebut nomor klasifikasi. Ada dua jenis notasi yaitu notasi murni (menggunakan huruf, angka, atau tanda lain secara konsisten) dan notasi campuran (menggunakan kombinasi huruf dan angka).[2]

2.1.3 Data Mining

Data mining merupakan proses yang memanfaatkan statistik, matematika, kecerdasan buatan, dan machine learning untuk mengekstraksi dan mengidentifikasi informasi serta pengetahuan yang berguna dari berbagai database besar (Wahyudin, 2019). Data mining dibagi menjadi beberapa kategori yaitu deskripsi, estimasi, prediksi, klasifikasi, pengelompokan, dan asosiasi.[5]

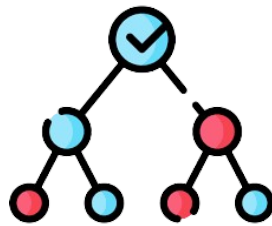
2.1.4 Bahasa Pemrograman Python



Gambar 2. Logo Bahasa Pemrograman Python

Python adalah bahasa pemrograman berorientasi objek yang dapat dioperasikan secara interaktif (Ridho & Niani, 2022). Bahasa ini memiliki struktur data tingkat tinggi. Python merupakan bahasa pemrograman interpretatif yang memiliki berbagai fungsi dan dirancang dengan fokus pada kejelasan serta kemudahan pemahaman kode. Python dianggap sebagai bahasa yang menggabungkan kemampuan dengan kejelasan sintaksis. Bahasa pemrograman Python dirancang khusus untuk memudahkan programmer dalam membuat program dengan efisiensi waktu, kemudahan pengembangan, dan kompatibilitas dengan berbagai sistem. Python dapat digunakan untuk membuat aplikasi mandiri maupun pemrograman skrip (Aqmila, 2022).[6]

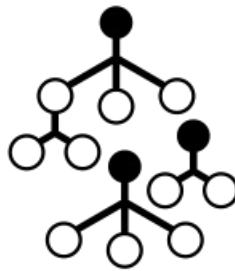
2.1.5 Decision Tree



Gambar 3. Logo Decision Tree

Decision Tree merupakan metode penelitian yang paling sering digunakan untuk masalah klasifikasi. Decision tree adalah sebuah struktur yang bisa digunakan untuk membagi kumpulan data besar menjadi himpunan-himpunan record yang lebih kecil melalui serangkaian aturan keputusan. Setiap simpul pada daun menandai label kelas. Simpul yang bukan simpul akhir terdiri dari simpul internal dan akar yang terdiri dari kondisi tes atribut pada sebagian record yang memiliki karakteristik yang berbeda. Simpul-simpul internal dan akar ditandai dengan bentuk oval dan simpul daun ditandai dengan segi empat.[3]

2.1.6 Random Forest



Gambar 4. Logo Random Forest

Random Forest adalah pengembangan dari metode Decision Tree yang menggunakan beberapa Decision Tree, dimana setiap Decision Tree telah dilakukan pelatihan menggunakan sampel individu dan setiap atribut dipecah pada pohon yang dipilih antara atribut subset yang bersifat acak. Random Forest memiliki beberapa kelebihan, yaitu dapat meningkatkan hasil akurasi jika terdapat data yang hilang, dan untuk resisting outliers, serta efisien untuk penyimpanan sebuah data. Selain itu, Random Forest mempunyai proses seleksi fitur dimana mampu mengambil fitur terbaik sehingga dapat meningkatkan performa terhadap model klasifikasi. Dengan adanya seleksi fitur tentu Random Forest dapat bekerja pada big data dengan parameter yang kompleks secara efektif.[4]

2.1.7 XGBoost



Gambar 5. Logo XGBoost

XGBoost merupakan algoritma pembelajaran mesin yang berbasis gradient tree boosting. Algoritma ini cocok digunakan untuk peramalan deret waktu karena mampu menangani masalah pembelajaran mesin dalam skala besar.[8]

2.1.8 Akurasi

Menurut Tedi (2022), akurasi adalah sejauh mana hasil estimasi, perhitungan, atau detail sesuai dengan nilai atau standar yang tepat. Akhirnya, akurasi menentukan seberapa dekat hasil estimasi dengan nilai yang sebenarnya atau yang diakui sebagai benar. Akurasi berarti mendapatkan nilai yang mendekati nilai aslinya. Memperkirakan seberapa tepat suatu estimasi dibandingkan dengan referensi yang berbeda disebut akurasi, seperti yang dijelaskan oleh Apipah (2015).[7]

2.2 Penelitian Terkait

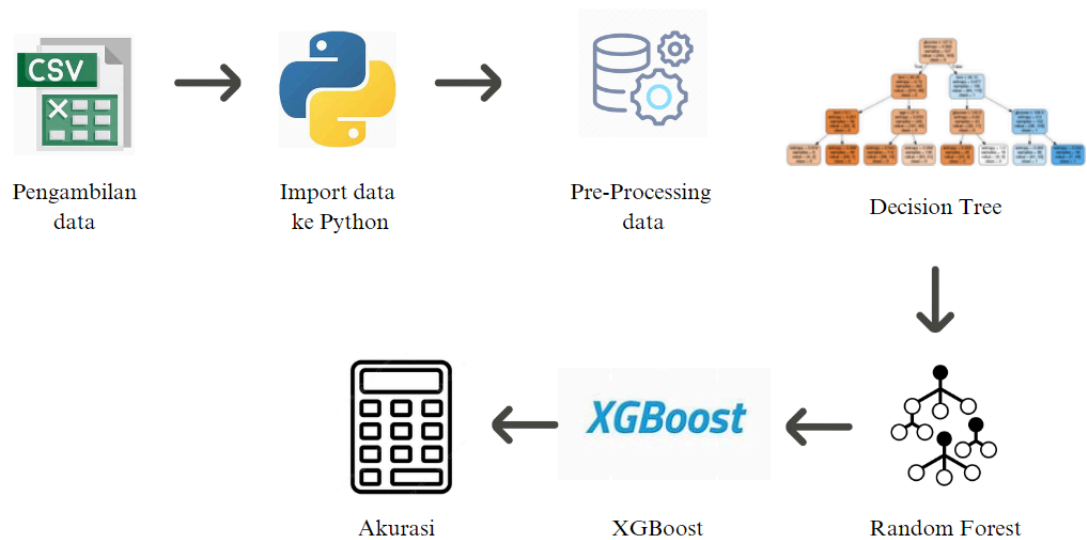
2.2.1 Penelitian yang dilakukan oleh Ivan Luis Simarmata dan I Wayan Supriana yang berjudul “*Music Genre Classification Using Random Forest Model*” (2023). Penelitian ini membahas tentang bagaimana cara mengklasifikasikan *Genre* musik dengan menggunakan random forest dan akan dilakukan perbandingan akurasi dengan menggunakan metode *Extreme Gradient Boosting*. Dengan sampel yang sama ditemukan bahwa metode *random forest* mendapatkan hasil akurasi yang lebih baik apabila dibandingkan dengan metode *Extreme Gradient Boosting*, namun dalam kasus tertentu yang mana terdapat beberapa fitur audio yang dihilangkan didapatkan bahwa metode *Extreme Gradient Boosting* mendapatkan nilai akurasi yang sedikit lebih baik bila dibandingkan dengan metode *random forest*. Hal menarik ditemukan dalam penelitian ini adalah kedua metode dapat menangani klasifikasi musik klasik dengan baik, sedangkan untuk musik rock kedua model masih kesulitan dalam mengklasifikasikan hasil yang akurat.

2.2.2 Penelitian selanjutnya dilakukan oleh Sally Lutfiani, Triando Hamonangan Saragih, Friska Abadi, Mohammad Reza Faisal dan Dwi Kartini dengan judul “Perbandingan Metode Extreme Gradient Boosting dan Metode Decision Tree Untuk Klasifikasi Genre Musik” (2023). Dataset yang digunakan dalam penelitian ini berasal dari GTZAN, Dalam pengujian penelitian ini masing-masing metode dilakukan percobaan dengan empat parameter. Metode *Extreme Gradient Boosting* menggunakan pengujian dalam parameter *Extreme Gradient Boosting Default*, *Learning rate*, *N-estimator*, dan parameter *gamma* yang mana keempat parameter tersebut menghasilkan tingkat akurasi sekitar 72% sedangkan dalam menguji metode *decision tree* menggunakan parameter *max depth*, *min sample split*, *min sample leaf*, dan parameter *max features* yang mana keempat parameter tersebut menghasilkan tingkat akurasi sekitar 51%.

2.2.3 Penelitian yang dilakukan oleh Muhammad Abid As Sarofi, Irhamah, dan Adatul Mukarromah dengan judul "Identifikasi Genre Musik dengan Menggunakan Metode Random Forest" (2020) bertujuan untuk mengidentifikasi genre lagu menggunakan metode Random Forest. Data yang digunakan dalam penelitian ini adalah GTZAN dataset yang diambil dari laman MARSYAS. Fitur ekstraksi yang digunakan adalah MFCC karena kemampuannya mengadaptasi pendengaran manusia. Model yang digunakan dalam penelitian ini menunjukkan performa klasifikasi yang tinggi dengan menggunakan KCV untuk membagi data training dan testing. Hasilnya, nilai akurasi dan F-score masing-masing adalah 0,9969 dan 0,9970 untuk data training, serta 0,8883 dan 0,8882 untuk data testing.

BAB III

METODOLOGI PENELITIAN



Gambar 6. Desain Sistem Klasifikasi Genre Musik

Penelitian ini dimulai dengan pengambilan data fitur audio dari Spotify yang diambil melalui github [berikut ini](#) yang kemudian disimpan dalam format .csv. Data tersebut mencakup berbagai atribut seperti track_id, track_name, track_artist, track_popularity, track_album_id, track_album_name, track_album_release_date, playlist_name, playlist_id, playlist_genre, playlist_subgenre, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentality, liveness, valence, tempo, dan duration_ms, yang masing-masing mewakili karakteristik audio dari lagu-lagu dengan genre berbeda. Data yang telah dikumpulkan kemudian diimpor ke dalam Python untuk analisis lebih lanjut. Langkah pertama dalam preprocessing data meliputi eksplorasi fitur audio berdasarkan genre, penghapusan outliers untuk memastikan data yang bersih dan relevan, serta analisis korelasi antara fitur dalam genre untuk memahami hubungan antar variabel.

Selanjutnya, data dipersiapkan untuk pelatihan dengan membaginya menjadi data latih dan data uji. Pada tahap implementasi, proses pengolahan model Decision Tree, Random Forest, dan XGBoost dimulai dengan pelatihan model Decision Tree menggunakan data latih, diikuti dengan evaluasi menggunakan data uji untuk menilai kinerja awal. Selanjutnya, model Random Forest dilatih dan dievaluasi dengan cara yang sama. Random Forest, sebagai metode ensemble, menggunakan banyak pohon keputusan untuk meningkatkan akurasi dan mengurangi overfitting dengan menggabungkan prediksi dari berbagai pohon keputusan yang dibuat dari subset data. Kemudian, model Gradient Boosting dengan XGBoost dilatih dan dievaluasi. XGBoost, sebagai versi yang lebih efisien dari Gradient Boosting, mengoptimalkan proses pelatihan dengan menggunakan teknik regularisasi yang lebih baik dan efisiensi komputasi yang lebih tinggi. Model ini membangun pohon keputusan secara berurutan, di mana setiap pohon baru mencoba memperbaiki kesalahan dari pohon sebelumnya. Analisis pentingnya variabel dilakukan untuk mengidentifikasi fitur-fitur yang paling berpengaruh dalam klasifikasi genre musik. Fitur-fitur ini dievaluasi berdasarkan kontribusi mereka terhadap keputusan yang dihasilkan oleh model. Akurasi dari setiap model dihitung menggunakan rumus perhitungan akurasi untuk menilai kinerja secara keseluruhan. Hasil dari penelitian ini diharapkan dapat memberikan wawasan tentang keefektifan algoritma klasifikasi dalam menentukan genre musik berdasarkan fitur audio yang ada.

BAB IV

HASIL DAN PEMBAHASAN

- Getting The Data

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
import xgboost as xgb

# Load the dataset
playlist_songs = pd.read_csv('/content/spotifyfulldataset.csv',
encoding='latin1')

# Extract feature names (adjust column indices if necessary)
feature_names = playlist_songs.columns[11:23]

# View the structure of the dataset
display(playlist_songs.info())
display(playlist_songs.describe())
display(playlist_songs.head())

# Custom function to display DataFrame structure similar to glimpse
in R
def glimpse(df):
    display(f"Rows: {df.shape[0]}, Columns: {df.shape[1]}")
    for col in df.columns:
        display(f"{col}: {df[col].dtype}")

# Apply the custom glimpse function
glimpse(playlist_songs)
```

Kode Di Atas bertujuan untuk membuat dan memeriksa struktur dataset lagu Spotify, serta mempersiapkan data untuk analisis lebih lanjut atau pembuatan model pembelajaran mesin.

Output :

	track_popularity	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms
count	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000
mean	42.477081	0.654850	0.698619	5.374471	-6.719499	0.565711	0.107068	0.175334	0.084747	0.190176	0.510561	120.881132	225799.811622
std	24.984074	0.145085	0.180910	3.611657	2.988436	0.495671	0.101314	0.219633	0.224230	0.154317	0.233146	26.903624	59834.006182
min	0.000000	0.000000	0.000175	0.000000	-46.448000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4000.000000
25%	24.000000	0.563000	0.581000	2.000000	-8.171000	0.000000	0.041000	0.015100	0.000000	0.092700	0.331000	99.960000	187819.000000
50%	45.000000	0.672000	0.721000	6.000000	-6.166000	1.000000	0.062500	0.080400	0.000016	0.127000	0.512000	121.984000	216000.000000
75%	62.000000	0.761000	0.840000	9.000000	-4.645000	1.000000	0.132000	0.255000	0.004830	0.248000	0.693000	133.918000	253585.000000
max	100.000000	0.983000	1.000000	11.000000	1.275000	1.000000	0.918000	0.994000	0.994000	0.996000	0.991000	239.440000	517810.000000

	track_id	track_name	track_artist	track_popularity	track_album_id	track_album_name	track_album_release_date	playlist_name	playlist_id	playlist_genre	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms	
0	6R807dlnrta1jY8h7YN	I Don't Care (with Justin Bieber) - Justin Bieber, Lud...	Ed Sheeran	66	2zC4Y0G3dK0R8G8Z52Cz	I Don't Care (with Justin Bieber) [Small Luxury...	2019-06-14	Pop Remix	37f9kQ2f10KJZD07fE8K9W	pop	--	6	-2.634	1	0.0583	0.1020	0.000000	0.0653	0.518	122.036	194754
1	0YTCVb2TWZg7b7C7H8ZP1	Memories - Maroon 5	Maroon 5	67	63HPS026A4J8W1X356C9H6	Memories (Official Remi...	2019-12-13	Pop Remix	37f9kQ2f10KJZD07fE8K9W	pop	--	11	-4.969	1	0.0373	0.0724	0.004210	0.3570	0.693	95.972	162800
2	1z1ng7V8dAHdH0EemDE79	All the Time - Don Diablo	Zaria Larson	70	1HcdngJdLccR0UE8g7Hv4	All the Time (Don Diablo Remi...	2019-07-05	Pop Remix	37f9kQ2f10KJZD07fE8K9W	pop	--	1	-3.432	0	0.0742	0.0794	0.000023	0.1100	0.613	124.008	176616
3	73P8d8HvQm8H8uG5C7	Call You Mine - The Chainsmokers	The Chainsmokers	69	1nq95d87yK8G020V8b8k6	Call You Mine - The Remi...	2019-07-19	Pop Remix	37f9kQ2f10KJZD07fE8K9W	pop	--	7	-3.778	1	0.1020	0.0287	0.000009	0.2040	0.277	121.956	169083
4	1e8PA6KJN8d8H8uG5C7	Someone You Loved - Lewis Capaldi	Lewis Capaldi	69	2n7Yv8d87yK8G020V8b8k6	Someone You Loved (Future Humans Remi...	2019-03-05	Pop Remix	37f9kQ2f10KJZD07fE8K9W	pop	--	1	-4.672	1	0.0359	0.0803	0.000000	0.0833	0.725	123.976	189052

Gambar 7. Dataset Spotify

Mendapatkan nilai count, mean sampai max pada tiap variabel dan juga menampilkan dataset lagu spotify.

```
from tabulate import tabulate
```

```
# Count the occurrences of each genre
genre_counts =
playlist_songs['playlist_genre'].value_counts().reset_index()
genre_counts.columns = ['playlist_genre', 'count']

# Display the table in a format similar to knitr::kable()
print(tabulate(genre_counts, headers='keys', tablefmt='pipe',
showindex=False))
```

Tujuan kode diatas adalah untuk menghitung jumlah frekuensi kemunculan pada setiap genre dalam kolom 'playlist_genre' dan menampilkan tabel dalam format terstruktur dengan mengimpor 'tabulate'

playlist_genre	count
edm	6043
rap	5746
pop	5507
r&b	5431
latin	5155
rock	4951

Gambar 8. Frekuensi Genre Musik

Mendapatkan dataset yang memiliki beberapa genre dengan frekuensi yang berbeda

- Exploring Audio Features by Genre

```
# Check the first few rows of the dataset to ensure it's loaded
correctly
display(playlist_songs.head())

# Extract feature names
feature_names = playlist_songs.columns[11:23]
display(f"Feature names: {feature_names}")

# Select the relevant columns and reshape the dataframe
df_long = pd.melt(playlist_songs, id_vars=['playlist_genre'],
value_vars=feature_names, var_name='feature', value_name='value')

# Filter out non-numeric values from the 'value' column
df_long_numeric = df_long[pd.to_numeric(df_long['value'],
errors='coerce').notna()]

# Display the first few rows of the new dataframe to ensure it has
the correct data
display(df_long_numeric.head())

# Set up the seaborn style
sns.set(style="whitegrid")

# Create the density plots
g = sns.FacetGrid(df_long_numeric, col='feature', col_wrap=3,
sharex=False, sharey=False, height=4, aspect=1.5)
g.map_dataframe(sns.kdeplot, x='value', hue='playlist_genre',
fill=True, common_norm=False, alpha=0.5)
g.add_legend()
```

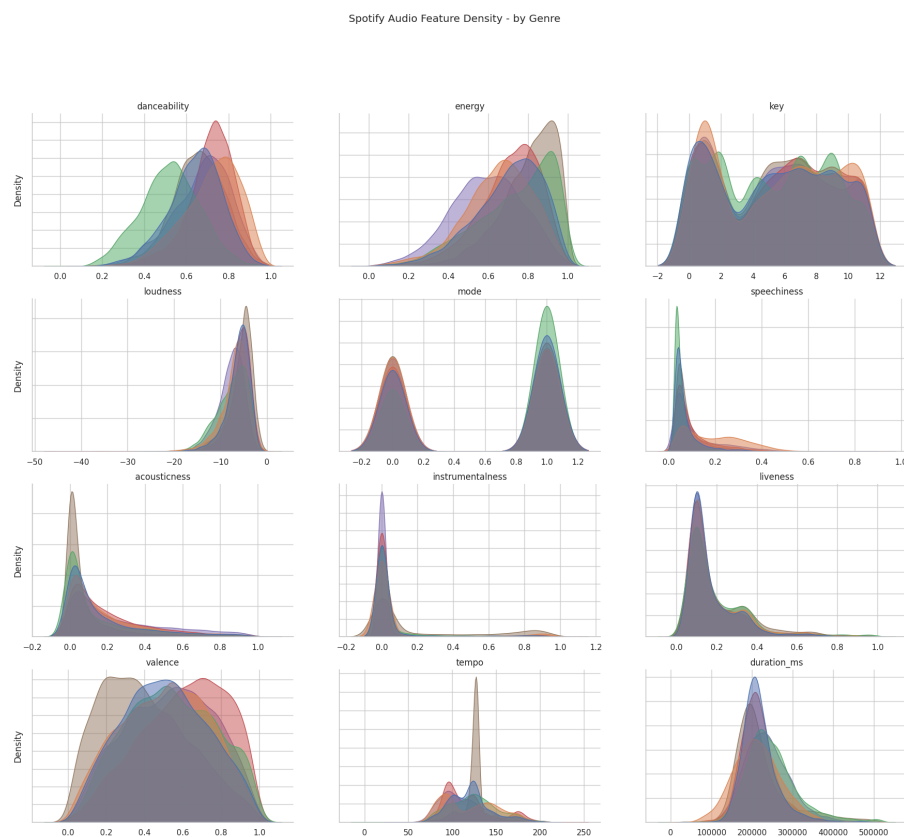
```
# Add titles and labels
g.set_titles('{col_name}')
g.set_axis_labels('', 'Density')
g.fig.suptitle('Spotify Audio Feature Density - by Genre', y=1.02)

# Remove y-axis text
for ax in g.axes.flatten():
    ax.yaxis.set_ticklabels([])

# Adjust layout
plt.subplots_adjust(top=0.9)
plt.show()
```

Tujuan kode diatas adalah menentukan kolom yang dianggap sebagai fitur audio yang relevan untuk menganalisis dan memverifikasi nama-nama fitur tersebut dan mengubah bentuk DataFrame untuk memudahkan visualisasi, diubah menjadi baris dengan dua kolom baru yaitu 'feature' dan 'value'. Tujuan selanjutnya adalah membuat plot densitas untuk setiap fitur audio, difasilitasi oleh 'seaborn.FaceGrid' untuk memisahkan plot berdasarkan fitur. Plot ini dibawah ini menggambarkan distribusi nilai fitur untuk setiap genre dengan menggunakan 'kdeplot'

Output :



Gambar 9. Spotify Audio Feature Density - by Genre

Setiap grafik menunjukkan distribusi fitur audio tertentu untuk genre musik yang berbeda. Fitur audio ini meliputi:

- Danceability: Seberapa menarik musiknya.
- Energy: Tingkat energi musiknya.

- **Key:** Kunci musiknya.
- **Loudness:** Keras suara musiknya.
- **Speechiness:** Seberapa banyak musiknya terdengar seperti ucapan.
- **Liveness:** Seberapa banyak musiknya terdengar seperti dimainkan secara langsung.
- **Acousticness:** Seberapa banyak musiknya terdengar seperti musik akustik.
- **Instrumentalness:** Seberapa banyak musiknya instrumental.
- **Valence:** Seberapa positif atau negatif musiknya.
- **Tempo:** Kecepatan tempo musiknya.
- **Duration:** Durasi lagu dalam milidetik.

Danceability: Genre musik elektronik dan dance memiliki danceability yang tinggi, sedangkan genre musik klasik dan jazz memiliki danceability yang rendah.

Energy: Genre musik rock dan metal memiliki energy yang tinggi, sedangkan genre musik klasik dan ambient memiliki energy yang rendah.

Key: Genre musik pop dan rock sering menggunakan kunci mayor, sedangkan genre musik blues dan jazz sering menggunakan kunci minor.

Loudness: Genre musik rock dan metal sering kali memiliki loudness yang tinggi, sedangkan genre musik klasik dan ambient sering kali memiliki loudness yang rendah.

Speechiness: Genre musik rap dan hip hop memiliki speechiness yang tinggi, sedangkan genre musik klasik dan instrumental memiliki speechiness yang rendah.

Liveness: Genre musik rock dan metal sering kali memiliki liveness yang tinggi, sedangkan genre musik elektronik dan dance sering kali memiliki liveness yang rendah.

Acousticness: Genre musik klasik dan folk memiliki acousticness yang tinggi, sedangkan genre musik elektronik dan dance memiliki acousticness yang rendah.

Instrumentalness: Genre musik klasik dan instrumental memiliki instrumentalness yang tinggi, sedangkan genre musik pop dan rock sering kali memiliki instrumentalness yang rendah.

Valence: Genre musik pop dan dance sering kali memiliki valence yang tinggi, sedangkan genre musik blues dan metal sering kali memiliki valence yang rendah.

Tempo: Genre musik dance dan elektronik sering kali memiliki tempo yang tinggi, sedangkan genre musik klasik dan ambient sering kali memiliki tempo yang rendah.

Duration: Genre musik klasik dan jazz sering kali memiliki durasi yang panjang, sedangkan genre musik pop dan rock sering kali memiliki durasi yang pendek.

- **Removing Outliers**

```
# Create a boxplot with outliers
plt.figure(figsize=(6, 8))
sns.boxplot(y='duration_ms', data=playlist_songs, color='red',
width=0.5)
plt.title('Duration')
plt.ylabel('Duration (ms)')
plt.xlabel('')
plt.gca().invert_yaxis() # Invert y-axis to have longer durations at
the top
plt.show()

# Identify and remove outliers
q1 = playlist_songs['duration_ms'].quantile(0.25)
q3 = playlist_songs['duration_ms'].quantile(0.75)
```

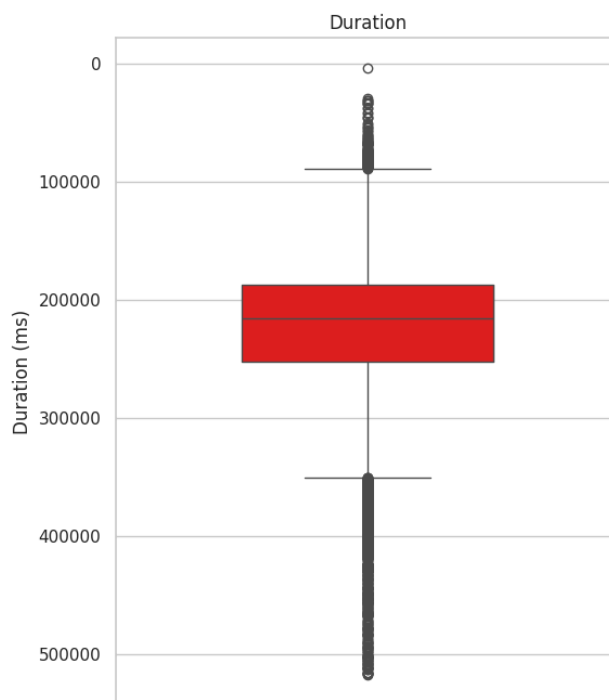
```

iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
playlist_songs_no_outliers =
playlist_songs[(playlist_songs['duration_ms'] >= lower_bound) &
(playlist_songs['duration_ms'] <= upper_bound)]

# Create a boxplot without outliers
plt.figure(figsize=(6, 8))
sns.boxplot(y='duration_ms', data=playlist_songs_no_outliers,
color='red', width=0.5)
plt.title('Duration, outliers removed')
plt.ylabel('Duration (ms)')
plt.xlabel('')
plt.gca().invert_yaxis() # Invert y-axis to have longer durations at
the top
plt.show()

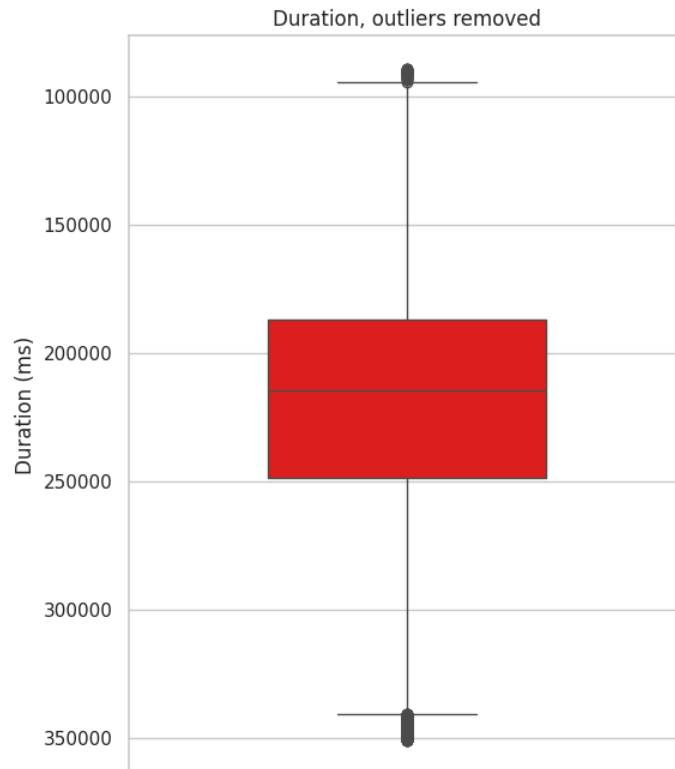
```

Kode diatas digunakan untuk membuat dua boxplot untuk durasi lagu dalam milidetik dari dataset `playlist_songs`. Boxplot pertama menyertakan outlier, sedangkan yang kedua menghapus outlier. Setelah membuat boxplot awal dengan seaborn dan membalik sumbu y agar durasi yang lebih lama ada di atas, kode ini menghitung kuartil pertama dan ketiga serta rentang interkuartil untuk menentukan batas bawah dan atas outlier. Data yang berada di luar batas ini dihapus, dan boxplot kedua dibuat dari data yang sudah dibersihkan dari outlier.



Gambar 10. Duration Outlier Boxplot

Memiliki median 100.000 milidetik dengan variabilitas sedang. Terdapat beberapa outlier yang menunjukkan durasi peristiwa yang jauh lebih pendek atau lebih lama dibandingkan dengan durasi normal.



Gambar 11. Duration Outliers removed Boxplot

Berdasarkan interpretasi boxplot di atas, dapat disimpulkan bahwa

- Distribusi data tidak simetris, dengan kecenderungan ke arah kanan.
- Ukuran tendensi sentral data adalah sekitar 200.000 ms.
- Ukuran penyebaran data adalah sekitar 75.000 ms.
- Terdapat satu nilai outlier pada data, yaitu dengan nilai sekitar 350.000 ms.

- **Correlation Between Features**

```
# Define function to remove outliers
def remove_outliers(df, column, coef=4):
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - coef * iqr
    upper_bound = q3 + coef * iqr
    return df[(df[column] >= lower_bound) & (df[column] <=
upper_bound)]

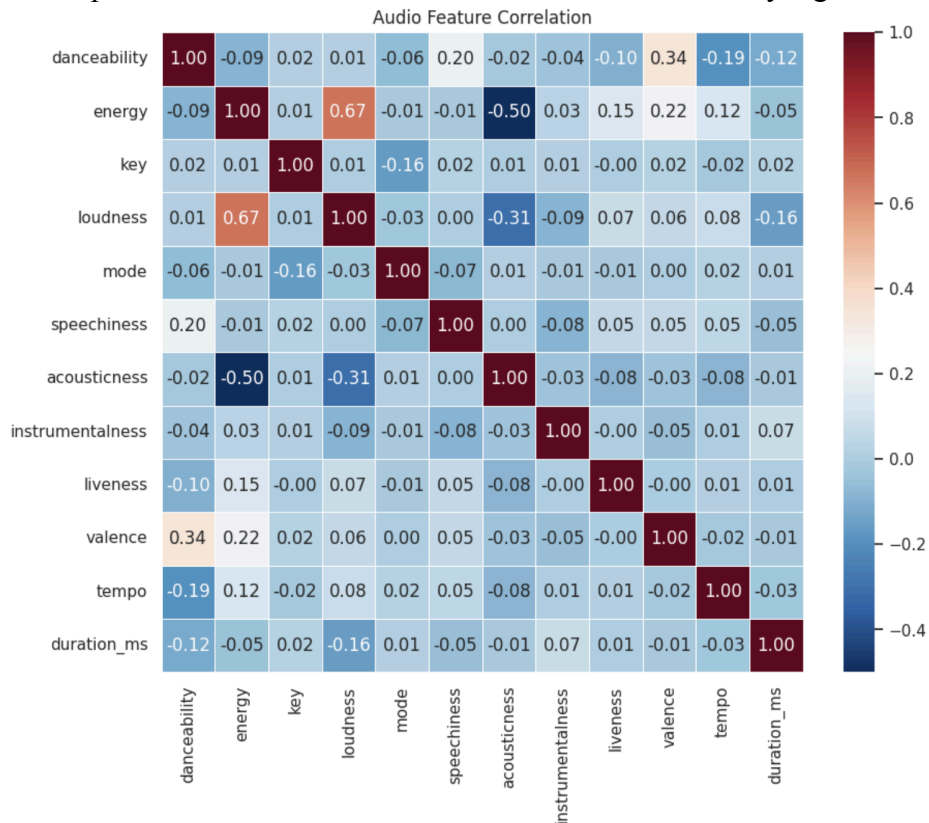
# Remove outliers from the selected features
playlist_songs_no_outliers = playlist_songs.copy()
for feature in feature_names:
    playlist_songs_no_outliers =
remove_outliers(playlist_songs_no_outliers, feature)

# Calculate the correlation matrix
correlation_matrix = playlist_songs_no_outliers[feature_names].corr()

# Create the correlation plot
plt.figure(figsize=(10, 8))
```

```
sns.heatmap(correlation_matrix, annot=True, cmap='RdBu_r', fmt=".2f",
linewidths=0.5)
plt.title('Audio Feature Correlation')
plt.show()
```

Kode ini mendefinisikan fungsi untuk menghapus outlier, menghapus outlier dari fitur-fitur tertentu dalam dataset `playlist_songs`, menghitung matriks korelasi, dan membuat plot korelasi. Fungsi `remove_outliers` menghitung batas bawah dan atas untuk mendeteksi outlier berdasarkan koefisien dari rentang interkuartil. Fitur-fitur yang dipilih diolah untuk menghapus outlier, dan kemudian matriks korelasi dihitung dari data yang sudah bersih. Plot korelasi dibuat menggunakan seaborn heatmap, menampilkan nilai korelasi antar fitur dalam format anotasi yang berwarna.



Gambar 12. Audio Feature Correlation Heatmap

Berdasarkan interpretasi nilai korelasi di atas, dapat disimpulkan bahwa terdapat beberapa hubungan yang kuat antara fitur-fitur audio.

- Lagu dengan danceability tinggi cenderung memiliki valence tinggi dengan nilai 0,34
 - Lagu dengan energy tinggi cenderung memiliki loudness tinggi dengan nilai 0,67
 - Lagu dengan loudness tinggi cenderung memiliki speechiness rendah dengan nilai 0
 - Lagu dengan acousticness tinggi cenderung memiliki instrumentalness rendah dengan nilai -0,03
- Correlation within Genres

```
# Assuming playlist_songs_no_outliers is already loaded as a
DataFrame

# Group by genre and calculate the median of numeric features
```

```

avg_genre_matrix = (
    playlist_songs_no_outliers
    .groupby('playlist_genre')
    .median(numeric_only=True)
    .reset_index()
)

# Extract the genres
genres = avg_genre_matrix['playlist_genre']

# Select numeric features and exclude 'mode'
features = avg_genre_matrix.drop(columns=['playlist_genre', 'mode'])

from sklearn.preprocessing import StandardScaler
# Scale the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Calculate the correlation matrix of the scaled median values
avg_genre_cor = np.corrcoef(scaled_features, rowvar=False)

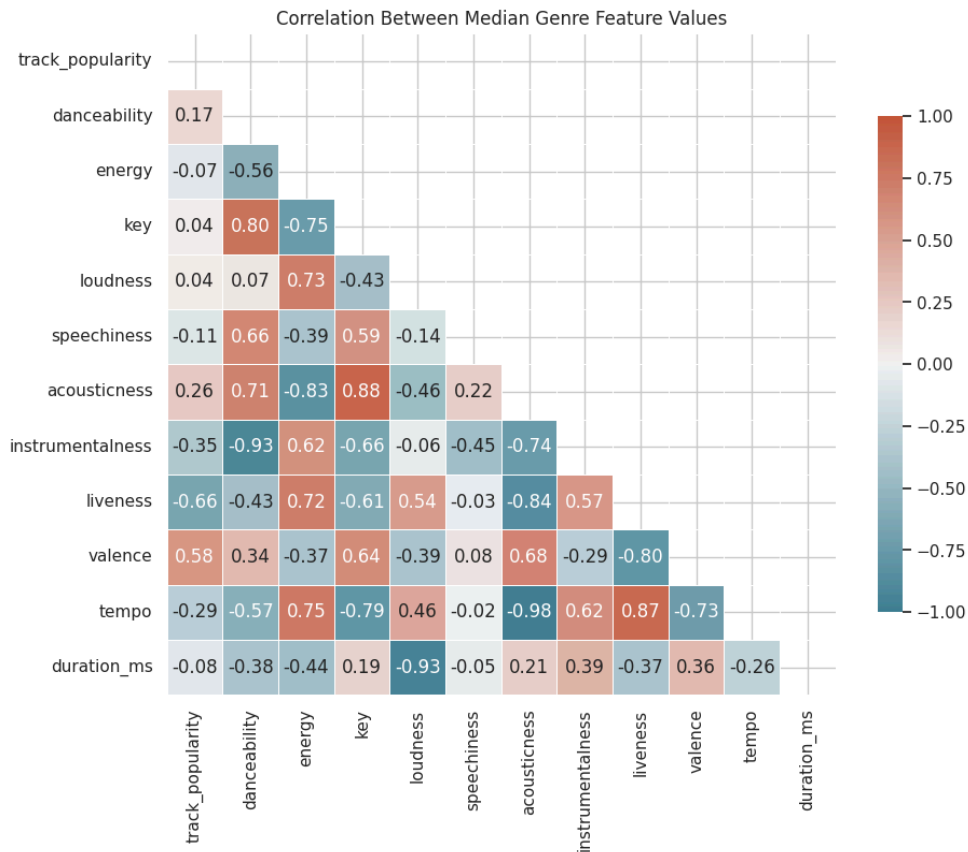
# Create a DataFrame for the correlation matrix with genres as labels
avg_genre_cor_df = pd.DataFrame(avg_genre_cor,
    columns=features.columns, index=features.columns)

# Print the correlation matrix
feature_names_reduced = [col for col in avg_genre_matrix.columns if
    col not in ['playlist_genre', 'mode']]

# Plotting the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(
    avg_genre_cor_df,
    annot=True,
    fmt='.2f',
    cmap=sns.diverging_palette(220, 20, as_cmap=True),
    vmax=1, vmin=-1,
    linewidths=0.5,
    cbar_kws={"shrink": .75},
    mask=np.triu(np.ones_like(avg_genre_cor_df, dtype=bool)) # Mask
    upper triangle
)
plt.title('Correlation Between Median Genre Feature Values')
plt.xticks(rotation=90) # Rotate x-axis labels for better
    readability
plt.yticks(rotation=0) # Ensure y-axis labels are horizontal
plt.show()

```

Kode ini menghitung korelasi antar fitur audio median per genre dalam dataset `playlist_songs_no_outliers`. Pertama, data dikelompokkan berdasarkan genre dan dihitung median dari fitur numeriknya. Fitur numerik dipilih dan fitur `mode` dikeluarkan. Fitur-fitur tersebut kemudian dinormalisasi menggunakan `StandardScaler` dari scikit-learn. Matriks korelasi dihitung dari fitur yang sudah dinormalisasi. Matriks korelasi ini kemudian diubah menjadi DataFrame dengan label fitur. Terakhir, matriks korelasi divisualisasikan menggunakan seaborn heatmap, dengan anotasi nilai korelasi, palet divergen untuk menunjukkan hubungan positif dan negatif, serta masking segitiga atas untuk mengurangi redundansi.



Gambar 13. Correlation Between Median Genre Feature Values Heatmap

Berdasarkan grafik korelasi di atas didapatkan interpretasi sebagai berikut:

1. Nilai korelasi positif tertinggi terdapat pada korelasi antara atribut acousticness dengan atribut key dengan nilai korelasi sebesar 0,88.
2. Nilai korelasi negatif tertinggi terdapat pada korelasi antara atribut tempo dengan atribut acousticness dengan nilai korelasi sebesar -0,98.
3. Nilai korelasi terendah terdapat pada korelasi antara atribut tempo dengan atribut speechiness dengan nilai korelasi sebesar -0,02.

- Preparing the Data for Training

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# Skala fitur numerik
playlist_songs_scaled = playlist_songs_no_outliers.copy()
numeric_features =
playlist_songs_no_outliers.select_dtypes(include=[np.number]).columns
scaler = StandardScaler()
playlist_songs_scaled[numeric_features] =
scaler.fit_transform(playlist_songs_no_outliers[numeric_features])

# Set seed untuk reproduktibilitas
np.random.seed(1234)

# Pisahkan data menjadi training dan testing sets
```

```

training_songs = np.random.choice(playlist_songs_scaled.index,
size=int(len(playlist_songs_scaled) * 0.80), replace=False)
train_set = playlist_songs_scaled.loc[training_songs,
['playlist_genre'] + feature_names_reduced]
test_set = playlist_songs_scaled.drop(training_songs).loc[:,
['playlist_genre'] + feature_names_reduced]

# Pisahkan fitur dan respon
train_resp = train_set.pop('playlist_genre')
test_resp = test_set.pop('playlist_genre')

# Konversi ke DataFrames (jika diperlukan)
train_set = pd.DataFrame(train_set, columns=feature_names_reduced)
test_set = pd.DataFrame(test_set, columns=feature_names_reduced)

# Optional: Konversi respon ke DataFrame
train_resp = pd.DataFrame(train_resp, columns=['playlist_genre'])
test_resp = pd.DataFrame(test_resp, columns=['playlist_genre'])

```

Berikut proses yang terjadi pada kode di atas:

1. Menskalakan fitur numerik dalam data.
2. Membagi data menjadi training set (80%) dan testing set (20%) secara acak.
3. Memisahkan fitur dan respon dalam masing-masing set.
4. Memastikan data dalam format DataFrame yang sesuai untuk proses pemodelan.

- **Modelling Decision Tree**

```

import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib
import matplotlib.pyplot as plt

# Assuming train_set and train_resp are already defined
# If not, you need to define them appropriately

# Set seed for reproducibility
np.random.seed(1111)

# Train the decision tree model
model_dt = DecisionTreeClassifier(random_state=1111)
model_dt.fit(train_set, train_resp)

# Define color palette
colors = {
    'edm': "#490B32",
    'latin': "#9A031E",
    'pop': "#FB8B24",
    'r&b': "#0F4C5C",
    'rap': "#5DA9E9",
    'rock': "#66717E"
}

# Get color for each node
def get_box_colors(clf, colors, class_names):
    box_colors = []

```

```

        for node in range(clf.tree_.node_count):
            if clf.tree_.children_left[node] ==
clf.tree_.children_right[node]: # leaf node
                class_idx = np.argmax(clf.tree_.value[node])
                class_name = class_names[class_idx]
                box_colors.append(colors.get(class_name, 'white'))
            else:
                box_colors.append('white')
        return box_colors

box_colors = get_box_colors(model_dt, colors, model_dt.classes_)

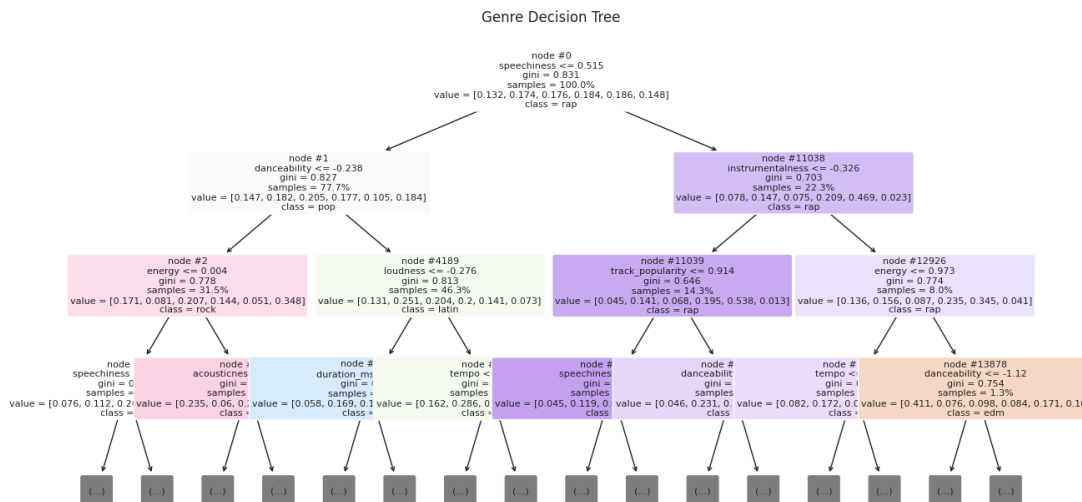
# Plot the decision tree
plt.figure(figsize=(15, 8))
plot_tree(model_dt,
          feature_names=feature_names_reduced,
          class_names=model_dt.classes_,
          filled=True,
          rounded=True,
          max_depth=3,
          fontsize=8,
          proportion=True,
          node_ids=True)

# Apply colors to nodes
ax = plt.gca()
for idx, node in
enumerate(ax.findobj(matplotlib.patches.FancyBboxPatch)):
    node.set_facecolor(box_colors[idx])

plt.title('Genre Decision Tree')
plt.show()

```

Kode ini melatih model pohon keputusan untuk klasifikasi genre musik dan memvisualisasikan hasilnya. Dataset pelatihan ('train_set' dan 'train_resp') digunakan untuk melatih model 'DecisionTreeClassifier' dari scikit-learn dengan seed untuk memastikan reproduktibilitas. Palet warna didefinisikan untuk setiap genre musik. Fungsi 'get_box_colors' menghasilkan warna untuk setiap node pohon berdasarkan kelas genre. Pohon keputusan divisualisasikan menggunakan 'plot_tree', dengan beberapa parameter untuk pengaturan tampilan, seperti nama fitur, nama kelas, pengisian warna, bentuk kotak yang dibulatkan, dan kedalaman maksimal yang ditampilkan. Node pohon kemudian diberi warna yang sesuai dengan genre masing-masing menggunakan matplotlib. Visualisasi ini memudahkan interpretasi keputusan model dalam mengklasifikasikan genre musik.



Gambar 14. Genre Decision Tree

Hasil visualisasi decision tree menunjukkan bagaimana fitur-fitur audio digunakan untuk mengklasifikasikan genre musik. Berikut adalah penjelasan keputusan yang diambil oleh pohon keputusan untuk masing-masing genre:

1. Pop : Kondisi 'speechiness ≤ 0.525 ' atau 'danceability ≤ -0.235 '. Jika salah satu dari kondisi ini terpenuhi, maka lagu diklasifikasikan sebagai genre 'Pop'.
2. Rap : Kondisi 'speechiness ≤ 0.525 ', 'instrumentalness ≤ -0.325 ', 'energy ≤ 0.662 ', 'danceability ≤ 0.97 ', 'acousticness ≤ 0.923 ' atau 'tempo ≤ 1.935 '. Jika semua kondisi di atas terpenuhi, maka lagu diklasifikasikan sebagai genre 'Rap'.
3. EDM : Kondisi 'speechiness ≤ 0.525 ', 'instrumentalness ≤ -0.325 ', 'energy ≤ 0.662 ', atau 'tempo ≤ -0.515 '. Jika semua kondisi di atas terpenuhi, maka lagu diklasifikasikan sebagai genre 'EDM'.
4. Rock : Kondisi 'speechiness ≤ -0.7 ', 'danceability ≤ -1 ' atau 'energy ≤ 0.004 '. Jika semua kondisi di atas terpenuhi, maka lagu diklasifikasikan sebagai genre 'Rock'.
5. Latin : Kondisi 'speechiness ≤ 0.525 ', 'danceability ≤ -0.235 ', 'duration_ms ≤ 0.447 ' atau 'energy ≤ -0.09 '. Jika semua kondisi di atas terpenuhi, maka lagu diklasifikasikan sebagai genre 'Latin'.
6. R&B : Kondisi 'speechiness ≤ 0.525 ', 'danceability ≤ -0.235 ', 'duration_ms ≤ 0.447 ', 'energy ≤ -0.624 '. Jika semua kondisi di atas terpenuhi, maka lagu diklasifikasikan sebagai genre 'R&B'.

Penjelasan ini memberikan gambaran tentang bagaimana pohon keputusan menggunakan fitur-fitur audio untuk membuat klasifikasi genre musik. Setiap kondisi merupakan keputusan yang diambil berdasarkan nilai-nilai fitur, dan jika semua kondisi untuk suatu jalur terpenuhi, maka lagu akan diklasifikasikan sesuai dengan genre yang telah ditentukan.

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier

# Prediksi menggunakan model decision tree
predict_dt = model_dt.predict_proba(test_set)

# Mendapatkan indeks dari probabilitas maksimum
```

```

max_id = predict_dt.argmax(axis=1)

# Mengonversi indeks ke label
pred = model_dt.classes_[max_id]

# Membuat DataFrame untuk membandingkan nilai sebenarnya dan nilai prediksi
compare_dt = pd.DataFrame({
    'true_value': test_resp.values.flatten(), # Konversi Series ke array 1D
    'predicted_value': pred,
    'model': 'decision_tree'
})

# Fungsi untuk menghitung akurasi model
def model_accuracy_calc(df, model_name):
    df['match'] = df['true_value'] == df['predicted_value']
    accuracy = df['match'].value_counts(normalize=True)
    return pd.DataFrame({'accuracy': [accuracy.get(True, 0)],
    'model': [model_name]})

# Menghitung akurasi untuk model decision tree
accuracy_dt = model_accuracy_calc(compare_dt, 'decision_tree')

print(accuracy_dt)

```

Kode diatas melakukan prediksi menggunakan model pohon keputusan (decision tree) dan menghitung akurasinya. Prediksi probabilitas kelas dibuat untuk data uji, kemudian indeks probabilitas maksimum dikonversi ke label kelas sebenarnya. Hasil prediksi dibandingkan dengan nilai sebenarnya dalam sebuah DataFrame. Akurasi model dihitung dengan menentukan persentase prediksi yang benar, dan hasil akurasi dicetak.

Output :

Mendapatkan nilai akurasi pada model decision tree :

```

      accuracy      model
0    0.40884  decision_tree

```

- Random Forest Classifier

```

df = playlist_songs
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
y = df['playlist_genre']
X =
df.drop(columns=['track_id','track_name','track_artist','track_popula
rity','track_album_id','track_album_name',
'track_album_release_date','playlist_name','playlist_id','playlist_ge
nre','playlist_subgenre'])
from sklearn.preprocessing import LabelEncoder

# Apply LabelEncoder to categorical features if needed
le = LabelEncoder()

```



```

for column in X.select_dtypes(include=['object']).columns:
    X[column] = le.fit_transform(X[column])

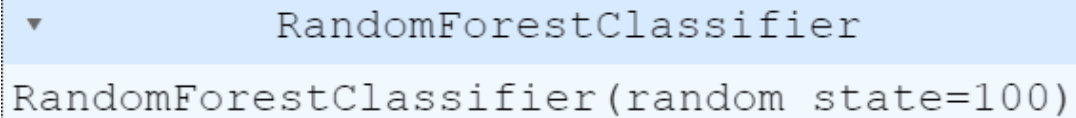
# Encode the target variable if needed
if y.dtype == 'object':
    y = le.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=100)

model_rf = RandomForestClassifier(random_state=100)

model_rf.fit(X_train, y_train)

```

Tujuan kode diatas adalah untuk melakukan klasifikasi genre playlist menggunakan algoritma random forest classifier dengan memisahkan fitur dan target, mengubah fitur kategorikal dalam X dengan mengubah menjadi bilangan bulat menggunakan LabelEncoder karena algoritma ini memerlukan input yang berupa bilangan bulat. Pembagian data latih dan data uji dengan perbandingan 70% data latih dan 30% data uji.



The screenshot shows a code cell in a Jupyter Notebook. The first line is a comment: `# Random Forest Classifier`. The second line is the class definition: `RandomForestClassifier`. The third line is the initialization of the model: `RandomForestClassifier(random_state=100)`. The cell is highlighted in blue.

Gambar 15. Random Forest Classifier

```

y_pred = model_rf.predict(X_test)

accuracy_rf = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy_rf}")

```

Output :

Didapatkan akurasi pada model random forest classifier yaitu
Accuracy: 0.5550253807106599

- Gradient Boosting with XGBoost

```

model_xgb = xgb.XGBClassifier(random_state=100,
use_label_encoder=False)

model_xgb.fit(X_train, y_train)

```

```

XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=None, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=None, max_leaves=None,
               min_child_weight=None, missing=nan, monotone_constraints=None,
               multi_strategy=None, n_estimators=None, n_jobs=None,
               num_parallel_tree=None, objective='multi:softprob', ...)

```

Gambar 16. XGBClassifier

```

# Make predictions
y_pred = model_xgb.predict(X_test)

# Evaluate the model
accuracy_xgb = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy_xgb}")

```

Output :

Didapatkan akurasi pada model XGBoost yaitu

Accuracy: 0.5536040609137056

- Feature Importance

```

# Extract feature importance scores
dt_importance = model_dt.feature_importances_

# Get the feature names
feature_names = X.columns

# Combine feature names with their importance scores
feature_importance_df = pd.DataFrame({'Feature': feature_names,
                                       'Importance': dt_importance})

# Sort the DataFrame by importance scores
feature_importance_df =
feature_importance_df.sort_values(by='Importance', ascending=False)

# Print feature names and their importance scores
for index, row in feature_importance_df.iterrows():
    print(f"Feature: {row['Feature']}, Score:
    {row['Importance']:.5f}")

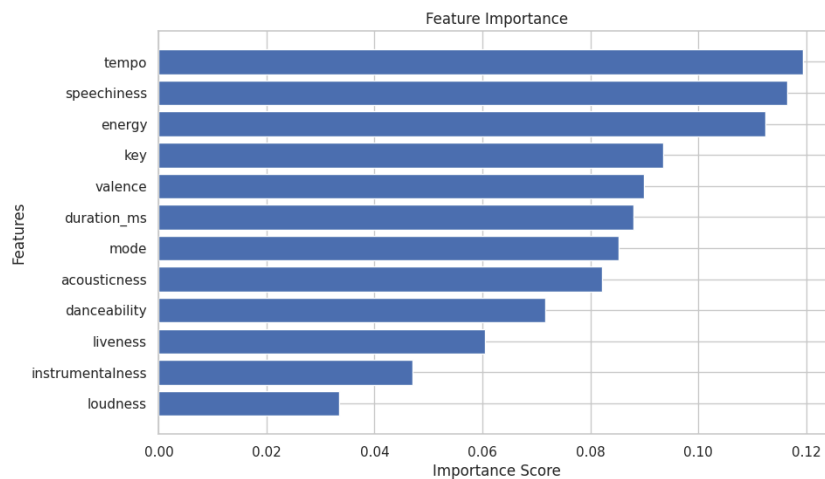
# Plot feature importance
plt.figure(figsize=(10, 6))
plt.barh(feature_importance_df['Feature'],
         feature_importance_df['Importance'])
plt.xlabel('Importance Score')
plt.ylabel('Features')
plt.title('Feature Importance')
plt.gca().invert_yaxis() # To display the highest importance at the
top
plt.show()

```

Tujuan dari kode di atas adalah untuk mengekstrak skor feature important, mengurutkan DataFrame berdasarkan skor important, dan memvisualisasikan skor fitur important dari model Decision Tree 'model_dt'

Output :

```
Feature: tempo, Score: 0.11947
Feature: speechiness, Score: 0.11655
Feature: energy, Score: 0.11244
Feature: key, Score: 0.09342
Feature: valence, Score: 0.08998
Feature: duration_ms, Score: 0.08807
Feature: mode, Score: 0.08520
Feature: acousticness, Score: 0.08215
Feature: danceability, Score: 0.07171
Feature: liveness, Score: 0.06041
Feature: instrumentalness, Score: 0.04707
Feature: loudness, Score: 0.03352
```



Gambar 17. Grafik Feature Importance

Dapat disimpulkan bahwa feature importance pada kategori 'tempo' (Fitur yang mengukur kecepatan musik) memiliki importance score yang paling tinggi dengan importance score yaitu sekitar 0,11 dibandingkan dengan features pada kategori 'loudness' (Fitur yang mengukur seberapa keras musik) dengan importance score yaitu sekitar 0,038.

- Accuracy Between Model

```
accuracy_results = {
    'Model': ['Decision Tree', 'Random Forest', 'XGBoost'],
    'Accuracy': [accuracy_dt, accuracy_rf, accuracy_xgb]
}

accuracy_df = pd.DataFrame(accuracy_results)
accuracy_df
```

Output :

No.	Model	Akurasi
1.	Decision Tree	0,40884
2.	Random Forest	0,555025
3.	XGBoost	0,553604

Tabel 1. Hasil Akurasi

Dari hasil 3 klasifikasi menggunakan Decision Tree, Random Forest, dan XGBoost. didapatkan nilai akurasi Decision Tree sebesar 0.40884, Random Forest sebesar 0.555025, dan XGBoost sebesar 0.553604 yang artinya bahwa dalam kasus ini, baik Random Forest maupun XGBoost memiliki performa yang lebih baik daripada Decision Tree dalam melakukan klasifikasi. Hal ini didukung oleh nilai akurasi yang lebih tinggi pada kedua model ensemble tersebut. Oleh karena itu, untuk dataset dan masalah klasifikasi yang sama, disarankan untuk mempertimbangkan penggunaan Random Forest atau XGBoost daripada Decision Tree untuk mendapatkan hasil yang lebih baik.

BAB V

KESIMPULAN

Penelitian berjudul "Klasifikasi Genre Musik dengan Menggunakan Decision Tree dan Random Forest Berdasarkan Fitur Audio dari Spotify" bertujuan untuk mengatasi masalah klasifikasi genre musik dengan memanfaatkan fitur audio yang diambil dari Spotify. Masalah utama yang diidentifikasi adalah menentukan model klasifikasi yang paling efektif dan akurat dalam memprediksi genre musik. Untuk itu, tiga model machine learning, yaitu Decision Tree, Random Forest, dan XGBoost, diterapkan dan dievaluasi. Hasil menunjukkan bahwa Random Forest memiliki akurasi tertinggi sebesar 0.555025, diikuti oleh XGBoost dengan akurasi 0.553604, sementara Decision Tree menunjukkan akurasi terendah sebesar 0.40884. Hal ini menunjukkan bahwa metode ensemble seperti Random Forest dan XGBoost lebih efektif dibandingkan metode Decision Tree tunggal dalam mengklasifikasikan genre musik berdasarkan fitur audio.

Selain mengukur akurasi, penelitian ini juga menilai kinerja model dalam hal waktu dan beban kerja memori selama pembentukan model. Random Forest, meskipun lebih akurat, membutuhkan waktu komputasi dan memori yang lebih besar dibandingkan Decision Tree. XGBoost, yang juga merupakan metode ensemble, menunjukkan performa serupa dengan Random Forest dalam hal akurasi tetapi lebih efisien dalam penggunaan memori dan waktu komputasi. Secara keseluruhan, penelitian ini menegaskan bahwa penggunaan model ensemble, khususnya Random Forest dan XGBoost, dapat memberikan hasil yang lebih akurat dalam klasifikasi genre musik, meskipun dengan biaya komputasi yang lebih tinggi dibandingkan dengan model Decision Tree tunggal.

DAFTAR PUSTAKA

- [1] Netti, S. Y. M., & Irwansyah, I. (2018). Spotify: Aplikasi Music Streaming untuk Generasi Milenial. *Jurnal Komunikasi*, 10(1), 1-16.
- [2] Widodo. (2016) Mengklasifikasi dan Menentukan Tajuk Subjek Bahan Perpustakaan. *Pustakawan Madya UPT Perpustakaan Universitas Sebelas Maret*, 1-13
- [3] Muzakir, A., & Wulandari, R. A. (2016). Model data mining sebagai prediksi penyakit hipertensi kehamilan dengan teknik decision tree. *Scientific Journal of Informatics*, 3(1), 19-26.
- [4] Devella, S., Yohannes, Y., & Rahmawati, F. N. (2020). Implementasi Random Forest untuk klasifikasi motif Songket Palembang berdasarkan SIFT. *JATISI (Jurnal Tek. Inform. dan Sist. Informasi)*, 7(2), 310–320
- [5] Naldy, E. T., & Andri, A. (2021). Penerapan Data Mining Untuk Analisis Daftar Pembelian Konsumen Dengan Menggunakan Algoritma Apriori Pada Transaksi Penjualan Toko Bangunan MDN. *Jurnal Nasional Ilmu Komputer*, 2(2), 89-101.
- [6] Triono, A., Budi, A. S., & Abdillah, R. (2023). Implementasi Peretasan Sandi Vigenere Chipper menggunakan Bahasa Pemrograman Python. *JOCITIS-Journal Science Informatica and Robotics*, 1(1), 01-09.
- [7] Laiya, J. W., & Manueke, S. (2022). Pentingnya Akurasi Data Dalam Mempertahankan Kinerja Perusahaan Pada PT. Massindo Solaris Nusantara. *Jurnal MABP*, 4(2), 38-51
- [8] Alhamdi, M. D. A., & Astuti, W. (2023). Peramalan Kebutuhan Obat Menggunakan XGBoost Studi Kasus pada Rumah Sakit XYZ: Forecasting Drug Needs Using XGBoost: Case Study at XYZ Hospital. *Indonesian Journal of Computer Science*, 12(5).
- [9] Fadhillah, K. (2021, January 18). *Mengenal Decision Tree dan Peran Pentingnya dalam Pengambilan Keputusan*. Jojonomic. Retrieved June 4, 2024, from <https://www.jojonomic.com/blog/decision-tree/>
- [10] Heng, R. (2024, February 15). *Statistik Spotify 2024 – Data tentang Artis, Penggunaan & Pendapatan*. Business 2 Community. Retrieved June 4, 2024, from <https://www.business2community.com/id/statistics/spotify>
- [11] Safira, S. (2023, July 3). *Peran Musik dalam Kehidupan Manusia: Mengapa Musik Begitu Penting?* hipwee. Retrieved June 4, 2024, from <https://www.hipwee.com/narasi/peran-musik-dalam-kehidupan-manusia-mengapa-musik-begitu-penting/>

- [12] Trivusi. (2022, September 17). *Algoritma Random Forest: Pengertian dan Kegunaannya*. Trivusi. Retrieved June 4, 2024, from <https://www.trivusi.web.id/2022/08/algoritma-random-forest.html>
- [13] Tianqi Chen dan Carlos Guestrin, 2016. *XGBoost: Sistem Pendorong Pohon yang Dapat Diskalakan*, <https://arxiv.org/pdf/1603.02754.pdf>

LAMPIRAN

Link Dataset : [Spotify_Dataset.csv](#)

Link Code Script : [GoogleColaboratoryPJBLDataMiningKelompok3](#)