

Associate Data Science

Bengkel Koding

2023-07-31

Table of contents

Pengantar	9
I Konsep	10
1 Data Science Tools	12
1.1 Python versi 3.10.10	12
1.2 Development Environment	12
1.2.1 VSCode Notebook	13
1.2.2 Jupyter Notebook	13
1.3 Install Libraries	14
2 Konsep Data Science	15
2.1 Workflow	15
2.2 Mengapa Data Scientist Dibutuhkan?	16
2.3 Apa Yang Dimaksud Big Data?	16
3 Pengembangan Model Aplikasi Data Science	17
3.1 Tahap Pengembangan Sistem AI Berbasis Big Data:	17
3.1.1 Pelatihan	18
3.1.2 Penggunaan	18
3.2 Software Development Life Cycle (SDLC)	18
3.3 Jenis-Jenis SDLC	20
3.3.1 Agile	20
3.3.2 Waterfall	20
3.3.3 Prototype	21
3.4 Rapid Application Development (RAD)	21
3.5 Kesimpulan	22
4 Data Understanding	23
4.1 Mengapa Kita Perlu Data Understanding	23
4.2 Data Understanding Documentation	24
5 Sumber, Susunan, Tipe dan Model Data	26
5.1 Sumber Data	27
5.1.1 Internal	27

5.1.2	External	27
5.2	Susunan Data	27
5.2.1	Structured	29
5.2.2	Unstructured	29
5.3	Tipe Data	29
5.3.1	Tipe Data Berdasarkan Sifatnya	29
5.3.2	Tipe Data Berdasarkan Cara Pengumpulan	30
5.3.3	Tipe Data Berdasarkan Seri Waktu	30
6	Pengambilan Data	31
6.1	Web Scraping	31
6.2	Application Program Interface (API)	31
6.3	Manual (Kaggle.com)	32
6.4	Direct Database	32
7	Telaah Data	35
7.1	Apa itu telaah data?	35
7.2	Mengapa perlu telaah data?	35
7.3	Bentuk Data	35
7.4	Sumber Data	36
7.5	Daftar Sumber Data Daring	36
7.6	Tipe Data	36
7.6.1	Berdasarkan sifat	36
7.6.2	Berdasarkan waktu	37
7.7	Pengambilan Data	37
7.8	Library Pandas	39
7.8.1	Memuat data ke Pandas	39
7.8.2	Menampilkan Data	40
7.8.3	Melihat Tipe Data	40
7.8.4	Deskripsi statistik data	40
7.8.5	Fungsi statistik dalam Pandas	41
7.8.6	Mentukan Outlier	41
7.8.7	Nilai Unik	42
7.8.8	Analisis dengan groupby	42
7.8.9	Korelasi	43
8	Validasi Data	44
8.1	Tugas Validasi Data	44
8.2	Manfaat Validasi Data	44
8.3	Laporan Dokumentasi Data Validasi	44
8.4	Validasi vs Verifikasi	44
8.5	Tahapan kritis dalam validasi	45
8.6	Teknik Validasi Data	45

8.7	Validasi Dengan Pandas	45
8.8	Visualisasi Data	45
8.8.1	Jenis Visualisasi Data	46
8.8.2	Library Visualisasi Data	46
8.8.3	Korelasi & Causation	47
8.8.4	Analisis Korelasi (Heatmap)	47
8.8.5	Histogram	47
8.8.6	Bar Chart	49
8.8.7	Pie Chart	49
9	Menentukan Objek atau Memilih Data	51
9.1	Menentukan Sumber Data	51
9.2	Menelaah Susunan Data	51
9.3	Menentukan Tipe dan Model Data yang dimiliki	51
9.4	Mengambil Data	51
9.5	Menelaah Data dalam Machine Learning	52
9.6	Contoh Source Code	52
10	Cleaning Data	54
10.1	Data Cleaning	54
10.2	Cara Mengatasi Missing Value	54
	- Menghapus baris atau kolom yang memiliki missing value	54
	- Mengisi nilai kosong dengan nilai rata-rata atau median	56
	- Mengisi nilai kosong dengan nilai yang sering muncul	56
10.3	Menangani Outlier	57
	- Menghapus Outlier	57
	- Menggantikan nilai outlier dengan nilai lain seperti nilai rata-rata atau median.	58
	- Mengisi nilai kosong dengan nilai yang sering muncul	59
10.4	- Menangani Format yang Tidak Konsisten	59
	- Mengubah format data menjadi format yang konsisten seperti mengubah for- mat tanggal menjadi format yang sama	60
	- Memperbaiki data yang salah ketik atau typo	60
10.5	Menangani Malformed Record	61
	- Menghapus record yang tidak sesuai dengan format atau struktur yang di- harapkan	61
	- Mengubah record yang tidak sesuai dengan format atau struktur yang diha- rakan	62
10.6	Kesimpulan	63
11	Transformasi Data	64
11.1	RekayasaFitur	64
11.1.1	Hands On Coding	64

11.2	Imputasi	66
11.2.1	Jenis-jenis Imputasi	66
11.2.2	Missing Not At Random	69
11.2.3	Teknik-Teknik Imputasi	70
11.3	Handling Outlier	82
11.3.1	Deteksi Outlier	82
11.3.2	Kategori Outlier	83
11.3.3	Mengatasi Outlier	84
11.4	Dokumentasi Fitur	92
11.4.1	Nama Fitur	93
11.4.2	Definisi Fitur	93
11.4.3	Tipe Data	94
11.4.4	Skala Data	94
11.4.5	Sumber Data	94
11.4.6	Keterangan	94
12	Pelabelan Data	95
12.1	Data Training	95
12.2	Metode Pelabelan Data	97
12.2.1	Internal Labeling	97
12.2.2	Synthetic labeling	97
12.2.3	Programmatic labeling	98
12.2.4	Outsourcing	98
12.2.5	Crowdsourcing	98
12.3	Penggunaan Data Labeling dalam Pengolahan Citra	99
12.3.1	Image Classification	100
12.3.2	Image Segmentation	101
12.3.3	Object Detection	102
12.3.4	Pengolahan Citra - Hands On Coding	102
12.4	Penggunaan Data Labeling dalam Natural Language Processing	105
12.4.1	Sentiment Analysis	106
12.4.2	Named Entity Recognition (NER)	107
12.4.3	Part of Speech Tagging (POS-tagging)	107
12.4.4	Hands On Coding	108
12.5	Analisa Kualitas dan Akurasi Data untuk Pelabelan	109
12.5.1	Definisi Data yang Berkualitas	109
12.5.2	Faktor-faktor yang mempengaruhi kualitas proses pelabelan data	110
12.5.3	Metode QA untuk mengukur kualitas data	110
12.5.4	Hands On Coding - Cronbach Alpha Test	111
12.6	Keamanan Pelabelan Data	112
12.6.1	Resiko Keamanan Outsourcing Data Labeling	112
12.6.2	Tiga area yang perlu menjadi perhatian untuk menjaga keamanan dokumentasi	113

13 Pemodelan Data Science	114
13.1 Pendahuluan	114
13.2 Tahap Pemodelan Machine Learning	114
13.2.1 Data Preparation	114
13.2.2 Model Training	115
13.2.3 Parameter Tuning	115
13.2.4 Saving Model	116
14 Evaluasi Data	117
14.1 Apa itu Evaluasi Model?	117
14.2 Metrik Evaluasi	117
14.2.1 Akurasi	117
14.2.2 Presisi dan Recall	119
14.2.3 F1-Score	120
14.2.4 Pahami Code berikut	121
14.2.5 Kurva ROC dan AUC	122
14.2.6 Amati dan pahami code berikut	124
15 Deployment	127
15.1 Pendahuluan	127
15.2 Instalasi Streamlit	127
15.3 Membuat Aplikasi Web dengan Streamlit	127
15.4 API Streamlit	128
15.4.1 Page Config	128
15.4.2 Write	128
15.4.3 Sidebar	128
15.4.4 Input User	129
15.4.5 Handling Input User	130
15.4.6 Pengecekan Dataframe	131
15.4.7 Prediksi	132
15.4.8 Hasil akhir	135
II Studi Kasus	136
Studi Kasus 1	138
Project Klasifikasi Sentimen Analisis	138
Klasifikasi Sentimen Analisis	138
Mencari Data Manual dengan Scraping	138
Mencari Data dengan mengunjungi website	141
Sentimen Analisis menggunakan data scraping youtube	141

Studi Kasus 2	145
Project Regresi Hotel Yogyakarta	145
Deskripsi	145
Tentang data	145
hotel_yogyakarta	145
hotel_room_yogyakarta	146
Data analysis	146
Import library	146
Load data	146
Menghapus Kolom	148
Merge Data	148
Target Processing	149
Validasi Data	151
Encoding Data	158
Export Data ke CSV	161
Data Analisis	161
Pemodelan Machine Learning	163
Studi Kasus 3	165
Project Clustering - Spotify Playlist Clustering	165
Project Overview	165
Spotify	165
Python	165
Streamlit	168
Clustering	168
K-Means	168
Application Flow	170
User Flow	170
Data Flow	170
Function Flow	170
Membuat Akun Spotify Developer	170
Daftar Sebagai Spotify Developer	170
Pergi ke dashboard spotify dan tekan create app	171
Isi form dengan data	173
Klik app yang sudah dibuat	173
Pergi ke settings, cari ClientID dan ClientSecret	173
Buat Repositori di Github	173
Login ke Github	173
Buat Repository Baru	175
Simpan Alamat Git (git remote)	175
Open Folder di Code Editor	175
Buat File Baru	175
Push File ke Repository	175

Pengambilan Dataset	177
Buka File Notebook	177
Import Library yang Dibutuhkan	177
Masukkan variable client_id, client_secret, dan playlist_url	181
Buat fungsi getToken()	181
Buat fungsi getAuthHeader()	182
Buat fungsi getAudioFeatures()	182
Buat fungsi getPlaylistItems()	183
Data Preprocessing	187
Clustering Menggunakan K-Means	190
Deploy ke Streamlit	191
Buat file bernama streamlit_app.py	191
Tambahkan fungsi dataProcessing()	192
Buat file requirements.txt	193
Upload ke github repository	194
Buat Akun Streamlit	194
Create new app, isi form	194
Kesimpulan	196

Pengantar

Selamat datang dalam modul pembelajaran online tentang Data Science. Modul ini dirancang untuk memberi Anda pemahaman yang komprehensif tentang konsep, metode, dan aplikasi yang terkait dengan Data Science. Dalam era digital yang terus berkembang, Data Science telah menjadi elemen kunci dalam pengambilan keputusan yang informasional dan cerdas di berbagai bidang, mulai dari bisnis hingga penelitian ilmiah.

Diharapkan dengan mengikuti modul ini, Anda akan dapat memahami dasar Data Science, menguasai metode analisis data, dan mengembangkan keterampilan pemrograman Python untuk mengolah data dan membuat model machine learning.

Part I

Konsep

Pada kursus Data Science berisi tentang modul-modul, sebagai berikut:

1. Data Science Tools
2. Data Understanding
3. Telaah Data
4. Validasi Data
5. Menentukan Objek atau Memilih Data
6. Cleaning Data
7. Transformasi Data
8. Pemodelan Data Science
9. Evaluasi Data
10. Deployment

1 Data Science Tools

Sama seperti pekerjaan pada umumnya, seorang Data Scientist juga membutuhkan alat-alat untuk melakukan pekerjaannya. Namun alat yang dimaksud adalah sekumpulan software yang digunakan untuk pekerjaan Data Scientist sehari-hari. Perlu diingat bahwa semua tools harus sudah diinstall sebelum pelaksanaan materi koding. Mari kita pelajari tools yang akan kita gunakan!

1.1 Python versi 3.10.10

- Buka browser, kunjungi <https://www.python.org/downloads/windows/>, cari bagian Python 3.10.10 - Feb 8, 2023, lalu pilih Download Windows Installer.
- Jalankan file yang di download, tunggu sampai proses installing selesai.
- Buka Command Prompt, lalu jalankan command “python –version”

Jika hasil dari command tersebut adalah “3.10.10” maka instalasi python sudah selesai.

```
C:\Users\Administrator>python --version
Python 3.10.10
```

Figure 1.1: Gambar3. Python di cmd

1.2 Development Environment

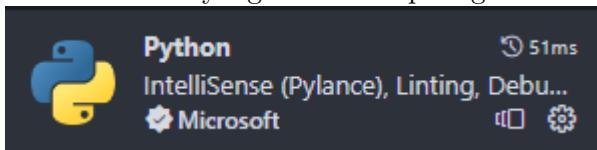
Environment adalah lingkungan yang kita tinggali selama proses koding. Di course ini kita bisa menggunakan salah satu dari tiga aplikasi, yaitu VSCode, Jupyter dan Google Colab. Cukup pilih salah satu dari ketiga opsi Kami merekomendasikan untuk menggunakan Jupyter notebook. Untuk alasannya, silahkan simak slide selanjutnya

1.2.1 VSCode Notebook

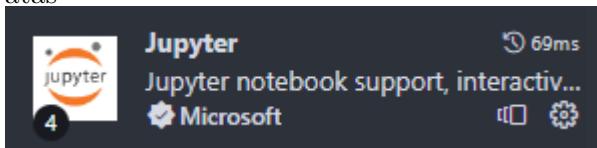
VSCode adalah sebuah code editor multifungsi yang cukup ringan dan handal. Kelebihan dari VSCode adalah fleksibilitas dari VSCode. Anda dapat mengkustomisasi tema, menambah ekstensi, dan mengubah semua settingan yang ada di VSCode. Anda mempunyai kendali penuh. Kekurangan dari VSCode adalah terkadang settingan VSCode terlalu kompleks untuk digunakan.

Berikut langkah-langkah untuk menginstalasi VScode Notebook :

- Buka browser, kunjungi <https://code.visualstudio.com/download#> pilih salah satu link yang berada di System Installer.
- Jalankan file yang di download, tunggu sampai proses installing selesai.
- Selanjutnya kita akan menginstall beberapa ekstensi.
- Buka VSCode, pergi ke bagian Extensions, (ctrl+shift+x) lalu ketik “Python”, lalu install ekstensi yang muncul di paling atas.



- Masih di tab Extensions, ketik “Jupyter”, lalu instal ekstensi yang muncul di paling atas



- Reload VSCode

1.2.2 Jupyter Notebook

Jupyter Notebook adalah aplikasi yang digunakan untuk memproses dan menampilkan teks, kode, dan data secara bersamaan. Jupyter sangat cocok untuk data science karena Jupyter sangat ringan, mudah di install, dan mudah untuk digunakan. Oleh karena itu, kita merekomendasikan anda untuk menggunakan Jupyter Notebook untuk pelatihan ini.

Berikut langkah-langkah untuk menginstalasi Jupyter Notebook :

- Buka Windows start menu, ketik “cmd”
- Klik kanan pada item command prompt, lalu pilih run as administrator

- Di dalam command prompt, ketik “pip install jupyterlab”
- Tunggu sampai proses selesai
- Ketik “jupyter-lab” untuk menjalankan aplikasi jupyter

1.3 Install Libraries

- Buka terminal (ctrl+shift+~) lalu ketik “pip install [package name]” Contoh : “pip install pandas”, dimana pandas adalah sebuah nama dari package Untuk membuka terminal di Jupyter Notebook, buka launcher baru (ctrl+shift+L) lalu pilih terminal.
- Package name diisi dengan list dibawah ini
 - numpy
 - scipy
 - pandas
 - matplotlib
 - seaborn
 - scikit

2 Konsep Data Science

Data science adalah bidang yang berkaitan dengan ekstraksi pengetahuan dan wawasan dari data. Data science menggabungkan konsep dan metode dari berbagai disiplin ilmu seperti statistika, matematika, komputer, dan pemodelan untuk menganalisis dan menginterpretasikan data secara sistematis.

2.1 Workflow

Di proses pekerjaan pengolahan data, Data Scientist bekerja untuk mengembangkan model terbaik dari data untuk menjawab permasalahan bisnis. Data Scientist mengambil data dengan bantuan Data Engineer dan DevOps Engineer, lalu menganalisa data tersebut. Hasil analisa data dikembalikan ke mereka atau dipresentasikan ke anggota lain.

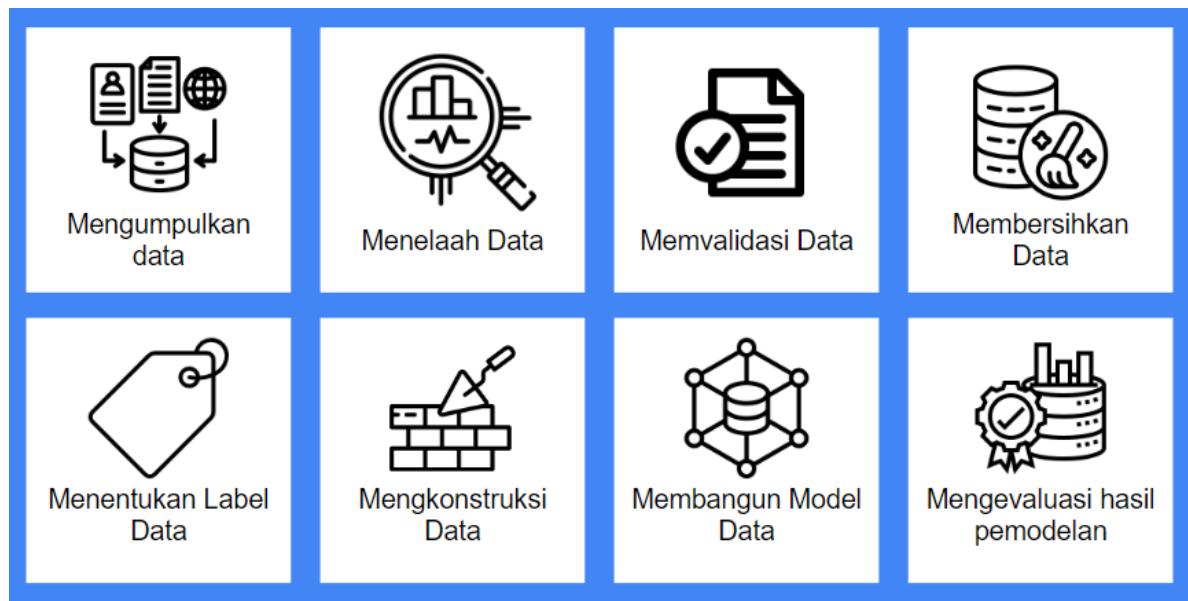


Figure 2.1: Gambar1. Contoh workflow data scientist

2.2 Mengapa Data Scientist Dibutuhkan?

Aplikasi Modern mempunyai skala yang jauh lebih besar dari aplikasi-aplikasi sebelumnya. Dalam satu waktu, jumlah user bisa mencapai ratusan ribu, bahkan sampai jutaan. Masing-masing user menghasilkan beberapa request ke server untuk mengambil data, dan menghasilkan data tersendiri. Hasilnya, data yang dibuat, disimpan, dan diproses sangatlah besar. Fenomena ini dinamakan: **BIG DATA**

2.3 Apa Yang Dimaksud Big Data?

Velocity	Volume	Variety
kecepatan data yang dihasilkan	Jumlah data yang diakumulasikan	Jenis/ragam data yang bermacam-macam
		

Big data mempunyai karakteristik yang dinamakan 5V

3 Pengembangan Model Aplikasi Data Science

Tahukah anda bahwa 70% pengembangan software gagal? Data science adalah bagian dari pengembangan software yang mempunyai tingkat kegagalan paling tinggi di angka 80% tingkat kegagalan.

GARTNER ESTIMATED	THROUGH 2020	THROUGH 2022	EXECUTIVE SURVEY
85% of big data projects fail (2017). The initial estimation was 60% (GARTNER 2016)	80% of AI projects will remain alchemy, run by wizards whose talents will not scale in the organization. (GARTNER 2018)	20% of analytic insights will deliver business outcomes. (GARTNER 2018)	77% respondents say that “business adoption” of big data and AI initiatives continues to represent a challenge for their organizations (NEWVANTAGE PARTNERS 2019)

Banyak sekali masalah yang dihadapi di dalam proses pengembangan software. Mulai dari perubahan requirement, maintenance yang kurang, hingga dokumentasi yang tidak lengkap. Oleh karena itu, pemahaman terhadap pengembangan model sangatlah penting, agar semua usaha kita untuk membuat sebuah aplikasi tidak sia sia, dan dapat membantu orang banyak.

3.1 Tahap Pengembangan Sistem AI Berbasis Big Data:

- **Pelatihan**, Adalah tahap dimana kita membangun sebuah model dari nol. Proses pelatihan meliputi pemilihan dataset, pemilihan algoritma, penyetelan parameter, testing, dan lain lain
- **Penggunaan**, Ketika model sudah dibuat dan siap untuk digunakan, model akan di deploy di internet agar dapat digunakan oleh semua orang.

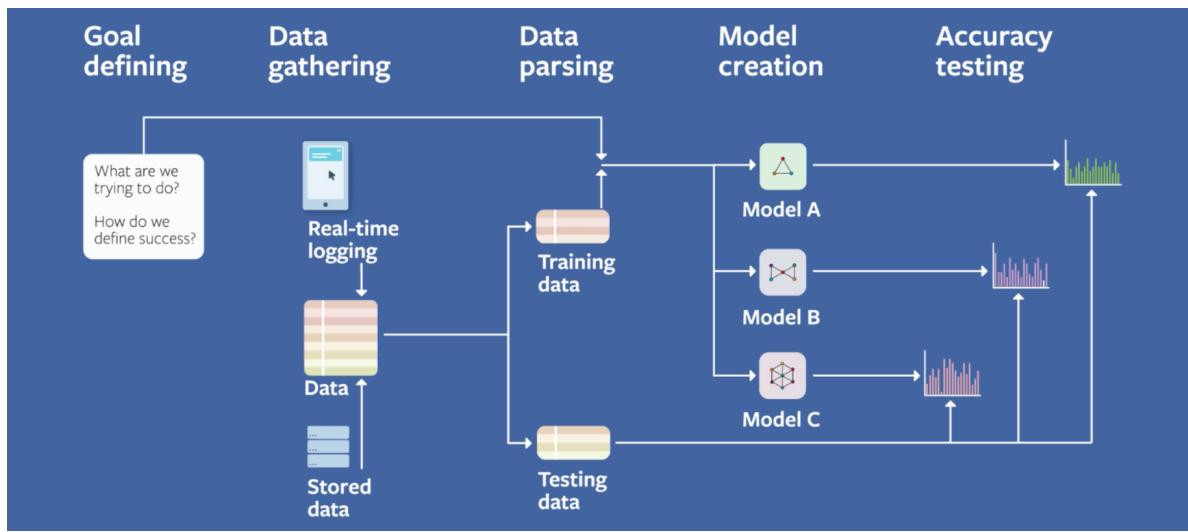


Figure 3.1: Gambar7. Flow dari proses training model machine learning

3.1.1 Pelatihan

3.1.2 Penggunaan

3.2 Software Development Life Cycle (SDLC)

Metode SDLC (Software Development Life Cycle) adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem rekayasa perangkat lunak.

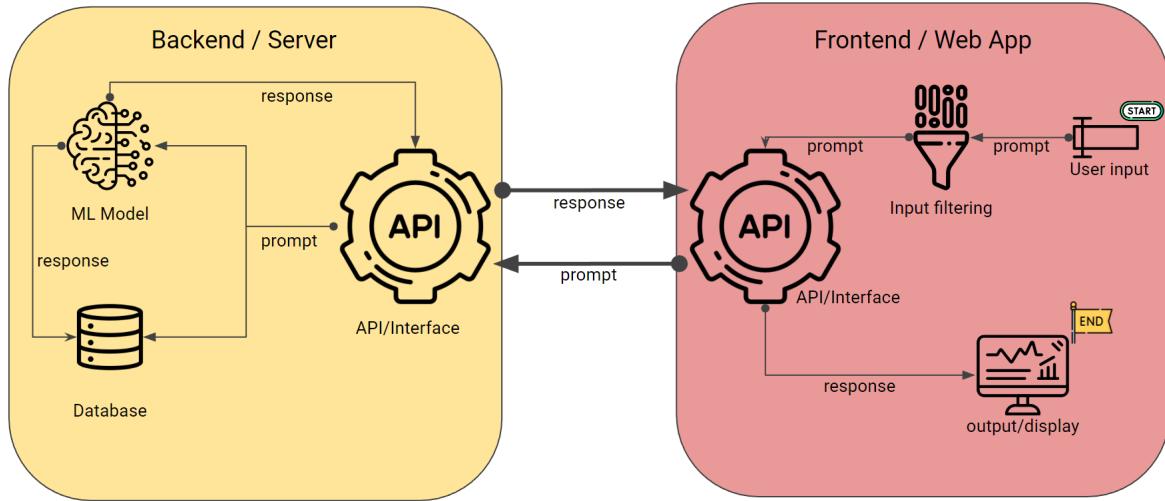


Figure 3.2: Gambar8. Flow dari proses penggunaan model machine learning



Proses logika yang digunakan oleh seorang analis sistem untuk mengembangkan sebuah sistem informasi yang melibatkan requirements, validation, training dan pemilik sistem proses yang memproduksi sebuah software dengan kualitas setinggi-tingginya tetapi dengan biaya yang serendah-rendahnya (Stackify)

3.3 Jenis-Jenis SDLC

Berikut adalah beberapa jenis dari SDLC:

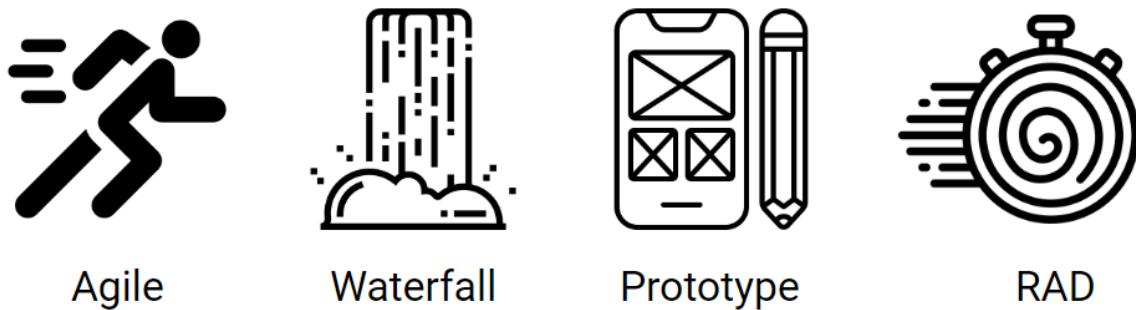


Figure 3.3: Gambar10. Jenis jenis SDLC

3.3.1 Agile

Metode pengembangan Agile adalah pendekatan kolaboratif dan iteratif dalam pengembangan perangkat lunak. Dalam metode ini, pengembangan dilakukan dalam iterasi kecil yang disebut “sprints”

Kelebihan

- Fleksibilitas tinggi - Kolaborasi erat antar tim - Aplikasi di-update secara rutin - Fokus kepada kebutuhan pengguna

Kekurangan

- Sulitnya merencanakan anggaran dan timeline - Memerlukan tingkat disiplin yang tinggi dan pengawasan konstan - Kurang efektif untuk project skala besar

3.3.2 Waterfall

Metode pengembangan Waterfall adalah pendekatan linier dan berurutan dalam pengembangan perangkat lunak. Dalam metode ini, setiap fase pengembangan (analisis kebutuhan, de-

sain, implementasi, pengujian, dan pemeliharaan) dilakukan secara berurutan, di mana setiap fase harus selesai sebelum melanjutkan ke fase berikutnya.

Kelebihan

- Struktur yang jelas
- Dokumentasi lengkap dan rapi
- Cocok untuk proyek yang kebutuhannya tidak berubah-ubah

Kekurangan

- Kurang fleksibel ketika menghadapi kebutuhan yang berubah
- Keterbatasan dalam kolaborasi dan feedback
- Risiko terlambatnya mendeteksi kesalahan/glitch

3.3.3 Prototype

Metode pengembangan prototyping adalah pendekatan dalam pengembangan perangkat lunak di mana prototipe perangkat lunak awal dibuat dan digunakan untuk mengumpulkan umpan balik dari pengguna dan pemangku kepentingan. Prototipe tersebut dapat berupa versi sederhana atau parsial dari perangkat lunak yang akhir.

Kelebihan

- Feedback didapatkan lebih awal
- Klarifikasi kebutuhan lebih jelas
- Meningkatkan kolaborasi tim

Kelemahan

- Membutuhkan banyak waktu dan sumber daya
- Potensi kesalahan di aspek desain dan efisiensi besar
- Kesulitan dalam timing untuk berhenti mengembangkan prototipe

3.4 Rapid Application Development (RAD)

Metode pengembangan Rapid Application Development (RAD) adalah pendekatan iteratif dan kolaboratif dalam pengembangan perangkat lunak yang menekankan kecepatan, fleksibilitas, dan keterlibatan pengguna. Dalam metode ini, pengembangan dilakukan dalam siklus pendek yang disebut “iterasi” dan melibatkan partisipasi aktif dari pengguna.

Kelebihan

- Waktu pengembangan cepat
- Keterlibatan pengguna secara sering dan langsung
- Fleksibilitas tinggi
- Tingginya kolaborasi terhadap developer dan pengguna

Kelemahan

- Ketergantungan terhadap pengguna tinggi
- Sulit untuk diskalakan
- Kualitas dan efisiensi arsitektur rendah

3.5 Kesimpulan

Kesimpulannya, Di dunia software development, tidak ada metode yang lebih baik atau buruk, hanya ada pengorbanan atau ‘tradeoff’.

Untuk melakukan proses development yang baik dan efisien, kita harus memahami kebutuhan, kondisi, dan sumber daya yang kita punya. Pemahaman yang baik memungkinkan kita untuk memilih tradeoff mana yang akan diambil. Setelah itu kita menyesuaikan tradeoff yang kita pilih dengan jenis SDLC yang sudah ada.

4 Data Understanding

Di dunia bisnis, proses data understanding dilakukan setelah problem bisnis didefinisikan. Tujuan utama dari data understanding adalah untuk memberikan gambaran utuh terhadap sebuah data. Setelah Data Understanding, kita dapat melanjutkan proses persiapan data.

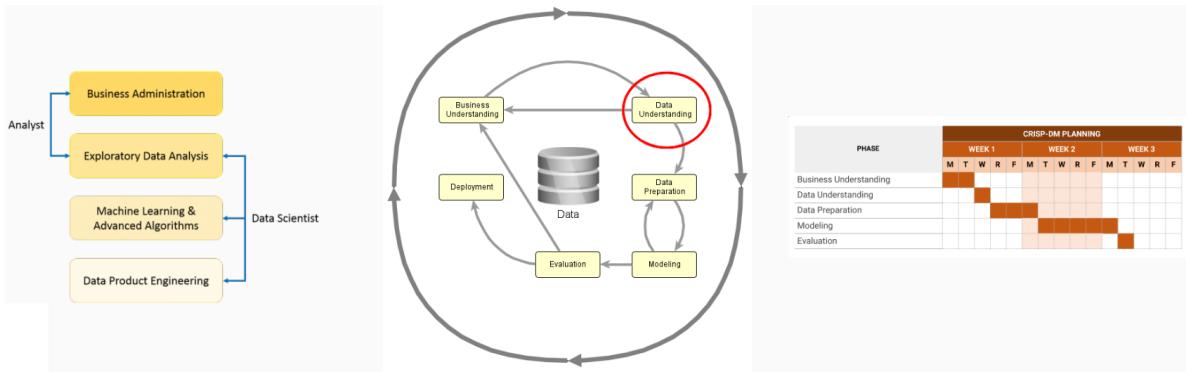


Figure 4.1: Gambar1. Data Understanding

4.1 Mengapa Kita Perlu Data Understanding

Di bidang machine learning, ada sebuah mantra yang semua data scientists harus ikuti; Garbage in, garbage out. Artinya, jika data yang dimasukkan ke model itu jelek, maka apapun yang terjadi, hasil dari model akan selalu jelek.

Semua data yang kita dapatkan belum tentu bagus dan bisa kita olah lebih lanjut karena:

- Maksud dan tujuan data berbeda
- Keadaan data terlalu terpisah, atau terlalu terintegrasi
- Kekayaan atau value data berbeda beda
- Keandalan atau reliability data berbeda beda

4.2 Data Understanding Documentation

Data Understanding Documentation adalah sebuah prosedur yang harus kita ikuti untuk mempelajari data yang akan kita gunakan dan menulis hasil pekerjaan kita secara terstruktur agar dapat dicerna oleh orang lain dengan mudah.

Data Understanding Documentation mempunyai 4 komponen, yaitu:

- Collection
 - Data Sources
 - Data Attribute
 - Data Count
 - Data Merge
- Description
 - Describing data
 - Data Quantity
 - Data Type
 - Key Attributes
 - Priority of Attributes
- Exploration
 - Hypothesis of Data
 - Attribute selection
 - Exploration of data characteristics
 - Compare with goal
 - Identify subset of data
- Quality
 - Missing values

- Data errors/corruptions
- Inconsistencies
- Outliers
- Noise

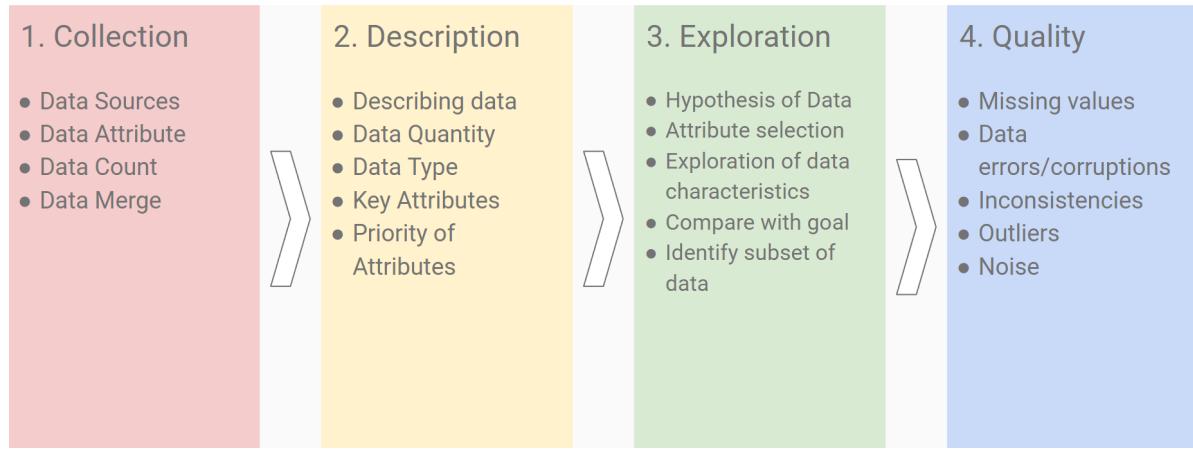


Figure 4.2: Gambar2. Komponen Data Understanding

5 Sumber, Susunan, Tipe dan Model Data

Data mempunyai berbagai jenis, bentuk, ukuran dan nilai. Sebagai Data Scientist, kita harus mengetahui sifat, jenis, asal, dan bentuk dari data yang kita olah.

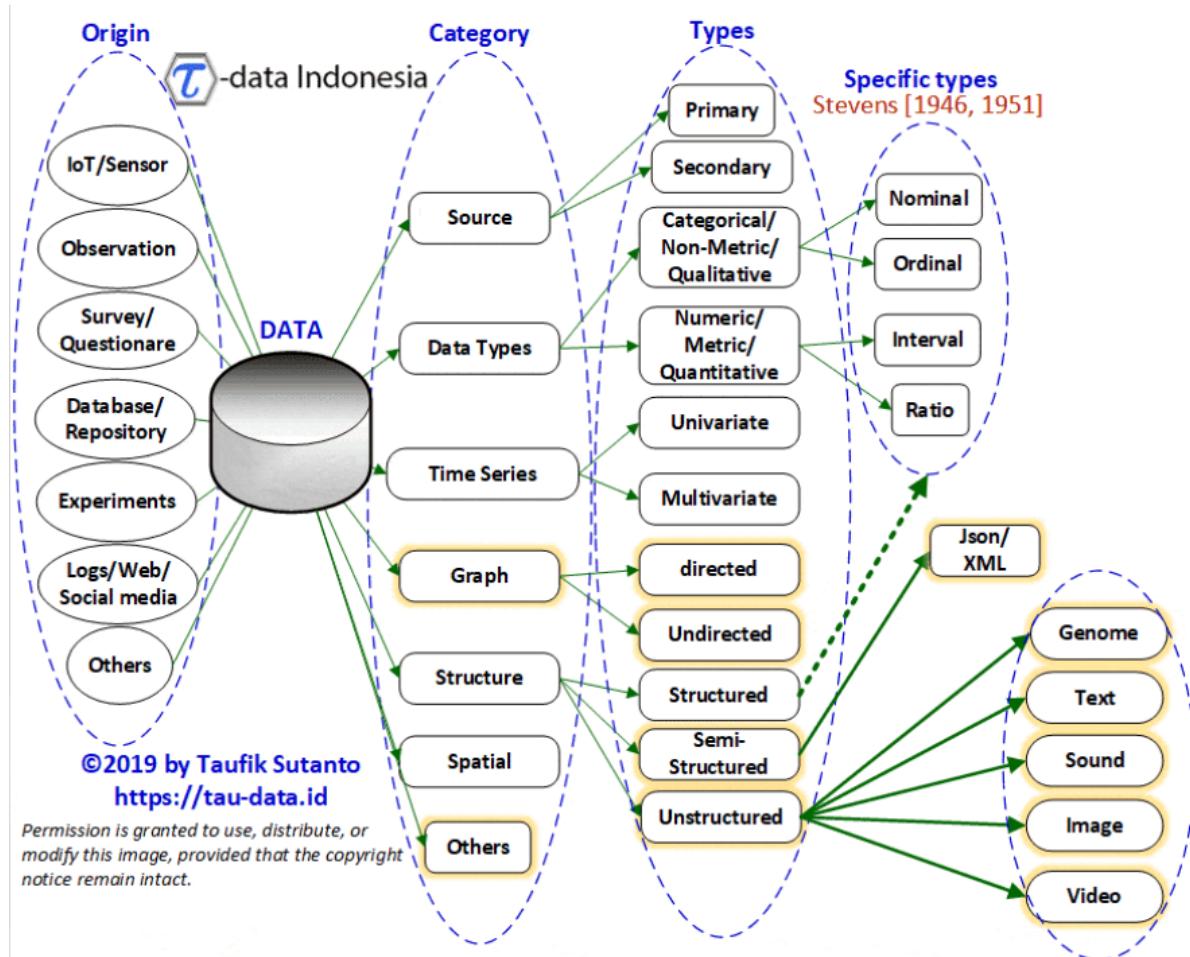


Figure 5.1: Gambar3. Susunan dan Jenis Data

5.1 Sumber Data

5.1.1 Internal

Data internal merupakan data yang sumbernya berasal dari dalam pihak yang dijadikan objek penelitian. Contoh data meliputi :

- Spreadsheet (excel, csv, json)
- Database (sql)
- Teks (txt, rtf)
- Media (video, audio)

5.1.2 External

Data eksternal merupakan data yang sumbernya berasal dari luar pihak objek yang diteliti. Contoh data meliputi :

- Website repository data (Kaggle, sklearn)
- Web page domain public (wikipedia, dbdata, data.go.id)
- Public Dataset (worldbank, UNICEF, WHO)

5.2 Susunan Data

Susunan data, juga dikenal sebagai struktur data, merujuk pada cara data disimpan, diatur, dan dihubungkan satu sama lain dalam suatu sistem komputasi. Ini melibatkan pemilihan format dan metode penyimpanan yang tepat agar data dapat diakses, dimanipulasi, dan dicari dengan efisien. Susunan data yang baik membantu meningkatkan efisiensi operasional, performa sistem, dan kemampuan pengambilan keputusan.

Datum adalah satuan terkecil, sebuah kumpulan teks dan angka. Pada bentuk ini, data tidak mempunyai nilai jual apapun.

Namun ketika kita menyusun beberapa datum menjadi satu kolom atau baris, kita dapat mendeskripsikan sebuah objek atau makna tertentu.

Kumpulkan banyak data maka kita menciptakan sebuah Informasi, dataset, atau konteks yang mudah dicerna oleh manusia.

Jika kita analisa sebuah dataset, kita akan mendapatkan sebuah hipotesis yang dapat dikonversi menjadi fakta atau knowledge setelah diverifikasi.

Dari sekumpulan knowledge, kita dapat menyusun sebuah decision atau keputusan yang sangat berpengaruh di kehidupan kita sehari-hari

Susunan Data dibagi menjadi 2, yaitu :

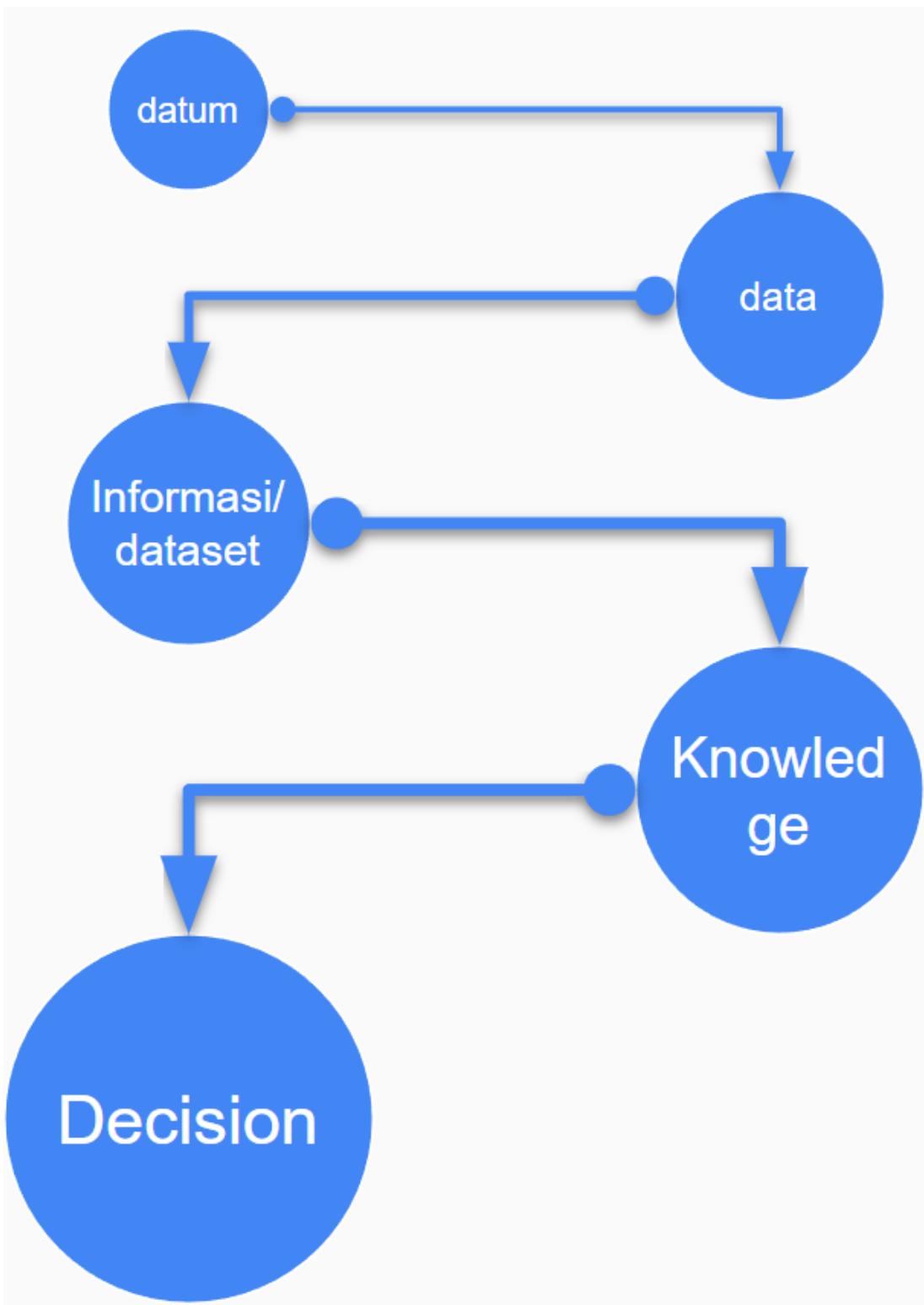


Figure 5.2: Gambar4. Struktur Data

5.2.1 Structured

Data terstruktur adalah jenis data yang memiliki format atau skema yang terorganisir dengan jelas. Setiap kolom dalam data terstruktur memiliki tipe data yang konsisten, dan setiap baris berisi entitas atau objek yang serupa.

Contoh data terstruktur meliputi :

- Tabular Data
- Object Oriented Data
- Time-series data

5.2.2 Unstructured

Data tidak terstruktur merujuk pada jenis data yang tidak memiliki format atau skema yang terorganisir dengan jelas. Data ini seringkali memiliki struktur yang tidak teratur atau tidak terprediksi, sehingga sulit untuk mengklasifikasikan, mengatur, atau memodelkannya secara tradisional.

Contoh data tidak terstruktur meliputi :

- Video atau audio
- Dokumen HTML
- Tweet atau postingan sosial media

5.3 Tipe Data

5.3.1 Tipe Data Berdasarkan Sifatnya

- **Data dikotomi**, merupakan data yang bersifat pilah satu sama lain, misalnya suku, agama, jenis kelamin, pendidikan, dan lain sebagainya.
- **Data diskrit**, merupakan data yang proses pengumpulan datanya dijalankan dengan cara menghitung atau membilang. Seperti, jumlah anak, jumlah penduduk, jumlah kematian dan sebagainya.
- **Data kontinum**, merupakan data pengumpulan datanya didapatkan dengan cara mengukur dengan alat ukur yang memakai skala tertentu. Seperti misalnya, Suhu, berat, bakat, kecerdasan, dan lainnya.

5.3.2 Tipe Data Berdasarkan Cara Pengumpulan

- **Data primer**, merupakan data yang didapatkan dari sumber pertama, atau dapat dikatakan pengumpulannya dilakukan sendiri oleh si peneliti secara langsung, seperti hasil wawancara dan hasil pengisian kuesioner (angket).
- **Data sekunder**, merupakan data yang didapatkan dari sumber kedua. Menurut Purwanto (2007), data sekunder yaitu data yang dikumpulkan oleh orang atau lembaga lain. Data sekunder adalah data yang digunakan atau diterbitkan oleh organisasi yang bukan pengolahnya (Soeratno dan Arsyad (2003;76).

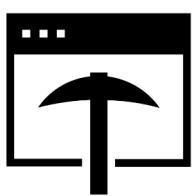
5.3.3 Tipe Data Berdasarkan Seri Waktu

- **Data Cross Section**, Data cross-section adalah data yang menunjukkan titik waktu tertentu. Contohnya laporan keuangan per 31 Desember 2020, data pelanggan PT. Data Indah bulan mei 2004, dan lain sebagainya.
- **Data Time Series / Berkala**, Data berkala adalah data yang datanya menggambarkan sesuatu dari waktu ke waktu atau periode secara historis. Contoh data time series adalah data perkembangan nilai tukar dollar amerika terhadap rupiah tahun 2016 - 2020.

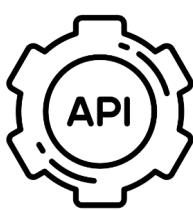
6 Pengambilan Data

Proses pengambilan data, juga dikenal sebagai proses ekstraksi data, merujuk pada langkah-langkah yang dilakukan untuk mengumpulkan, mengakses, dan memperoleh data dari sumber yang relevan

Ada beberapa Cara untuk mengambil data, yaitu:



Web Scraping



API



Manual



Direct
Database

6.1 Web Scraping

Web scraping artinya mengekstraksi data secara langsung dari suatu halaman web.

Langkah-langkah umum (contoh detil dapat dilihat di <https://realpython.com/beautiful-soup-web-scraping-python/>)

- Tentukan URL halaman web (HTML) yang akan di-scrape.
- Gunakan fungsi `requests.get` untuk mengakses URL tersebut. Teks HTML akan tersimpan pada atribut `text` dari object yang dikembalikan `requests.get`.
- Lakukan parsing pada HTML dengan library `beautifulsoup` untuk memperoleh tabel data yang diinginkan (dengan mengekstraksi elemen-elemen HTML yang relevan).

6.2 Application Program Interface (API)

API adalah sebuah alat untuk memudahkan website dan pengguna saling bertukar informasi. API disediakan oleh berbagai website atau perusahaan seperti Kaggle dan Twitter. Biasanya

API yang dimiliki perusahaan bersifat private, sehingga dibutuhkan sebuah token khusus untuk mengaksesnya. Namun ada beberapa API publik yang dapat diakses oleh siapa saja seperti PokeAPI.

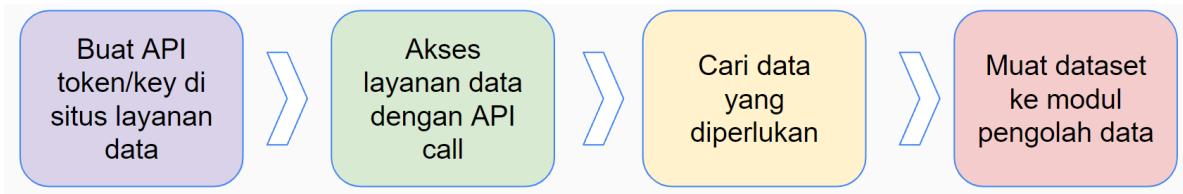


Figure 6.1: Gambar6. Contoh Penggunaan API

6.3 Manual ([Kaggle.com](#))

- Kita akan mengakses data dari “Goal Dataset – Top 5 European Leagues” dari Kaggle.
- Kunjungi Kaggle.com dan login (buat akun jika perlu)
- Lakukan pencarian “goal dataset top 5 European leagues”
- Klik “Goal Dataset – Top 5 European Leagues”

6.4 Direct Database

Data juga dapat bersumber dari basis data relasional (RDB, Mysql, Postgres) organisasi. Berikut langkah-langkahnya :

- Install sqlalchemy (pip install sqlalchemy), pandas (pip install pandas), dan mysql connector (pip install mysql-connector-python)
- Import semua library
- Tentukan username (root), password (kosong), port (3306), dan nama dari database (emp), lalu masukkan ke variable url.
- Buat engine menggunakan function create_engine, masukkan url sebagai parameteranya
- Buat query yang akan dijalankan di database, masukkan ke variable sql

← goal dataset top 5 european leagues

Date

- Last 90 days 18

Viewed By You

- Viewed 1
- Not Viewed 195

Dataset Size

- small 18
- medium 3

Dataset File Types

- csv 15
- xlsx 2
- sqlite 1

[More](#)

Dataset License

- Other 11
- Commercial 9
- Non-Commercial 1



Football Data: Expected Goals and Other Metrics

by Sergi Lehkyi
a year ago • 1 MB • ▲ 93

[Top European Leagues](#) Advanced Stats starting from 2014, includes xG metrics



The Beautiful Game - Analysis of Football Events

by Ahmed Youssef
3 years ago • 2m to run • R • ▲ 102

This [dataset](#) includes information on **9,074** matches from Europe's [top five leagues](#): the Premier League



Goal Dataset - Top 5 European Leagues

by shreyansh khandelwal
a month ago • 174 KB • ▲ 6

[Goal Dataset - Top 5 European Leagues](#)



Football Events

by Alin Secareanu

Figure 6.2: Gambar7. Contoh Pencarian Data di Kaggle

- Jalankan query menggunakan function execute
- Tampung hasil query di dataframe
- Lakukan operasi data di dataframe

7 Telaah Data

7.1 Apa itu telaah data?

Telaah data merujuk pada proses pengumpulan, pembersihan, eksplorasi, dan analisis data untuk mendapatkan pemahaman yang lebih yang terkandung dalam data tersebut. Tujuan dari telaah data adalah mendukung pengambilan keputusan berdasarkan bukti yang ditemukan dalam data.

7.2 Mengapa perlu telaah data?

1. Data dari masing-masing sumber belum tentu dapat langsung dipakai karena:
 - maksud dan tujuan data berbeda-beda
 - keadaan asal terpisah-pisah atau justru terintegrasi secara ketat.
 - tingkat kekayaan (richness) berbeda-beda
 - tingkat keandalan (reliability) berbeda-beda
2. Data understanding memberikan gambaran awal tentang:
 - kekuatan data
 - kekurangan dan batasan penggunaan data
 - tingkat kesesuaian data dengan masalah bisnis yang akan dipecahkan
 - ketersediaan data (terbuka/tertutup, biaya akses, dsb.)

7.3 Bentuk Data

Data dapat memiliki berbagai bentuk, diantaranya:

- Spreadsheet(excel, csv, dll)
- Database(SQL, Oracle, dll)
- Text file(txt, doc, pdf, dll)
- Multimedia documents(audio, video, gambar, dll)

7.4 Sumber Data

1. Sumber Internal
 - Data yang dihasilkan oleh perusahaan sendiri
 - Data yang dihasilkan oleh perusahaan lain yang terkait dengan perusahaan
2. Sumber Eksternal
 - Repozitori data publik
 - Halaman web publik

7.5 Daftar Sumber Data Daring

- [Portal Satu Data Indonesia](#)
- [Portal Data Jakarta](#)
- [Portal Data Bandung](#)
- [Badan Pusat Statistik](#)
- [Badan Informasi Geospasial](#)
- [UCI Machine Learning repository](#)
- [Kaggle](#)
- [World Bank Open Data](#)

7.6 Tipe Data

7.6.1 Bedasarkan sifat

- **Data dikotomi**, merupakan data yang bersifat pilah satu sama lain, misalnya suku, agama, jenis kelamin, pendidikan, dan lain sebagainya.
- **Data diskrit**, merupakan data yang proses pengumpulan datanya dijalankan dengan cara menghitung atau membilang. Seperti, jumlah anak, jumlah penduduk, jumlah kematian dan sebagainya.
- **Data kontinu**, merupakan data pengumpulan datanya didapatkan dengan cara mengukur dengan alat ukur yang memakai skala tertentu. Seperti misalnya, Suhu, berat, bakat, kecerdasan, dan lainnya.

7.6.2 Bedasarkan waktu

- **Data Cross Section**, adalah data yang menunjukkan titik waktu tertentu. Contohnya laporan keuangan per 31 Desember 2020, data pelanggan PT.Data Indah bulan mei 2004, dan lain sebagainya.
- **Data Time Series / Berkala**, adalah data yang datanya menggambarkan sesuatu dari waktu ke waktu atau periode secara historis. Contoh data time series adalah data perkembangan nilai tukar dollar amerika terhadap rupiah tahun 2016 - 2020

7.7 Pengambilan Data

- Pengambilan data secara manual.
- Pengambilan data melalui API
 - Contoh melalui API Kaggle (pip install kaggle)(kaggle datasets download -d mauryansshivam/paytm-revenue-users-transactions)
 - Contoh melalui API Portal Data Bandung (<http://data.bandung.go.id/index.php/portal/api/1>)
- Pengambilan data melalui akses langsung ke basis data relasional yang ada.
- Pengambilan data melalui web scraping.
 - Contoh pengambilan data melalui web scraping kometar video youtube

```
#| code-fold: true

# Jalankan instalasi library berikut jika belum terinstall
# pip install google-api-python-client

import csv
from googleapiclient.discovery import build

# Set up the API key and YouTube Data API service
## Masukkan API key yang sudah dibuat di Google Cloud Platform
## Contoh : AIzaSyCFjru8d0ZbGtZUi_AQu1Cz1MLoANaY22k
API_KEY = ""
youtube = build("youtube", "v3", developerKey=API_KEY)

def scrape_comments(video_id):
    # Get the video details
    video_response = youtube.videos().list(
        part="snippet",
        id=video_id
```

```

).execute()

video_title = video_response['items'][0]['snippet']['title']
print("Scraping comments for video:", video_title)

# Get the video comments
comments = []
next_page_token = None

while True:
    comment_response = youtube.commentThreads().list(
        part="snippet",
        videoId=video_id,
        maxResults=100,
        pageToken=next_page_token
    ).execute()

    for item in comment_response["items"]:
        comment = item["snippet"]["topLevelComment"]["snippet"]["textDisplay"]
        comments.append(comment)

    next_page_token = comment_response.get("nextPageToken")

    if not next_page_token:
        break

return comments

# Test the function
## Masukkan ID video youtube yang ingin diambil komentarnya
## Contoh : 5kAF9QV5nYQ
video_id = ""
comments = scrape_comments(video_id)

# Save comments to CSV file
filename = "comments.csv"
with open(filename, "w", newline="", encoding="utf-8") as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(["Comment"])
    writer.writerows(zip(comments))

```

```
print("Comments saved to", filename)
```

7.8 Library Pandas



- Pandas adalah library yang digunakan untuk melakukan manipulasi, analisis, dan visualisasi data.
- Pandas menyediakan struktur data dan fungsi high-level untuk membuat pekerjaan dengan data terstruktur/tabular lebih cepat, mudah, dan ekspresif.

7.8.1 Memuat data ke Pandas

1. Nyalakan Jupyter Notebook di folder kerja Anda.
2. Buka atau buat baru suatu skrip ipynb (Python 3)
3. Import pandas dan numpy. (Pastikan sudah terinstal sebelumnya).
4. Load file CSV yang sudah diunduh sebelumnya (Mengambil Data secara Manual) ke dalam sebuah DataFrame
5. Gunakan perintah `read_csv(dataset)`

```
import pandas as pd
# Contoh csv
df = pd.read_csv('titanic_dataset.csv')

# Contoh excel
df = pd.read_excel('titanic_dataset.xlsx')

# Contoh sqlite
import sqlite3
conn = sqlite3.connect('titanic_dataset.sqlite3')
df = pd.read_sql_query("SELECT * FROM titanic", conn)
```

7.8.2 Menampilkan Data

Method `head()` dan `tail()` pada DataFrame membantu kita menampilkan 5 beberapa baris pertama/terakhir dari data yang kita muat. Dengan memberikan argumen pada method tersebut kita juga bisa mengatur jumlah data yang ditampilkan

```
# 5 baris pertama  
df.head()  
  
# 5 baris terakhir  
df.tail()  
  
# 10 baris pertama  
df.head(10)
```

7.8.3 Melihat Tipe Data

Method `dtypes` pada DataFrame membantu kita menampilkan tipe data dari setiap kolom pada data yang kita muat. Jika terdapat tipe data yang tidak tepat maka sebaiknya dilakukan pengecekan nilai pada kolom tersebut. Jika memang sudah benar, maka kita bisa mengubah tipe data tersebut menjadi tipe data yang tepat.

```
# Melihat tipe data  
df.dtypes  
  
# Mengubah tipe data menjadi int  
df['nama_kolom'] = df['nama_kolom'].astype('int')
```

7.8.4 Deskripsi statistik data

Method `describe()` pada DataFrame membantu kita menampilkan deskripsi statistik dari data yang kita muat. Deskripsi statistik yang ditampilkan adalah deskripsi statistik untuk kolom-kolom dengan tipe data numerik. Jika terdapat kolom dengan tipe data selain numerik, maka deskripsi statistik tersebut tidak akan ditampilkan. Untuk tetap menampilkan deskripsi statistik dari kolom non-numerik, kita bisa menambahkan argumen `include='all'` pada method `describe()`.

```
# Deskripsi statistik data  
df.describe()
```

```
# Deskripsi statistik data termasuk kolom non-numerik  
df.describe(include='all')
```

7.8.5 Fungsi statistik dalam Pandas

Pandas menyediakan fungsi statistik untuk menghitung nilai rata-rata, median, standar deviasi, dan variansi. Fungsi-fungsi tersebut adalah:

Fungsi	Keterangan
count	Jumlah observasi non-NUL
sum	Jumlah
mean	Rata-rata
mad	Deviasi absolut rata-rata
median	Nilai tengah
min	Nilai minimum
max	Nilai maksimum
mode	Modus (nilai yang paling sering muncul)
abs	Nilai absolut (nilai mutlak)
prod	Hasil kali dari nilai-nilai
quantile	Kuantil sampel (nilai pada persentil), kuartil pertama = quantile(0.25)
std	Standar deviasi
var	Varians
sem	Standar error mean (standar error dari rata-rata)
skew	Skewness (kecondongan distribusi data)
kurt	Kurtosis (tingkat “tumpul” atau “tajam” distribusi data)
cumsum	Akumulasi jumlah
cumprod	Akumulasi hasil kali
cummax	Nilai maksimum akumulasi
cummin	Nilai minimum akumulasi

7.8.6 Mentukan Outlier

Outlier adalah data yang memiliki nilai ekstrim dibandingkan dengan data lainnya. Outlier dapat mempengaruhi hasil analisis data sehingga perlu ditangani dengan baik. Untuk mengetahui apakah suatu data merupakan outlier atau bukan, kita bisa menggunakan beberapa metode, diantaranya:

- Tukey Fences (IQR)

- **Plus:** IQR adalah metode yang tahan terhadap nilai ekstrim dan lebih baik digunakan pada data dengan distribusi yang condong (skewed) karena mengandalkan kuartil (quartiles) yang tidak terpengaruh oleh ekstrem nilai.
 - **Minus:** IQR tidak dapat digunakan pada data dengan distribusi normal karena mengandalkan kuartil (quartiles).
- Z-Score
 - **Plus:** Z-score dapat digunakan pada data dengan distribusi normal karena mengandalkan mean dan standard deviasi.
 - **Minus:** Z-score tidak tahan terhadap nilai ekstrim.
 - Metode Hampiran Berbasis Probabilitas (Contoh: grubb's test)
 - **Plus:** Metode hampiran berbasis probabilitas seperti Grubbs' Test dapat memberikan kepercayaan statistik dalam menentukan apakah sebuah data benar-benar outlier atau hanya perbedaan alami dalam distribusi data.
 - **Minus:** Memerlukan asumsi bahwa data harus berdistribusi normal. Selain itu, metode ini hanya cocok untuk mengidentifikasi satu outlier dalam satu arah (outlier yang memiliki nilai ekstrim di atas atau di bawah rata-rata).

7.8.7 Nilai Unik

Method `unique()` pada Pandas digunakan untuk mengetahui nilai unik dari suatu kolom. Method ini mengembalikan nilai unik dari suatu kolom dalam bentuk array. Method `value_counts()` pada Pandas digunakan untuk menghitung berapa kali suatu nilai muncul dalam suatu kolom. Method ini mengembalikan Series yang berisi frekuensi setiap nilai yang muncul dalam suatu kolom.

```
# Penggunaan unique
df["nama_kolom"].unique()

# Penggunaan value_counts
df["nama_kolom"].value_counts()
```

7.8.8 Analisis dengan groupby

Method `groupby()` pada Pandas digunakan untuk melakukan grouping berdasarkan kolom tertentu. Method ini mengembalikan objek DataFrameGroupBy.

```
# Penggunaan groupby dengan fungsi mean
df.groupby("kolom1")["kolom2"].mean()
```

```
# menghitung statistik deskriptif dari setiap kelompok data
df.groupby("kolom1")["kolom2"].describe()

# agregasi kustom dengan fungsi agg
df.groupby("kolom1")["kolom2"].agg([sum, min, max])

#Menggunakan operasi agregasi kustom (menghitung rasio nilai maksimum dan minimum dalam setiap kelompok)
df.groupby("kolom1")["kolom2"].agg(lambda x: x.max() / x.min())
```

7.8.9 Korelasi

Korelasi adalah salah satu metode statistik yang digunakan untuk mengetahui seberapa besar hubungan antara dua variabel dengan variabel lainnya. Korelasi memiliki nilai berkisar antara -1 hingga 1. Nilai 1 menunjukkan hubungan yang sempurna, nilai -1 menunjukkan hubungan yang sempurna pula dengan arah yang berlawanan, dan nilai 0 menunjukkan tidak ada hubungan antara kedua variabel tersebut. Kita dapat menggunakan method `corr()` pada Pandas untuk menghitung korelasi dari setiap pasang kolom pada suatu DataFrame.

```
# Menghitung korelasi
df.corr()

# Terdapat 3 pilihan metode perhitungan korelasi, yaitu:
# pearson (default), kendall, dan spearman
df.corr(method="kendall")
```

8 Validasi Data

8.1 Tugas Validasi Data

- Periksa/Nilai Kualitas Data
- Periksa/Nilai Tingkat Kecukupan Data
- Periksa/Nilai Kesesuaian Data
- Periksa/Nilai Konsistensi Data

8.2 Manfaat Validasi Data

- Tidak merusak perhitungan pada tahapan selanjutnya.
- Memvisualisasikan sebaran data, mendekripsi pola, dan mengidentifikasi anomali.
- Membantu menjelaskan dan mengkomunikasikan hasil analisis dengan lebih jelas.

8.3 Laporan Dokumentasi Data Validasi

Laporan dokumentasi data validasi, setidaknya memiliki parameter berikut:

- **Kebenaran**, misal di Indonesia isian Gender yang diakui hanya 2 P/W; Agama hanya 6 (Islam, Protestan, Katholik, Hindu, Budha, Konghucu)
- **Kelengkapan**, misal data provinsi seluruh Indonesia (34 prov), namun hanya sebagian yg ada.
- **Konsistensi**, misal penulisan STM atau SMK;

8.4 Validasi vs Verifikasi

- **Validasi**: memastikan bahwa data yang diinputkan sesuai dengan ketentuan yang berlaku.
- **Verifikasi**: memastikan bahwa data yang diinputkan sesuai dengan data yang ada.

8.5 Tahapan kritikal dalam validasi

- Tipe Data (integer, float, string)
- Ekspresi Konsisten (mis. Jalan, Jl., Jln.)
- Format Data (mis. utk tgl “YYYY-MM-DD” vs “DD-MM-YYYY.”)
- Nilai Null/Missing Values
- Misspelling/Type
- Invalid Data (gender: L/P: L; Laki-laki; P: Pria/Perempuan?)

8.6 Teknik Validasi Data

- **Manual:** melihat data secara langsung, misalnya melihat data di Excel.
- **Statistik:** menggunakan statistik deskriptif, misalnya melihat jumlah data, nilai maksimum, nilai minimum, dan lain-lain.
- **Visualisasi:** menggunakan visualisasi data, misalnya melihat sebaran data menggunakan histogram, boxplot, dan lain-lain.

8.7 Validasi Dengan Pandas

Method `info()` dapat digunakan untuk melihat informasi data frame, seperti jumlah baris, kolom, nilai non-NUL, tipe data, dan total penggunaan memori. Method ini sangat berguna dalam melakukan validasi tahap awal pada data.

```
import pandas as pd
df = pd.read_csv('spreadsheet.csv')

# Penggunaan info
df.info()
```

8.8 Visualisasi Data

- Visualisasi data dapat membantu dalam memvalidasi data.
- Memvisualisasikan sebaran data, mendeteksi pola, dan mengidentifikasi anomali.
- Visualisasi yang baik dapat menceritakan sebuah cerita tentang data Anda dengan cara yang tidak dapat dilakukan oleh sebuah kalimat.

8.8.1 Jenis Visualisasi Data

- Perbandingan (Comparison)
 - Bar Chart
 - Line Chart
 - Combo Chart
- Komposisi (Composition)
 - Pie Chart
 - Stacked Bar Chart
 - Treemap
 - Waterfall Chart
- Distribusi (Distribution)
 - Histogram
 - Box Plot
 - Violin Plot
- Hubungan (Relationship)
 - Scatter Plot
 - Bubble Chart
 - Heatmap

8.8.2 Library Visualisasi Data

Terdapat **dua** library populer untuk visualisasi data di Python, yaitu:

- **Matplotlib**: library yang paling populer untuk visualisasi data di Python. Umumnya diberi alias `plt`.



- **Seaborn**: library yang dibangun di atas Matplotlib, sehingga memiliki sintaks yang mirip. Umumnya diberi alias `sns`.



8.8.3 Korelasi & Causation

- Korelasi merupakan suatu pengukuran sejauh mana nilai saling ketergantungan antar variabel.
- Causation merupakan hubungan antara sebab dan akibat antara dua variabel
- Penting untuk mengetahui perbedaan antara keduanya dan bahwa korelasi tidak mendeskripsikan sebab-akibat.
- Menentukan korelasi jauh lebih sederhana, menentukan sebab memerlukan analisis lebih lanjut

8.8.4 Analisis Korelasi (Heatmap)

- Korelasi Pearson
 - Mengukur hubungan linier antara dua variabel
- Korelasi Spearman
 - Mengukur hubungan monotonik antara dua variabel

Visualisasi korelasi dapat dilakukan dengan menggunakan **heatmap**.

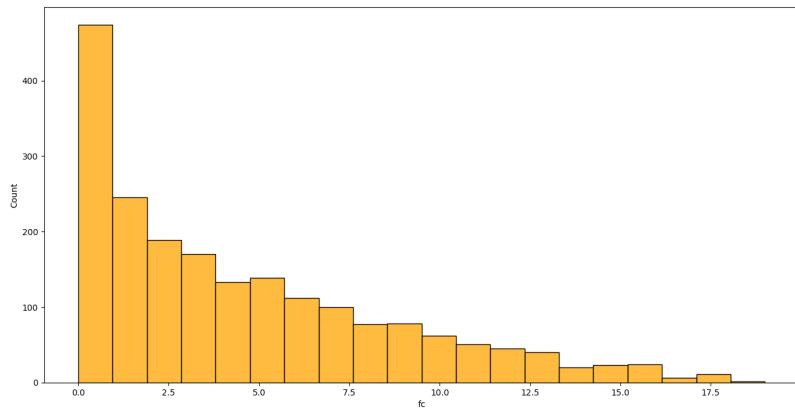
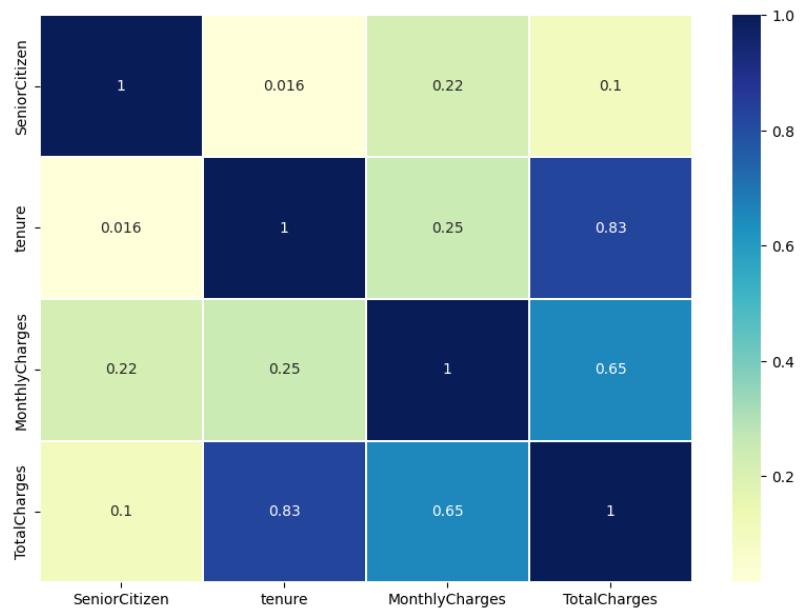
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 7))
sns.heatmap(df.corr('pearson'), annot=True, cmap='YlGnBu', linewidths=0.2)
```

8.8.5 Histogram

Histogram dapat membantu dalam melihat persebaran data.

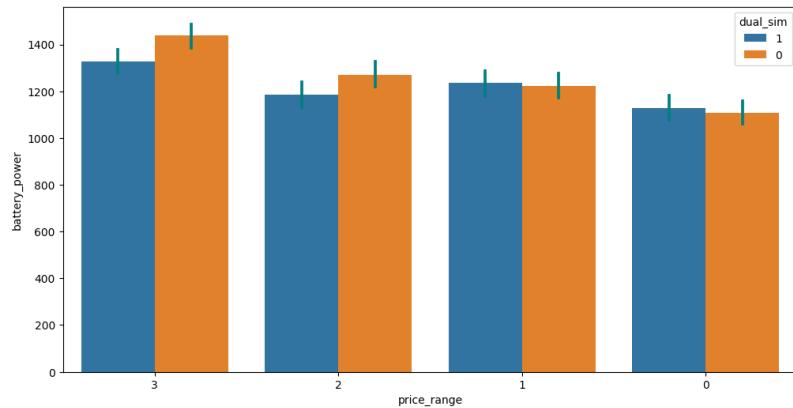
```
sns.histplot(data = df, x= 'fc' ,color='orange', kde=False)
```



8.8.6 Bar Chart

Bar chart digunakan untuk membandingkan data-data yang berbentuk kategorikal.

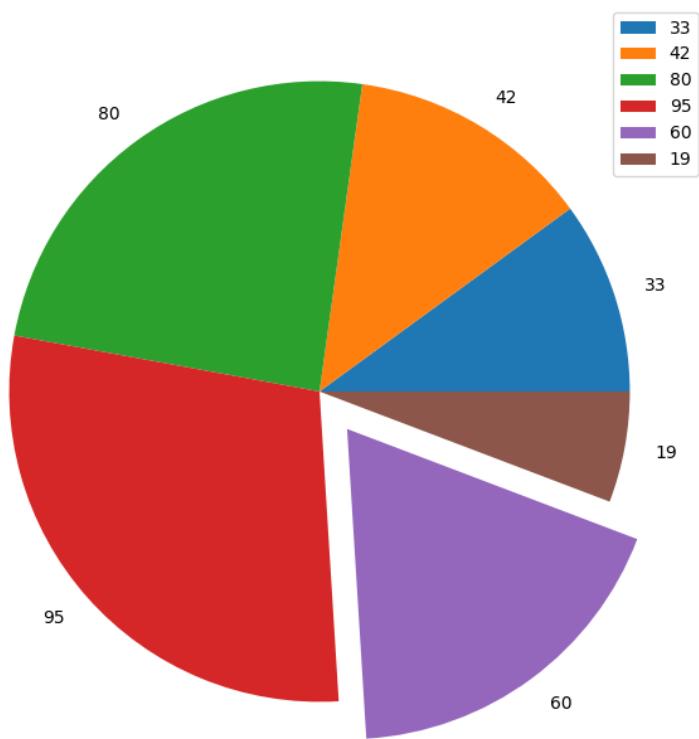
```
sns.barplot(x='price_range',y='battery_power',data = df, order=[3,2,1,0], hue='dual_sim',
```



8.8.7 Pie Chart

Pie chart membantu memvisualisasikan komposisi elemen-elemen dalam sebuah data.

```
ax = plt.gca()
plt.pie(x=x, explode=[0, 0, 0, 0, 0.15, 0], labels=x)
plt.legend(x, loc='upper right')
plt.show()
```



50

9 Menentukan Objek atau Memilih Data

9.1 Menentukan Sumber Data

Sumber data merupakan tempat atau lokasi sebuah data disimpan dan atau diakses untuk dapat dilakukan analisis serta penggunaan lainnya. Pemilihan sumber data harus mempertimbangkan sebuah kendala, akurasi, ketersediaan, serta relevansi data dengan tujuan analisis atau model dari machine learning yang akan digunakan atau dibangun.

9.2 Menelaah Susunan Data

- Melakukan pemeriksaan struktur data untuk memahami bagaimana data diorganisir.
- Melihat dimensi data, jumlah atribut/kolom, dan jumlah sampel/baris.
- Menilai apakah data terstruktur (misalnya, data tabel) atau tidak terstruktur (misalnya, data teks atau gambar).

9.3 Menentukan Tipe dan Model Data yang dimiliki

- Identifikasi tipe data untuk setiap atribut (numerik, kategorikal, teks, tanggal, dll.).
- Penentuan model data yang sesuai untuk analisis atau model machine learning berdasarkan tipe data.
- Pemahaman tentang apakah data bersifat kontinu, diskret, ordinal, atau nominal.

9.4 Mengambil Data

- Proses pengambilan data dari sumber data ke lingkungan analisis atau pengembangan model machine learning.
- Menggunakan berbagai metode seperti mengimpor file data (misalnya CSV, Excel), mengakses database, atau menggunakan API untuk mengambil data dari sumber online.

9.5 Menelaah Data dalam Machine Learning

- Melakukan eksplorasi data (data exploration) untuk memahami karakteristik data secara lebih mendalam.
- Visualisasi data dengan grafik atau plot untuk memahami pola, distribusi, dan korelasi antara atribut.
- Identifikasi missing value, outlier, dan data yang tidak konsisten untuk diatasi sebelum analisis atau pemodelan.

9.6 Contoh Source Code

Berikut adalah contoh sederhana menggunakan Python untuk mengambil data dari file CSV, menelaah susunan data, menentukan tipe data, dan melakukan eksplorasi data menggunakan pandas dan matplotlib.

Silahkan Unduh dataset berikut [Iris_Dataset](#). Pastikan dataset disimpan dalam file CSV dengan nama “iris_dataset.csv” dalam folder yang sama dengan script Python.

```
import pandas as pd
import matplotlib.pyplot as plt

# Mengambil data dari file CSV
data_path = 'iris_dataset.csv'
df = pd.read_csv(data_path)

# Menelaah susunan data
print("Dimensi data:", df.shape)
print("Info data:")
print(df.info())
print("Sepuluh data pertama:")
print(df.head(10))

# Menentukan tipe data dan model data yang dimiliki
print("Tipe data untuk setiap atribut:")
print(df.dtypes)

# Preprocessing data (jika diperlukan)
# Tidak ada preprocessing yang diperlukan dalam contoh ini

# Mengambil statistik deskriptif untuk data numerik
print("Statistik deskriptif:")
```

```

print(df.describe())

# Eksplorasi data dengan visualisasi
plt.figure(figsize=(10, 6))
plt.scatter(df['sepal_length'], df['sepal_width'])
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Scatter Plot: Sepal Length vs. Sepal Width')
plt.show()

plt.figure(figsize=(8, 6))
df['species'].value_counts().plot(kind='bar')
plt.xlabel('Species')
plt.ylabel('Count')
plt.title('Bar Plot: Species Distribution')
plt.show()

```

Pastikan bahwa dataset “Iris” (iris_dataset.csv) berisi kolom dengan nama “sepal_length”, “sepal_width”, “petal_length”, “petal_width”, dan “species”. Jika dataset tersebut memiliki atribut lain atau format yang berbeda, Anda perlu menyesuaikan kode di atas sesuai dengan dataset yang digunakan.

Contoh di atas akan membaca dataset Iris, menampilkan informasi dasar tentang dataset, menampilkan sepuluh baris pertama dari data, menentukan tipe data untuk setiap atribut, menampilkan statistik deskriptif untuk data numerik, dan melakukan eksplorasi data dengan visualisasi menggunakan scatter plot untuk melihat hubungan antara panjang dan lebar kelopak bunga serta bar plot untuk melihat distribusi spesies bunga.

Harap dicatat bahwa contoh di atas hanya merupakan contoh sederhana, dan dalam penerapan sebenarnya, langkah-langkah analisis dan eksplorasi data bisa lebih canggih dan lengkap sesuai dengan kompleksitas dan karakteristik data.

Pemahaman mendalam tentang sumber data, susunan data, tipe data, dan karakteristik data adalah langkah kritis sebelum membangun model machine learning yang efektif.

10 Cleaning Data

10.1 Data Cleaning

Pada proses machine learning, terdapat tahapan **preprocessing**. Preprocessing merupakan tahapan yang penting dalam machine learning. Hasil dari preprocessing dapat mempengaruhi nilai akurasi dari sebuah model. Tujuan dari Preprocessing adalah untuk memastikan data siap untuk diproses dan atau digunakan pada machine learning. Dalam melakukan preprocessing memiliki beberapa tantangan, diantara adalah **missing value, outlier, format tidak konsisten, dan malformed record**

10.2 Cara Mengatasi Missing Value

- Mising Value merupakan data yang kosong atau tidak lengkap.
- Tantangan dalam mengatasi missing value adalah bagaimana mengisi nilai kosong tersebut.
- Strategi yang dapat dilakukan diantara lain :
 - Menghapus baris atau kolom yang memiliki missing value
 - Mengisi nilai kosong dengan nilai rata-rata atau median
 - Mengisi nilai kosong dengan nilai yang sering muncul

Pilih strategi yang paling sesuai tergantung pada tipe data dan tujuan analisis

- **Menghapus baris atau kolom yang memiliki missing value**
- **Tanpa Menggunakan Dataset**

```
import pandas as pd

# Contoh dataset dengan nilai yang hilang
data = {
    'A': [1, 2, None, 4],
    'B': [5, None, 7, 8],
```

```

    'C': [9, 10, 11, None]
}
df = pd.DataFrame(data)

# Tampilkan dataset sebelum penghapusan
print("Dataset sebelum penghapusan:")
print(df)

# Hapus baris yang memiliki nilai yang hilang
df_cleaned_rows = df.dropna()
print("\nDataset setelah menghapus baris yang memiliki nilai yang hilang:")
print(df_cleaned_rows)

# Hapus kolom yang memiliki nilai yang hilang
df_cleaned_cols = df.dropna(axis=1)
print("\nDataset setelah menghapus kolom yang memiliki nilai yang hilang:")
print(df_cleaned_cols)

```

- Menggunakan Dataset

untuk mengunduh dataset agar dapat digunakan pada source code [disini](#)

```

import pandas as pd

# Load dataset Titanic dari file CSV
titanic_df = pd.read_csv("titanic.csv")

# Tampilkan informasi mengenai dataset, termasuk jumlah nilai yang hilang di setiap kolom
print("Informasi tentang dataset Titanic:")
print(titanic_df.info())

# Hapus baris yang memiliki nilai yang hilang
titanic_cleaned_rows = titanic_df.dropna()
print("\nDataset Titanic setelah menghapus baris yang memiliki nilai yang hilang:")
print(titanic_cleaned_rows)

# Hapus kolom yang memiliki nilai yang hilang
titanic_cleaned_cols = titanic_df.dropna(axis=1)
print("\nDataset Titanic setelah menghapus kolom yang memiliki nilai yang hilang:")
print(titanic_cleaned_cols)

```

- Mengisi nilai kosong dengan nilai rata-rata atau median

```
import pandas as pd

# Load dataset Titanic dari file CSV
titanic_df = pd.read_csv("titanic.csv")

# Tampilkan informasi mengenai dataset, termasuk jumlah nilai yang hilang di setiap kolom
print("Informasi tentang dataset Titanic sebelum pengisian nilai kosong:")
print(titanic_df.info())

# Mengisi nilai kosong pada kolom 'Age' dengan nilai median dari kolom tersebut
age_median = titanic_df['Age'].median()
titanic_df['Age'].fillna(age_median, inplace=True)

# Mengisi nilai kosong pada kolom 'Fare' dengan nilai rata-rata dari kolom tersebut
fare_mean = titanic_df['Fare'].mean()
titanic_df['Fare'].fillna(fare_mean, inplace=True)

# Tampilkan informasi tentang dataset setelah pengisian nilai kosong
print("\nInformasi tentang dataset Titanic setelah pengisian nilai kosong:")
print(titanic_df.info())
```

- Mengisi nilai kosong dengan nilai yang sering muncul

```
import pandas as pd

# Load dataset Titanic dari file CSV
titanic_df = pd.read_csv("titanic.csv")

# Tampilkan informasi mengenai dataset, termasuk jumlah nilai yang hilang di setiap kolom
print("Informasi tentang dataset Titanic sebelum pengisian nilai kosong:")
print(titanic_df.info())

# Mengisi nilai kosong pada kolom 'Embarked' dengan nilai yang sering muncul (mode) dari k
embarked_mode = titanic_df['Embarked'].mode()[0]
titanic_df['Embarked'].fillna(embarked_mode, inplace=True)

# Tampilkan informasi tentang dataset setelah pengisian nilai kosong
print("\nInformasi tentang dataset Titanic setelah pengisian nilai kosong:")
```

```
print(titanic_df.info())
```

10.3 Menangani Outlier

Outlier merupakan data yang jauh dari nilai rata-rata atau nilai normal. Outlier dapat mempengaruhi hasil analisis dan machine learning.

- Strategi untuk mengatasi Outlier :
 - Menghapus outlier.
 - Menggantikan nilai outlier dengan nilai lain seperti nilai rata-rata atau median.
 - Menggunakan teknik scaling atau normalisasi.

Pilih strategi yang paling sesuai tergantung pada tipe data dan tujuan analisis.

- Menghapus Outlier

```
import pandas as pd
from sklearn.datasets import load_iris
from scipy import stats

# Load dataset Iris dari scikit-learn
data = load_iris()
iris_df = pd.DataFrame(data.data, columns=data.feature_names)

# Tampilkan informasi tentang dataset Iris sebelum penghapusan outlier
print("Informasi tentang dataset Iris sebelum penghapusan outlier:")
print(iris_df.describe())

# Definisikan fungsi untuk menghapus outlier berdasarkan z-score
def remove_outliers_zscore(df, z_threshold=3):
    z_scores = stats.zscore(df)
    return df[(z_scores < z_threshold).all(axis=1)]

# Hapus outlier dari dataset Iris berdasarkan z-score
iris_cleaned = remove_outliers_zscore(iris_df)

# Tampilkan informasi tentang dataset Iris setelah penghapusan outlier
print("\nInformasi tentang dataset Iris setelah penghapusan outlier:")
print(iris_cleaned.describe())
```

Pada contoh di atas, digunakan metode z-score untuk mendeteksi outlier dalam dataset Iris. Outlier adalah data yang memiliki z-score lebih besar dari z_threshold yang telah ditentukan (z_threshold=3). Fungsi remove_outliers_zscore digunakan untuk menghapus baris yang mengandung outlier berdasarkan z-score, yaitu baris yang memiliki setidaknya satu fitur (kolom) dengan z-score melebihi z_threshold. Penting untuk berhati-hati karena penghapusan outlier dapat mempengaruhi hasil analisis atau model machine learning. Selain z-score, ada banyak teknik deteksi outlier lainnya seperti IQR dan pendekatan berbasis model machine learning atau domain knowledge.

- Menggantikan nilai outlier dengan nilai lain seperti nilai rata-rata atau median.

```
import pandas as pd

# membaca dataset publik
data = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.da

# menentukan batas outlier
q1 = data[0].quantile(0.25)
q3 = data[0].quantile(0.75)
iqr = q3 - q1
lower_bound = q1 - (1.5 * iqr)
upper_bound = q3 + (1.5 * iqr)

# menggantikan outlier dengan nilai rata-rata
data[0] = data[0].apply(lambda x: data[0].mean() if x < lower_bound or x > upper_bound else

# menggantikan outlier dengan nilai median
data[0] = data[0].apply(lambda x: data[0].median() if x < lower_bound or x > upper_bound else

# menampilkan dataset yang telah diubah
print(data)
```

Pada contoh kode di atas, dataset publik yang digunakan adalah Iris Dataset yang tersedia di UCI Machine Learning Repository. Pertama-tama, kita menentukan batas outlier dengan menghitung kuartil pertama (Q1), kuartil ketiga (Q3), dan rentang antar kuartil (IQR). Kemudian, kita menentukan batas bawah dan batas atas untuk menentukan nilai outlier.

Kemudian, kita menggantikan nilai outlier dengan nilai rata-rata atau median. Untuk menggantikan dengan nilai rata-rata, kita menggunakan fungsi apply() pada kolom yang menghasilkan nilai rata-rata jika nilai kurang dari batas bawah atau lebih dari batas atas, sedangkan untuk menggantikan dengan nilai median, kita juga menggunakan fungsi apply() pada

kolom yang menghasilkan nilai median jika nilai kurang dari batas bawah atau lebih dari batas atas.

Terakhir, kita menampilkan dataset yang telah diubah dengan fungsi print().

- Mengisi nilai kosong dengan nilai yang sering muncul

```
import pandas as pd

# Load dataset Titanic dari file CSV
titanic_df = pd.read_csv("titanic.csv")

# Tampilkan informasi mengenai dataset, termasuk jumlah nilai yang hilang di setiap kolom
print("Informasi tentang dataset Titanic sebelum pengisian nilai kosong:")
print(titanic_df.info())

# Mengisi nilai kosong pada kolom 'Embarked' dengan nilai yang sering muncul (mode) dari k
embarked_mode = titanic_df['Embarked'].mode()[0]
titanic_df['Embarked'].fillna(embarked_mode, inplace=True)

# Tampilkan informasi tentang dataset setelah pengisian nilai kosong
print("\nInformasi tentang dataset Titanic setelah pengisian nilai kosong:")
print(titanic_df.info())
```

Pada contoh di atas, menggunakan metode mode() dari pandas untuk menghitung nilai yang sering muncul (mode) pada kolom ‘Embarked’, lalu mengisi nilai kosong pada kolom tersebut dengan nilai mode yang dihitung. Penggunaan parameter inplace=True memastikan perubahan dilakukan langsung pada DataFrame asli. Setelah mengisi nilai kosong, hasilnya dapat diperiksa untuk memastikan tidak ada lagi nilai yang hilang pada kolom ‘Embarked’.

Perlu diingat, pengisian nilai kosong dengan nilai yang sering muncul merupakan salah satu teknik imputasi yang umum. Teknik lainnya termasuk pengisian dengan nilai rata-rata, nilai median, atau menggunakan model machine learning untuk memprediksi nilai yang hilang berdasarkan data lainnya. Pilihan teknik imputasi tergantung pada karakteristik dataset dan tujuan analisis atau model machine learning yang ingin diimplementasikan.

10.4 - Menangani Format yang Tidak Konsisten

Format Tidak Konsisten terjadi ketika data memiliki format yang sama atau tidak sesuai dengan format yang diharapkan. Sering terjadi pada pemformatan tanggal, bulan, dan tahun.

- Strategi dalam mengatasi Format Tidak Konsisten
 - Mengubah format data menjadi format yang konsisten seperti mengubah format tanggal menjadi format yang sama
 - Memperbaiki data yang salah ketik atau typo

Pilih strategi yang paling sesuai tergantung pada tipe data dan tujuan analisis

- Mengubah format data menjadi format yang konsisten seperti mengubah format tanggal menjadi format yang sama

```
import pandas as pd

# Contoh dataset dengan kolom tanggal dalam format yang berbeda
data = {
    'Tanggal': ['2021-08-01', '02-08-2021', '2021/08/03', '20210804']
}
df = pd.DataFrame(data)

# Tampilkan dataset sebelum perubahan format
print("Dataset sebelum perubahan format:")
print(df)

# Ubah format tanggal menjadi format yang sama (YYYY-MM-DD)
df['Tanggal'] = pd.to_datetime(df['Tanggal'], errors='coerce').dt.strftime('%Y-%m-%d')

# Tampilkan dataset setelah perubahan format
print("\nDataset setelah perubahan format:")
print(df)
```

- Memperbaiki data yang salah ketik atau typo

```
import pandas as pd
from sklearn.datasets import load_iris

# Load dataset Iris dari scikit-learn
data = load_iris()
iris_df = pd.DataFrame(data.data, columns=data.feature_names)
iris_df['species'] = data.target_names[data.target]
```

```

# Contoh data dengan salah ketik atau typo
iris_df.iloc[0, 0] = 5.1
iris_df.iloc[1, 1] = 3.6

# Tampilkan dataset sebelum pembersihan data
print("Dataset sebelum pembersihan data:")
print(iris_df)

# Koreksi data salah ketik atau typo
# Misalnya, jika nilai 3.6 pada kolom 'sepal width (cm)' seharusnya 3.0
iris_df.loc[iris_df['sepal width (cm)'] == 3.6, 'sepal width (cm)'] = 3.0

# Tampilkan dataset setelah pembersihan data
print("\nDataset setelah pembersihan data:")
print(iris_df)

```

10.5 Menangani Malformed Record

Malformed Record terjadi saat ketika data tidak memenuhi format atau struktur yang diharapkan.

- Strategi yang dapat dilakukan diantara lain :
 - Menghapus record yang tidak sesuai dengan format atau struktur yang diharapkan
 - Mengubah record yang tidak sesuai dengan format atau struktur yang diharapkan

Pilih strategi yang paling sesuai tergantung pada tipe data dan tujuan analisis.

- Menghapus record yang tidak sesuai dengan format atau struktur yang diharapkan

```

import pandas as pd

# membaca dataset
data = pd.read_csv("nama_file.csv")

# mengecek dan menghapus record yang tidak sesuai dengan format atau struktur yang diharapkan
for i, row in data.iterrows():
    if not format_check(row):
        data.drop(i, inplace=True)

```

```
# menampilkan dataset yang telah diubah
print(data)
```

Kode di atas menggunakan library pandas untuk membaca dataset dari file csv dan melakukan pengecekan format atau struktur pada setiap record dalam dataset dengan fungsi format_check(). Jika record tidak sesuai dengan format atau struktur yang diharapkan, maka record tersebut dihapus dari dataset menggunakan fungsi drop(). Fungsi iterrows() digunakan untuk mengiterasi setiap record dalam dataset. Setelah proses penghapusan selesai, dataset yang telah diubah ditampilkan menggunakan fungsi print(). Penting untuk menyesuaikan kode dengan format atau struktur dataset yang digunakan dan memastikan menggunakan fungsi format_check() yang sesuai.

- Mengubah record yang tidak sesuai dengan format atau struktur yang diharapkan

```
#| code-fold: true
import pandas as pd

# membaca dataset dari file csv
data = pd.read_csv('nama_file.csv')

# fungsi untuk melakukan pengecekan format atau struktur pada setiap record dalam dataset
def format_check(record):
    # implementasi pengecekan format atau struktur pada satu record
    # return True jika format atau struktur sesuai, False jika tidak sesuai

    # melakukan iterasi pada setiap record dalam dataset
    for index, row in data.iterrows():
        # cek apakah format atau struktur record sesuai dengan yang diharapkan
        if not format_check(row):
            # jika tidak sesuai, hapus record dari dataset
            data = data.drop(index)

# menampilkan dataset yang telah diubah
print(data)
```

Pada contoh di atas, dataset dibaca dari file csv menggunakan fungsi read_csv() dari library pandas. Kemudian, dilakukan iterasi pada setiap record dalam dataset menggunakan fungsi iterrows(). Pada setiap iterasi, record dicek dengan fungsi format_check() untuk memastikan bahwa format atau struktur record sesuai dengan yang diharapkan. Jika tidak sesuai, record

tersebut dihapus dari dataset menggunakan fungsi drop(). Setelah proses penghapusan selesai, dataset yang telah diubah ditampilkan menggunakan fungsi print(). Harap diingat bahwa fungsi format_check() harus disesuaikan dengan format atau struktur yang diharapkan pada dataset yang digunakan.

10.6 Kesimpulan

Preprocessing merupakan tahapan penting dalam proses machine learning. Tantangan seperti contoh diatas dapat diatasi dengan berbagai strategi, dengan memilih strategi yang tepat dan pemahaman tipe atau jenis data yang ada akan makin memudahkan dalam melakukan preprocessing.

11 Transformasi Data

Transformasi Data adalah proses pembuatan atau modifikasi variabel atau fitur dari sebuah dataset untuk membuat dataset lebih mudah digunakan dan atau lebih akurat, sehingga dapat membuat model yang lebih akurat.

Konstruksi Data dibagi menjadi empat aktivitas utama, yaitu:

- Rekayasa Fitur
- Imputasi
- Handling Outlier
- Dokumentasi Fitur

11.1 RekayasaFitur

Rekayasa fitur adalah proses penambahan atau modifikasi fitur dengan mengaplikasikan penghitungan matematik, statistika, atau pengetahuan terhadap fitur.

Sebagai contoh, anda dapat membuat fitur baru bernama average atau rata rata yang mengambil nilai dari fitur-fitur lain. Atau anda dapat membuat fitur kategori baru dengan mengolah data dari fitur-fitur lain.

Diharapkan fitur-fitur yang di modifikasi atau ditambah dapat menambah kualitas dataset sehingga dapat menghasilkan model yang lebih akurat dan efisien.

11.1.1 Hands On Coding

Kita akan melakukan proses feature enginnering ke sebuah dataset dibawah ini
Jangan lupa untuk menginstall library pandas menggunakan command `pip install pandas`

```
from spacy import displacy
import spacy
from matplotlib import pyplot as plt
import cv2
from sklearn.preprocessing import MinMaxScaler
import time
```

```
import random
import plotly.express as px
from sklearn.impute import KNNImputer
from keras.optimizers import Adam
from keras.layers import Dense
from keras.models import Sequential
from scipy.stats import pearsonr
import math
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df = pd.DataFrame({'Country': ['Afghanistan', 'Cameroon', 'Indonesia', 'Guatemala'],
                   'UrbanPopulation': [100000000, 100000000, 100000000, 100000000],
                   'RuralPopulation': [8738685, 28829316, 11242817, 117874900, 8119648], 'TotalPopulation': [100000000, 100000000, 100000000, 100000000, 100000000]}
df.head()
```

Dataset ini mempunyai 4 fitur:

- Country, Nama negara
 - UrbanPopulation, populasi manusia yang hidup di daerah urban (perumahan kota)
 - RuralPopulation, populasi manusia yang hidup di daerah rural (pinggiran kota)
 - SlumPopulation, populasi manusia yang hidup di daerah slum (pemukiman kumuh)

Pertama kita akan menghitung presentase jumlah populasi yang hidup di slum (pemukiman kumuh) dari populasi yang hidup di urban (perumahan kota). Nilai ini didapat menggunakan rumus:

$$SlumPercentage = \frac{SlumPopulation}{UrbanPopulation} * 100$$

```
df['SlumPopulation'] = round(
    (df['SlumPopulation']/(df['UrbanPopulation']))*100, 2)
df.head()
```

Selanjutnya, kita menggabungkan dua fitur (urban dan rural) menjadi satu, dan mengubah nilainya dari jumlah penduduk ke presentase penduduk. Rumusnya cukup simple:

$$UrbanPercentage = \frac{UrbanPopulation}{UrbanPopulation + RuralPopulation} * 100$$

```
df['UrbanPopulation'] = round(  
    (df['UrbanPopulation']/(df['UrbanPopulation']+df['RuralPopulation']))*100, 2)  
df.head()
```

Dan untuk sentuhan akhir, kita akan hapus kolom SlumPopulation karena nilainya sudah ter-representasikan di kolom UrbanPopulation

```
df = df.drop(columns=['RuralPopulation'])  
df.head()
```

Hasilnya, nilai atau value di dataset lebih mudah dibaca, dan dapat direpresentasikan menggunakan fitur yang lebih sedikit. Dataset sudah siap untuk diproses lebih lanjut.

11.2 Imputasi

Imputasi adalah proses penggantian nilai data yang hilang dengan data yang baru. Seperti contoh rekayasa fitur sebelumnya, nilai NaN termasuk data yang perlu kita olah.

Dalam bab ini kita akan mempelajari beberapa hal terkait imputasi antara lain:

- Jenis-jenis imputasi

- Teknik imputasi

11.2.1 Jenis-jenis Imputasi

Jenis data yang hilang dikelompokkan menjadi 3, yaitu - Missing completely at random (MCAR) - Missing at random (MAR) - Missing not at random (MNAR)

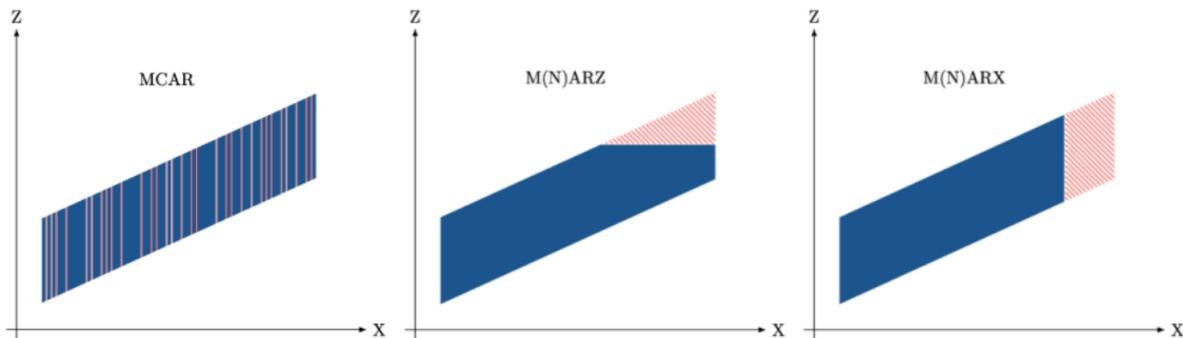


Figure 11.1: Gambar1. Jenis-jenis Imputasi

11.2.1.1 Missing Completely At Random (MCAR)

Jika probabilitas hilangnya data dalam suatu fitur **sama antara satu data dengan yang lain**. Asumsi ini dapat diuji dengan memisahkan data yang hilang dan yang lengkap serta

memeriksa karakteristik data. Jika karakteristik data tidak sama untuk kedua fitur, asumsi MCAR tidak berlaku

Complete data		Incomplete data	
Age	IQ score	Age	IQ score
25	133	25	
26	121	26	121
29	91	29	91
30	105	30	
30	110	30	110
31	98	31	
44	118	44	118
46	93	46	93
48	141	48	
51	104	51	
51	116	51	116
54	97	54	

Figure 11.2: Gambar2. Contoh MCAR

11.2.1.2 Missing At Random (MAR)

Kemungkinan data yang hilang **dipengaruhi oleh variabel lain, namun tidak dipengaruhi oleh variabel yang hilang**. Sebagai contoh, untuk data di samping, hanya peserta dengan umur yang dibawah 31 yang nilainya hilang. Berarti fitur age mempengaruhi probabilitas missing data IQ score.

Complete data		Incomplete data	
Age	IQ score	Age	IQ score
25	133	25	
26	121	26	
29	91	29	
30	105	30	
30	110	30	
31	98	31	
44	118	44	118
46	93	46	93
48	141	48	141
51	104	51	104
51	116	51	116
54	97	54	97

Figure 11.3: Gambar3. Contoh MAR

11.2.2 Missing Not At Random

Kemungkinan data yang hilang **tidak dipengaruhi oleh fitur lain, namun dipengaruhi oleh fitur pada data yang hilang**. Sebagai contoh, untuk data di samping, ada kemungkinan bahwa data IQ score yang hilang hanya data yang nilainya dibawah 110. Sedangkan variabel age tidak berpengaruh atas hilangnya data IQ score karena age yang kecil dan besar sama-sama mempunyai data yang hilang

Complete data		Incomplete data	
Age	IQ score	Age	IQ score
25	133	25	133
26	121	26	121
29	91	29	
30	105	30	
30	110	30	110
31	98	31	
44	118	44	118
46	93	46	
48	141	48	141
51	104	51	
51	116	51	116
54	97	54	

Figure 11.4: Gambar4. Contoh MNAR

11.2.3 Teknik-Teknik Imputasi

Perlu diingat bahwa jika 70% data hilang/missing, maka semua fitur(kolom) dan data (row) harus dihapus.

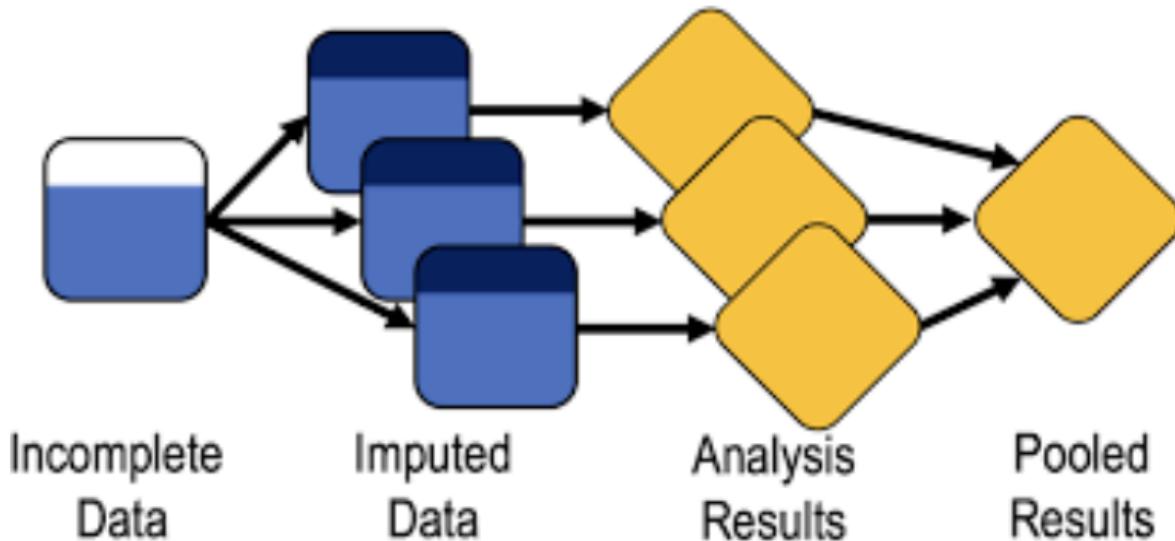


Figure 11.5: Gambar5. Flow Proses Imputasi

Data yang hilang harus dimutusakan berdasarkan jenis datanya, yaitu :

Numerik

- Mean/Median
- Arbitrary
- End of tail
- Regresi
- KNN

Kategorik

- Frequent/Modus
- KNN

11.2.3.1 Mean

Dataset imputasi mean adalah metode untuk mengisi nilai yang hilang dalam dataset dengan menggunakan nilai rata-rata (mean) dari variabel yang bersangkutan. Ketika ada nilai yang hilang dalam suatu variabel dalam dataset, imputasi mean menggantikan nilai-nilai yang hilang tersebut dengan nilai rata-rata dari seluruh nilai yang ada dalam variabel tersebut.

Kelebihan

- Mudah dan cepat.
- Bekerja efektif untuk dataset numerik berukuran kecil.
- Cocok untuk variabel numerik.
- Cocok untuk data missing completely at random (MCAR).
- Dapat digunakan dalam produksi (mis. dalam model deployment)

Kekurangan

- Tidak memperhitungkan korelasi antar fitur, berfungsi pada tingkat kolom.
- Kurang akurat.
- Tidak memperhitungkan probabilitas/ketidakpastian.
- Tidak cocok utk >5% missing data.

11.2.3.1.1 Hands On Coding

Pertama kita akan buat sebuah dataset menggunakan pandas dataframe

```
# buat dataset dengan format dataframe
df = pd.DataFrame({'age': [25, 26, 29, 30, 30, 31, 44, 46],
                    'IQ': [np.NaN, 121, 91, np.NaN, 110, np.NaN, 118, 93]})
display(df)
```

Lalu kita akan hitung mean atau rata rata dari dataset

```
mean = df['IQ'].mean()
print(f'Mean: {mean}, dibulatkan menjadi {round(mean)}')
display(df)
```

Lalu kita masukkan nilai mean tersebut ke data yang kosong

```
# masukkan nilai mean ke missing value
df['IQ'] = df['IQ'].fillna(round(mean))
display(df)
```

11.2.3.2 Arbiter

Teknik imputasi arbiter (arbiter imputation) adalah metode untuk mengisi nilai yang hilang dalam dataset dengan menggunakan hasil gabungan dari beberapa metode imputasi yang berbeda. Dalam metode ini, beberapa teknik imputasi yang berbeda diterapkan pada dataset yang sama, dan hasil dari setiap teknik imputasi digabungkan menjadi satu nilai yang digunakan untuk mengisi nilai yang hilang.

Kelebihan - Sangat mudah dan cepat

- Cocok untuk missing dataset dengan asumsi tidak missing at random

Kekurangan - Mengganggu variansi dan distribusi variable original

- Dapat membentuk outlier
- Semakin besar nilai arbitrary, maka semakin besar distorsi

11.2.3.2.1 Hands On Coding

Kita akan gunakan dataset yang sama seperti sebelumnya

```
# buat dataset dengan format dataframe
df = pd.DataFrame({'age': [25, 26, 29, 30, 30, 31, 44, 46],
                    'IQ': [np.NaN, 121, 91, np.NaN, 110, np.NaN, 118, 93]})
display(df)
```

Cukup masukkan nilai arbiter ke data yang kosong. Dalam koding ini, nilai arbiter adalah 130.

```
df['IQ'] = df['IQ'].fillna(130)
display(df)
```

11.2.3.3 End Of Tail

Teknik imputasi end of tail adalah metode untuk mengisi nilai yang hilang dalam dataset dengan menggunakan nilai ekstrem (tail) dari distribusi data yang ada. Metode ini didasarkan pada asumsi bahwa nilai yang hilang cenderung berada di ekor distribusi data. Namun untuk teknik ini kita harus mengikuti sebuah ketentuan khusus, yaitu:

Jika distribusi data bersifat normal, maka gunakan rumus :

$$\sigma = \sqrt{\frac{\sum |x - \mu|^2}{N}}$$

Jika distribusi data bersifat skewed, maka gunakan rumus IQR proximity

$$IQR = Q_3 - Q_1$$

11.2.3.3.1 Distribusi Normal

Data yang terdistribusi secara normal, juga dikenal sebagai distribusi Gaussian atau kurva lonceng, mengacu pada distribusi statistik di mana titik-titik data secara simetris terdistribusi di sekitar nilai rata-rata, menciptakan kurva berbentuk lonceng yang khas.

Untuk menghitung data yg mempunyai distribusi normal, kita harus mengetahui standar deviasinya. Berikut rumus untuk menghitung standar deviasi :

$$\mu + 3 * \sigma$$

Keterangan:

= Standar Deviasi

Σ = jumlah

x = data yang dihitung

= Mean/rata rata

N = Jumlah data

11.2.3.3.2 Distribusi Normal - Hands On Coding

Pertama kita akan buat sebuah dataset yang cukup besar jika dibandingkan dengan dataset yang diatas.

Jangan lupa untuk menginstall library yang dibutuhkan :

```
- pip install numpy  
- pip install pandas  
- pip install matplotlib  
- pip install keras - pip install tensorflow
```

```
age = [25, 26, 29, 30, 30, 31, 44, 46, 22, 33,  
      35, 27, 21, 23, 45, 47, 41, 38, 37, 21, 24]  
data = [85, 90, 95, 95, 100, np.nan, 100, 110, 105, 105, 110,  
       np.nan, 110, 110, 115, 115, 115, 120, np.nan, 125, 130]  
  
# make a dataframe  
df = pd.DataFrame({'age': age, 'IQ': data})  
display(df)
```

Selanjutnya kita akan mengecek jenis distribusi data menggunakan library matplotlib

```
data = [85, 90, 95, 95, 100, np.nan, 100, 110, 105, 105, 110,  
       np.nan, 110, 110, 115, 115, 115, 120, np.nan, 125, 130]  
  
plt.hist(data, bins=5)
```

```
plt.xlabel('Data')
plt.ylabel('Frequency')
plt.show()
```

Dapat dilihat bahwa grafik histogram membentuk seperti gunung atau lonceng, dengan puncak tepat di tengah-tengah grafik. Inilah salah satu karakteristik dataset dengan distribusi normal.

Selanjutnya, kita akan mencari mean dari dataset

```
# nilai kosong dihilangkan dari data terlebih dahulu
data = [85, 90, 95, 95, 100, 100, 110, 105, 105,
        110, 110, 110, 115, 115, 115, 120, 125, 130]
mean = np.mean(data)
print(f'Mean dari dataset adalah: ', mean)
```

Lalu, kita akan menghitung jarak antara nilai x dan mean

```
total = 0
for i in data:
    calc = (107.5-i)**2
    total = total + calc

print(f'jarak antara nilai x dan mean adalah ', total)
```

Mari kita hitung standar deviasi nya

```
# math.sqrt adalah fungsi untuk melakukan operasi akar pangkat
# round adalah fungsi untuk membulatkan hasil operasi)
stddev = round(math.sqrt(total/len(data)), 2)
print(f'nilai standar deviasi adalah ', stddev)
```

Hitung nilai imputasi End of Tail

```
imp = round(mean + 3 * stddev, 1)
print(f'nilai imputasi end of tail adalah ', imp)
```

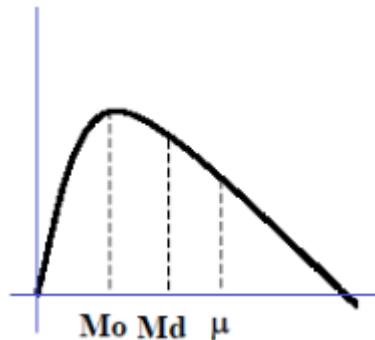
Dari kalkulasi nilai imputasi end of tail, diperoleh nilai 142.2.
Kita akan memasukkan nilai ini ke dalam dataset

```
df['IQ'] = df['IQ'].fillna(imp)
display(df)
```

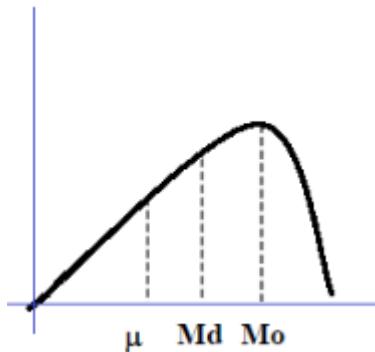
11.2.3.3.3 Distribusi Skewed

Distribusi skew atau skewness mengacu pada karakteristik asimetri dalam distribusi data. Dalam distribusi skew, ekor distribusi data cenderung condong ke salah satu sisi, baik ke kanan (positif) atau ke kiri (negatif), dibandingkan dengan pusat distribusi.

Dalam distribusi skew positif, ekor distribusi condong ke kanan, sementara nilai-nilai yang lebih kecil cenderung berada di sebelah kiri. Ini menghasilkan ekor yang panjang di sisi kanan distribusi.



Dalam distribusi skew negatif, ekor distribusi condong ke kiri, dengan nilai-nilai yang lebih besar cenderung berada di sebelah kiri. Ini menghasilkan ekor yang panjang di sisi kiri distribusi. Nilai rata-rata akan lebih kecil daripada median dalam distribusi ini.



11.2.3.3.4 Distribusi Skewed - Hands On Coding

Pertama-tama mari kita buat data baru yang terdiri dari 21 data dengan 3 missing value. Tugas kita adalah mengisi missing value dengan imputasi end of tail.

```
age = [25, 26, 29, 30, 30, 31, 44, 46, 22, 33,  
      35, 27, 21, 23, 45, 47, 41, 38, 37, 21, 24]  
data = [125, 130, 125, 95, 115, np.nan, 100, np.nan, 130, 110,  
       90, 110, 120, 115, 105, 85, 115, 110, 120, 100, np.nan]
```

```
df = pd.DataFrame({'age': age, 'IQ': data})
display(df)
```

Selanjutnya kita akan mengecek jenis distribusi data menggunakan library matplotlib

```
data = [125, 130, 125, 95, 115, 100, 130, 110, 90,
        110, 120, 115, 105, 85, 115, 110, 120, 100]

fig = px.histogram(df, x='IQ')
fig.show()
```

Distribusi data adalah skew negatif karena puncak dari data berada di sebelah kanan titik tengah. Mari kita hitung nilai imputasi menggunakan rumus IQR.

Inter-Quartile-Range (IQR) adalah sebuah nilai yang digunakan untuk mengukur sebaran data dalam sebuah distribusi.

$$IQR = Q_3 - Q_1$$

$$IQR_{max} = Q_3 + 3 * IQR$$

$$IQR_{min} = Q_1 + 3 * IQR$$

Pertama-tama, kita akan hitung nilai precentile dari dataset, yang dapat kita kalkulasi dengan mudah menggunakan fungsi np.percentile dari library numpy.

```
median = np.median(data)
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)

print(f'Median: {median}')
print(f'Q1: {q1}')
print(f'Q3: {q3}')
```

Lalu kita akan menghitung nilai IQR, IQRmin, dan IQRmax

```
iqr = q3 - q1
iqrmin = q1 + 3 * iqr
iqrmax = q3 + 3 * iqr

print(f'IQR: {iqr}')
print(f'IQRmin: {iqrmin}')
print(f'IQRmax: {iqrmax}')
```

Anda boleh memilih nilai diantara IQRmin dan IQRmax. Namun untuk contoh ini, kita akan ambil nilai IQRmax saja, lalu kita masukkan ke dataset.

```
df['IQ'] = df['IQ'].fillna(iqrmax)
display(df)
```

11.2.3.4 Regresi Linier

Teknik imputasi regresi adalah metode untuk mengisi nilai kosong menggunakan algoritma regresi. Algoritma regresi akan memprediksi nilai kosong berdasarkan hubungan fitur nilai kosong dengan fitur lainnya.

Kelebihan - Sederhana dan mudah dipahami

- Menggabungkan hubungan antar variabel
- Cocok untuk data yang besar dan bersifat numerik

Kekurangan - Hanya berlaku untuk data yang linear

- Sensitif terhadap outlier
- Tidak dapat menangani data non-numerik
- Bergantung kepada dua fitur

11.2.3.4.1 Regresi Linier - Hands On Coding

Kelemahan utama dari teknik regresi ini adalah dataset harus mempunyai distribusi linear. Mengambil contoh dataset age (umur) dan IQ. Jika dataset tersebut adalah linear, semakin tinggi umur seseorang, maka semakin tinggi IQ orang tersebut.

Maka dari itu, kita harus mengecek jenis data yang akan kita kerjakan, apakah data tersebut bersifat linear atau tidak. Untuk mengecek dataset, kita gunakan grafik scatter plot.

```
age = [25, 26, 29, 30, 30, 31, 44, 46, 22, 33,
       35, 27, 21, 23, 45, 47, 41, 38, 37, 21, 24]
data = [125, 130, 125, 95, 115, np.nan, 100, np.nan, 130, 110,
        90, 110, 120, 115, 105, 85, 115, 110, 120, 100, np.nan]

df = pd.DataFrame({'age': age, 'IQ': data})

# hapus data NaN
df = df.dropna()

# buat grafik scatter
fig = px.scatter(df, x='age', y='IQ',
```

```
color='age', hover_data=['age', 'IQ'])  
fig.show()
```

Secara garis besar, scatter plot membentuk sebuah garis diagonal dari pojok kiri atas ke pojok kiri bawah.

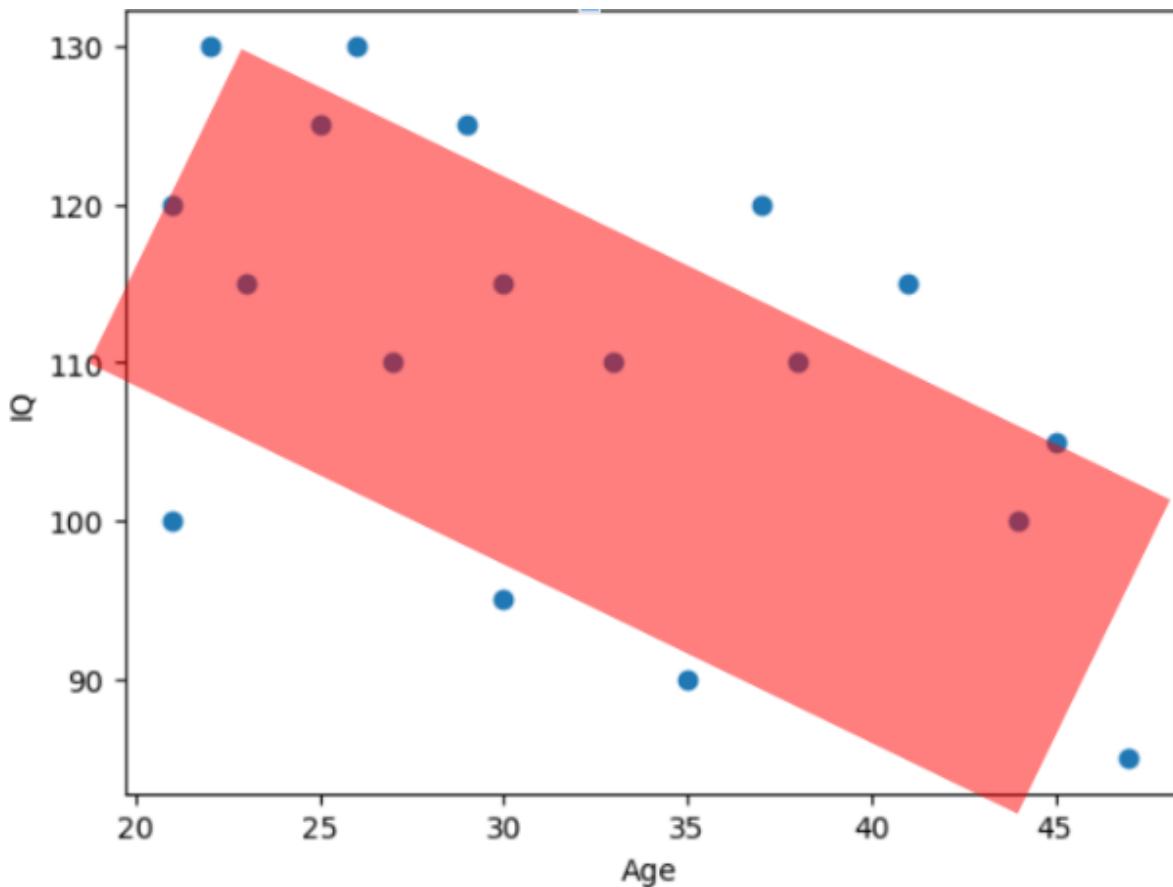


Figure 11.6: Gambar9. Data bersifat Linear

Untuk menambah keyakinan kita bahwa dataset bersifat linear, kita dapat melakukan pengecekan dataset menggunakan pearson correllation. Kita harus menginstall library scipy terlebih dahulu dengan menjalankan perintah `pip install scipy` di terminal atau command prompt.

Pearson correlation adalah sebuah rumus yang berfungsi untuk menghitung kekuatan dan arah hubungan antara dua variable. nilai -1 mengindikasikan bahwa korelasi bersifat negatif, semakin kecil nilai variable x, maka nilai variable y akan semakin besar. nilai 0 mengindikasikan bahwa tidak ada korelasi linear antara variable. nilai 1 mengindikasikan bahwa korelasi bersifat positif.

```
corr, _ = pearsonr(df['age'], df['IQ'])
print(f'Pearson's correlation: {round(corr, 2)}')
```

nilai pearson -0.53 menandakan bahwa hubungan antara variabel umur dan iq adalah linear negatif yang mempunyai intensitas lemah.

Selanjutnya, kita akan membuat model regresi. Sebelumnya, kita harus menginstall library keras dengan cara menjalankan perintah pip install keras di terminal atau command prompt.

```
model = Sequential()
model.add(Dense(1, input_shape=(1,)))
model.compile(Adam(learning_rate=0.8), 'mean_squared_error')

model.fit(df['age'], df['IQ'], epochs=1000, verbose=0)

pred = model.predict(df['age'])

plt.scatter(df['age'], df['IQ'])
plt.plot(df['age'], pred, color='red')
plt.xlabel('Age')
plt.ylabel('IQ')
plt.show()
```

Lalu buat hasil prediksi dari model regresi. Nilai yang kosong adalah IQ dengan umur 27, 31, dan 37. Maka kita masukkan ketiga nilai tersebut ke model untuk di prediksi nilai IQ nya.

```
hasilprediksi = model.predict([27, 31, 37]).round(0).astype(int)
print(hasilprediksi)
```

Hasil prediksi menunjukkan angka 115, 112, dan 107. Maka kita akan masukkan nilai IQ tersebut kedalam dataset

```
age = [25, 26, 29, 30, 30, 31, 44, 46, 22, 33,
       35, 27, 21, 23, 45, 47, 41, 38, 37, 21, 24]
data = [125, 130, 125, 95, 115, np.nan, 100, np.nan, 130, 110,
        90, 110, 120, 115, 105, 85, 115, 110, 120, 100, np.nan]

data[5] = hasilprediksi[0][0]
data[7] = hasilprediksi[1][0]
data[20] = hasilprediksi[2][0]
df = pd.DataFrame({'age': age, 'IQ': data})
```

```
display(df)
```

11.2.3.5 Frequent

Imputasi frequent adalah teknik imputasi yang hanya bisa digunakan di jenis data kategorik. Kita mengambil nilai kategorik yang paling sering muncul, dan memasukkannya ke data yang kosong.

Kelebihan - Cocok untuk data dengan missing at random.

- Mudah dan cepat diterapkan.
- Cocok utk data yang memiliki skew
- Dapat digunakan dalam produksi (mis. dalam model deployment).

Kelemahan - Mendistorsi relasi label dengan frekuensi tertinggi vs variabel lain.

- Menghasilkan over-representation jika banyak data yang missing.

11.2.3.5.1 Frequent - Hands On Coding

Langkah pertama adalah memuat dataset yang mempunyai jenis data kategorik. Oleh karena itu, fitur IQ kita ganti oleh nilai ‘rendah’, ‘sedang’, dan ‘tinggi’

```
age = [25, 26, 29, 30, 30, 31, 44, 46, 22, 33,  
      35, 27, 21, 23, 45, 47, 41, 38, 37, 21, 24]  
IQ = ['rendah', 'sedang', 'sedang', np.nan, 'sedang', 'rendah', np.nan, 'tinggi', 'sedang'  
      'rendah', 'sedang', 'tinggi', 'sedang', np.nan, 'rendah', 'tinggi', 'sedang', 'tinggg  
  
df = pd.DataFrame({'age': age, 'IQ': IQ})  
display(df)
```

Lalu kita hitung frekuensi dari data kategorik

```
freq = df['IQ'].value_counts()  
print(freq)
```

Kategori sedang mempunyai frekuensi paling tinggi dengan nilai 8. Jadi, nilai imputasi yang akan kita gunakan adalah ‘sedang’

Selanjutnya, kita masukkan data ‘sedang’ ke dataset.

```
df['IQ'] = df['IQ'].fillna('sedang')  
display(df)
```

11.2.3.6 K-Nearest Neighbor (KNN)

Imputasi menggunakan K-Nearest Neighbors (KNN) adalah sebuah metode untuk mengisi data kosong dengan mempertimbangkan nilai terdekat dari fitur lain di kategori yang sama.

Kelebihan - Lebih akurat vs mean/median/most frequent.

Kekurangan - Biaya komputasi mahal (karena KNN bekerja dengan menyimpan seluruh dataset pelatihan dalam memori).

-Sensitif terhadap outlier dalam data (tidak seperti SVM).

11.2.3.6.1 K-Nearest Neighbor (KNN) - Hands On Coding

Dataset yang kita gunakan sama dengan contoh diatas

```
age = [25, 26, 29, 30, 30, 31, 44, 46, 22, 33,  
      35, 27, 21, 23, 45, 47, 41, 38, 37, 21, 24]  
IQ = ['rendah', 'sedang', 'sedang', np.nan, 'sedang', 'rendah', np.nan, 'tinggi', 'sedang'  
      'rendah', 'sedang', 'tinggi', 'sedang', np.nan, 'rendah', 'tinggi', 'sedang', 'tinggi'  
  
df = pd.DataFrame({'age': age, 'IQ': IQ})  
display(df)
```

Algoritma KNN tidak bisa menghitung data berjenis string, maka kita harus konversi dari string (huruf/kata) ke integer (angka).

```
df['IQ'] = df['IQ'].map({'rendah': 1, 'sedang': 2, 'tinggi': 3})  
display(df)
```

Lalu kita membuat model KNN. Hasil dari KNN bisa langsung dimasukkan ke dataset.

```
imputer = KNNImputer(n_neighbors=3)  
# isi missing value dengan KNN, lalu dibulatkan  
df = pd.DataFrame(np.round(imputer.fit_transform(df)), columns=df.columns)  
display(df)
```

11.2.3.7 Kesimpulan

- Sebelum melakukan imputasi kita harus mengetahui jenis missing data, tipe data, dan distribusi data

- Tidak ada metode imputasi yang sempurna, masing masing teknik mempunyai kelebihan dan kelemahan yang unik
- Distribusi data sangat berpengaruh terhadap efisiensi imputasi
- Imputasi dapat dilakukan menggunakan function dari library sklearn, feature_engine, dan keras. Namun akan jauh lebih baik jika kita menghitung/coding secara manual untuk mengetahui cara kerja algoritma tsb sebelum menggunakan function dari library.

11.3 Handling Outlier

Outlier adalah sebuah data yang mempunyai pola atau letak yang menyimpang sangat jauh dari rata rata dataset atau pola yang diharapkan dari sebuah dataset.

Outlier dapat terjadi karena berbagai alasan, seperti kesalahan pengukuran, kesalahan entri data, variasi alami, atau peristiwa langka. Outlier dapat memiliki dampak signifikan pada analisis statistik, model pembelajaran mesin, dan interpretasi data, yang dapat mengarah pada hasil yang bias atau kesimpulan yang tidak akurat.

Outlier harus kita atasi agar data bisa kita proses secara efisien.

11.3.1 Deteksi Outlier

Outlier dapat dideteksi menggunakan beberapa metode, antara lain

11.3.1.1 Visualisasi

Adalah sebuah teknik untuk memvisualisasikan sebuah data menjadi suatu bentuk yang dapat dilihat secara menyeluruh sehingga kita dapat menganalisa bentuk data, ukuran data, data point dan mendekksi outlier.

Beberapa bentuk visualisasi yang sering digunakan untuk mendekksi outlier yaitu - Histogram

- Scatter plot
- Box plot

Kita akan menggunakan library plotly express untuk melakukan visualisasi, jadi jangan lupa untuk menginstall plotly dengan menjalankan pip `install plotly` di terminal atau cmd

Dataset yang kita gunakan adalah dataset lagu yang diambil dari spotify. Sumber dari dataset adalah <https://www.kaggle.com/datasets/vatsalmavani/spotify-dataset>

Namun karena dataset ukurannya sangat besar, maka kita akan gunakan subset dataset berjumlah 100 data yang diambil secara random. Dataset versi ini dapat

```
di download di https://github.com/rif42/AssociateDataScientist/blob/master/Module7-  
AssociateDataScientist/data_sampled_100.csv
```

Jika sudah di download, masukkan data ke dalam directory lalu muat dataset tersebut.

```
data = pd.read_csv('data_sampled_100.csv')  
data.head()
```

Dataset ini berisi lagu-lagu yang ada di Spotify. Setiap lagu mempunyai fitur yang unik seperti popularitas, durasi, loudness, acousticness, speechiness, danceability, dll. Anda bisa melihat deskripsi data secara detail di <https://developer.spotify.com/documentation/web-api/reference/get-several-audio-features>

Selanjutnya kita akan lakukan proses visualisasi data menggunakan plot histogram dengan library plotly. Fitur yang kita visualisasikan adalah loudness atau volume lagu terhadap popularitas dari lagu yang ada di dataset.

Berikut diagram scatter plot dari dataset:

```
fig = px.scatter(data, x='loudness', y='popularity', color='loudness',  
                  hover_data=['artists', 'name', 'loudness', 'popularity'])  
fig.show()
```

Berikut diagram histogram dari dataset:

```
fig = px.histogram(data, x='loudness')  
fig.show()
```

Berikut diagram box plot dari dataset:

```
fig = px.box(data, x='loudness')  
fig.show()
```

Ada beberapa data yang bisa disebut outlier, salah satunya adalah lagu *Thursday Afternoon - 2005 digital remaster*, oleh Brian Eno yang mempunyai nilai loudness -31.8808. Alasannya adalah tingkat loudness nya jauh lebih kecil daripada yang lain.

11.3.2 Kategori Outlier

Variate dan univariate adalah dua jenis data dalam statistik. Outlier adalah observasi atau nilai yang secara signifikan berbeda dari pola atau pola umum data yang lain.

11.3.2.1 Variate Data

Variate data merujuk pada set data yang terdiri dari beberapa variabel atau fitur. Contoh umum variate data adalah dataset yang terdiri dari beberapa kolom, di mana setiap kolom mewakili variabel yang berbeda. Misalnya, jika kita memiliki dataset tentang mahasiswa yang mencakup variabel seperti tinggi, berat badan, dan usia, maka kita memiliki variate data. Outlier dalam variate data merujuk pada observasi atau nilai yang di luar kisaran yang diharapkan dalam setidaknya satu variabel.

11.3.2.2 Univariate Data

Univariate data merujuk pada set data yang hanya memiliki satu variabel atau fitur. Contoh umum univariate data adalah dataset yang hanya terdiri dari satu kolom, seperti data tinggi badan seseorang. Outlier dalam univariate data merujuk pada observasi atau nilai yang sangat ekstrem atau jauh dari rentang nilai yang diharapkan.

```
df = pd.DataFrame({'age': [25, 26, 29, 30, 30, 31, 44, 46],  
                  'IQ': [np.NaN, 121, 91, np.NaN, 110, np.NaN, 118, 93]})  
display(df)
```

Masih ingat contoh data di imputasi data diatas? Data berisi fitur umur dan nilai IQ. Masing masing fitur hanya mempunyai 1 buah nilai. Inilah yang dimaksud dengan univariate data; data di dalam fitur hanya mempunyai satu jenis nilai.

11.3.3 Mengatasi Outlier

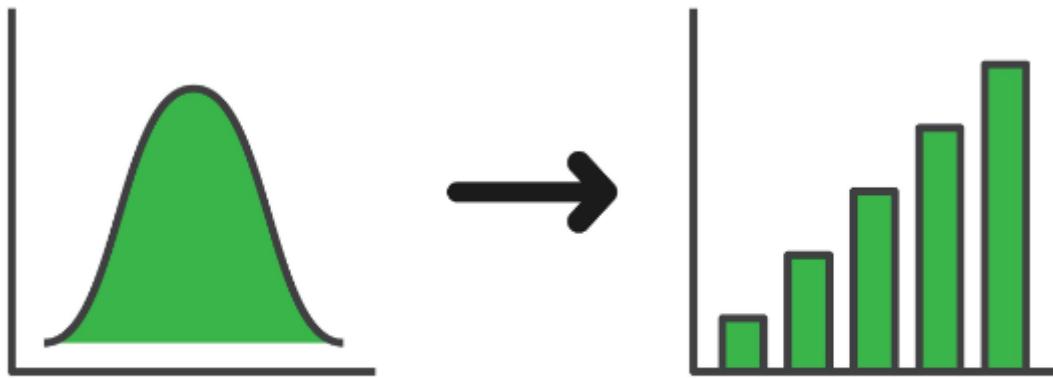
Tidak semua outlier harus dihapus atau dihilangkan. Pada contoh data lagu di spotify, meskipun letaknya jauh dari yang lain, data ini masih valid sebagai lagu. Karena adalah lagu *Thursday Afternoon - 2005 digital remaster*, oleh Brian Eno mempunyai genre instrumental/ambient. Umumnya, tujuan utama genre ini adalah sebagai music background, sehingga tidak diperlukan vokal atau melodi yang keras.

Namun terkadang outlier harus dihilangkan atau dihapus karena nilainya terlalu jauh dengan yang lain, yang dapat merubah value dari dataset secara keseluruhan, lalu dapat merubah hasil dari training data.

11.3.3.1 Discretization/Binning

Discretization atau binning adalah proses mengubah data kontinu menjadi data diskrit dengan cara membagi rentang nilai kontinu menjadi beberapa interval atau kelompok yang disebut

“bin” atau “bucket”. Tujuan utama dari discretization adalah mengurangi kompleksitas data kontinu dengan mengelompokkan nilainya ke dalam kategori atau range tertentu.



Discretization Process

Figure 11.7: Gambar10. Grafik Proses Binning Dataset

Kelebihan - Dapat diterapkan pada data kategorik dan numerik.

- Model lebih robust dan mencegah overfitting.

Kekurangan - Meningkatnya biaya kinerja perhitungan.

- Mengorbankan informasi.
- Untuk kolom data numerik, dapat menyebabkan redundansi untuk beberapa algoritma.
- Untuk kolom data kategorik, label dengan frekuensi rendah berdampak negatif pada robustness model statistik.

11.3.3.1.1 Discretization/Binning - Hands On Coding

Kita akan menggunakan dataset lagu spotify dengan fitur loudness untuk melakukan proses binning. Binning dilakukan menggunakan fungsi `pd.cut`. Fungsi ini akan membagi semua nilai yang ada di dalam sebuah fitur menjadi 3 (atau angka lain yang anda inginkan)

```
# muat data
df = pd.read_csv('./data_sampled_100.csv')

# binning data menjadi 5 kategori
df['loudness'] = pd.cut(df['loudness'], bins=3, labels=[

    'sunyi', 'standar', 'bising'])

# urutkan data berdasarkan fitur loudness
```

```

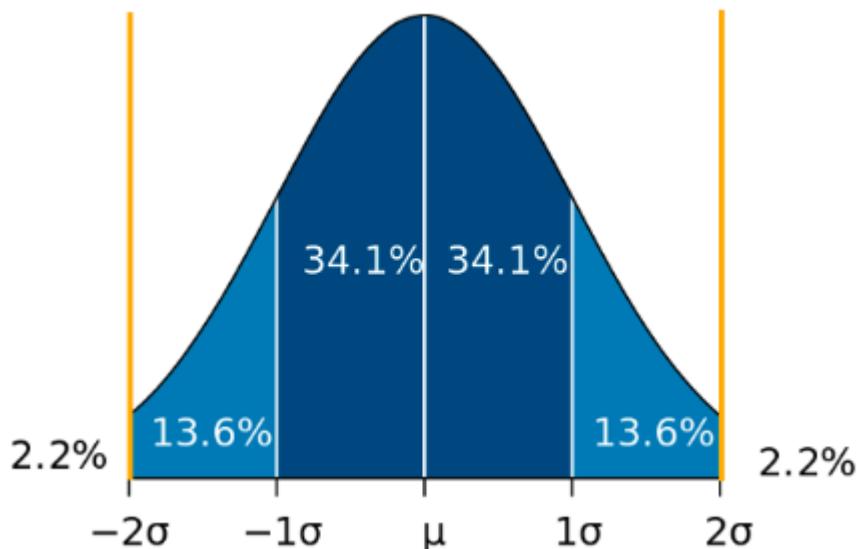
df_sorted = df.sort_values(by='loudness')

# visualisasikan data menggunakan histogram
fig = px.histogram(df_sorted, x='loudness')
fig.show()

```

11.3.3.2 Trimming

Trimming adalah proses penghapusan data yang dianggap sebagai outlier. Trimming biasanya dilakukan berdasarkan presentase data yang akan di trim, contohnya 5%.



Kelebihan

- Cepat dan mudah
- Dapat memperbaiki rata-rata data

Kekurangan

- Hilangnya data yang dapat mengandung informasi
- Dapat menyebabkan bias terhadap variansi data

11.3.3.2.1 Trimming - Hands On Coding

Sangat penting untuk melakukan proses trimming hanya pada data tidak dibutuhkan/invalid. Oleh karena itu, proses trimming seharusnya dimulai dari angka yang sangat kecil, semisal 0.0001%. Jika outlier masih nampak, maka kita tambah presentasenya sedikit demi sedikit.

Dataset yang kita gunakan masih sama, yaitu lagu spotify dengan fitur loudness. Namun karena ukuran dataset kita cukup kecil, maka kita bisa gunakan presentase trimming 1%.

Pertama kita muat data, lalu kita sortir dataset.

```
df = pd.read_csv('./data_sampled_100.csv')
df_sorted = df.sort_values(by='loudness')
df.head()
```

Selanjutnya, kita ambil 1% dari data tersebut, hanya dari **tail** atau ujung belakang dari dataset. Nilai ini nantinya akan menjadi batas dari trimming kita.

```
q = df['loudness'].sort_values().quantile(0.01)
print(q)
```

Jadi batas dari trimming kita adalah -27.3332. Selanjutnya kita akan hilangkan data yang melebihi nilai ini.

```
df = df[df['loudness'] > q]
fig = px.scatter(df, x='loudness', y='popularity', color='loudness',
                  hover_data=['artists', 'name', 'loudness', 'popularity'])
fig.show()
```

Bisa dilihat bahwa lagu Thursday Afternoon - 2005 digital remaster, oleh Brian Eno yang mempunyai loudness kurang dari batas trimming sudah hilang.

11.3.3.3 Winsorizing

Winsorizing adalah proses penggantian data outlier dengan nilai-nilai yang berada dalam distribusi yang ditentukan.

Kelebihan

- Mempertahankan informasi
- Mengurangi efek dari outlier
- Mudah diimplementasikan

Kekurangan

- Dapat menghasilkan bias
- Pemilihan presentil dapat mempengaruhi hasil analisis data

11.3.3.3.1 Winsorizing - Hands On Coding

Kita akan menggunakan dataset dan fitur yang sama, yaitu lagu spotify dengan fitur loudness. Pertama kita akan muat data, lalu tentukan batas data yang akan kita winsorize. Kita akan menggunakan nilai yang sama dengan metode sebelumnya(trimming) yaitu 1% dan batas trimming -27.3332.

```
df = pd.read_csv('./data_sampled_100.csv')
df_sorted = df.sort_values(by='loudness')
df.head()
```

Perbedaan utama antara trimming dan winsorizing adalah, di proses trimming, nilai dibawah batas dihilangkan, namun di proses winsorizing, nilai dibawah batas digantikan dengan nilai diantara quartil 1 dan quartil 3.

Untuk itu, kita akan menggunakan box plot karena box plot sudah memberikan nilai high fence dan low fence secara langsung.

```
fig = px.box(df, x='loudness')
fig.show()
```

Selanjutnya, kita akan menghitung berapa banyak data yang ada dibawah batas trimming

```
q = df['loudness'].sort_values().quantile(0.01)
outlier = df[df['loudness'] < q]
outlier
```

Diketahui ada 1 buah outlier. Selanjutnya kita akan menentukan nilai q1 dan q3 sebagai batasan nilai winsorizing kita.

```
q1 = np.percentile(df['loudness'], 25)
q3 = np.percentile(df['loudness'], 75)
print(f'q1 = ', q1)
print(f'q3 = ', q3)
```

Nilai q1 dan q3 ini akan kita gunakan untuk memberi batasan terhadap angka random yang akan kita generate sebagai nilai winsorizing kita.

```
# nilai n menyesuaikan jumlah data outlier
outliercount = len(outlier)

# generate random number
random.seed(time.time_ns())
```

```

winsorized_outlier = [random.randint(int(q1), int(q3))
                      for i in range(outliercount)]
print(winsorized_outlier)

```

Ingat, nilai ini di generate secara random, jadi nilai akan berubah setiap code di run. Selanjutnya, kita akan memasukkan nilai winsorizing ke outlier, lalu masukkan outlier ke dataset kita.

```

for i in range(len(winsorized_outlier)):
    outlier.iloc[i, 12] = winsorized_outlier[i]

# masukkan outlier ke dataset induk
for i in range(len(outlier)):
    id = outlier.iloc[i, 8] # ambil ID dari data outlier
    # cari index dari data outlier di dataset
    index = df[df['id'] == id].index[0]
    df.iloc[index, 12] = outlier.iloc[i, 12] # gantikan dataset dengan outlier

df = df.sort_values(by='loudness')
display(df)

```

11.3.3.4 Imputing

Imputing adalah proses penggantian data outlier dengan nilai-nilai yang diprediksi atau d'estimasi berdasarkan karakteristik data. Teknik imputasi sudah di bahas secara detail di bab sebelumnya

Kelebihan

- Mempertahankan informasi dan ukuran sampel
- Meningkatkan akurasi analisis

Kekurangan

- Pemilihan metode dan implementasi bisa cukup sulit
- Berpotensi merusak distribusi data

11.3.3.5 Normalization

Adalah metode untuk mengubah skala nilai dalam dataset sehingga nilainya berkisar antara 0 dan 1. Untuk melakukan normalisasi data, kita membagi data berdasarkan nilai minimum dan maksimum dari data. Proses normalisasi baik digunakan untuk dataset yang mempunyai distribusi data non-normal atau tidak beraturan.

Rumus dari normalisasi adalah:

$$normalized_x = \frac{x - min(x)}{max(x) - min(x)}$$

Kelebihan

- Mempertahankan informasi dan ukuran sampel
- Meningkatkan akurasi analisis
- Mudah diimplementasikan

Kekurangan

- Metode harus sesuai dengan karakteristik data
- Tidak menghilangkan outlier secara langsung, hanya mengurangi efek dari outlier

11.3.3.5.1 Normalization - Hands On Coding

Kita akan menggunakan dataset lagu spotify dengan fitur loudness, mirip seperti bab-bab sebelumnya.

Pertama kita cek batasan-batasan data dari fitur loudness

```
df = pd.read_csv('./data_sampled_100.csv')
df['loudness'].describe()
```

Diketahui bahwa batas minimum data adalah 31.808000, dan batas maksimum data adalah -2.478000

Batas-batas data ini akan kita ubah menjadi 0 - 1 menggunakan metode normalization. Sebelumnya, kita cek distribusi dari data menggunakan plot histogram.

```
fig = px.histogram(df, x='loudness')
fig.show()
```

Lalu kita akan gunakan function minmax scaler untuk melakukan proses normalisasi terhadap data loudness.

```
# buat objek scaler
scaler = MinMaxScaler()

# transformasi data tempo menggunakan objek scaler
df['loudness'] = scaler.fit_transform(df[['loudness']])

# grafik histogram untuk fitur tempo
fig = px.histogram(df, x='loudness')
```

```
fig.show()
```

Selanjutnya, kita cek batas-batas data dari data yang sudah di normalisasi.

```
df['loudness'].describe()
```

Data telah ter-normalisasi! Namun, apa efek dari normalisasi selain mengubah rentang data? Efek yang paling umum adalah jarak data dari kedua ujung menjadi lebih sama-rata. Hal ini dapat menambah akurasi model machine learning.

11.3.3.6 Standarization / Z-Score

Standarisasi data adalah suatu proses dalam analisis data yang mengubah variabel-variabel menjadi memiliki rata-rata nol dan standar deviasi satu. Dalam standarisasi data, setiap nilai data dikurangi dengan rata-rata dari seluruh data, kemudian hasilnya dibagi dengan standar deviasi data. Dengan melakukan hal ini, nilai-nilai data akan berada pada skala yang relatif terhadap variabilitas data. Proses standarisasi menggunakan rumus sebagai berikut:

$$\text{standardized}_x = \frac{x - \text{mean}(x)}{\text{standarddeviation}(x)}$$

Kelebihan

- Mean dan standar deviasi tidak berubah
- Tidak sensitif terhadap outlier

Kekurangan

- Tidak dapat menentukan batasan data
- Hasil dapat berupa angka negatif

11.3.3.6.1 Standardization / Z-score - Hands On Coding

Kita akan menggunakan dataset lagu spotify dengan fitur loudness, mirip seperti bab-bab sebelumnya.

Pertama kita cek batasan-batasan data dari fitur loudness

```
df = pd.read_csv('./data_sampled_100.csv')
df['loudness'].describe()
```

```
fig = px.histogram(df, x='loudness')
fig.show()
```

Lalu kita akan gunakan function z-score scaler untuk melakukan proses normalisasi terhadap data loudness.

```
mean = round(np.mean(df['loudness']), 2)
std = round(np.std(df['loudness']), 2)
normalized = round((df['loudness']-mean)/std, 2)

print(f'mean = ', mean)
print(f'std deviation = ', std)
normalized.describe()
```

Selanjutnya kita buat diagram histogramnya untuk mengecek perubahan distribusi data.

```
fig = px.histogram(normalized, x="loudness")
fig.show()
```

Jika kita bandingkan histogram data asli dengan data yang di standarisasi, distribusi data tidak berbeda jauh. Selain itu, batas data minimal mengecil dari -31 menjadi -3, dan batas data maksimal membesar dari -2 menjadi 1.

11.4 Dokumentasi Fitur

Dokumentasi data dapat menjembatani kesenjangan antara transaksi (pembuatan data) dan analisis (konsumsi data). Dokumentasi data yang baik memungkinkan pengguna, ataupun rekan tim untuk memahami siapa/apa/kapan/di mana/bagaimana/mengapa data tersebut dibentuk ataupun dikonsumsi.

Secara umum, dokumentasi fitur dari sebuah dataset mempunyai beberapa poin, yaitu :

- Nama fitur
- Definisi
- Tipe data
- Skala
- Sumber data
- Keterangan

Mari kita lakukan sebuah dokumentasi fitur dari dataset spotify yang kita gunakan di slide sebelumnya. Dengan menggunakan code data.info() kita bisa mengetahui banyak hal tentang dataset kita

```
df = pd.read_csv('./data_sampled_100.csv')
df.info()
```

11.4.1 Nama Fitur

Fitur atau kolom adalah satuan data yang mendeskripsikan aspek tertentu dalam data. Di dalam dataset ini, kita mempunyai beberapa fitur, antara lain :

- Valence
- Year
- Acousticness
- Artists
- Danceability
- Duration_ms
- Energy
- Explicit
- ID
- Instrumentalness
- Key
- Liveness
- Loudness
- Mode
- Name
- Popularity
- Release_date
- Speechiness
- Tempo

11.4.2 Definisi Fitur

Setiap fitur mempunyai definisi atau penjelasan makna tertentu. Definisi yang ada di bawah hanya sebagian dari dataset saja. Untuk definisi keseluruhan data bisa dilihat di <https://developer.spotify.com/documentation/web-api/reference/get-several-audio-features>. Berikut definisi fitur dari dataset :

- **name**, Nama lagu
- **artists**, Nama artis pembuat lagu
- **year**, Tahun rilis lagu
- **popularity**, Tingkat popularitas lagu. Nilai popularitas adalah antara 0 dan 100, dengan 100 menjadi nilai maksimal.
- **key**, adalah kunci lagu dari lagu tersebut. kunci lagu direpresentasikan menggunakan angka menurut Pitch Class Notation Standard. Misalnya, 0 = C, 1 = C /D , 2 = D, dan seterusnya. Jika tidak ada kunci, maka nilai -1.
- **mode**, Mode dari lagu tersebut. 1 = Major, 0 = Minor.

11.4.3 Tipe Data

Tipe data sangatlah penting karena tipe data menentukan operasi atau function apa yang bisa dilakukan terhadap data tersebut. Sebagai contoh, untuk menentukan nilai random, kita tidak bisa menggunakan angka float (pecahan), kita harus menggunakan angka integer (bulat). Kita dapat mengecek tipe data dari sebuah dataset menggunakan function `data.info()`

```
df = pd.read_csv('./data_sampled_100.csv')
df.info()
```

Definisi dari tipe data di dataset :

- **float**, angka pecahan
- **int**, angka bulat
- **object**, kombinasi antara teks dan angka

11.4.4 Skala Data

Skala atau rentang data adalah batasan-batasan data yang ada di sebuah dataset. Skala data bisa diperoleh menggunakan kode `data.describe()`

```
df = pd.read_csv('./data_sampled_100.csv')
df.describe()
```

11.4.5 Sumber Data

Data diperoleh dari kaggle dengan link <https://www.kaggle.com/datasets/vatsalmavani/spotify-dataset>

Kaggle adalah platform dataset open source. Namun dataset spotify ini dapat di scrape secara mandiri dengan mendaftar sebagai developer di spotify dan mengambil data menggunakan API spotify.

Setelah di download data diambil 100 baris secara random untuk meningkatkan performa dari coding.

11.4.6 Keterangan

Data mempunyai beberapa lagu yang tidak valid, jadi harus dilakukan proses data cleaning.

12 Pelabelan Data

Pelabelan data (data labeling) adalah proses menandai atau memberi label pada data dengan kelas atau kategori yang sesuai.

Kuantitas & kualitas data pelatihan yang secara langsung menentukan keberhasilan suatu algoritma AI sehingga tidak mengherankan jika rata-rata 80% waktu yang dihabiskan untuk proyek AI membahas data pelatihan yang mencakup proses pelabelan data.

Keakuratan model AI Anda berkorelasi langsung dengan kualitas data yang digunakan untuk melatihnya. Hal ini menjadi satu alasan mengapa proses pelabelan data merupakan bagian integral dari alur kerja persiapan data dalam membangun model AI yang handal.

12.1 Data Training

Machine learning dibagi menjadi dua, yaitu supervised dan unsupervised learning.

Unsupervised learning menggunakan dataset tanpa label untuk menemukan sebuah pola, struktur atau hubungan antar fitur untuk melatih algoritma machine learning.

Supervised learning menggunakan dataset yang mempunyai label untuk melatih algoritma machine learning memprediksi nilai atau mengklasifikasikan sesuatu.

Namun, tidak semua dataset mempunyai label. Dan tidak semua dataset yang tidak mempunyai label dapat di training menggunakan unsupervised learning. Jadi, kita harus melakukan proses labeling dataset.

Ketika sebuah dataset diberi sebuah label, maka label tersebut sebagai dasar kebenaran atau ground truth.

Pelabelan dataset sangatlah penting untuk proses machine learning. Proses labeling data dapat mempengaruhi kualitas dan akurasi dari sebuah model.

Selain itu, pelabelan dataset dapat meningkatkan kualitas dari dataset, sehingga dataset dapat diproses oleh lebih banyak algoritma machine learning.

Contoh lain dari pelabelan dataset deteksi email spam :

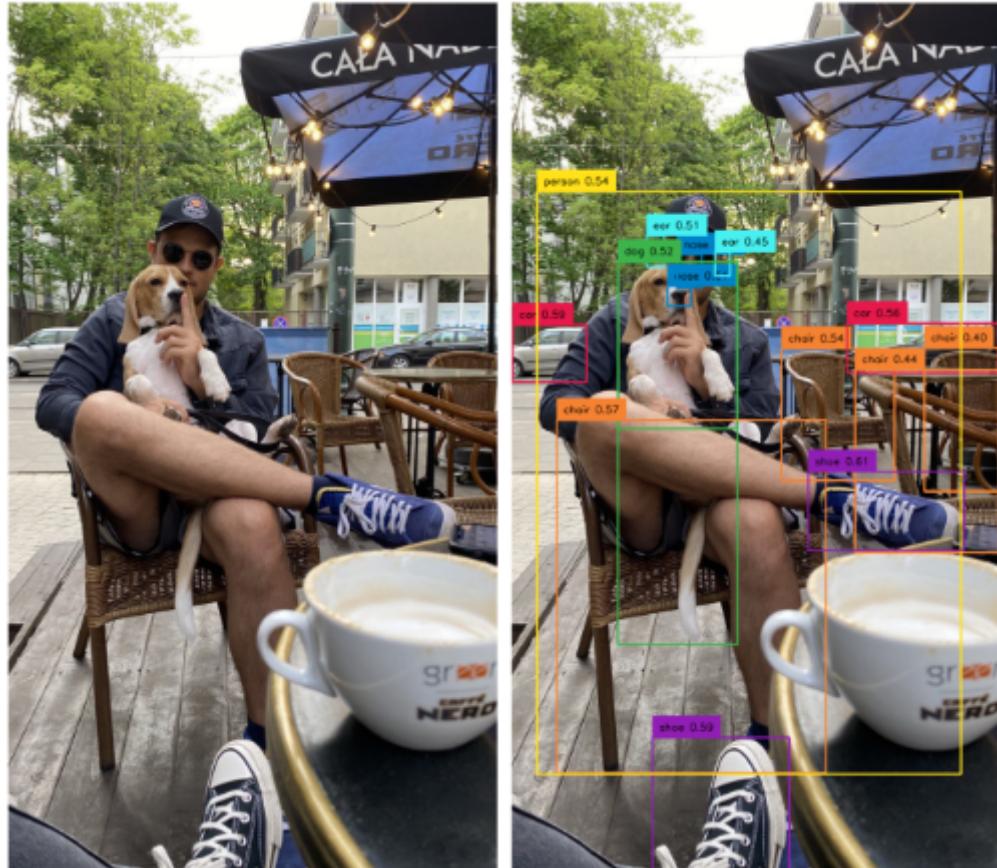


Figure 12.1: Gambar12. Pelabelan data gambar

BENAR			SALAH		
Email subject	Email content	label	Email subject	Email content	Label
Urgent: Claim your prize!	Congratulations! You have won a free vacation.	Spam	Urgent: Claim your prize!	Congratulations! You have won a free vacation.	Spam
Important Meeting Reminder	Hello, Just a friendly reminder about our meeting tomorrow.	Not spam	Important Meeting Reminder	Hello, Just a friendly reminder about our meeting tomorrow.	Spam
Your Order Confirmation	Thank you for your purchase. Order details enclosed.	Not spam	Your Order Confirmation	Thank you for your purchase. Order details enclosed.	Spam

Kesimpulannya, jika pelabelan dataset tidak akurat, maka hasil prediksi yang dibuat oleh algoritma machine learning tidak akan akurat

12.2 Metode Pelabelan Data

Data labeling adalah langkah yang sangat penting dalam proses pengembangan model machine learning. Proses ini terlihat simpel, namun sebenarnya tidak semudah itu untuk diimplementasikan. Sebagai data scientist/data engineer, kita harus mempertimbangkan semua faktor dan metode yang ada untuk menentukan implementasi yang terbaik. Seperti semua hal di dunia programming, setiap metode pelabelan data mempunyai kelebihan dan kekurangan. Berikut beberapa metode pelabelan data yang sering digunakan :

12.2.1 Internal Labeling

Proses labeling data oleh data scientist, data engineer, atau staf expert lain yang bekerja di perusahaan.

Kekuatan

- Kualitas labeling tinggi
- Akurasi labeling tinggi
- Kemanan data terkontrol

Kelemahan

- Membutuhkan waktu dan tenaga yang banyak
- Cukup mahal

12.2.2 Synthetic labeling

Proses labeling data yang dilakukan dengan men-generate data baru (dengan pattern yang mirip) dari data yang sudah ada. Contohnya adalah imagen (image generator) dimana sebuah

gambar diputar dan digeser sehingga menghasilkan data baru. Proses ini biasanya dilakukan untuk melatih sebuah model machine learning

Kekuatan

- Cepat
- Tidak membutuhkan tenaga yang banyak

Kelemahan

- Tidak semua algoritma bisa menerima semua jenis generated data

12.2.3 Programmatic labeling

Proses labeling data yang dilakukan oleh algoritma atau mesin, umumnya algoritma machine learning berbentuk klasifikasi, clustering, atau regresi.

Kekuatan

- Proses labeling cepat dan dapat diskalakan dengan mudah
- Tidak membutuhkan tenaga yang banyak

Kelemahan

- Kualitas labeling tergantung pada kualitas dataset dan penerapan algoritma machine learning
- Cukup mahal tergantung dengan skala dan penggunaan hardware
- Proses training model bisa lama

12.2.4 Outsourcing

Outsourcing adalah kegiatan merekrut perusahaan atau organisasi untuk melakukan proses data labeling. Salah satu pertimbangan utama adalah spesialisasi dan kompetensi dari sebuah perusahaan atau organisasi tersebut.

Kekuatan

- Simple
- Hasil labeling akurat

Kelemahan

- Mahal
- Memakan waktu lama

12.2.5 Crowdsourcing

Crowdsourcing adalah proses dimana pekerjaan data labeling didistribusikan ke khalayak publik, umumnya freelancer melalui platform website. Contohnya adalah project ReCaptcha di mana manusia diminta untuk mengisi captcha, lalu hasilnya dimasukkan ke algoritma machine learning untuk memprediksi captcha.

Kekuatan

- Paling efisien, dapat mengolah banyak data dalam waktu singkat

Kelemahan

- Perlu mengembangkan workflow yang cocok untuk freelancer
- Akurasi tergantung dengan QA dan workflow
- Potensi kebocoran data tinggi

12.3 Penggunaan Data Labeling dalam Pengolahan Citra

Pengolahan citra adalah kegiatan memanipulasi data yang berbentuk gambar atau video. Di bidang machine learning, pengolahan citra adalah salah satu bidang yang paling penting. Model machine learning pengolah citra yang sering dibuat meliputi: klasifikasi, segmentasi, deteksi objek, dan estimasi pose. Semua proses pembuatan model machine learning sangat erat kaitannya dengan pelabelan data yang digunakan di data training.

Tujuan akhir dari semua model yang telah disebutkan adalah melabeli sebuah gambar atau video dengan sesuatu.

12.3.1 Image Classification



CAT

Image classification model adalah jenis model dalam bidang pengolahan citra yang digunakan untuk mengklasifikasikan gambar ke dalam berbagai kategori atau kelas yang telah ditentukan sebelumnya. Tujuan dari model ini adalah untuk mengidentifikasi dan memprediksi kelas atau label yang sesuai dengan gambar yang diberikan.

12.3.2 Image Segmentation

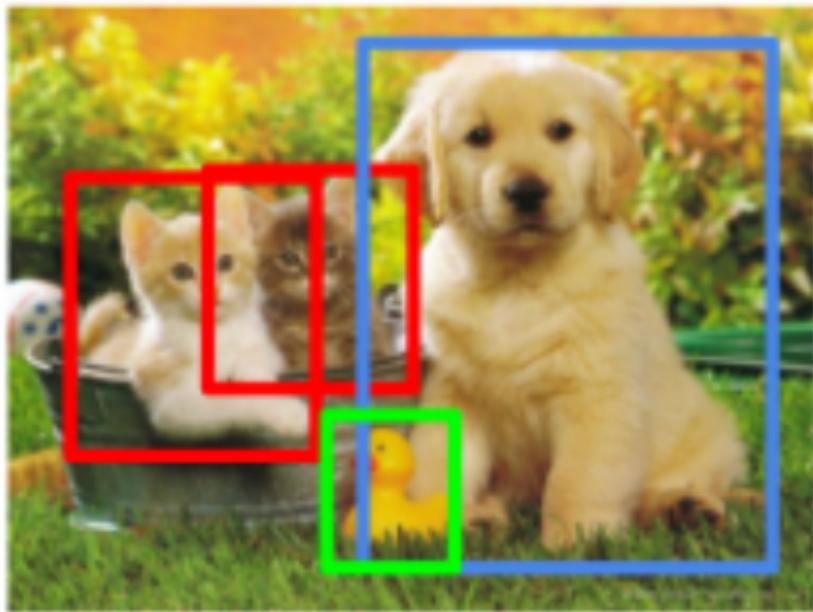


CAT

Image segmentation adalah proses dalam pengolahan citra yang membagi atau memisahkan gambar menjadi beberapa bagian/segmen yang lebih kecil. Setiap segmen mewakili area yang memiliki karakteristik visual yang serupa, seperti warna, tekstur, atau bentuk. Tujuan dari image segmentation adalah untuk memahami struktur internal gambar dan mengidentifikasi objek yang ada di dalamnya.

Kemiripan antara segmentasi dan klasifikasi adalah objek atau hasil prediksi hanya mempunyai satu nilai

12.3.3 Object Detection



CAT, DOG, DUCK

Deteksi objek adalah proses dalam pengolahan citra yang bertujuan untuk mengidentifikasi dan memetakan objek-objek tertentu dalam sebuah gambar atau video. Tujuan utama dari deteksi objek adalah untuk menemukan dan menandai lokasi serta batas-batas objek yang ada dalam sebuah gambar.

Perbedaan object detection dengan metode-metode sebelumnya yaitu objek atau hasil prediksi mempunyai lebih dari satu hasil

12.3.4 Pengolahan Citra - Hands On Coding

Kita akan membuat model object detection menggunakan library OpenCV. Kita juga akan menggunakan algoritma classifier bernama Haar Cascade. Untuk file lengkapnya bisa dilihat di sini [link paper](#).

Algoritma ini menggunakan sebuah sistem machine learning dimana kita memberikan gambar positif dan negatif. Gambar positif adalah gambar objek yang kita ingin klasifikasikan, sedangkan gambar negatif adalah gambar objek yang tidak ingin kita klasifikasikan.

Karena keterbatasan waktu, kita akan menggunakan model yang sudah jadi untuk melakukan deteksi objek pada gambar yang kita inginkan. Pada kasus ini, kita akan mendekripsi rambu lalu lintas stop.

12.3.4.1 Download Library

Library bisa di download menggunakan pip install opencv-python dan pip install matplotlib

12.3.4.2 Import Library

buat file python dan import package-package sebagai berikut:

- import cv2
- import matplotlib.pyplot as plt

12.3.4.3 Download Model dan Gambar Contoh

Download pre-trained model yang mempunyai format .xml di [link ini](#). Jangan lupa download gambar stop sign.



Figure 12.2: Gambar17. Contoh gambar stop sign

12.3.4.4 Ekstrak Model dan Gambar ke Directory

12.3.4.5 Tampilkan Gambar Menggunakan OpenCV dan Matplotlib

```
# buka gambar menggunakan opencv
img = cv2.imread("image.jpg")

# OpenCV membuka gambar menggunakan metode BRG (blue red green)
# namun kita ingin membuka dengan metode RGB (red green blue)
# kita harus konversi dari BRG ke RGB
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# kita juga membutuhkan gambar versi grayscale (hitam putih)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# menampilkan gambar menggunakan matplotlib
plt.subplot(1, 1, 1)
plt.imshow(img_rgb)
plt.show()
```

12.3.4.6 Buat Algoritma Deteksi Objek

```
# load model yang sudah dilatih untuk mendeteksi stop sign
# model berbentuk file xml
stop_data = cv2.CascadeClassifier('stop_data.xml')

# buat ukuran kotak minimum agar ukuran kotak yang terdeteksi tidak terlalu kecil
found = stop_data.detectMultiScale(img_gray,
                                    minSize=(5, 5))

# hitung jumlah objek yang ditemukan.
amount_found = len(found)

# jika objek tidak ditemukan maka tidak dilakukan apa-apa
if amount_found != 0:

    # jika objek yang ditemukan lebih dari satu, maka :
    for (x, y, width, height) in found:

        # kita gambar sebuah kotak hijau di objek yang ditemukan
```

```
cv2.rectangle(img_rgb, (x, y),  
             (x + height, y + width),  
             (0, 255, 0), 5)  
  
# tampilkan hasil citra menggunakan plt  
plt.subplot(1, 1, 1)  
plt.imshow(img_rgb)  
plt.show()
```

12.3.4.7 Eksperimen!

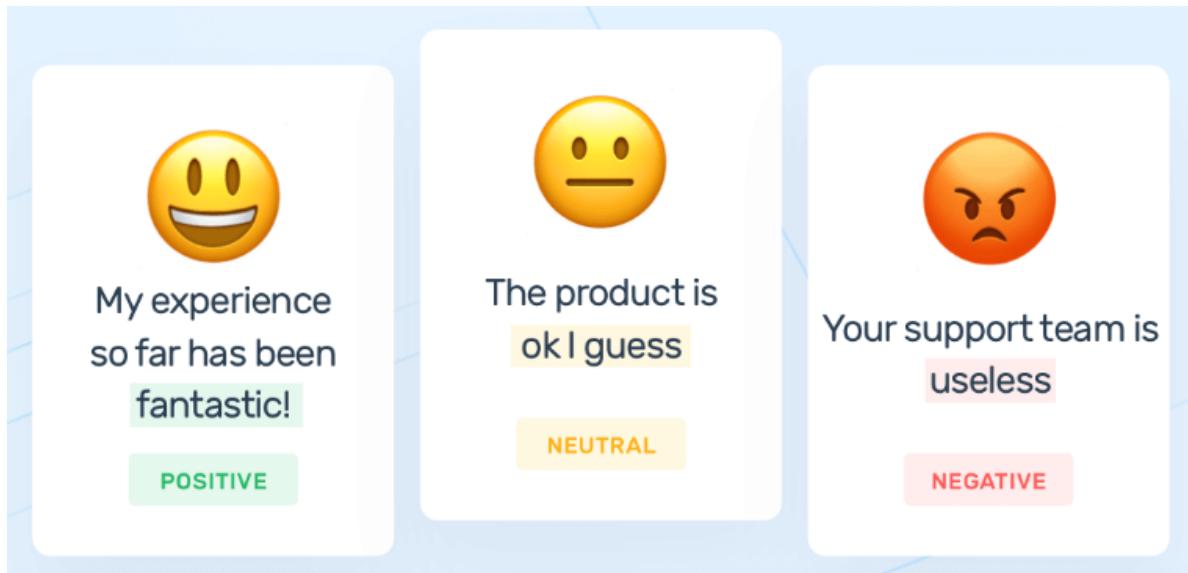
Cari gambar rambu lalu lintas stop di internet atau ambil foto secara langsung, lalu masukkan gambar tersebut ke coding. Terapkan algoritma dan lihat apakah algoritma deteksi objek berjalan dengan sempurna.

Petunjuk : edit line `cv2.imread("[nama file foto]")` untuk menggunakan gambar atau foto anda sendiri

12.4 Penggunaan Data Labeling dalam Natural Language Processing

Natural Language Processing atau NLP mengacu pada analisis bahasa manusia dan bentuknya selama interaksi baik dengan manusia lain maupun dengan mesin. Menjadi bagian dari linguistik komputasi awalnya, NLP telah berkembang lebih lanjut dengan bantuan Artificial Intelligence dan Deep Learning.

12.4.1 Sentiment Analysis



Adalah sebuah proses pemberian label sentimen terhadap sebuah teks (kata, kalimat, atau paragraf) berdasarkan ekspresi perasaan manusia. Tujuan utamanya adalah untuk mengkategorikan teks berdasarkan sentimen yang diekspresikan oleh teks.

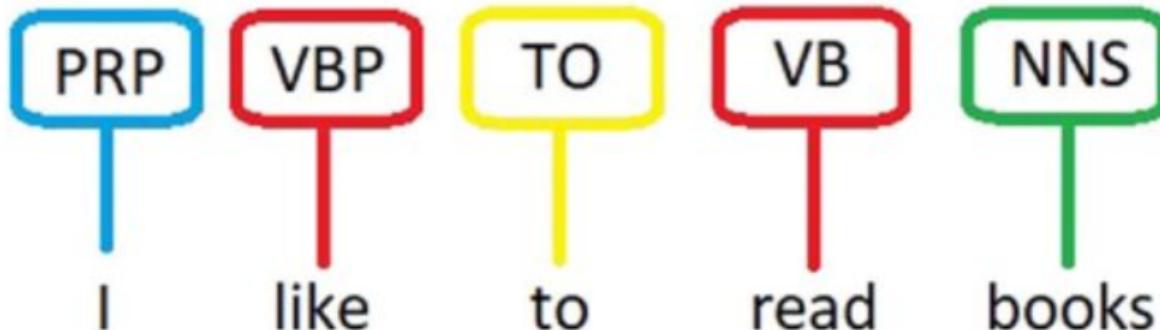
Teks dianalisa dengan mempertimbangkan beberapa faktor, antara lain : pemilihan kata, konteks, subjektivitas, dan beberapa faktor lain. Lalu sebuah sentimen berupa perasaan (marah, sedih, bahagia, bingung) dilabelkan ke teks tersebut.

12.4.2 Named Entity Recognition (NER)

Barack Hussein Obama (born August 4, 1961) is an American attorney and politician who served as the 44th President of the United States from January 20, 2009, to January 20, 2017. A member of the Democratic Party, he was the first African American to serve as president. He was previously a United States Senator from Illinois and a member of the Illinois State Senate.

Adalah sebuah proses pemberian label entitas dalam sebuah teks. Entitas adalah sebuah kategori luas yang dapat diatur oleh pembuat algoritma. Contoh dari entitas antara lain, nama seseorang, lokasi, negara, organisasi, besaran, jumlah uang, dan lain lain. Tujuan utama dari NER adalah mengekstrak informasi entitas dari sebuah teks. Dalam proses labeling, algoritma harus memperhatikan konteks, struktur kata, dan arti tersirat dalam sebuah teks untuk mengidentifikasi sebuah entitas secara akurat.

12.4.3 Part of Speech Tagging (POS-tagging)



Adalah sebuah proses pemberian label kategori tata-bahasa (grammar) dalam sebuah teks. Pada umumnya sebuah kalimat terdiri dari beberapa kata yang termasuk dalam kategori grammar kata benda (noun), kata sifat (adjective), keterangan (adverb), kata ganti (pronoun), dan lain-lain.

Tujuan utama POS-tagging adalah untuk menentukan kategori grammar dari sebuah kata

di dalam sebuah konteks yang ada di teks. POS-tagging sangatlah penting karena grammar atau tata-bahasa dapat mengubah makna dari sebuah kalimat secara keseluruhan.

12.4.4 Hands On Coding

Dalam praktikum ini kita akan membuat algoritma Named Entity Recognition menggunakan library spaCy dan sebuah artikel.

Download Library dan Dataset yang Dibutuhkan

- python -m pip install -U pip setuptools wheel
- pip install -U spacy
- python -m spacy download en_core_web_sm

12.4.4.1 Import library

- import spacy
- from spacy import displacy

12.4.4.2 Masukkan Model NER SpaCy

```
NER = spacy.load("en_core_web_sm")
```

12.4.4.3 Tentukan Artikel yang Akan di Proses. Artikel Harus Berbahasa Inggris.

```
rawtext = "Asus Zenfone 10 will please everyone who finds the current flagship cellphones
```

12.4.4.4 Masukkan Artikel ke Fungsi NER, Lalu tampilkan hasilnya

```
text1 = NER(rawtext)

for word in text1.ents:
    print(word.text, '-', word.label_)
```

12.4.4.5 Kesimpulan

Hasil dari NER adalah sebuah kata yang dideteksi dan kategori dari kata tersebut. Namun bisa dilihat bahwa hasil dari algoritma tidak 100% akurat. Ada beberapa hasil yang labelnya tidak sesuai, contohnya “Asus - PERSON”. Algoritma melabeli kata Asus sebagai sebuah nama orang, padahal Asus seharusnya adalah sebuah organisasi (ORG).

12.4.4.6 Eksperimen!

Cari teks dari artikel, wikipedia, berita, lalu masukkan teks tersebut ke algoritma NER dan analisa hasil dan akurasi dari algoritma.

Petunjuk : masukkan teks ke dalam variable rawtext sebagai string (“ ”)

12.5 Analisa Kualitas dan Akurasi Data untuk Pelabelan

Akurasi dalam pelabelan data mengukur seberapa dekat pelabelan dengan ground truth, atau seberapa baik fitur berlabel dalam data set konsisten dengan kondisi dunia nyata. Dalam model pemrosesan bahasa alami (NLP) contohnya adalah seberapa akurat model memberikan label sentimen terhadap sebuah teks.

Kualitas dalam pelabelan data adalah tentang akurasi dataset secara keseluruhan. Apakah pekerjaan semua pemberi label terlihat sama? Apakah pelabelan secara konsisten akurat di seluruh data set?

12.5.1 Definisi Data yang Berkualitas

Proses pelabelan data selalu bertumpu pada kualitas data. Trash in, trash out adalah sebuah mantra yang populer di dunia pengolahan data. Artinya, data yang tidak berkualitas tidak dapat menghasilkan produk yang berkualitas.

Maka dari itu, data harus kita tentukan kualitas dan akurasinya agar kita dapat menentukan ekspektasi terhadap hasil dari pelabelan data yang akan kita lakukan. Pada umumnya data yang berkualitas mempunyai :

- Tidak ada nilai kosong/korup
- Nilai unik tiap entry
- Variasi seimbang
- Metode pengambilan data yang tangguh
- Dokumentasi lengkap
- Format yang rapi

12.5.2 Faktor-faktor yang mempengaruhi kualitas proses pelabelan data

12.5.2.1 Pemahaman Data

Data mempunyai banyak fitur dan value. Pengetahuan tentang fitur data dan nilai-nilai yang ada didalamnya akan sangat membantu untuk menentukan proses pelabelan data.

12.5.2.2 Kompetensi Developer/Trainer

Developer harus mempunyai pengetahuan yang luas di bidang machine learning atau bidang yang relevan. Developer harus bisa menentukan kualitas dataset, langkah-langkah data pre-processing, algoritma yang sesuai, dan cara untuk mengukur akurasi pelabelan.

12.5.2.3 Ketangguhan Workflow

Proses pelabelan data pasti mempunyai tahap-tahap yang ditetapkan oleh kepemimpinan. Proses atau workflow tersebut harus tahan terhadap gangguan-gangguan seperti human error, machine error, perubahan requirements, ambiguitas, sehingga proses dapat berjalan lancar selamanya. Salah satu proses terpenting yaitu QA, yang akan kita bahas di slide selanjutnya

12.5.3 Metode QA untuk mengukur kualitas data

12.5.3.1 Consensus Algorithm

Adalah sebuah metodologi untuk mencapai sebuah keputusan mengenai kualitas data dengan cara mengumpulkan persetujuan (consensus) dari semua atau sebagian orang yang melakukan proses data labeling.

12.5.3.2 Benchmarking dan Gold Standard

Adalah proses membandingkan hasil data labeling dari beberapa model dengan mengaplikasikan model tersebut ke dataset yang sering digunakan. Tujuan utamanya adalah untuk memperoleh batas bawah (baseline) atau reference point untuk mengevaluasi kualitas dari model. Gold standard adalah sebuah hasil pelabelan dari sebuah model berkualitas tinggi terhadap dataset berkualitas tinggi. Gold standard merepresentasikan teknologi terbaik, aplikasi terakurat, dan dataset berkualitas paling tinggi, hasilnya adalah nilai akurasi yang paling tinggi diantara riset atau percobaan lain.

12.5.3.3 Cronbach Alpha Test

Adalah sebuah metode untuk mengukur tingkat konsistensi dari beberapa variabel yang mempunyai variabel laten yang sama. Variabel laten adalah variabel yang tidak mempunyai metrik atau ukuran secara tersendiri. Untuk mengukur variabel laten, diperlukan beberapa variabel lain yang saling berhubungan dan mempunyai konsistensi tinggi. Cronbach Alpha berfungsi untuk mengukur variabel lain ini.

Contohnya adalah kita kana mengukur tingkat ekstroversi seseorang. Tingkat ekstroversi ini adalah variabel laten karena kita tidak bisa mengukur tingkat ekstroversi dengan sendirinya. Maka diperlukan kuesioner dengan 5 pertanyaan. Hasil dari 5 pertanyaan inilah yang akan kita tes dengan Cronbach Alpha Test untuk menentukan apakah mereka merepresentasikan tingkat ekstroversi seseorang.

12.5.4 Hands On Coding - Cronbach Alpha Test

Rumus dari Cronbach Alpha test adalah:

$$\alpha = \frac{k}{k-1} \left(1 - \frac{\sum s_i^2}{s_X^2}\right)$$

α = koefisien reliabilitas k = jumlah item set s_i^2 = nilai variance setiap item i dimana $i = 1, 2, \dots, k$, s_X^2 = nilai variance dari semua item

Nilai diatas 0.7 termasuk cukup bagus untuk sebuah paper.

Sebagai contoh, kita akan membuat sebuah sistem untuk mengecek kepuasan pelanggan terhadap 5 produk baru kita. Pelanggan diberikan sebuah kuesioner dan pelanggan akan memberi rating dari 1-5 terhadap 5 produk baru kita.

12.5.4.1 Install dan Import Library

- pip install numpy
- pip install scipy

12.5.4.2 Buat Data Menggunakan NumPy

Anda dapat menambahkan data baru

```
data = np.array([[4,3,5,2,4], [5,4,4,3,5], [3,2,3,4,3], [4,4,5,5,4]])
print(data)
```

12.5.4.3 Hitung Rata-Rata dari Setiap Item/Fitur

```
item_means = np.mean(data, axis=0)
print(item_means)
```

12.5.4.4 Hitung Variansi dari Total Skor dan Skor Setiap Item

```
total_var = np.var(np.sum(data, axis=1))
item_var = np.var(data, axis=0, ddof=1)
print(f'total variance : ', total_var)
print(f'item variance : ', item_var)
```

12.5.4.5 Hitung Rata-Rata dari Setiap Item/Fitur

```
num_items = len(item_means)
cronbach_alpha = (num_items / (num_items - 1)) * (1 - (np.sum(item_var) / total_var))
print(f'Nilai Cronbach Alpha adalah : ', cronbach_alpha)
```

Hasil nilai Cronbach Alpha adalah 0.4. Nilai ini menunjukkan bahwa konsistensi dari kuesioner yang terdiri dari 5 fitur (item 1 - 5) tergolong rendah dan tidak mencerminkan kepuasan pelanggan secara akurat.

Pada umumnya, nilai Cronbach Alpha yang tergolong bagus adalah diatas 0.7

12.6 Keamanan Pelabelan Data

Pelabelan data adalah sebuah pekerjaan yang memakan banyak waktu dan tenaga. Proses pelabelan data mempunyai banyak cabang dan workflow, dan di setiap langkah workflow selalu ada resiko kebocoran data.

Terlebih jika pekerjaan pelabelan data dikerjakan oleh organisasi lain yang berada di luar kendali kita. Oleh karena itu, kita harus membuat sebuah workflow yang dapat mencegah kebocoran data, serta paham tentang prinsip-prinsip dasar tentang keamanan data.

12.6.1 Resiko Keamanan Outsourcing Data Labeling

- Mengakses data dari jaringan yang tidak aman atau menggunakan perangkat tanpa perlindungan malware

- Mengunduh atau simpan sebagian data (mis., screen capture, flash drive)
- Memberi label data saat berada di tempat umum
- Tidak memiliki pelatihan, konteks, atau akuntabilitas terkait dengan aturan keamanan untuk pekerjaan labeling
- Bekerja di lingkungan fisik atau digital yang tidak disertifikasi untuk mematuhi peraturan data (mis., HIPAA, SOC 2).

12.6.2 Tiga area yang perlu menjadi perhatian untuk menjaga keamanan dokumen

- Orang dan Tenaga Kerja: Ini dapat mencakup pemeriksaan latar belakang untuk pekerja dan mungkin mengharuskan pemberi label untuk menandatangani perjanjian kerahasiaan (NDA) atau dokumen serupa yang menguraikan persyaratan keamanan data.
- Teknologi dan Jaringan: Pekerja mungkin diminta untuk menyerahkan perangkat yang mereka bawa ke tempat kerja, seperti ponsel atau tablet.
- Fasilitas dan Ruang Kerja: Pekerja dapat duduk di tempat yang menghalangi orang lain untuk melihat pekerjaan mereka.

13 Pemodelan Data Science

13.1 Pendahuluan

Pemodelan data science adalah proses membangun model yang dapat digunakan untuk memprediksi nilai dari suatu variabel berdasarkan nilai variabel lainnya. Model yang dibangun dapat berupa model statistika, model machine learning, atau model deep learning. Pemodelan data science merupakan salah satu tahapan penting dalam proses data science.

Pada modul ini, data yang akan dijadikan contoh adalah data kamar hotel di Yogyakarta. Data ini berisi informasi mengenai kamar hotel di Yogyakarta yang terdapat pada website [Traveloka](#).

13.2 Tahap Pemodelan Machine Learning

13.2.1 Data Preparation

Umumnya data terlebih dahulu dibagi menjadi dua bagian, yaitu data training dan data testing. **Data training** digunakan untuk membangun model, sedangkan **data testing** digunakan untuk menguji performa model. Data training dan data testing harus memiliki karakteristik yang sama. Data training dan data testing dapat dibagi secara acak. Data training dan data testing dapat dibagi dengan perbandingan **80:20** atau **70:30** dimana data testing seharusnya mendapatkan data yang lebih banyak dibandingkan testing agar model dapat dilatih dengan lebih baik.

```
import pandas as pd
from sklearn.model_selection import train_test_split

# membaca dataset
df = pd.read_csv('./dataset/kamar-hotel-yogyakarta.csv')

# Memisahkan fitur dan label
x = df.iloc[:, 1:]
y = df.iloc[:, 0]
# Alternatif
```

```

# x = df.drop('harga', axis=1)
# y = df['harga']

# membagi data menjadi data training dan data testing dengan perbandingan 80:20
# Random state digunakan untuk mengatur agar pembagian data menjadi sama setiap kali dijalankan
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=123)

# menampilkan ukuran data training dan data testing
trainRatio = round(x_train.shape[0]/len(df), 2)*100
testRatio = round(x_test.shape[0]/len(df), 2)*100

print(f'Train set: {x_train.shape[0]} ({trainRatio}%)')
print(f'Test set: {x_test.shape[0]} ({testRatio}%)')

```

13.2.2 Model Training

Algoritma yang digunakan dalam modul ini adalah Algoritma Random Forest. Algoritma Random Forest merupakan algoritma yang digunakan untuk melakukan klasifikasi dan regresi. Algoritma Random Forest merupakan pengembangan dari algoritma Decision Tree. Algoritma Random Forest mengambil keputusan berdasarkan hasil voting dari beberapa pohon keputusan.

Library yang digunakan untuk membangun model Random Forest adalah **sklearn**. Library ini berisi algoritma untuk membangun model machine learning seperti Random Forest, Gradient Boosting, dan lainnya.

```

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor()

rf.fit(x_train, y_train)

# menampilkan skor akurasi dari model
print(f'Skor R2: {rf.score(x_test, y_test)}')

```

13.2.3 Parameter Tuning

Parameter tuning dilakukan untuk menemukan parameter terbaik yang dapat digunakan untuk membangun model. Parameter tuning dapat dilakukan dengan menggunakan GridSearchCV atau RandomizedSearchCV yang terdapat pada library **sklearn**. Grid Search meru-

pakan teknik untuk mencari parameter terbaik dengan cara mencoba semua kombinasi dari parameter yang diberikan, maka dari itu waktu komputasi akan lebih panjang. Untuk mempercepat waktu komputasi, dapat digunakan Random Search yang akan mencari parameter terbaik secara acak. Untuk detail mengenai parameter yang digunakan pada algoritma Random Forest dapat dilihat pada [dokumentasi sklearn](#).

```
from sklearn.model_selection import RandomizedSearchCV

# menentukan parameter yang akan dicoba
rfParams = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_depth': [None, 5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None],
}

# mencari parameter terbaik
rfRandom = RandomizedSearchCV(rf, rfParams, random_state=123, n_jobs=-1)

rfRandom.fit(x_train, y_train)

# menampilkan parameter terbaik
print(f'Parameter terbaik: {rfRandom.best_params_}')

# menampilkan skor akurasi dari model
print(f'Skor R2: {rfRandom.score(x_test, y_test)})
```

13.2.4 Saving Model

Model yang telah dibangun dapat disimpan dengan menggunakan library **pickle**. Library ini digunakan untuk menyimpan objek Python ke dalam file. Model yang telah disimpan dapat digunakan kembali tanpa harus membangun model dari awal.

```
import pickle

# refit model dengan parameter terbaik
rfModel = rfRandom.best_estimator_.fit(x_train, y_train)

# menyimpan model
pickle.dump(rfModel, open('rfModel.pkl', 'wb'))
```

14 Evaluasi Data

14.1 Apa itu Evaluasi Model?

Merupakan salah satu tahap penting dalam proses machine learning yang memiliki tujuan untuk **memastikan model dapat menghasilkan prediksi yang akurat**.

14.2 Metrik Evaluasi

- Akurasi
- Presisi
- Recall
- F1-Score

14.2.1 Akurasi

- Merupakan metrik yang mengukur sejauh mana model dapat melakukan prediksi dengan benar.
- Formula atau rumus akurasi yaitu
- Semakin tinggi nilai akurasi, maka semakin baik model tersebut.

Seberapa Penting Nilai Akurasi

- Akurasi dapat digunakan untuk mengevaluasi perfoma sebuah model
- Akurasi memberikan gambaran seberapa baik model yang digunakan secara keseluruhan
- Dengan nilai akurasi, dapat mengetahui seberapa akurat dalam memprediksi kelas data.

Kelebihan dan Keterbatasan Akurasi

- Akurasi memiliki kelebihan sebagai matrik evaluasi yang sederhana dan sangat mudah dimengerti.
- Keterbatasan akurasi dalam mengatasi ketidakseimbangan kelas pada sebuah dataset.

Pahami Code berikut

- Perhatikan contoh code di bawah, kira-kira menghasilkan nilai Akurasi berapa persen?
- Anda dapat mencoba code di samping menggunakan dataset yang berbeda, lalu fahami dan lihat nilai Akurasinya.
- Pada dataset yang digunakan, dibagi menjadi 80:20 pada tahap split data. Selanjutnya menggunakan algoritma KNN dengan nilai K=3. selanjutnya akan dilakukan prediksi pada data uji dan akurasi model yang dihitung dengan membandingkan hasil prediksi dengan label sebenarnya pada data uji menggunakan fungsi “accuracy_score”

```
# Impor library yang diperlukan
from sklearn.datasets import load_iris #dataset dari sklearn
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Muat dataset Iris
#dapat dirubah menggunakan dataset yang lain
iris = load_iris()
X = iris.data
y = iris.target

# Bagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Inisialisasi model K-Nearest Neighbors (KNN) dengan k=3
model = KNeighborsClassifier(n_neighbors=3)

# Latih model menggunakan data latih
model.fit(X_train, y_train)

# Lakukan prediksi pada data uji
y_pred = model.predict(X_test)

# Hitung akurasi model dengan membandingkan prediksi dengan label sebenarnya pada data uji
accuracy = accuracy_score(y_test, y_pred)

# Tampilkan hasil akurasi
print("Akurasi model: {:.2f}%".format(accuracy * 100))
```

14.2.2 Presisi dan Recall

Presisi dan Recall merupakan metrik untuk mengukur performa pada model tertentu. Semakin tinggi nilai presisi dan recall, maka semakin baik model pada kelas tertentu.

Presisi

- Presisi mengukur sejauh mana, model **dapat melakukan identifikasi dengan benar** pada kelas tertentu.
- Formula atau rumus presisi sebagai berikut

Recall

- Recall mengukur sejauh mana model **dapat menemukan kembali** kelas tertentu.
- Formula atau rumus dari recall

Pahami Code berikut

Pada dataset yang digunakan, dibagi menjadi 80:20 pada tahap split data. Selanjutnya menggunakan algoritma KNN dengan nilai K=3. selanjutnya model akan melakukan prediksi pada data uji dan presisi serta recall dari model menggunakan fungsi **precision_score** dan **recall_score** dengan parameter **average='macro'**

```
# Impor library yang diperlukan
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import precision_score, recall_score

# Muat dataset Iris
iris = load_iris()
X = iris.data
y = iris.target

# Bagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Inisialisasi model K-Nearest Neighbors (KNN) dengan k=3
model = KNeighborsClassifier(n_neighbors=3)
```

```

# Latih model menggunakan data latih
model.fit(X_train, y_train)

# Lakukan prediksi pada data uji
y_pred = model.predict(X_test)

# Hitung presisi model
precision = precision_score(y_test, y_pred, average='macro')

# Hitung recall model
recall = recall_score(y_test, y_pred, average='macro')

# Tampilkan hasil presisi dan recall
print("Presisi model: {:.2f}".format(precision))
print("Recall model: {:.2f}".format(recall))

```

14.2.3 F1-Score

- Merupakan metrik untuk menggabungkan presisi dan recall
 - Formula F1-Score
 - Semakin tinggi dari nilai F1-Score, maka semakin baik model pada kelas tersebut.
- #### Mengapa F1-Score penting?** {unnumbered}
- F1-Score cocok digunakan saat terdapat ketidakseimbangan kelas pada sebuah dataset.
 - F1-Score memberikan bobot yang seimbang antara presisi dan recall.

Pahami Code berikut

Pada dataset yang digunakan, dibagi menjadi 80:20 pada tahap **split data**. Selanjutnya menggunakan algoritma KNN dengan nilai K=3. selanjutnya model tersebut melakukan prediksi pada data uji dan F1-score dari model dihitung menggunakan fungsi **f1_score** dengan parameter **average='macro'**

```

# Impor library yang diperlukan
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score

```

```

# Muat dataset Iris
iris = load_iris()
X = iris.data
y = iris.target

# Bagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Inisialisasi model K-Nearest Neighbors (KNN) dengan k=3
model = KNeighborsClassifier(n_neighbors=3)

# Latih model menggunakan data latih
model.fit(X_train, y_train)

# Lakukan prediksi pada data uji
y_pred = model.predict(X_test)

# Hitung F1-score model
f1score = f1_score(y_test, y_pred, average='macro')

# Tampilkan hasil F1-score
print("F1-score model: {:.2f}".format(f1score))

```

14.2.4 Pahami Code berikut

- Apa yang berbeda dari ketiga source code sebelumnya?
- Bagaimana hasil nilai akurasi, presisi, recall, dan F1-Score?

```

# Impor library yang diperlukan
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Muat dataset Iris
iris = load_iris()
X = iris.data
y = iris.target

# Bagi data menjadi data latih dan data uji

```

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Inisialisasi model K-Nearest Neighbors (KNN) dengan k=3
model = KNeighborsClassifier(n_neighbors=3)

# Latih model menggunakan data latih
model.fit(X_train, y_train)

# Lakukan prediksi pada data uji
y_pred = model.predict(X_test)

# Hitung akurasi model
accuracy = accuracy_score(y_test, y_pred)

# Hitung presisi model
precision = precision_score(y_test, y_pred, average='macro')

# Hitung recall model
recall = recall_score(y_test, y_pred, average='macro')

# Hitung F1-score model
f1score = f1_score(y_test, y_pred, average='macro')

# Tampilkan hasil akurasi, presisi, recall, dan F1-score
print("Akurasi model: {:.2f}%".format(accuracy * 100))
print("Presisi model: {:.2f}".format(precision))
print("Recall model: {:.2f}".format(recall))
print("F1-score model: {:.2f}".format(f1score))

```

14.2.5 Kurva ROC dan AUC

- ROC dan AUC merupakan metrik evaluasi model pada sebuah machine learning.
- ROC atau Receiver Operating Characteristic merupakan grafik yang menggambarkan sebuah performa model pada berbagai threshold.
- AUC atau Area Under the Curve merupakan luas daerah di bawah kurva ROC yang menggambarkan performa dari keseluruhan model.

Kurva ROC

- ROC Curve adalah grafik yang menggambarkan trade-off antara True Positive Rate (TPR) dan False Positive Rate (FPR)
- TPR adalah rasio data positif yang benar diprediksi oleh model, dibandingkan dengan total data positif
- FPR adalah rasio data negatif yang salah diprediksi sebagai positif oleh model, dibandingkan dengan total data negatif
- Semakin dekat kurva ROC ke sudut kiri atas, semakin baik performa model

Kurva AUC

- AUC adalah metrik evaluasi yang mengukur performa keseluruhan model
- AUC menghitung luas daerah di bawah kurva ROC
- Nilai AUC berada dalam rentang 0 hingga 1, dengan nilai terbaik adalah

Pahami Code berikut

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score

# Muat dataset Iris
iris = load_iris()
X = iris.data
y = iris.target

# Bagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Inisialisasi model Logistic Regression
model = LogisticRegression()

# Latih model menggunakan data latih
model.fit(X_train, y_train)

# Dapatkan probabilitas prediksi untuk kelas positif
y_prob = model.predict_proba(X_test)[:, 1]
```

```

# Hitung nilai AUC (Area Under the Curve)
auc_score = roc_auc_score(y_test, y_prob)

# Hitung nilai FPR (False Positive Rate) dan TPR (True Positive Rate) untuk kurva ROC
fpr, tpr, _ = roc_curve(y_test, y_prob)

# Plot kurva ROC
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = {:.2f})'.format(auc_
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()

```

14.2.6 Amati dan pahami code berikut

- Apa yang berbeda dari ketiga source code sebelumnya?
- Bagaimana hasil dari code tersebut?

```

import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_c

# Muat dataset Iris
iris = load_iris()
X = iris.data
y = iris.target

# Bagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Inisialisasi model Logistic Regression
model = LogisticRegression()

```

```

# Latih model menggunakan data latih
model.fit(X_train, y_train)

# Lakukan prediksi pada data uji
y_pred = model.predict(X_test)

# Menghitung akurasi model
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi Model: {:.2f}".format(accuracy))

# Menghitung presisi model
precision = precision_score(y_test, y_pred, average='macro')
print("Presisi Model: {:.2f}".format(precision))

# Menghitung recall model
recall = recall_score(y_test, y_pred, average='macro')
print("Recall Model: {:.2f}".format(recall))

# Menghitung F1-score model
f1score = f1_score(y_test, y_pred, average='macro')
print("F1-Score Model: {:.2f}".format(f1score))

# Dapatkan probabilitas prediksi untuk kelas positif
y_prob = model.predict_proba(X_test)

# Hitung nilai AUC (Area Under the Curve) untuk setiap kelas
auc_scores = []
for i in range(len(iris.target_names)):
    auc_score = roc_auc_score(y_test == i, y_prob[:, i])
    auc_scores.append(auc_score)
    print("AUC Class {}: {:.2f}".format(iris.target_names[i], auc_score))

# Plot kurva ROC untuk setiap kelas
plt.figure()
for i in range(len(iris.target_names)):
    fpr, tpr, _ = roc_curve(y_test == i, y_prob[:, i])
    plt.plot(fpr, tpr, lw=2, label='ROC curve Class {} (area = {:.2f})'.format(iris.target_names[i], auc_scores[i]))
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])

```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
```

15 Deployment

15.1 Pendahuluan

Deployment model data science dilakukan untuk memastikan model yang dibuat dapat digunakan oleh orang lain. Deployment model data science dapat dilakukan dengan berbagai cara, salah satunya adalah dengan menggunakan **Streamlit**. Streamlit adalah sebuah framework yang dapat digunakan untuk membuat aplikasi web dengan menggunakan bahasa pemrograman Python. Dengan menggunakan Streamlit, deployment model data science dapat dilakukan dengan mudah dan cepat.

15.2 Instalasi Streamlit



Streamlit

Untuk menginstall Streamlit, dapat dilakukan dengan menggunakan perintah berikut pada command prompt atau terminal:

```
pip install streamlit
```

15.3 Membuat Aplikasi Web dengan Streamlit

Setelah Streamlit terinstall, langkah selanjutnya adalah membuat aplikasi web dengan menggunakan Streamlit. Untuk membuat aplikasi web dengan Streamlit, dapat dilakukan dengan cara membuat file python baru dengan nama app.py. Kemudian, pada file tersebut, tuliskan kode berikut:

```
# streamlit umumnya diinisialisasi dengan 'st'  
import streamlit as st
```

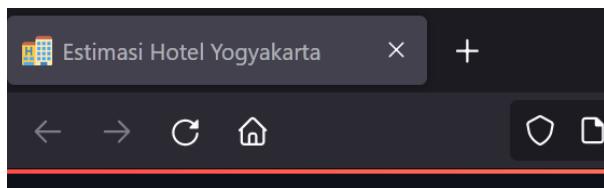
15.4 API Streamlit

API Streamlit dapat digunakan untuk membuat aplikasi web dengan Streamlit. API Streamlit dapat dilihat pada [dokumentasi Streamlit](#). Berikut adalah beberapa API Streamlit yang dapat digunakan untuk membuat aplikasi web dengan Streamlit:

15.4.1 Page Config

`st.set_page_config` dapat digunakan untuk mengatur konfigurasi halaman. Beberapa konfigurasi yang dapat diatur adalah judul halaman, layout halaman, dan lain-lain. Berikut adalah contoh penggunaan `st.set_page_config` untuk mengatur judul halaman:

```
st.set_page_config(  
    page_title="Estimasi Hotel Yogyakarta",  
    page_icon=':hotel:' # :hotel: merupakan nama emoji  
)
```



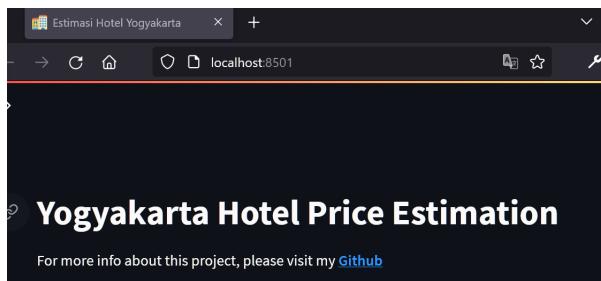
15.4.2 Write

`st.write` dapat digunakan untuk menampilkan teks, dataframe, dan visualisasi. Format penulisan dalam method ini adalah format markdown. Berikut adalah contoh penggunaan `st.write` untuk menampilkan teks:

```
st.title('Yogyakarta Hotel Price Estimation')  
st.write(  
    'For more info about this project, please visit my [**Github**] (https://github.com/Lio
```

15.4.3 Sidebar

`st.sidebar` dapat digunakan untuk membuat sidebar. Sidebar bisa digunakan sekedar untuk informasi tambahan atau bahkan bisa sebagai input user. Berikut adalah contoh penggunaan `st.sidebar` untuk membuat sidebar:



```
st.sidebar.header('User Input Features')
```

15.4.4 Input User

Untuk memasukkan elemen input user dalam sidebar dapat dilakukan dengan menggunakan `st.sidebar.slider`, `st.sidebar.selectbox` dan lain sebagainya . Berikut adalah contoh input data pada sidebar:

```
def user_input_features():
    starRating = st.sidebar.slider('Star Rating', 0, 5, 3)
    builtYear = st.sidebar.slider('Built Year', 1900, 2023, 1960)
    size = st.sidebar.slider('Room Size (m2)', 2.0,
                           100.0, 50.0, 0.1, format='%.0.1f')
    occupancy = st.sidebar.slider('Occupancy', 1, 5, 3)
    childAge = st.sidebar.slider('Child Age', 0, 18, 9)
    childOccupancy = st.sidebar.slider('Child Occupancy', 0, 5, 2)
    breakfast = st.sidebar.checkbox('Breakfast Included')
    wifi = st.sidebar.checkbox('Wifi Included')
    refund = st.sidebar.checkbox('Free Cancellation / Refund')
    livingRoom = st.sidebar.checkbox('Living Room')
    hotelFacilitie = st.sidebar.multiselect(
        'Hotel Facilities', (hotelFacilities))
    roomFacilitie = st.sidebar.multiselect(
        'Room Facilities', (roomFacilities))
    pointInterest = st.sidebar.multiselect(
        'Point of Interest', (nearestPoint))

    # Handle Checkbox
    breakfast = 1 if breakfast else 0
    wifi = 1 if wifi else 0
    refund = 1 if refund else 0
```

```

livingRoom = 1 if livingRoom else 0

# Handle Multiselect
hotelFacilitie = ','.join(hotelFacilitie)
roomFacilitie = ','.join(roomFacilitie)
pointInterest = ','.join(pointInterest)

data = {
    'starRating': starRating,
    'builtYear': builtYear,
    'size': size,
    'baseOccupancy': occupancy,
    'maxChildAge': childAge,
    'maxChildOccupancy': childOccupancy,
    'isBreakfastIncluded': breakfast,
    'isWifiIncluded': wifi,
    'isRefundable': refund,
    'hasLivingRoom': livingRoom,
    'hotelFacilities': hotelFacilitie,
    'roomFacilities': roomFacilitie,
    'nearestPoint': pointInterest
}
features = pd.DataFrame(data, index=[0])
return features

st.sidebar.header('User Input Features')
df = user_input_features()

```

- **slider** digunakan untuk memasukkan elemen input user berupa angka
- **checkbox** digunakan untuk memasukkan elemen input user berupa boolean
- **multiselect** digunakan untuk memasukkan elemen input user berupa list

15.4.5 Handling Input User

Setelah input user diterima, input user perlu diubah menjadi bentuk yang dapat digunakan oleh model. Untuk model ini perlu dilakukan multi-hot encoding untuk kolom fasilitas hotel, fasilitas kamar, dan *point of interests*. Berikut code untuk melakukan multi-hot encoding:

```

# create function to create dataframe with 0 and 1 value
def create_df(dfOri, df_name, df, prefix):

```

```

value = prefix+dfOri[df_name][0]
for i in range(0, len(df.columns)):
    column_name = df.columns[i]
    if column_name in value:
        df.loc[0, column_name] = 1
    else:
        df.loc[0, column_name] = 0
return df

# create empty dataframe for hotelFacilities, roomFacilities, nearestPoint, with column name
roomFacilities_df = pd.DataFrame(columns=roomFacilities)
hotelFacilities_df = pd.DataFrame(columns=hotelFacilities)
nearestPoint_df = pd.DataFrame(columns=nearestPoint)

create_df(df, 'roomFacilities', roomFacilities_df, 'Room_')
create_df(df, 'hotelFacilities', hotelFacilities_df, 'Hotel_')
create_df(df, 'nearestPoint', nearestPoint_df, 'Point_')

df = df.drop(['hotelFacilities', 'roomFacilities', 'nearestPoint'], axis=1)
df = pd.concat([df, hotelFacilities_df, roomFacilities_df, nearestPoint_df], axis=1)

# change all column data type to unit8 except the first column
df = df.astype({col: 'float64' for col in df.columns[:2]})
df = df.astype({col: 'uint8' for col in df.columns[2:]})

```

Code diatas digunakan untuk membuat data input user menjadi sama pada data yang dilakukan training. Hal ini dilakukan agar model dapat melakukan prediksi dengan benar.

15.4.6 Pengecekan Dataframe

Setelah data input user diubah menjadi dataframe, perlu dilakukan pengecekan apakah dataframe tersebut sudah sesuai dengan dataframe yang digunakan untuk training. Berikut adalah code untuk mengecek dataframe:

```

# check df column order with model column order using colOri, if not the same print the warning
# colOri merupakan kolom pada data training yang di export menggunakan pickle
colOri = colOri[1:]
if df.columns.tolist() == colOri.all():
    st.info("Column order is correct.")
else:
    mismatched_columns = [(idx, df_col, model_col) for idx, (df_col, model_col) in enumerate(df.columns) if df_col != model_col]

```

```
if len(mismatched_columns) > 0:  
    st.warning("The order of the columns is not the same as the model. Mismatched columns:  
    for idx, df_col, model_col in mismatched_columns:  
        st.write(f"At index {idx}: DataFrame column '{df_col}' - Model column '{model_col}'")
```

 Warning

Urutan kolom hasil input user **harus sama** dengan urutan kolom pada data training. Jika tidak, maka akan terjadi error atau membuat hasil prediksi menjadi tidak valid.

15.4.7 Prediksi

Untuk melakukan prediksi, model perlu di-load terlebih dahulu. Berikut adalah code untuk melakukan prediksi:

```
# Load Model  
xgbModel = pickle.load(open('Model/xgbModel.pkl', 'rb'))  
svrModel = pickle.load(open('Model/svrModel.pkl', 'rb'))  
rfModel = pickle.load(open('Model/rfModel.pkl', 'rb'))
```

Dalam contoh ini digunakan 3 model, yaitu XGBoost, Support Vector Regression, dan Random Forest. Untuk melakukan prediksi, dapat dilakukan dengan menggunakan code berikut:

```
st.write('Press button below to predict :')  
model = st.selectbox('Select Model', ('XGBoost', 'Random Forest', 'SVR'))  
  
if model == 'XGBoost' and st.button('Predict'): #  
    bar = st.progress(0)  
    status_text = st.empty()  
    for i in range(1, 101):  
        status_text.text("%i%% Complete" % i)  
        bar.progress(i)  
        time.sleep(0.01)  
  
    # Formatting the prediction  
    prediction = xgbModel.predict(df)  
  
    formaString = "Rp{:.2f}"  
    prediction = float(prediction[0])  
    formatted_prediction = formaString.format(prediction)
```

```

time.sleep(0.08)

# print the prediction
st.subheader('Prediction')
st.metric('Price (IDR)', formatted_prediction)

# empty the progress bar and status text
time.sleep(0.08)
bar.empty()
status_text.empty()

elif model == 'Random Forest' and st.button('Predict'):
    bar = st.progress(0)
    status_text = st.empty()
    for i in range(1, 101):
        status_text.text("%i%% Complete" % i)
        bar.progress(i)
        time.sleep(0.01)

    # Formatting the prediction
    prediction = rfModel.predict(df)

    formaString = "Rp{:.2f}"
    prediction = float(prediction[0])
    formatted_prediction = formaString.format(prediction)
    time.sleep(0.08)

    # print the prediction
    st.subheader('Prediction')
    st.metric('Price (IDR)', formatted_prediction)

    # empty the progress bar and status text
    time.sleep(0.08)
    bar.empty()
    status_text.empty()

elif model == 'SVR' and st.button('Predict'):
    bar = st.progress(0)
    status_text = st.empty()
    for i in range(1, 101):
        status_text.text("%i%% Complete" % i)

```

```
bar.progress(i)
time.sleep(0.01)

# Formatting the prediction
prediction = svrModel.predict(df)

formaString = "Rp{:.2f}"
prediction = float(prediction[0])
formatted_prediction = formaString.format(prediction)
# prediction = rfModel.predict(df)
time.sleep(0.08)

# print the prediction
st.subheader('Prediction')
st.metric('Price (IDR)', formatted_prediction)

# empty the progress bar and status text
time.sleep(0.08)
bar.empty()
status_text.empty()
```

15.4.8 Hasil akhir

Untuk contoh hasil akhir dapat dilihat pada link berikut: [Streamlit hotel Yogyakarta](#)



Part II

Studi Kasus

Berikut daftar studi kasus yang dapat digunakan untuk mempelajari materi pada modul ini:

1. Project Klasifikasi Sentimen Analisis
2. Project Regresi Prediksi Harga Kamar Hotel
3. Project Clustering - Spotify Playlist Clustering

Studi Kasus 1

Project Klasifikasi Sentimen Analisis

Klasifikasi Sentimen Analisis

Klasifikasi dapat berbentuk berupa Sentimen Analisis. Sentimen Analisis merupakan gabungan dari data mining dan text mining. Secara sederhananya merupakan proses mengolah berbagai opini dan argumen dari berbagai media sosial dalam berbagai aspek seperti jasa, produk, atau sesuai dengan isi konten pada media sosial tersebut .

Untuk dapat mengerjakan sebuah sentimen analisis, diperlukan sebuah dataset yang berisi banyak opini positif, negatif, dan atau netral. Maka dari itu pengambilan dataset merupakan proses yang penting dalam tahapan sentimen analisis. Dataset yang baik harus memiliki ukuran yang cukup besar dengan jumlah yang cukup banyak untuk meminimalkan kesalahan secara perhitungan algoritma.

Untuk mendapatkan sebuah dataset yang berisikan opini di media sosial dapat dilakukan dengan berbagai cara. Salah satunya adalah mencari dataset yang sudah jadi pada situs penyedia dataset seperti kaggle, dataset tersebut dapat dikategorikan sebagai dataset publik. Selain dengan menggunakan dataset publik, data opini dapat dibuat dengan cara scraping atau crawling. Scraping merupakan teknik yang sering digunakan untuk mendapatkan atau mengekstrak sebuah informasi pada sebuah media sosial dan atau website secara otomatis tanpa harus melakukan penyalinan secara manual.

Mencari Data Manual dengan Scraping

Berikut adalah cara mendapatkan data untuk sentimen analisis menggunakan Komentar Youtube. Dapat menggunakan API Key berikut **AIzaSyDkCRF4cmM__TtyBznV9aKptHNZqooyucqU**

```
# yang harus di instal  
  
!pip install google-api-python-client  
!pip install google-auth google-auth-oauthlib google-auth-httplib2  
!pip install pickle
```

```
!pip install sastrawi  
!pip install textblob
```

Code diatas merupakan library yang harus terinstall.

```
#library yang digunakan, jika dirasa kurang penting dapat dihapus  
import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
import string  
import re  
from nltk import word_tokenize  
import nltk  
nltk.download('punkt')  
nltk.download('stopwords')  
from nltk.corpus import stopwords  
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory  
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer  
from sklearn.model_selection import train_test_split  
from sklearn.ensemble import RandomForestClassifier  
from wordcloud import WordCloud, STOPWORDS  
from sklearn.metrics import ConfusionMatrixDisplay  
from textblob import TextBlob  
from sklearn.metrics import confusion_matrix  
from sklearn.metrics import accuracy_score
```

Setalah mengimport semua library, selanjutnya masukan code berikut

```
from google.colab import drive  
drive.mount('/content/drive')  
  
import pandas as pd  
  
## Call the "build()" function from the Python-client  
from googleapiclient.discovery import build  
  
api_key = input("API KEY: ")  
youtube = build("youtube", "v3", developerKey=api_key)  
url = input("VIDEOURL: ")  
  
def get_comments(url):
```

```

# Get the ID of the video by splitting the URL
single_video_id = url.split("=")[1].split("&")[0]
# Use the list() method to extract a JSON with key information
# from the video.
video_list=youtube.videos().list(part="snippet",id=single_video_id).execute()
channel_id= video_list["items"][0]["snippet"]["channelId"]
title_single_video= video_list["items"][0]["snippet"]["title"]
playlist_id = None
forUserName = None

nextPageToken_comments = None
commentsone=[]

while True:
    #Request the first 50 videos of a channel. This is the full dictionary. The result
    #PageToken at this point is "None"
    pl_request_comment= youtube.commentThreads().list(part=["snippet","replies"],
                                                       videoId=single_video_id,
                                                       maxResults=50,
                                                       pageToken= nextPageToken_comments)
    pl_response_comment = pl_request_comment.execute()

    ## Send the amount of views and the URL of each video to the videos empty list tha
    for i in pl_response_comment["items"]:
        vid_comments = i["snippet"]["topLevelComment"]["snippet"]["textOriginal"]
        comm_author = i["snippet"]["topLevelComment"]["snippet"]["authorDisplayName"]
        comm_author_id = i["snippet"]["topLevelComment"]["snippet"]["authorChannelId"]
        comm_date = i["snippet"]["topLevelComment"]["snippet"]["publishedAt"]
        comm_likes = i["snippet"]["topLevelComment"]["snippet"]["likeCount"]
        new_var=i.get("replies","0")

        commentsone.append({
            "comm_date":comm_date,
            "author":comm_author,
            "author_id":comm_author_id,
            "likes":comm_likes,
            "comment":vid_comments,
            "video_id":single_video_id
        })

```

```

nextPageToken_comments = pl_response_comment.get("nextPageToken")

if not nextPageToken_comments:
    break

for i in commentsone[:10]:
    print(i["comment"])

pd.DataFrame.from_dict(commentsone).to_csv(f"/content/drive/MyDrive/comments/dataset.csv")

get_comments(url)

```

pada source code scraping diatas, file yang telah di scraping akan dimasukan kedalam Google Drive dengan nama file **comments**, lalu nama file akan menjadi **dataset.csv**

untuk menampilkan hasil scraping dapat menggunakan perintah seperti berikut.

```

df = pd.read_csv('/content/drive/MyDrive/comments/dataset.csv')
df.head(500)
df.count()

```

Mencari Data dengan mengunjungi website

Data sentimen analisis bisa didapatkan melalui website penyedia data seperti kaggle dan UC Irvine Machine Learning Repository.

Sentimen Analisis menggunakan data scraping youtube

gunakan code berikut untuk melakukan sentimen analisis

```

import nltk
import pandas as pd
data = pd.read_csv("/content/drive/MyDrive/comments/dataset.csv")
data = data.dropna()
print(data.head())

data_nw = data.drop(['comm_date','author', 'author_id','likes','video_id'], axis=1 )
data_nw

```

Fungsi code diatas adalah untuk menghapus atau melakukan drop pada kolom ‘comm_date’,“author”, ‘author_id’,“likes”,‘video_id’

```
data_nw.to_csv("/content/drive/MyDrive/comments/dataset_drop.csv") #Fungsinya untuk menyimpan  
membuka file hasil dari dataset_drop.csv  
  
data_baru = pd.read_csv("/content/drive/MyDrive/comments/dataset_drop.csv")  
data_baru.head()  
  
def caseFolding(comment):  
    comment = comment.lower()  
    comment = comment.strip(" ")  
    comment = re.sub(r'[?|$|.!]',r'', comment)  
    comment = re.sub(r'[^a-zA-Z0-9 ]',r'', comment)  
    return comment  
  
data_baru['comment'] = data_baru['comment'].apply(caseFolding)
```

Fungsi dari code diatas adalah untuk menghilangkan tanda baca serta angka yang tidak dibutuhkan dan mengubah huruf kapital menjadi huruf kecil. Lalu data disimpan ke dalam kolom **comment**.

Selanjutnya simpan menggunakan perintah berikut

```
data_baru.to_csv("/content/drive/MyDrive/comments/dataset_bersih.csv")
```

Selanjutnya data yang sudah bersih, dapat diberi label negatif, positif, dan atau netral pada samping kolom comments. Label dapat di beri nama **sentiment**.

Selanjutnya dapat dilakukan uji akurasi menggunakan Algoritma atau model yang sesuai dan yang di inginkan.

Sebagai contoh menggunakan Algoritma KNN, maka dapat menggunakan code berikut.

```
import pandas as pd  
  
data = pd.read_csv('/content/drive/MyDrive/comments/dataset_bersih.csv')  
X = data['comment']  
y = data['sentimen']
```

```

# Lakukan preprocessing pada teks
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

nltk.download('stopwords')

def preprocessing(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'[^w\s]', '', text)
    text = text.strip()
    tokens = nltk.word_tokenize(text)
    stop_words = set(stopwords.words('indonesian'))
    filtered_tokens = [token for token in tokens if token not in stop_words]
    stemmer = PorterStemmer()
    stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]
    return ' '.join(stemmed_tokens)

X = X.apply(preprocessing)

# Lakukan vectorization pada teks
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(X)

# Lakukan pembagian dataset menjadi data training dan data testing
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Lakukan training model KNN
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

```

```
# Lakukan prediksi pada data testing
y_pred = knn.predict(X_test)

# Lakukan evaluasi model
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confu

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
cm = confusion_matrix(y_test, y_pred)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 Score:', f1)
print('Confusion Matrix:\n', cm)
```

Studi Kasus 2

Project Regresi Hotel Yogyakarta

Deskripsi

Project ini merupakan contoh project Data Science yang menggunakan data hotel di Yogyakarta. Project ini bertujuan untuk memprediksi harga kamar hotel berdasarkan fitur-fitur yang ada. Project ini menggunakan 3 algoritma yaitu, **XGBoost**, **Random Forest**, dan **SVM**. Project ini juga menggunakan teknik *hyperparameter tuning* untuk meningkatkan performa model. Kemudian model tersebut dilakukan deployment menggunakan **streamlit**

Tentang data

Data ini diperoleh dengan teknik *scrapping* pada website **Traveloka**. Data ini berbentuk sqlite yang berisikan 2 tabel bernama `hotel_yogyakarta` dan `hotel_room_yogyakarta`

`hotel_yogyakarta`

Berikut detail column pada tabel `hotel_yogyakarta`. **Dimensi (378, 12)**

- `id`: Unique id hotel
- `type`: Tipe penginapan
- `name`: Nama hotel
- `starRating`: Rating bintang hotel
- `builtYear`: Tahun dibuatnya hotel
- `description`: Deskripsi tentang hotel
- `link`: URL menuju halaman hotel di Traveloka
- `address`: Alamat hotel
- `city`: Kota hotel
- `image`: URL gambar hotel
- `facilities`: Daftar fasilitas pada hotel
- `nearestPointofInterests`: Area populer / fasilitas umum disekitar hotel

hotel_room_yogyakarta

Berikut detail column pada tabel hotel_room_yogyakarta. **Dimensi (1199, 16)**

- **id**: Unique id hotel
- **hotelId**: Id hotel
- **roomType**: Tipe kamar hotel
- **description**: deskripsi kamar hotel
- **bedDescription**: deskripsi kasur kamar
- **size**: Ukuran kamar (m^2)
- **originalRate**: Harga kamar per malam
- **baseOccupancy**: Kapasitas kamar
- **maxChildAge**: Umur maksimal anak-anak
- **maxChildOccupancy**: Kapasitas kamar untuk anak-anak
- **numExtraBeds**: Jumlah kasur tambahan
- **isBreakfastIncluded**: Fasilitas sarapan
- **isWifiIncluded**: Fasilitas WiFi
- **isRefundable**: Fasilitas refund
- **hasLivingRoom**: Fasilitas ruang keluarga
- **facilities**: Daftar fasilitas lainnya pada kamar

Data analysis

Import library

```
from sqlite3 import connect
import pickle
import pandas as pd
import json
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MultiLabelBinarizer
```

Load data

Buat koneksi ke **database sqlite**, lalu baca tabel **hotel_yogyakarta** dan **hotel_room_yogyakarta** menjadi dataframe pandas.

```

# SQLite3 connection
con = connect('./dataset/hotel-directories-ORI.sqlite3')
df_sql_hotel = pd.read_sql_query("SELECT * from hotel_yogyakarta", con=con)
df_sql_room = pd.read_sql_query("SELECT * from hotel_room_yogyakarta", con=con)
con.close()

# Table columns
print('Kolom Tabel Hotel :')
print(df_sql_hotel.columns)
print("Total Baris :", df_sql_hotel.shape[0])
print("Total Kolom :", df_sql_hotel.shape[1])

print('-' * 50)

print('Kolom Tabel Kamar:')
print(df_sql_room.columns)
print("Total Baris :", df_sql_room.shape[0])
print("Total Kolom :", df_sql_room.shape[1])

```

Untuk melihat hasil dataframe dapat dilakukan menggunakan code berikut.

```

print(f"Dimensi : {df_sql_hotel.shape}")
df_sql_hotel.sample(2)

print(f"Dimensi : {df_sql_room.shape}")
df_sql_room.sample(2)

# Fungsi menghitung unique value
def check_unique(df):
    count = 0
    for i in df.columns:
        if df[i].nunique() == 1:
            count += 1
            print(f'{i}: {df[i].nunique()}')
        else:
            print(f'{i}: {df[i].nunique()}')
    if count == 0:
        print('No columns with only one unique value')

```

Gunakan fungsi `check_unique()` untuk mengecek apakah terdapat data dengan unique value kurang dari 2. Jika ada, maka data tersebut tidak akan digunakan.

```
check_unique(df_sql_hotel)
```

```
check_unique(df_sql_room)
```

- Nilai penghubung kedua tabel adalah **id** pada data **hotel** dan **hotelId** pada data **kamar**.
- Terdapat beberapa beberapa kolom yang tidak digunakan pada analisis ini
 - **Hotel**: name, description, link, address, dan image.
 - **Kamar**: id, roomType, description, dan bedDescription.
- Terdapat beberapa kolom dengan total nilai unik kurang dari 2
 - **Hotel**: type dan city.
 - **Kamar**: bedDescription dan numExtraBeds
- Setelah dilakukan penghapusan kolom selanjutnya tabel akan di-*merge* menjadi satu dataframe.
- **starRating** memiliki 8 nilai unik, perlu di teliti lebih lanjut untuk detailnya.

Menghapus Kolom

```
hotelDrop = ['name', 'description', 'link', 'address', 'image', 'type', 'city']
roomDrop = ['id', 'roomType', 'description', 'bedDescription', 'numExtraBeds']

df_hotel = df_sql_hotel.drop(hotelDrop, axis=1)
df_room = df_sql_room.drop(roomDrop, axis=1)

print('Total Hotel Table Data : ', df_hotel.shape[0])
print('Total Hotel Table Column : ', df_hotel.shape[1])
print('-' * 30)
print('Total Room Table Data : ', df_room.shape[0])
print('Total Room Table Column : ', df_room.shape[1])
```

Merge Data

```
# Rename column
df_hotel.rename(columns={'id': 'hotelId'}, inplace=True)
df_hotel.rename(columns={'facilities': 'hotelFacilities'}, inplace=True)
df_room.rename(columns={'facilities': 'roomFacilities'}, inplace=True)
```

- Menyamakan nama `id` pada tabel `hotel` dan `hotelId` pada tabel `kamar`.
- Menambah prefix `hotel` dan `room` pada tiap kolom `facilities` masing-masing tabel.

```
# merge hotel dan room data
df = pd.merge(df_hotel, df_room, on='hotelId', how='inner')

# remove id column
df.drop(columns=['hotelId'], inplace=True)

# re arrange column
df = df[['originalRate', 'starRating', 'builtYear', 'size', 'baseOccupancy', 'maxChildAge',
         'maxChildOccupancy', 'isBreakfastIncluded', 'isWifiIncluded', 'isRefundable',
         'hasLivingRoom', 'hotelFacilities', 'roomFacilities', 'nearestPointOfInterests']]

df
```

- Dimensi pasca penggabungan adalah `(1199, 14)`
- Menghapus kolom `hotelId` karena sudah tidak diperlukan lagi.
- Mengubah urutan kolom untuk mempermudah analisis, target kolom berada di kiri dan kolom array berada di kanan.

Target Processing

```
for i in range(0, 5):
    print(df.loc[i, 'originalRate'])

• Data berbentuk JSON atau Dictionary, maka perlu diubah menjadi nilai harga dengan 1 nilai.
• Karena semua data menggunakan matauang IDR, maka currency tidak diperlukan.

# Extract the amount from originalRate using a lambda function
df['rate'] = df['originalRate'].apply(lambda x: json.loads(x)['amount'])
df['tax'] = df['originalRate'].apply(lambda x: json.loads(x)['tax'])
df = df.drop(columns=['originalRate'])

# check tipe data
print('Tipe data harga :', df['rate'].dtype)
print('Tipe data pajak :', df['tax'].dtype)
```

```

# ubah tipe data
df['rate'] = df['rate'].astype('int')
df['tax'] = df['tax'].astype('int')

print('Tipe data harga :', df['rate'].dtype)
print('Tipe data pajak :', df['tax'].dtype)

```

Rasio Pajak

Tahap ini bertujuan untuk mengetahui rasio pajak dari harga kamar hotel. Rasio pajak ini akan digunakan untuk menghitung harga kamar hotel setelah dikenakan pajak.

```

# create series for original rate
original_rate = df['rate']
# create series for tax
tax = df['tax']
# create dataframe for original rate, tax, and tax rate
df_rate = pd.DataFrame({'original_rate': original_rate, 'tax': tax})
df_rate['tax_rate'] = df_rate['tax'] / df_rate['original_rate'] * 100

df_rate

# count number of data with tax rate 20% and under 21%, also over 21%
count0 = 0
count_20 = 0
countBetween = 0
for i in range(len(df_rate['tax_rate'])):
    if df_rate['tax_rate'][i] < 20 and df_rate['tax_rate'][i] > 0:
        countBetween += 1
    elif df_rate['tax_rate'][i] >= 20:
        count_20 += 1
    elif df_rate['tax_rate'][i] == 0:
        count0 += 1
print('Median pajak : ', df_rate['tax_rate'].median())
print('null / 0% pajak: ', count0)
print('Pajak diantara 0 - 20%:', countBetween)
print('Rasio pajak diatas 20% :', count_20)

```

```

# plot for tax rate and give the total value on the top of the bar if the value is 0% it w
sns.set(rc={'figure.figsize': (10, 7)})
sns.set_style('darkgrid')
ax = sns.histplot(df_rate['tax_rate'], kde=False, color='dodgerblue', bins=9)
ax.set(xlabel='Tax Rate (%)', ylabel='Count')
ax.set_title('Tax Rate Distribution')
total = len(df_rate['tax_rate'])
for p in ax.patches:
    height = p.get_height()
    if height != 0:
        ax.text(p.get_x() + p.get_width() / 2.,
                 height + 15,
                 '{:1.2f}%'.format(100 * height / total),
                 ha="center")
plt.show()

# Menghapus kolom pajak
df = df.drop(columns=['tax'])

# Mengubah target kolom menjadi di awal
# sekedar untuk merapikan dataframe
df = df[['rate']] + [col for col in df.columns if col != 'rate']]
df.columns

```

- Pajak hotel di Yogyakarta ada di kisaran 20-22% dengan median 21%
- Karena 94% data memiliki pajak dikisaran tersebut maka nilai pajak dianggap 21% (*median*) secara keseluruhan.

Validasi Data

Pengecekan Rating Bintang

```

# starRating Distribution
value = df.starRating.value_counts()
print('OriginalRate Distribution by starRating')
print(value)

# starRating Distribution by percentage
value_percentage = value / len(df) * 100

```

```

# create a list of tuples where each tuple contains the value and index of each element in
value_percentage_list = [(value_percentage[i], i)
                           for i in value_percentage.index]

# sort the list by the value in descending order
value_percentage_list_sorted = sorted(value_percentage_list, reverse=True)

fig, ax = plt.subplots(1, 2, figsize=(20, 10), dpi=80,
                      gridspec_kw={'width_ratios': [1, 0.5]})
sns.histplot(df, x="rate", hue='starRating', palette='bright',
              ax=ax[0]).set(title='OriginalRate Distribution by starRating')

# starRating percentage plot
sns.barplot(x=value_percentage.index, y=value_percentage.values,
             palette='bright', ax=ax[1]).set(title='starRating percentage')

# add the percentage text using the sorted list
for container in ax[1].containers:
    for bar in container.patches:
        v = bar.get_height()
        bar_center = bar.get_x() + bar.get_width() / 2
        ax[1].text(bar_center, v + 0.5,
                   f'{v:.2f}%', color='black', fontweight='bold', ha='center')
fig.tight_layout()

```

- Rating bintang memiliki beberapa nilai dengan 0.5 (desimal), tetapi nilai tersebut hanya memiliki persentase jumlah data yang sedikit, maka dari itu rating tersebut dihilangkan angka desimalnya dari rating seharusnya.

```

# ubah starRating dengan angka bulat
df['starRating'] = df['starRating'].replace(2.5, 2)
df['starRating'] = df['starRating'].replace(3.5, 3)

# starRating Distribution
value = df.starRating.value_counts()
print('OriginalRate Distribution by starRating')
print(value)

# starRating Distribution by percentage
value_percentage = value / len(df) * 100

```

```

# create a list of tuples where each tuple contains the value and index of each element in
value_percentage_list = [(value_percentage[i], i)
                           for i in value_percentage.index]

# sort the list by the value in descending order
value_percentage_list_sorted = sorted(value_percentage_list, reverse=True)

fig, ax = plt.subplots(1, 2, figsize=(20, 10), dpi=80,
                      gridspec_kw={'width_ratios': [1, 0.5]})
sns.histplot(df, x="rate", hue='starRating', palette='bright',
              ax=ax[0]).set(title='OriginalRate Distribution by starRating')

# starRating percentage plot
sns.barplot(x=value_percentage.index, y=value_percentage.values,
             palette='bright', ax=ax[1]).set(title='starRating percentage')

# add the percentage text using the sorted list
for container in ax[1].containers:
    for bar in container.patches:
        v = bar.get_height()
        bar_center = bar.get_x() + bar.get_width() / 2
        ax[1].text(bar_center, v + 0.5,
                   f'{v:.2f}%', color='black', fontweight='bold', ha='center')
fig.tight_layout()

```

- Persentase persebaran data tiap rating bintang sudah lebih baik setelah dilakukan pengubahan nilai rating bintang.
- Terlihat pada plot yang **kiri** bahwa terdapat ekor yang sangat panjang, ini menunjukkan adanya outlier pada kolom harga(**rate**)

```
df.info()
```

- Kolom **starRating** masih bertipe data **float** walau sudah tidak memiliki angka desimal, maka perlu akan menjadi **int**
- Kolom **builtYear** harus diganti ke tipe data **int**
- Kolom **size** harus diganti ke tipe data **float** (tipe data dasar dari tabel sqlite adalah **float**)
- Terdapat nilai **null** pada kolom **builtYear** dan **size** yang harus ditangani

```

df['starRating'] = df['starRating'].astype('int')
print('Tipe data starRating :', df['starRating'].dtype)

```

Data Cleaning

15.4.0.0.1 * Check Duplicate Data

```
# show index who has duplicate value
print('Total duplicated row = ', df.duplicated().sum())
# print duplicated data list index 1
df[df.duplicated(keep=False)]  
  
# drop duplicate data
df = df.drop_duplicates(keep='first')
df.shape
```

- Dengan menggunakan parameter `keep = 'first'` maka data yang duplikat akan dihapus kecuali data pertama yang muncul.

15.4.0.0.2 * Check Null

```
# Jumlah baris data
jumlah_baris_ori = df.shape[0]  
  
# crate dataframe for null value
df_null = pd.DataFrame(df.isnull().sum(), columns=['null_value'])
df_null['null_value_percentage'] = df_null['null_value'] / len(df) * 100
df_null
```

- Terdapat 2 data yang memiliki nilai `null` dengan persentase yang cukup tinggi, yaitu kolom `builtYear` dan `size`. Oleh karena itu data tersebut akan diubah dengan nilai median per rating hotel.

```
# create new dataframe for null value rows
df_null_rows = df[df.isnull().any(axis=1)]
df_null_rows  
  
# ubah sementara null value menjadi 0
df['builtYear'] = df['builtYear'].fillna(0)
df['size'] = df['size'].fillna(0)  
  
# ubah tipe data
df['builtYear'] = df['builtYear'].astype('int32')
```

```
df['size'] = df['size'].astype('float')

print('Tipe data builtYear :', df['builtYear'].dtype)
print('Tipe data size :', df['size'].dtype)
```

i Note

Kolom yang memiliki nilai `null` akan membuat tipe data menjadi `object`, maka dari itu pada penelitian ini akan diisi dengan nilai 0 terlebih dahulu, kemudian kolom tersebut diubah tipe datanya

```
# ubah nilai 0 pada kolom builtYear menjadi median tiap starRating
for i in df['starRating'].unique():
    df.loc[(df['starRating'] == i) & (df['builtYear'] == 0),
           'builtYear'] = df[df['starRating'] == i]['builtYear'].median()

# ubah nilai 0 pada kolom size menjadi median tiap starRating
for i in df['starRating'].unique():
    df.loc[(df['starRating'] == i) & (df['size'] == 0),
           'size'] = df[df['starRating'] == i]['size'].median()

# crate dataframe for null value
df_null = pd.DataFrame(df.isnull().sum(), columns=['null_value'])
df_null['null_value_percentage'] = df_null['null_value'] / len(df) * 100
df_null
```

- Data sudah tidak memiliki nilai `null`

Statistik Deskriptif

```
df.describe()
```

- Pada kolom `builtYear` terdapat nilai minimum 1 yang tidak mungkin terjadi, maka data tersebut akan dihapus.
- nilai median pada `rate` dan `size` terpaut cukup jauh dengan nilai maximum, ini menunjukkan adanya outlier pada kolom `rate`.

15.4.0.0.1 * Built Year Data Handling

```
# Cek nilai unique pada kolom builtYear dibawah 2000
print('Nilai unique builtYear dibawah 2000 :')
print(df[df['builtYear'] < 2000]['builtYear'].unique())

print('Nilai unique bulitYear dibawah 1900 :')
print(df[df['builtYear'] < 1900]['builtYear'].unique())
```

- Terdapat data yang memiliki nilai `builtYear` yang tidak mungkin terjadi, maka data tersebut akan dihapus.
- Tidak terdapat hotel dibawah tahun 1900, maka dari itu data yang disimpan adalah data diatas tahun 1900.

```
# menghapus baris yang memiliki nilai dibawah 1900 pada kolom builtYear
df = df[df['builtYear'] > 1900]

# Cek nilai unique pada kolom builtYear dibawah 2000
print('Nilai unique builtYear dibawah 2000 :')
print(df[df['builtYear'] < 2000]['builtYear'].unique())
```

Outlier Handling

15.4.0.0.1 * Rate Data

```
# Statistik Harga
print('Harga')
print(f'maximum value : {df.rate.max()}')
print(f'minimum value : {df.rate.min()}')
print(f'skew value : {round(df.rate.skew(), 2)}')

# Distribusi harga
sns.set_style('darkgrid')
plt.figure(figsize=(20, 10), dpi=80)
sns.displot(df, x="rate", kind="kde", fill=True).set(
    title='OriginalRate Distribution')
plt.show()
```

- Kolom `rate` memiliki nilai `skew` yang cukup tinggi, selain itu dari plot terlihat memiliki ekor yang cukup panjang. Ini menunjukkan adanya outlier pada kolom `rate`.
- Penghapusan outlier dilakukan dengan menggunakan metode IQR.

Note

Untuk penjelasan lebih lanjut mengenai *skew* dapat dilihat [disini](#)

```
# Hitung outlier pada kolom rate
Q1 = df['rate'].quantile(0.25)
Q3 = df['rate'].quantile(0.75)
IQR = Q3 - Q1

print('Batas bawah :', Q1 - (1.5 * IQR))
print('Batas atas :', Q3 + (1.5 * IQR))

# Hitung jumlah outlier
total_outlier = len(df[(df['rate'] < (Q1 - (1.5 * IQR))) | (df['rate'] > (Q3 + (1.5 * IQR)))]
print('Jumlah outlier :', total_outlier)
```

- Terdapat 91 data outlier pada kolom **rate**.

```
# Hapus outlier
df = df[(df['rate'] > (Q1 - (1.5 * IQR))) & (df['rate'] < (Q3 + (1.5 * IQR)))]
df.describe()
```

- Nilai maksimum sudah cukup menurun setelah dilakukan penghapusan outlier.
- Nilai maksimum pada kolom **size** juga ikut menurun.

```
# Statistik Harga
print('Harga')
print(f'maximum value : {df.rate.max()}')
print(f'minimum value : {df.rate.min()}')
print(f'skew value : {round(df.rate.skew(), 2)}')

# Distribusi harga
sns.set_style('darkgrid')
plt.figure(figsize=(20, 10), dpi=80)
sns.displot(df, x="rate", kind="kde", fill=True).set(
    title='OriginalRate Distribution')
plt.show()
```

- Nilai skew pada kolom **rate** sudah menjadi lebih baik setelah dilakukan penghapusan outlier.

15.4.0.0.2 * Size Data

```
# Statistik Size
print('Size')
print('maximum value : {}'.format(df['size'].max()))
print('minimum value : {}'.format(df['size'].min()))
print('skew value : {}'.format(df['size'].skew()))

# Distribusi size
plt.figure(figsize=(20, 10), dpi=80)
sns.set_style('darkgrid')
sns.jointplot(data=df, x='size', y='rate')
plt.show()
```

- Nilai skew sudah sangat mendekati angka 1, ini menunjukkan bahwa distribusi data sudah sangat baik.

```
# Hasil data cleaning
print('Total baris data awal :', jumlah_baris_ori)
print('Total baris data yang dihapus :', jumlah_baris_ori - df.shape[0])
print('Total baris data setelah cleaning :', df.shape[0])

df.isnull().sum()
```

Encoding Data

```
df = df.reset_index(drop=True)
df.info()
```

- Kolom `hotelfacilities`, `roomfacilities`, dan `nearestPointofInterests` merupakan sebuah fitur dengan multi label. Oleh karena itu data tersebut akan dilakukan *multi-hot encoding*.
- Proses tersebut akan dilakukan dengan library `sklearn.preprocessing.MultiLabelBinarizer`

i Note

Untuk penjelasan lebih lanjut tentang *multi-label* dan *multi-class* dapat dilihat [disini](#)

Check data format

```
df['hotelFacilities'].head(2)

df['roomFacilities'].head(2)

df['nearestPointOfInterests'].head(2)
```

- `roomfacilities` dan `hotelFacilities` memiliki format yang sama, yaitu `list` yang berisi `string`.
- `nearestPointofInterests` memiliki format yang berbeda, yaitu `list` yang berisi `dictionary/json` yang berisi `string` dan `float`.

Data Preprocessing

```
# create a MultiLabelBinarizer object
mlb = MultiLabelBinarizer()

# Daftar kolom hasil encoding akan diexport menjadi file pkl yang akan digunakan pada
# aplikasi streamlit.

# reformat kolom hotelFacilities
df['hotelFacilities'] = df['hotelFacilities'].apply(eval)

# multi label binarizer untuk kolom hotelFacilities dengan preifx Hotel_
# hotel_facilities = pd.DataFrame(mlb.fit_transform(
#     df['hotelFacilities']), columns=[f'Hotel_{col}' for col in mlb.classes_])

# hotelNewCol = hotel_facilities.shape[1]
# print('Jumlah kolom :', hotel_facilities.shape[1])

# export hotel_facilities with pickle
# hotelFacilities = hotel_facilities.columns.tolist()
# pickle.dump(hotelFacilities, open('hotelFacilities.pkl', 'wb'))

# hotel_facilities.head(2)

# reformat kolom roomFacilities
df['roomFacilities'] = df['roomFacilities'].apply(eval)
```

```

# multi label binarizer untuk kolom roomFacilities dengan preifx Room_
room_facilities = pd.DataFrame(mlb.fit_transform(df['roomFacilities']), columns=[f'Room_{col}' for col in mlb.classes_])

roomNewCol = room_facilities.shape[1]
print('Jumlah kolom :', roomNewCol)

# export room_facilities with pickle
roomFacilities = room_facilities.columns.tolist()
pickle.dump(roomFacilities, open('roomFacilities.pkl', 'wb'))

room_facilities.head(2)

# reformat kolom nearestPointOfInterests
df['nearestPointOfInterests'] = df['nearestPointOfInterests'].apply(
    lambda x: [item['landmarkType'] for item in json.loads(x)])

# multi label binarizer untuk kolom nearestPointOfInterests dengan preifx Point_
pointOfInterests = pd.DataFrame(mlb.fit_transform(
    df['nearestPointOfInterests']), columns=[f'Point_{col}' for col in mlb.classes_])

pointNewCol = pointOfInterests.shape[1]
print('Jumlah kolom :', pointNewCol)

# export pointOfInterests with pickle
pointInterests = pointOfInterests.columns.tolist()
pickle.dump(pointInterests, open('pointInterests.pkl', 'wb'))

pointOfInterests.head(2)

```

15.4.0.0.1 * Menggabungkan hasil encoding

```

# Total kolom encoding
totalNewCol = hotelNewCol + roomNewCol + pointNewCol
print('Total kolom encoding :', totalNewCol)

# menghapus kolom hotelFacilities, roomFacilities, dan nearestPointOfInterests
df = df.drop(columns=['hotelFacilities', 'roomFacilities', 'nearestPointOfInterests'])

```

```
print('df shape :', df.shape)
print('hotel_facilities shape :', hotel_facilities.shape)
print('room_facilities shape :', room_facilities.shape)
print('pointOfInterests shape :', pointOfInterests.shape)

df = pd.concat([df, hotel_facilities, room_facilities,
                pointOfInterests], axis=1)
df.head()

df.info()
```

Export Data ke CSV

```
df.to_csv('kamar-hotel-yogyakarta.csv', index=False)

col = df.columns

# export col with pickle
pickle.dump(col, open('col.pkl', 'wb'))

print(col)
```

Data Anlisis

```
value = df.starRating.value_counts()
print(value)

# starRating Distribution by percentage
value_percentage = value / len(df) * 100

# create a list of tuples where each tuple contains the value and index of each element in
value_percentage_list = [(value_percentage[i], i)
                           for i in range(len(value_percentage))]

# sort the list by the value in descending order
value_percentage_list_sorted = sorted(value_percentage_list, reverse=True)
```

```

fig, ax = plt.subplots(1, 2, figsize=(20, 10), dpi=80,
                      gridspec_kw={'width_ratios': [1, 0.5]})
sns.histplot(df, x="rate", hue='starRating', palette='bright',
             ax=ax[0]).set(title='Rate Distribution by starRating')

sns.barplot(x=value_percentage.index, y=value_percentage.values,
             palette='bright', ax=ax[1]).set(title='starRating percentage')

# add the percentage text using the sorted list
for i, (v, index) in enumerate(value_percentage_list_sorted):
    ax[1].text(index, v + 0.5, str(round(v, 2)) + '%',
               color='black', fontweight='bold', ha='center')

fig.tight_layout()

filtered_0 = df[df['starRating'] == 0.0]
filtered_1 = df[df['starRating'] == 1.0]
filtered_2 = df[df['starRating'] == 2.0]
filtered_3 = df[df['starRating'] == 3.0]
filtered_4 = df[df['starRating'] == 4.0]
filtered_5 = df[df['starRating'] == 5.0]

print('Skew value for every starRating')
print(df.groupby('starRating')['rate'].skew())

# OriginalRate Distribution by starRating using hisplot inside subplot
fig, ax = plt.subplots(2, 3, figsize=(20, 10), dpi=80,
                      gridspec_kw={'width_ratios': [1, 1, 1]})

sns.histplot(filtered_0, x="rate", ax=ax[0, 0]).set(
    title='Rate Distribution by 0 starRating')
sns.histplot(filtered_1, x="rate", ax=ax[0, 1]).set(
    title='Rate Distribution by 1 starRating')
sns.histplot(filtered_2, x="rate", ax=ax[0, 2]).set(
    title='Rate Distribution by 2 starRating')
sns.histplot(filtered_3, x="rate", ax=ax[1, 0]).set(
    title='Rate Distribution by 3 starRating')
sns.histplot(filtered_4, x="rate", ax=ax[1, 1]).set(
    title='Rate Distribution by 4 starRating')
sns.histplot(filtered_5, x="rate", ax=ax[1, 2]).set(
    title='Rate Distribution by 5 starRating')

fig.tight_layout()

```

```
# Statistik Harga tiap rating bintang hotel

dfRateStat = df.groupby('starRating').agg(
    {'rate': ['mean', 'std', 'min', 'max', lambda x: x.quantile(0.25), 'median', lambda x: x.quantile(0.75)]})

# change the column name from index 4 and 6
dfRateStat = dfRateStat.rename(
    columns={'<lambda_0>': '25%', '<lambda_1>': '75%'})
dfRateStat
```

- Pada kota Yogyakarta tidak terdapat banyak hotel bintang 5.
- Mayoritas hotel di Yogyakarta adalah hotel bintang 0.
- Terdapat hotel bintang 2 yang memiliki harga setara dengan hotel bintang 5.

i Note

Dari tabel statistik tersebut mengindikasikan beberapa nilai yang tidak wajar. Untuk penelitian selanjutnya bisa dilakukan pembersihan data lebih mendalam lagi.

i Note

Hasil dari *multi-hot encoding* juga belum dilakukan **pembersihan data**, selain itu dengan banyaknya hasil kolom juga dapat dilakukan reduksi dimensi, contohnya menggunakan PCA. Oleh karena itu untuk penelitian selanjutnya bisa dilakukan pembersihan data lebih mendalam lagi dan dilakukan reduksi dimensi.

Masih banyak informasi-informasi yang dapat di ambil dari data ini, seperti:

- Perbandingan harga hotel bintang 5 dengan hotel bintang 0.
- Landmark apa yang paling banyak dicari oleh pengunjung hotel?
- Fasilitas apa yang sudah menjadi standar pada hotel bintang 3?
- Dan masih banyak lagi.

Pemodelan Machine Learning

```
import numpy as np
import xgboost as xgb
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import train_test_split, HalvingGridSearchCV
from sklearn.svm import SVR
```

```
from sklearn.ensemble import RandomForestRegressor  
from sklearn.metrics import mean_squared_error, r2_score  
from IPython import display
```

Studi Kasus 3

Project Clustering - Spotify Playlist Clustering

Project Overview

Project ini berfungsi untuk membuat sebuah aplikasi untuk melakukan proses clustering terhadap lagu-lagu yang ada di dalam sebuah playlist spotify.

Lagu-lagu tersebut dikelompokan berdasarkan karakteristik lagu (audio feature) yang kita dapatkan dari spotify web API.

Perlu diingat bahwa untuk project ini, kita akan buat algoritmanya dahulu, mengetesnya di notebook kita, lalu kita akan mengubahnya menjadi file python dan mengupload nya di Streamlit.

Spotify

Spotify adalah platform musik digital yang menyediakan layanan streaming musik dan podcast. Musik dan podcast di-stream melalui internet tanpa perlu mengunduhnya secara permanen. Pengguna dapat membuat playlist pribadi dan menyimpan lagu favorit di Library.

Python

Python adalah bahasa pemrograman serba guna yang dikembangkan pada awal 1990-an oleh Guido van Rossum. Tidak perlu mendeklarasikan tipe data variabel, dan nilai variabel dapat berubah saat program berjalan. Python mengutamakan kejelasan dan menghindari penggunaan tanda kurung kurawal atau titik koma. Python menyediakan pustaka bawaan (standard library) yang kaya, serta banyak modul dan pustaka dari pihak ketiga yang memperluas fungsionalitasnya.



Figure 15.1: Gambar1. Logo Spotify

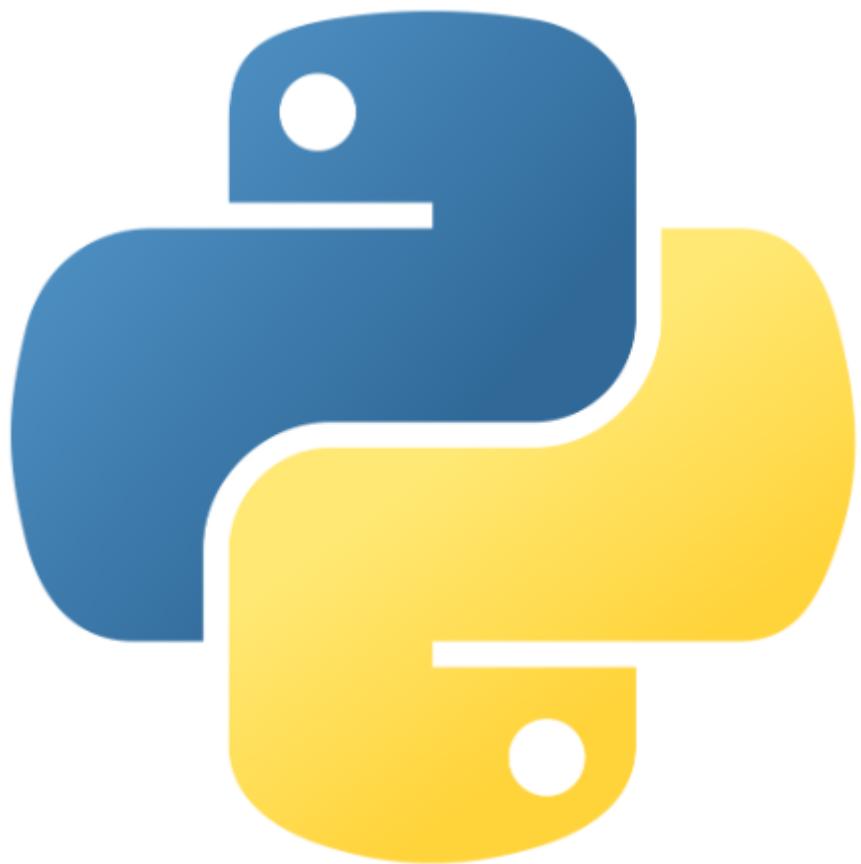


Figure 15.2: Gambar2. Logo Python



Figure 15.3: Gambar3. Logo Streamlit

Streamlit

Streamlit adalah framework open-source untuk mengembangkan aplikasi web interaktif dengan menggunakan bahasa pemrograman Python. Tujuannya adalah menyederhanakan proses pembuatan aplikasi web dengan memungkinkan pengembang untuk membuat aplikasi dengan mudah menggunakan kode Python yang sederhana dan familiar.

Clustering

Clustering adalah proses pengelompokan data atau objek-objek serupa menjadi kelompok-kelompok yang lebih homogen berdasarkan kesamaan fitur atau karakteristik tertentu.

K-Means

Algoritma k-means adalah metode clustering yang mengelompokkan data menjadi beberapa kelompok berdasarkan jaraknya ke pusat kelompok yang ditentukan secara iteratif.

Untuk penjelasan Algoritma Clustering K-Means, bisa mengunjungi video ini <https://www.youtube.com/watch?>

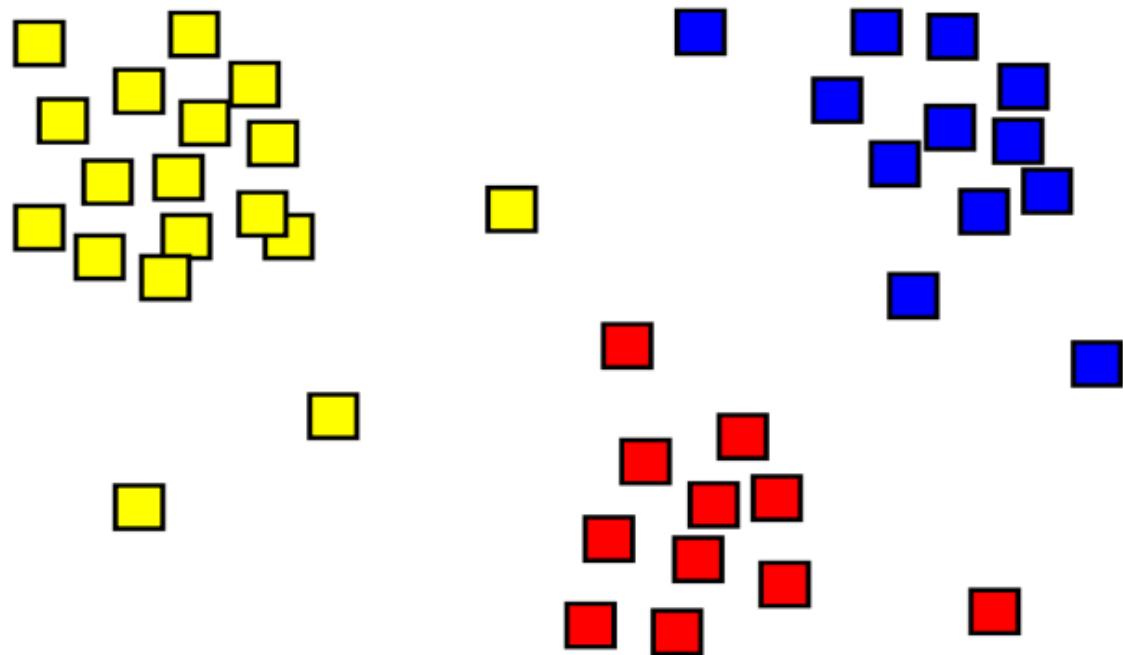


Figure 15.4: Gambar4. Contoh Grafik Clustering

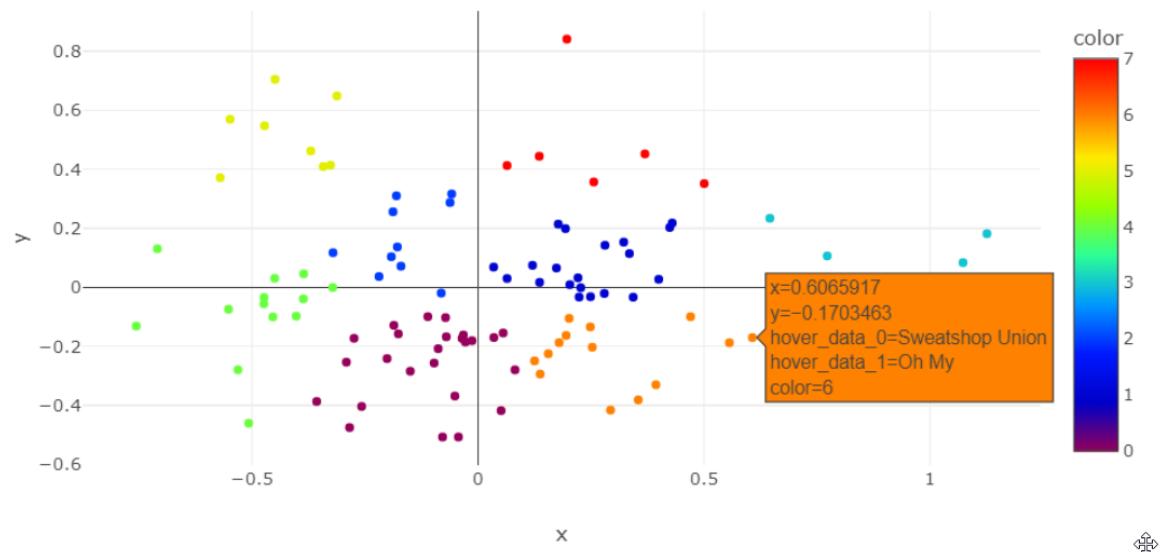


Figure 15.5: Gambar5. Contoh Clustering Menggunakan Scatter Plot

Application Flow

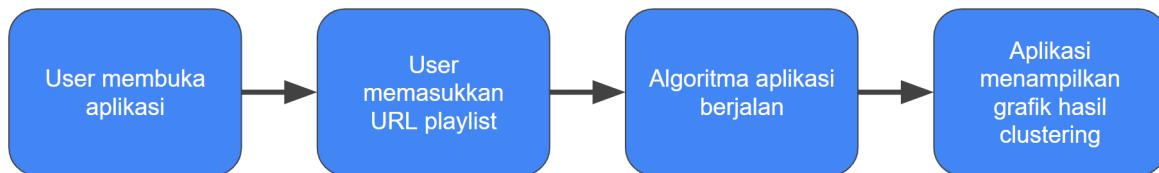
User flow adalah interaksi antara user dan aplikasi ini harus kita pelajari dan petakan secara detail agar aplikasi bisa digunakan dengan user secara nyaman.

Data flow adalah alur dari data yang ada di dalam aplikasi.

Function flow adalah struktur dari fungsi yang ada di dalam aplikasi

User Flow

User flow dari aplikasi yang akan kita buat cukup simple :



Data Flow

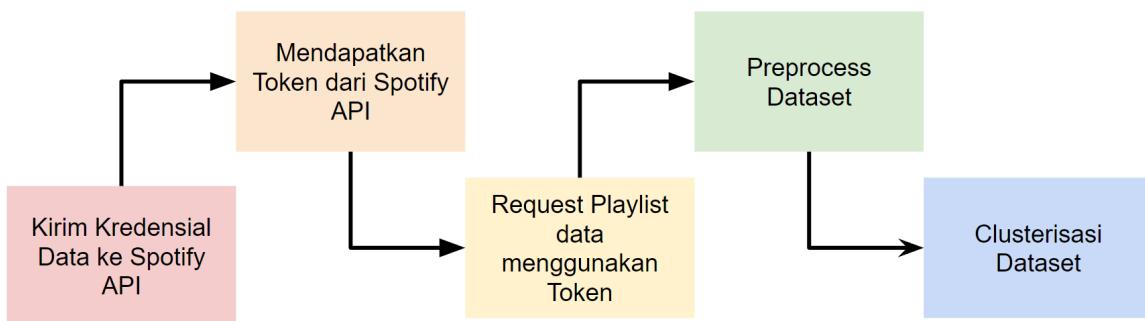


Figure 15.6: Gambar7. Data Flow

Function Flow

Membuat Akun Spotify Developer

Daftar Sebagai Spotify Developer

Untuk mengakses web API spotify, kita harus mendaftar sebagai spotify developer. Berikut langkahnya:

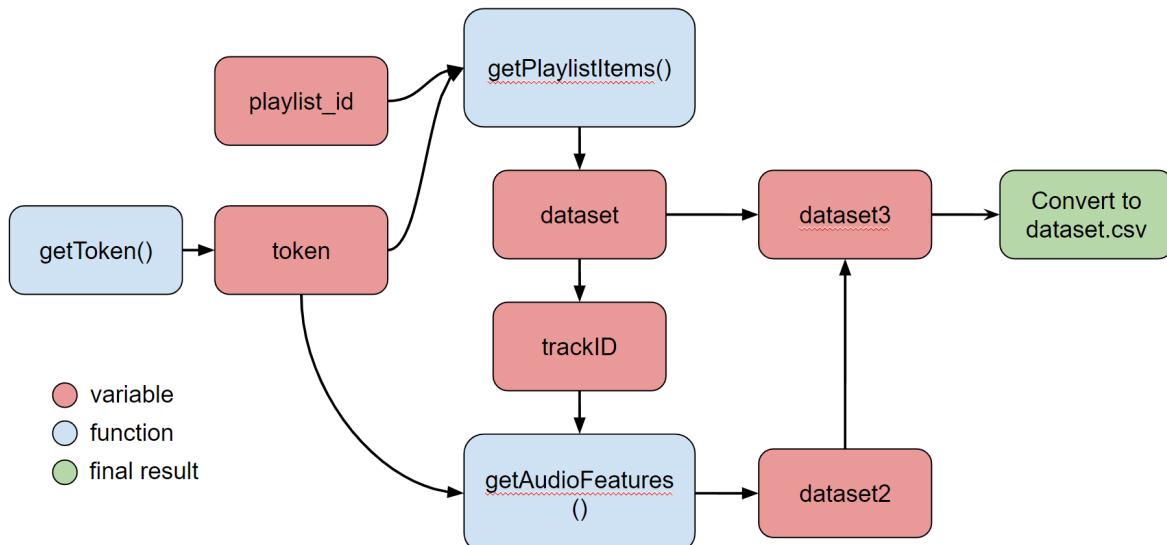


Figure 15.7: Gambar8. Function Flow

- Pergi ke <https://developer.spotify.com>
 - Klik tombol daftar di pojok kanan atas
 - Isi form, klik daftar, verifikasi email
 - Login dengan akun terverifikasi
 - Pergi ke dashboard
- Jika anda sudah bisa masuk dashboard, berarti anda sudah terdaftar menjadi developer spotify! Ikuti langkah-langkah selanjutnya.

Pergi ke dashboard spotify dan tekan create app

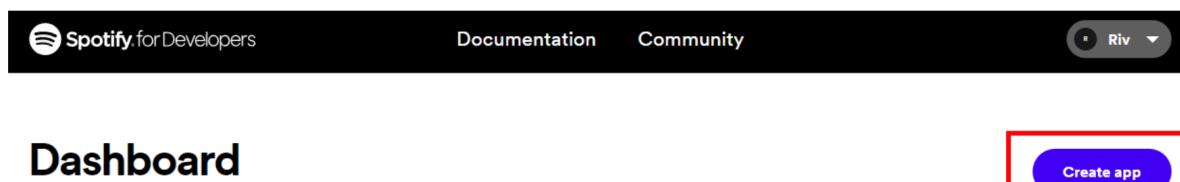


Figure 15.8: Gambar9. Dashboard Spotify Developer

Create app

App name

App description

Website

Redirect URI

A URI where users can be redirected after authentication success or failure

I understand and agree with Spotify's [Developer Terms of Service](#) and [Design Guidelines](#)

Save **Cancel**

Figure 15.9: Gambar10. Isi Form di halaman Create App

Isi form dengan data

App name, nama dari aplikasi, misal spotify_playlist_clusterization

App description, deskripsi singkat dari aplikasi yang akan dibuat

Website, website personal dari pembuat aplikasi

Redirect URI, umumnya user akan ditujukan ke URI ini ketika mengalami kegagalan request. namun untuk project ini, kita dapat mengisinya dengan apapun, seperti URL akun github atau linkedin

Klik app yang sudah dibuat

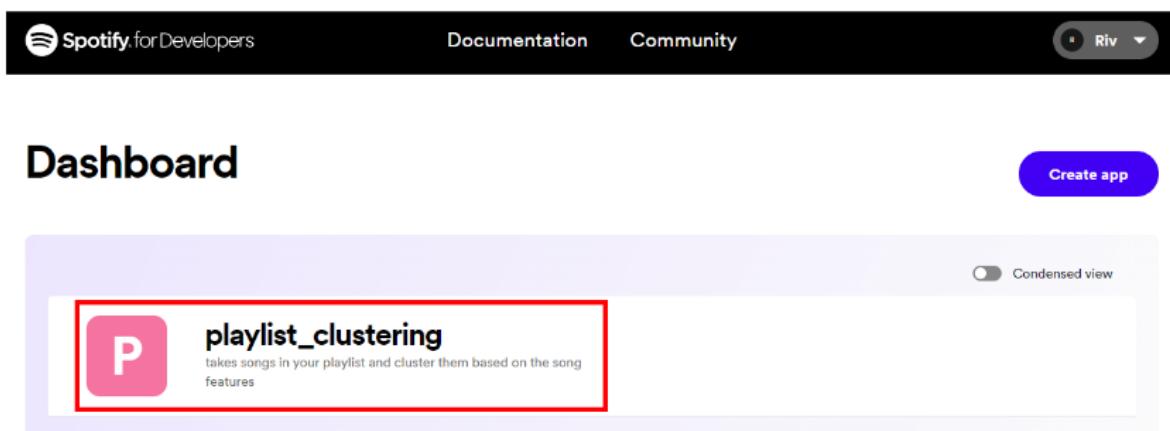


Figure 15.10: Gambar11. Daftar App di Spotify Developer

Pergi ke settings, cari ClientID dan ClientSecret

Buat Repositori di Github

Github adalah aplikasi untuk mengontrol versioning dari koding kita. Github akan melacak perubahan yang kita buat di dalam file koding, sehingga kita dapat membandingkan atau kembali ke versi sebelumnya. Dalam project ini, Github juga digunakan untuk proses deployment di Streamlit.

Login ke Github

Pergi ke Github.com dan login atau daftar akun baru. Jika anda membuat akun baru, jangan lupa verifikasi akun setelah dibuat. Jika anda sudah login, maka dashboard akun anda akan tampil seperti berikut:

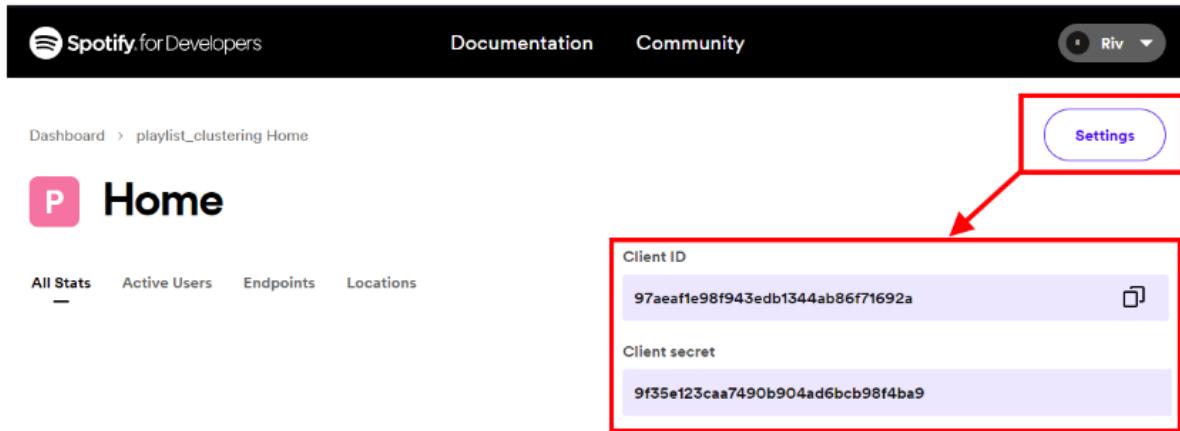


Figure 15.11: Gambar12. Client ID dan Client Secret dari App

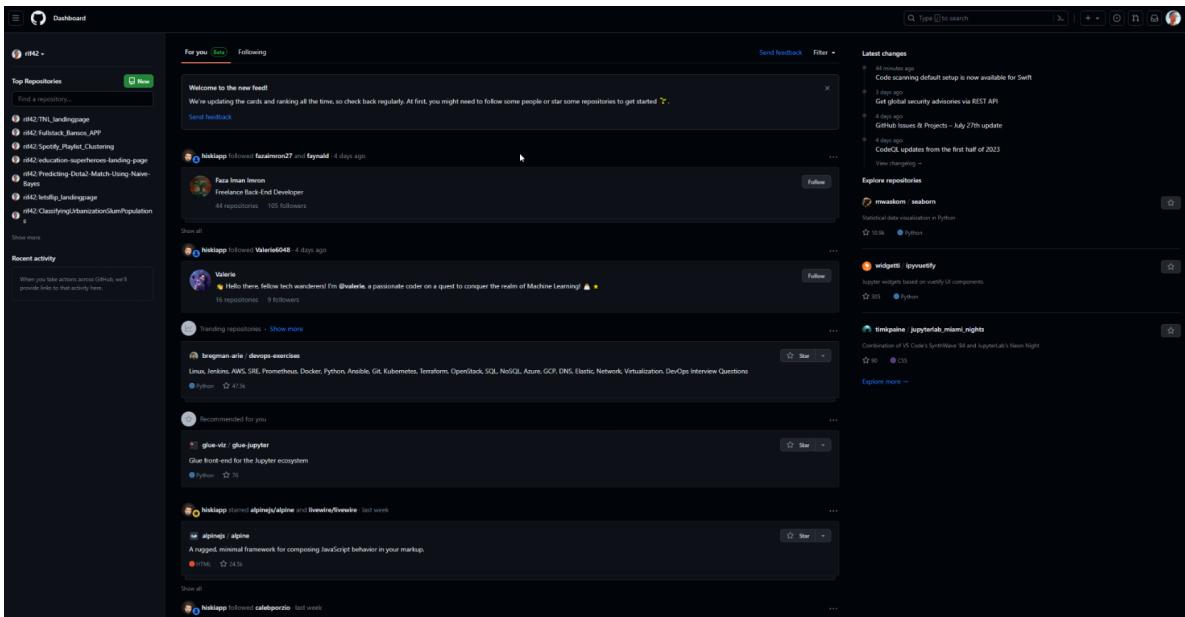


Figure 15.12: Gambar13. Dashboard Github

Buat Repository Baru

Di side tab sebelah kiri dashboard anda klik tombol new untuk membuat sebuah repository baru. Repository adalah sebuah tempat untuk menyimpan code-code anda.

Isi kolom seperti contoh, nama dan deskripsi bebas. Pastikan bahwa repository bersifat public. Lalu klik create repository

Simpan Alamat Git (git remote)

Setelah membuat repositori, halaman ini akan muncul. Copy alamat git (<https://github.com/rif42/githubtest0.git>) dan simpan data ini

Open Folder di Code Editor

Buka code editor anda (contoh VScode atau IntelliJ), buka terminal, lalu open folder dan pilih sebuah folder.

Buat File Baru

Buat file baru, dan namai file nya spotify_clustering.ipynb. File ini adalah sebuah notebook, mirip seperti jupyter notebook. Format notebook sangat bagus untuk eksperimen coding. Setelah file dibuat, tidak perlu diisi apapun.

Push File ke Repository

Buka terminal dengan cara menekan (ctrl + shift + ') atau membuka terminal > new terminal di top bar vscode. Lalu ketik command berikut:

- `git add .`, command ini berfungsi untuk menyimpan semua perubahan dalam file
- `git commit -m "first"`, command ini berfungsi untuk menyimpan commit, sebuah langkah terakhir untuk menyimpan semua perubahan yang ada di dalam repository. Teks yang ada di dalam tanda petik adalah message atau deskripsi dari commit.
- `git remote add origin [url]`, URL disini adalah alamat git yang didapat di langkah 4c (<https://github.com/rif42/githubtest0.git>)
- `git push origin master`, mengirim semua commit ke repository secara final

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ()*

Owner * **Repository name ***

 rif42 /
 githubtest0 is available.

Great repository names are short and memorable. Need inspiration? How about [musical-fiesta](#) ?

Description (optional)
just a test

 **Public**
Anyone on the internet can see this repository. You choose who can commit.

 **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

Figure 15.13: Gambar14. Github Repo

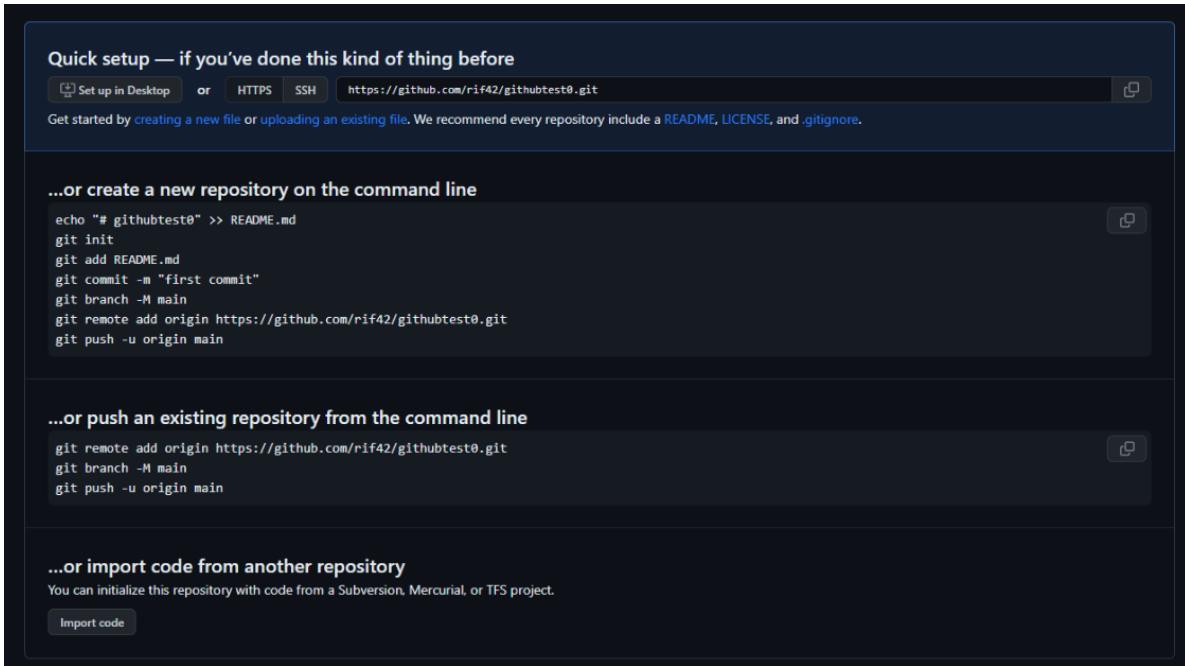


Figure 15.14: Gambar15. Github Remote

Pengambilan Dataset

Dataset dari aplikasi yang kita gunakan adalah lagu-lagu yang ada didalam playlist dari user. Untuk mengambil lagu tersebut, kita perlu membuat sebuah script untuk berkomunikasi dengan spotify web API dan mengambil data yang kita butuhkan.

Buka File Notebook

Buka code editor anda dan temukan file notebook yang telah dibuat di langkah sebelumnya. Pastikan bahwa git repository telah aktif dengan menulis `git status` di terminal.

Import Library yang Dibutuhkan

Jika library belum terinstall, maka jalankan command ini di terminal, lalu import

- pip install streamlit
- pip install pandas
- pip install numpy
- pip install scikit-learn
- pip install plotly-express

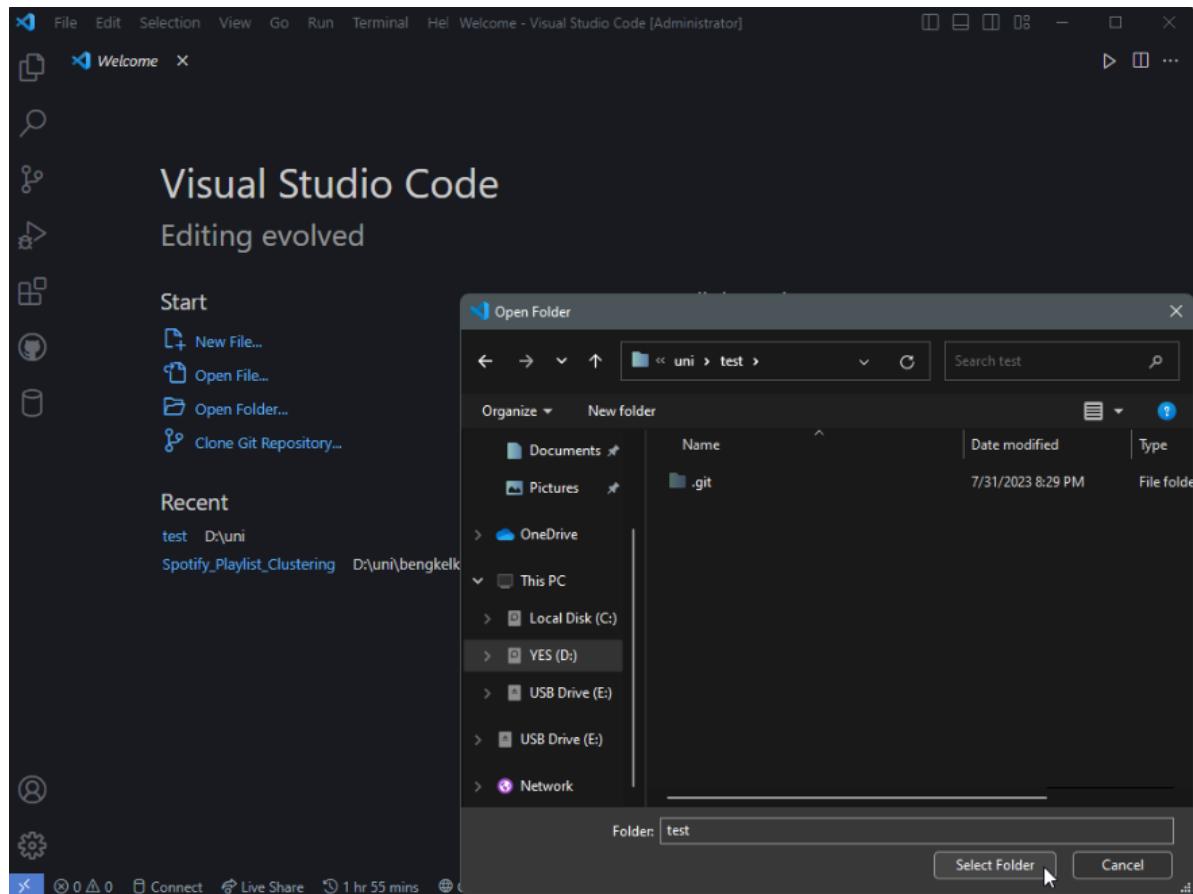


Figure 15.15: Gambar16. Open Folder di VSCode

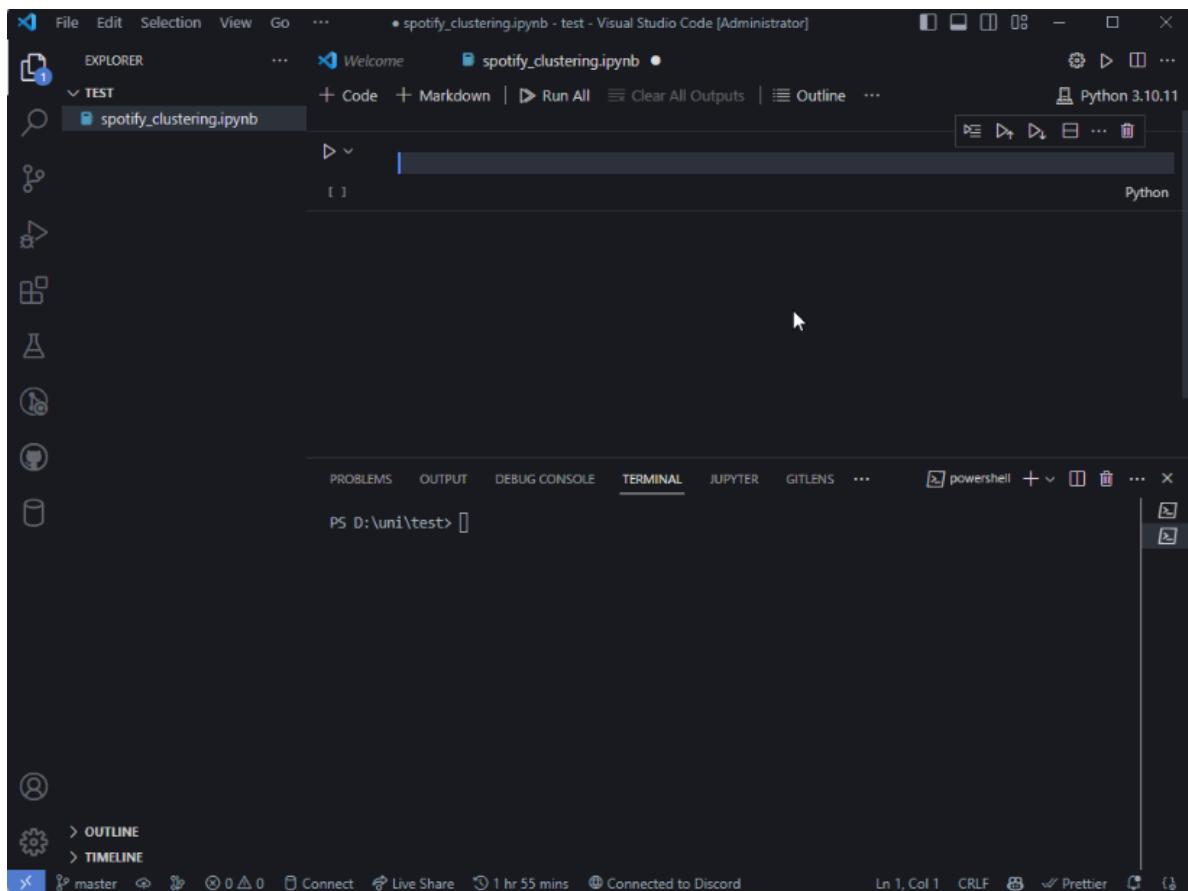


Figure 15.16: Gambar17. Buat File Baru

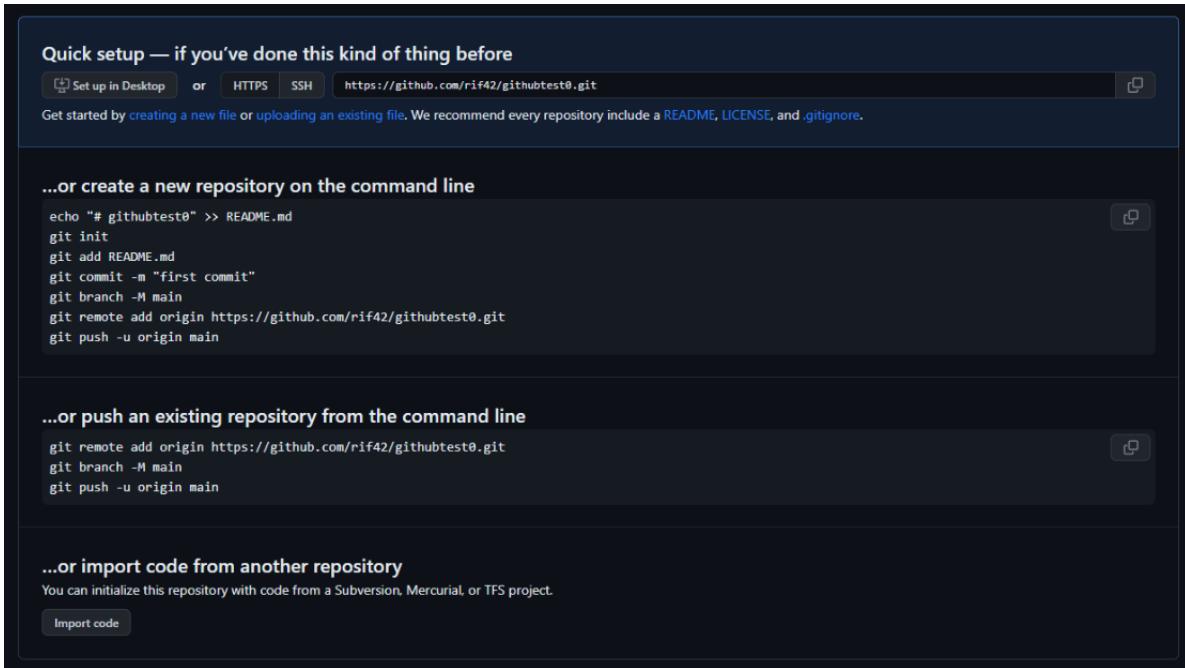


Figure 15.17: Gambar18. Buka file notebook (.ipynb)

- pip install nbformat
- pip install ipykernel

Library yang tidak disebutkan diatas adalah library built in (seperti base64, json, csv), kita tidak perlu menginstall nya, kita hanya perlu mengimportnya saja

```
import streamlit as st
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import plotly.express as px
import base64
from requests import post, get
import json
import csv
from sklearn import preprocessing
```

Masukkan variable client_id, client_secret, dan playlist_url

```
client_id = '97aeaf1e98f943edb1344ab86f71692a' ##ganti variabel dengan client_id milik anda
client_secret = '9f35e123caa7490b904ad6bcb98f4ba9' ##ganti variabel dengan client_secret milik anda
playlistId = '1dtCMTYzAOzwKXqkIxPJNS'

## 37i9dQZF1DXbrUpGvoi3TS - 1(similar sad songs)
## 1dtCMTYzAOzwKXqkIxPJNS - 2(old songs, rock, rap)
## OIN7IWKmIfwlEysGyWUuRg - 3(mix of modern electronic, pop, and rock)

dataset = []
dataset2 = []
dataset3 = []
```

Ganti isi variable dengan client id dan client secret yang diperoleh dari akun spotify developer yang kita bahas di slide 14.

Anda dapat menggunakan contoh URL playlist atau masukkan URL playlist anda sendiri. Setelah itu, buat 3 variable kosong untuk menampung hasil pengolahan data kita di slide selanjutnya.

Buat fungsi getToken()

Kita akan menggunakan key Client ID dan Client Secret untuk mendapatkan akses token. Token ini berfungsi untuk memperbolehkan seseorang mengakses dan menggunakan spotify web API Pertama kita gabungkan string Client ID dan Client Secret, lalu kita encode key tersebut menggunakan algoritma enkripsi base64, dan kita kirim key tersebut ke server spotify web API. Panggil fungsi tersebut untuk mengecek apakah token bisa terambil atau tidak.

```
def getToken():
    # gabungkan client_id dan client_secret
    auth_string = client_id + ':' + client_secret

    # encode ke base64
    auth_b64 = base64.b64encode(auth_string.encode('utf-8'))

    # url untuk mengambil token
    url = 'https://accounts.spotify.com/api/token'

    # header untuk mengambil token - sesuai dengan guide dari spotify
    headers = {
        'Authorization': 'Basic ' + auth_b64.decode('utf-8'),
```

```

    'Content-Type': 'application/x-www-form-urlencoded'
}

# data untuk mengambil token - sesuai dengan guide dari spotify
data = {'grant_type': 'client_credentials'}

# kirim request POST ke spotify
result = post(url, headers=headers, data=data)

# parse response ke json
json_result = json.loads(result.content)
token = json_result['access_token']

# ambil token untuk akses API
return token

## panggil fungsi getToken() dibawah ini

```

Buat fungsi getAuthHeader()

Fungsi ini berguna untuk mengambil token dan memasukkan token ke sebuah objek header request. Header request adalah sebuah metode identifikasi dan otorisasi di dalam API. Bearer berarti kita adalah sebuah klien yang meminta data. Di dalam objek header request, kita menyematkan token yang kita dapatkan untuk menandakan bahwa kita sudah mempunyai izin untuk mengakses API.

Fungsi ini akan dipanggil nanti ketika kita akan me-request data dari API

```

## pengambilan token untuk otorisasi API
def getAuthHeader(token):
    return {'Authorization': 'Bearer ' + token}

```

Buat fungsi getAudioFeatures()

Fungsi ini berguna untuk mengambil data karakteristik lagu. Fungsi ini mengambil token dan ID track (sebuah lagu).

Token didapatkan dari pemanggilan fungsi getToken() dan getAuthHeader(). ID track didapatkan dari list lagu yang diambil oleh fungsi getPlaylistItems(). Data karakteristik lagu yang dihasilkan oleh fungsi ini akan disimpan di variabel dataset2.

```

## pengambilan audio features dari track (lagu)
def getAudioFeatures(token, trackId):
    # endpoint untuk akses playlist
    url = f'https://api.spotify.com/v1/audio-features/{trackId}'
    # ambil token untuk otorisasi, gunakan sebagai header
    headers = getAuthHeader(token)
    result = get(url, headers=headers) # kirim request GET ke spotify
    json_result = json.loads(result.content) # parse response ke json

    # ambil data yang diperlukan dari response
    audio_features_temp = [
        json_result['danceability'],
        json_result['energy'],
        json_result['key'],
        json_result['loudness'],
        json_result['mode'],
        json_result['speechiness'],
        json_result['acousticness'],
        json_result['instrumentalness'],
        json_result['liveness'],
        json_result['valence'],
        json_result['tempo'],
    ]
    dataset2.append(audio_features_temp)

```

Buat fungsi getPlaylistItems()

Fungsi ini berguna untuk mengambil lagu-lagu yang ada di playlist. Fungsi ini mengambil token dan playlistID sebagai parameter nya. Token didapatkan dari pemanggilan fungsi getToken() dan getAuthHeader(). playlistID adalah variabel yang berisi URL playlist spotify yang nantinya diisi oleh user. Untuk mengambil data dari spotify web API, kita harus meng-input URL yang benar, disertai parameter (limit, market, fields) yang dibutuhkan. Semua variabel, ditambah header akan digabungkan dan membuat request ke web API

```

def getPlaylistItems(token, playlistId):
    # endpoint untuk akses playlist
    url = f'https://api.spotify.com/v1/playlists/{playlistId}/tracks'
    limit = '&limit=100' # batas maksimal track yang diambil
    market = '?market=ID' # negara yang tempat aplikasi diakses
    # format data dari track yang diambil
    fields = '&fields=items%28track%28id%2Cname%2Cartists%2Cpopularity%2Cduration_ms%2C+albu'

```

```

url = url+market+fields+limit # gabungkan semua parameter
# ambil token untuk otorisasi, gunakan sebagai header
headers = getAuthHeader(token)
result = get(url, headers=headers) # kirim request GET ke spotify
json_result = json.loads(result.content) # parse response ke json
# print(json_result)

```

Masih di fungsi yang sama, Hasil request kita yang disebut dengan response, akan ditampung di variabel json_result. Namun kita hanya mengambil beberapa fitur saja. Selanjutnya fitur-fitur tersebut kita masukkan ke variabel dataset.

```

# ambil data yang diperlukan dari response
for i in range(len(json_result['items'])):
    playlist_items_temp = []
    playlist_items_temp.append(json_result['items'][i]['track']['id'])
    playlist_items_temp.append(
        json_result['items'][i]['track']['name'].encode('utf-8'))
    playlist_items_temp.append(
        json_result['items'][i]['track']['artists'][0]['name'].encode('utf-8'))
    playlist_items_temp.append(
        json_result['items'][i]['track']['popularity'])
    playlist_items_temp.append(
        json_result['items'][i]['track']['duration_ms'])
    playlist_items_temp.append(
        int(json_result['items'][i]['track']['album']['release_date'][0:4]))
    dataset.append(playlist_items_temp)

```

Variabel dataset tadi berisikan lagu-lagu yang ada di dalam sebuah playlist. Sekarang kita akan ambil karakteristik lagu-lagu tersebut. Untuk mengambil lagu menggunakan fungsi getAudioFeatures() kita membutuhkan track ID dan token. Jadi, kita akan membuat sebuah for loop di dalam dataset, mengambil trackID nya saja (menggunakan array index 0), lalu kita panggil fungsi getAudioFeatures() dan sematkan trackID dan token sebagai parameternya.

```

for i in range(len(dataset)):
    getAudioFeatures(token, dataset[i][0])

```

Hasil dari fungsi getPlaylistItems() disimpan di variabel dataset. Hasil dari fungsi getAudioFeatures() disimpan di variabel dataset2. Selanjutnya, kita akan menggabungkan isi dari kedua variabel kedalam variabel dataset3, lalu meng-export nya menjadi file .csv

```

# gabungkan dataset dan dataset2
for i in range(len(dataset)):
    dataset3.append(dataset[i]+dataset2[i])

print(dataset3)
# convert dataset3 into csv
with open('dataset.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["id", "name", "artist", "popularity", "duration_ms", "year", "danceability",
                    "speechiness", "acousticness", "instrumentalness", "liveness", "valence"])
    writer.writerows(dataset3)

```

Untuk langkah terakhir, tulis coding dibawah untuk menjalankan semua fungsi dan menghasilkan dataset.csv

```
token = getToken() print('access token : '+token) getPlaylistItems(token, playlistId)
```

Jika kita gabungkan semua coding, maka coding akan menjadi seperti yang dibawah :

```

# pengambilan track (lagu) dari playlist
def getPlaylistItems(token, playlistId):
    # endpoint untuk akses playlist
    url = f'https://api.spotify.com/v1/playlists/{playlistId}/tracks'
    limit = '&limit=100' # batas maksimal track yang diambil
    market = '?market=ID' # negara yang tempat aplikasi diakses
    # format data dari track yang diambil
    fields = '&fields=items%28track%28id%2Cname%2Cartists%2Cpopularity%2C+duration_ms%2C+acousticness%2C+instrumentalness%2C+liveness%2C+valence%29%29'
    url = url+market+fields+limit # gabungkan semua parameter
    # ambil token untuk otorisasi, gunakan sebagai header
    headers = getAuthHeader(token)
    result = get(url, headers=headers) # kirim request GET ke spotify
    json_result = json.loads(result.content) # parse response ke json
    # print(json_result)

    # ambil data yang diperlukan dari response
    for i in range(len(json_result['items'])):
        playlist_items_temp = []
        playlist_items_temp.append(json_result['items'][i]['track']['id'])
        playlist_items_temp.append(
            json_result['items'][i]['track']['name'].encode('utf-8'))
        playlist_items_temp.append(
            json_result['items'][i]['track']['artists'][0]['name'].encode('utf-8'))

```

```

playlist_items_temp.append(
    json_result['items'][i]['track']['popularity'])
playlist_items_temp.append(
    json_result['items'][i]['track']['duration_ms'])
playlist_items_temp.append(
    int(json_result['items'][i]['track']['album']['release_date'][0:4]))
dataset.append(playlist_items_temp)

# ambil audio features dari semua track di dalam playlist
for i in range(len(dataset)):
    getAudioFeatures(token, dataset[i][0])

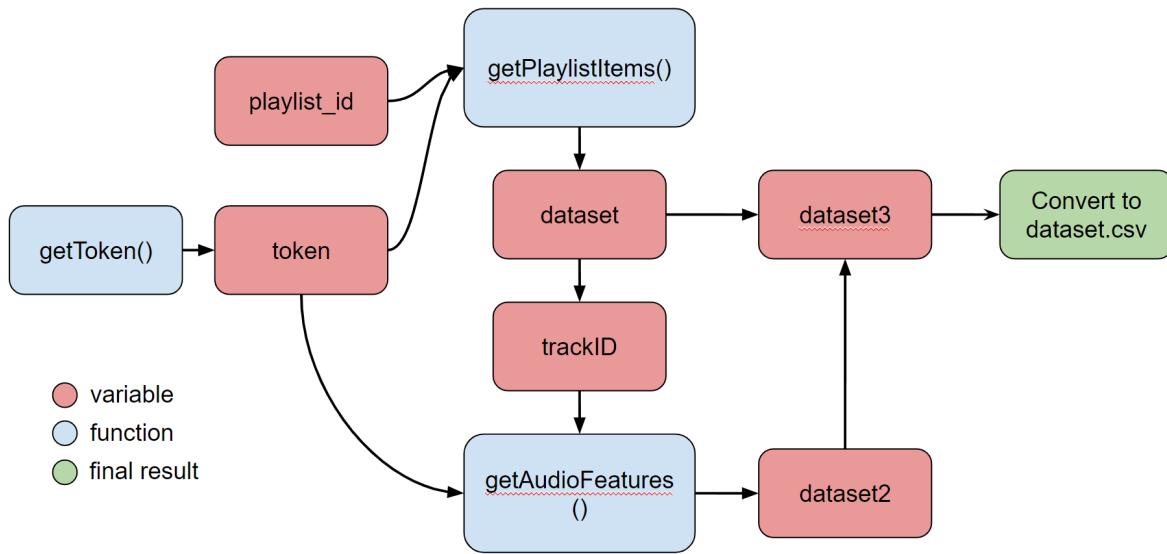
# gabungkan dataset dan dataset2
for i in range(len(dataset)):
    dataset3.append(dataset[i]+dataset2[i])

print(dataset3)
# convert dataset3 into csv
with open('dataset.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["id", "name", "artist", "popularity", "duration_ms", "year", "danceability",
                    "speechiness", "acousticness", "instrumentalness", "liveness", "valence"])
    writer.writerows(dataset3)

token = getToken()
print('access token : '+token)
getPlaylistItems(token, playlistId)

```

Jika anda masih bingung, bisa melihat gambar function flow dibawah ini :



Data Preprocessing

Spotify web API terkenal dengan format dan prosedur API yang sangat tangguh, sehingga bug atau kesalahan dataset jarang ditemui. Namun kita tetap harus mengecek dataset kita.

Data Understanding

Sebagai seorang data scientist, kita perlu memahami dataset kita secara sepenuhnya. Berikut adalah penjelasan singkat dari beberapa fitur di dalam dataset yang akan kita gunakan. Untuk deskripsi sepenuhnya, bisa mengunjungi <https://developer.spotify.com/documentation/web-api/reference/get-several-audio-features>.

- Acousticness, Tingkat intensitas akustik di dalam lagu tersebut. 0.0 = Lagu tersebut tidak akustik, 1.0 = Lagu tersebut sangat akustik.
- Loudness, Kekuatan suara keseluruhan dari sebuah lagu dalam desibel (dB). Nilai kekuatan suara dihitung rata-rata sepanjang seluruh lagu dan bermanfaat untuk membandingkan kekuatan suara relatif antara lagu-lagu. Kekuatan suara adalah kualitas dari suara yang merupakan korelasi psikologis utama dari kekuatan fisik (amplitudo). Nilai-nilai biasanya berkisar antara -60 hingga 0 dB.
- Tempo, Rata rata kecepatan atau tempo dari sebuah lagu, diukur menggunakan beats (ketukan) per minute (BPM).

Muat Data

Ketika coding pengambilan dataset dijalankan, sebuah file csv bernama dataset.csv akan muncul. File ini berisi dataset yang akan kita olah. Muat dataset menggunakan pandas dataframe dan check apakah dataset sudah benar.

```
import pandas as pd
import numpy as np

from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

import plotly.express as px

## muat dataset
data = pd.read_csv('dataset.csv')
data.head()
```

Decode - Encode ke base64

Dataset harus di encode-decode agar semua karakter ter-standarisasi dan file bisa dibaca dan diproses, bebas dari korupsi data. Karena data yang diambil dari spotify web API di encode menggunakan bas64, maka kita akan men-decode data tersebut menggunakan algoritma yang sama.

Namun ketika di decode, muncul efek samping seperti penambahan ('b) di depan data dan (') dibelakang data.

Kita akan menghapus efek samping tersebut menggunakan teknik string slicing.

```
## Hapus karakter yang tidak perlu pada kolom artist dan name
data['artist'] = data['artist'].map(lambda x: str(x)[2:-1])
data['name'] = data['name'].map(lambda x: str(x)[2:-1])

data.head()
```

Cek Nilai Kosong

Sebagai efek samping dari encode-decode, terkadang kolom nama kosong. Maka kita harus menemukan data tersebut dan menghapusnya, agar algoritma dapat memproses data yang benar.

```

##delete empty string in name column
data = data[data['name'] != '']

##reset index
data = data.reset_index(drop=True)
data.head()

```

Feature Selection

Tujuan utama kita adalah untuk melakukan clustering terhadap karakteristik lagu. Dalam dataset, ada beberapa fitur yang tidak mencerminkan karakteristik dari sebuah lagu, maka kita harus menghapus fitur-fitur tersebut. anda bisa melihat deskripsi dari fitur tersebut di section data understanding di slide sebelumnya.

```

## drop name artist and year column
data2 = data.copy()
data2 = data2.drop(['artist', 'name', 'year', 'popularity', 'key','duration_ms', 'mode', 'tempo'])

data2.head()

```

Normalisasi MinMax

beberapa fitur mempunyai batasan nilai yang berbeda-beda dan jaraknya cukup jauh. sebagai contoh, di fitur loudness, batasan nilai adalah -60 sampai -3, sedangkan di variabel instrumentalness, batasan nilai adalah 0 sampai 3.490000e-01. Untuk data min max selengkapnya bisa dicek di code dibawah.

nilai spread yang cukup besar ini dapat mempengaruhi hasil akhir clusterisasi, maka kita harus melakukan proses standarisasi agar semua fitur mempunyai batasan yang sama, yaitu 0 sampai 1

```

from sklearn import preprocessing

## normalize all data to 0 and 1
x = data2.values ##returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
data2 = pd.DataFrame(x_scaled)

## convert to dataframe

```

```
data2.columns = ['acousticness', 'danceability', 'energy', 'instrumentalness', 'loudness', 'li  
## data2.head()  
data2.describe()
```

Dimensionality Reduction using Principal Component Analysis (PCA)

Setelah kita menghapus fitur yang tidak terpakai, kita masih mempunyai 9 fitur. Kita tidak dapat melakukan proses clustering dengan 9 fitur, karena proses clustering hanya bisa dilakukan dengan maksimal 2 fitur. Oleh karena itu diperlukan metode dimensionality reduction dengan Principal Component Analysis untuk mengkondensasi 9 fitur menjadi 2 fitur. Nilai intrinsik yang ada di 9 fitur tidak akan hilang, nilai tersebut akan direpresentasikan ke bentuk yang lebih kecil, menjadi 2 fitur.

```
## Lakukan PCA untuk mengurangi jumlah fitur menjadi 2 fitur saja  
pca = PCA(n_components=2)  
pca.fit(data2)  
pca_data = pca.transform(data2)  
  
## buat dataframe dari hasil pca  
pca_df = pd.DataFrame(data=pca_data, columns=['x', 'y'])  
pca_df.head()  
  
## plot pca_df using plotly  
fig = px.scatter(pca_df, x='x', y='y', title='PCA')  
fig.show()
```

Clustering Menggunakan K-Means

Clustering adalah proses pengelompokan data atau objek-objek serupa menjadi kelompok-kelompok yang lebih homogen berdasarkan kesamaan fitur atau karakteristik tertentu. Algoritma k-means adalah metode clustering yang mengelompokkan data menjadi beberapa kelompok berdasarkan jaraknya ke pusat kelompok yang ditentukan secara iteratif.

Dataset akan kita ubah bentuknya menjadi list menggunakan fungsi zip. Lalu kita aplikasikan algoritma K-Means menggunakan parameter n-init=10 dan max_iter=1000. N-init berfungsi untuk menambah konsistensi hasil dan max_iter=1000 untuk menambah akurasi clustering.

Lalu kita buat graf scatter plot menggunakan library plotly-express

```

## rubah bentuk data ke list
data2 = list(zip(pca_df['x'], pca_df['y']))

## fit kmeans model
kmeans = KMeans(n_init=10, max_iter=1000).fit(data2)

## make scatter plot using plotly
fig = px.scatter(pca_df, x='x', y='y', color=kmeans.labels_, color_continuous_scale='rainbow')
fig.show()

```

Deploy ke Streamlit

Streamlit adalah framework open-source untuk mengembangkan aplikasi web interaktif dengan menggunakan bahasa pemrograman Python.

Tujuannya adalah menyederhanakan proses pembuatan aplikasi web dengan memungkinkan pengembang untuk membuat aplikasi dengan mudah menggunakan kode Python yang sederhana dan familiar.

Buat file bernama streamlit_app.py

Langkah pertama adalah membuat sebuah file baru bernama streamlit_app.py. Kita akan masukkan semua koding yang telah kita buat sebelumnya ke dalam satu file ini.

Kita telah membahas proses pengambilan dataset, dan telah dijelaskan bahwa kita harus membuat beberapa fungsi seperti getToken(), getPlaylistItems(). Masukkan semua fungsi-fungsi tersebut ke dalam file streamlit_app.py.

Selain itu, pada bab data preprocessing, terdapat banyak potongan code untuk pemrosesan data. Masukkan semua potongan code tersebut ke dalam sebuah function bernama dataProcessing().

Akhirnya, berikut fungsi-fungsi yang ada di dalam streamlit_app.py:

- **getToken()**, untuk mengambil token
- **getAuthHeader()**, untuk mengisi header request dengan token yang didapat
- **getAudioFeatures()**, untuk mendapatkan data karakteristik lagu
- **getPlaylistItems()**, untuk mengambil lagu-lagu yang ada di dalam playlist
- **dataProcessing()**, untuk memproses dataset dan menghasilkan clustering

Tambahkan fungsi dataProcessing()

Tambahkan `dataProcessing()` (pemanggilan function `dataProcessing()`) di bagian paling akhir function `getAudioFeatures()`. Gunanya untuk melanjutkan pemrosesan data secara otomatis.

Selain itu, fungsi ini berisi beberapa fungsi-fungsi khusus streamlit yang berguna untuk menampilkan grafik dan hasil dari coding.

Berikut beberapa contoh fungsi-fungsi streamlit:

- `st.write()`, untuk menulis teks atau paragraph. Masukkan teks ke dalam parameter fungsi untuk menampilkannya. Anda dapat menggunakan sintaks markdown (seperti `#` untuk header, dan `**` untuk bold) untuk memanipulasi format teks.
- `st.plotly_chart()`, untuk menampilkan grafik menggunakan library plotly. Masukkan variable grafik (fig) ke parameter fungsi untuk meggunakannya.
- `st.text_input()`, untuk mendapatkan input yang dimasukkan ke text box oleh user. Masukkan teks ke dalam parameter fungsi untuk memberi label dari text box. Hasil input dari user akan dimasukkan ke sebuah variabel yang dipakai oleh fungsi.
- `st.button()`, untuk membuat sebuah tombol. Gunakan fungsi `if` untuk menjalankan sesuatu jika tombol ditekan. Masukkan teks ke dalam parameter fungsi untuk memberi label dari button.

Untuk coding dari fungsi `dataProcessing()` dan fungsi driver streamlit bisa di simak dibawah ini :

```
def dataProcessing():
    data = pd.read_csv('dataset.csv')
    data
    st.write("## Preprocessing Result") # streamlit widget

    data = data[['artist', 'name', 'year', 'popularity', 'key', 'mode', 'duration_ms', 'acousticness',
                'danceability', 'energy', 'instrumentalness', 'loudness', 'liveness', 'speechiness']]
    data = data.drop(['mode'], axis=1)
    data['artist'] = data['artist'].map(lambda x: str(x)[2:-1])
    data['name'] = data['name'].map(lambda x: str(x)[2:-1])
    st.write("### Data to be deleted:")
    data[data['name'] == '']
    data = data[data['name'] != '']

    st.write("## Normalization Result") # streamlit widget
    data2 = data.copy()
    data2 = data2.drop(['artist', 'name', 'year', 'popularity', 'key', 'duration_ms'], axis=1)
    x = data2.values
```

```

min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
data2 = pd.DataFrame(x_scaled)
data2.columns = ['acousticness', 'danceability', 'energy', 'instrumentalness',
                 'loudness', 'liveness', 'speechiness', 'tempo', 'valence']
data2

st.write("## Dimensionality Reduction with PCA") # streamlit widget
pca = PCA(n_components=2)
pca.fit(data2)
pca_data = pca.transform(data2)
pca_df = pd.DataFrame(data=pca_data, columns=['x', 'y'])
fig = px.scatter(pca_df, x='x', y='y', title='PCA')
st.plotly_chart(fig) # output plotly chart using streamlit

st.write("## Clustering with K-Means") # streamlit widget
data2 = list(zip(pca_df['x'], pca_df['y']))
kmeans = KMeans(n_init=10, max_iter=1000).fit(data2)
fig = px.scatter(pca_df, x='x', y='y', color=kmeans.labels_,
                  color_continuous_scale='rainbow', hover_data=[data.artist, data.name])
st.plotly_chart(fig) # output plotly chart using streamlit

st.write("Enjoy!")

# streamlit widgets
st.write("# Spotify Playlist Clustering")
client_id = st.text_input("Enter Client ID")
client_secret = st.text_input("Enter Client Secret")
playlistId = st.text_input("Enter Playlist ID")

# streamlit widgets
if st.button('Create Dataset!'):
    token = getToken()
    getPlaylistItems(token, playlistId)

```

Buat file requirements.txt

Fungsi dari file ini adalah untuk memberitahu streamlit package apa yang harus di install sebelum di deploy.

Isi file requirements.txt dengan

```
scikit-learn  
plotly-express
```

Setelah itu, save file.

Upload ke github repository

Jika semua coding sudah selesai, maka kita harus meng-upload semua coding kita ke github.

- login ke github
- di dashboard github, klik **New** untuk membuat repository baru
- beri nama repositori, deskripsi, dan **pastikan bahwa repository bersifat public**. Lalu klik create repository
- kembali ke code editor, masukkan file streamlit_app.py ke dalam folder
- ketik `git init`
- ketik `git add .`
- ketik `git remote add origin https://github.com/[username]/[repository_name].git`

Buat Akun Streamlit

- Pergi ke streamlit.io
- Buat akun/masuk menggunakan akun
- Sangat direkomendasikan masuk menggunakan akun github

Create new app, isi form

- **Repository**, Nama repositori dari github yang akan di upload ke streamlit. format [user github]/[nama repository], contoh : `rif42/spotifyclustering`
- **Branch**, Branch atau cabang dari repositori yang akan digunakan. Secara default akan diisi master

[← Back](#)

Deploy an app

Repository [Paste GitHub URL](#)

Branch

Main file path

App URL (Optional)

Domain is available

[Advanced settings...](#)

[Deploy!](#)

Figure 15.18: Gambar19. Form create new app di streamlit

- **Main File Path**, Letak file yang akan dijadikan host dari aplikasi streamlit. Pastikan file sudah masuk didalam directory dan repository. Pastikan juga nama dari file sudah benar, sesuai dengan yang ada di repository.
- **App URL**, domain atau URL yang akan digunakan untuk mengakses aplikasi yang sudah di deploy.

Klik create app dan tunggu sampai streamlit selesai melakukan proses deploy

Kesimpulan

Untuk project machine learning clustering ini, kita menggunakan dataset lagu yang diambil dari spotify.

Untuk melakukan proses pengambilan lagu, kita membutuhkan akses API dari spotify. Pertama kita harus mendaftar sebagai spotify developer di web resmi spotify. Lalu kita mengambil sebuah token menggunakan kredensial yang kita dapat dari akun developer kita. Sebuah token memungkinkan kita untuk mengambil data dari spotify web API secara langsung. user menginput url playlist dan aplikasi akan mengambil semua lagu didalam playlist tersebut dan membuat dataset secara otomatis.

Dataset yang dihasilkan sudah cukup solid karena spotify web API memang dikenal mempunyai kualitas tinggi. Namun kita tetap melakukan pengecekan konsistensi data. Ketika data diambil dari web, beberapa lagu/artist mempunyai karakter yang tidak dapat ditampilkan seperti huruf kanji, maka kita harus encode-decode karakter tersebut ke standar utf-8. Namun terkadang proses dari encode-decode menghasilkan sebuah string kosong. Walaupun fitur yang lain tidak kosong, data ini tidak bisa kita identifikasi, jadi kita akan hapus data ini.

Setelah di preproses, kita akan melakukan proses dimensionality reduction menggunakan principal component analysis (PCA). Proses clustering menggunakan scatter plot mengharuskan data mempunyai 2 fitur. Tujuan dari proses PCA adalah mengurangi jumlah fitur dari 9 menjadi 2 tanpa mengurangi makna yang ada didalam dataset.

Kita melakukan clustering dataset menggunakan algoritma K-Means dengan titik cluster yang diatur secara otomatis. Adapun parameter yang kita masukkan ke algoritma yaitu **n_init=10** untuk menambah konsistensi hasil dan **max_iter=1000** untuk menambah akurasi clustering.

Hasilnya dataset terbagi menjadi 7 buah cluster (secara otomatis), dan masing masing lagu di dalam cluster mempunyai keunikan tersendiri. Keunikan tersebut dapat direpresentasikan dengan tempo lagu, jenis melodi, jenis instrumen, overall ‘vibe’ dari lagu, dan lain lain. Jumlah cluster dapat diatur menggunakan parameter **n_cluster**.

Untuk melakukan proses deploy, pertama kita buat github repository dulu. Isi github repository dengan file python (file py, bukan ipynb) yang akan kita gunakan untuk proses deploy. File python ini berisi fungsi-2fung