



Kampus
Merdeka
INDONESIA JAYA

PYTHON PROGRAMMING FOR DATA SCIENCE

Deden Wahiddin, S.Kom., M.Kom.

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BUANA PERJUANGAN KARAWANG
2024

DAFTAR ISI

BAB I FUNDAMENTAL DATA SCIENCE	8
1.1 PENDAHULUAN	8
1.2 DATA SCIENCE.....	8
1.3 STRUKTUR	8
1.4 PYTHON	8
BAB II INSTALASI DAN PENGATURAN.....	9
2.1 ANCONDA.....	9
2.2 JUPYTER NOTEBOOK	11
2.3 PENGATURAN	11
BAB III LISTS AND DICTIONARIES	13
3.1 STRUKTUR	13
3.2 TUJUAN	13
3.3 LIST	13
3.4 OPERASI MANIPULASI DAFTAR YANG BERBEDA	14
3.5 PERBEDAAN LIST DAN TUPLE	15
3.6 KAMUS (DICTIONARY).....	16
3.6.1 MEMBUAT KAMUS	16
3.6.2 BEBERAPA OPERASI DENGAN KAMUS	16
3.7 KESIMPULAN.....	17
BAB IV PAKET, FUNGSI, dan PERULANGAN	18
4.1 STRUKTUR	18
4.2 TUJUAN	18
4.3 FUNGSI HELP()	18
4.4 IMPORT PAKET.....	18
4.5 MEMANGGIL FUNGSI	18
4.6 MELEWATI PARAMETER DALAM SUATU FUNGSI.....	19
4.7 PARAMETER YANG TIDAK DIKETAHUI DALAM SUATU FUNGSI	19
4.8 FUNGSI LAMDA	19
4.9 WHILE dan LOOP	19
4.10 KESIMPULAN.....	20
BAB V FUNDAMENTAL NUMPY	21
5.1 STRUKTUR	21
5.2 TUJUAN	21
5.3 IMPOR PAKET NUMPY.....	21
5.4 MENGGUNAKAN ARRAY NUMPY DARIPADA LIST	21
5.5 ATRIBUT ARRAY NUMPY.....	22
5.6 MEMBUAT ARRAY NUMPY.....	22
5.7 MENGAKSES ELEMEN DALAM ARRAY NUMPY	22
5.8 PEMOTONGAN (SLICING) DALAM ARRAY NUMPY	23
5.9 KESIMPULAN.....	24

BAB VI PANDAS DAN DATAFRAME	25
6.1 STRUKTUR	25
6.2 TUJUAN	25
6.3 IMPOR PANDAS	25
6.4 STRUKTUR DATA PANDAS	25
6.4.1 SERIES.....	25
6.4.2 DATAFRAME	26
6.5 FUNGSI DATAFRAME	28
6.6 MENANGANI DATA HILANG DI DATAFRAME	29
6.7 KESIMPULAN.....	30
BAB VII KOMUNIKASI BASIS DATA.....	31
7.1 STRUKTUR	31
7.2 TUJUAN	31
7.3 SQLAlchmy	31
7.3.1 INSTALASI	31
7.3.2 IMPOR PAKET SQLAlchemy	32
7.3.3 PENGUNAAN SQLAlchemy	32
7.3.4 SQLALCHEMY ENGINE CONFIGURATION	33
7.3.5 MEMBUAT TABEL DI DATABASE	33
7.3.6 INPUT DATA	34
7.3.7 UPDATE DATA	34
7.3.8 JOIN TWO TABLES	34
7.3.9 INNER JOIN	35
7.3.10 LEFT JOIN	36
7.3.11 RIGHT JOIN	36
7.4 KESIMPULAN.....	36
BAB VIII STATISTIK DALAM ILMU DATA	37
8.1 STRUKTUR	37
8.2 TUJUAN	37
8.3 STATISTIK DALAM ILMU DATA	37
8.4 JENIS DATA/VARIABEL STATISTIK	37
8.5 MEAN, MEDIAN, DAN MODUS.....	38
8.5.1 MEAN	38
8.5.2 MEDIAN	38
8.5.3 MODUS.....	38
8.6 DASAR PROBABILITAS	39
8.7 DISTRIBUSI STATISTIK	39
8.8 DISTRIBUSI POISSON	39
8.9 BINOMIAL DISTRIBUTION.....	39
8.10 NORMAL DISTRIBUTION	40
8.11 KOEFISIEN KORELASI PEARSON.....	40
8.12 FUNGSI DENSITAS PROBABILITAS	41
8.13 CONTOH.....	41

8.14	INFERENSI STATISTIK DAN PENGUJIAN HIPOTESIS	41
8.15	KESIMPULAN.....	44
BAB IX PYTHON	45
9.1	STRUKTUR	45
9.2	TUJUAN	45
9.3	IMPORTING TEXT DATA.....	45
9.4	IMPOR CSV	46
9.5	IMPOR EXCEL DATA.....	46
9.6	IMPOR JSON DATA	47
9.7	PICKLED DATA.....	47
9.8	DATA TERKOMPRESI.....	47
9.9	KESIMPULAN.....	47
BAB X PEMBERSIHAN DATA	48
10.1	STRUKTUR	48
10.2	TUJUAN	48
10.3	MENGENAL DATA.....	48
10.4	MENGHAPUS NILAI HILANG	49
10.5	MENGISI NILAI KOSONG	50
10.6	NORMALISASI DATA	50
10.7	PARSE DATES	51
10.8	PENGKODEAN KARAKTER	52
10.9	CLEANING INCONSISTENT DATA	53
10.10	KESIMPULAN	53
BAB XI VISUALISASI DATA	54
11.1	STRUKTUR	54
11.2	TUJUAN	54
11.3	BAR CHART.....	54
11.4	LINE CHART.....	55
11.5	HISTOGRAMS.....	55
11.6	SCATTER PLOT.....	55
11.7	STACKED PLOT	56
11.8	BOX PLOT	56
11.9	KESIMPULAN.....	57
BAB XII DATA PRE-PROCESSING	58
12.1	STRUKTUR	58
12.2	TUJUAN	58
12.3	STUDI KASUS.....	58
12.4	IMPORT LIBRARY	58
12.5	IMPORT DATASET	58
12.6	ANALISIS DATA	60
12.6.1	KORELASI	60
12.6.2	EKSPLORASI VARIABEL I	60
12.6.3	EKSPLORASI VARIABEL 2.....	61

12.6.4 EKSPLORASI VARIABEL 2,3,4.....	62
12.7 DATA PREPROCESSING.....	62
12.7.1 FEATURE ENGINEERING	62
12.7.2 OUTLIER DETECTION.....	63
12.8 KESIMPULAN.....	64
BAB XIII SUPERVISED MACHINE LEARNING	65
13.1 STRUKTUR	65
13.2 TUJUAN.....	65
13.3 DAFTAR ISTILAH.....	65
13.4 MACHINE LEARNING (ML).....	65
13.4.1 UNSUPERVISED LEARNING.....	65
13.4.2 SEMI-SUPERVISED LEARNING.....	66
13.4.3 REINFORCEMENT LEARNING	66
13.4.4 DAFTAR ALGORITMA	66
13.4.5 SUPERVISED LEARNING	66
13.4.6 SUPERVISED ML FUNDAMENTALS	66
13.4.7 KLASIFIKASI	66
13.4.8 MEMECAHKAN MASALAH KLASIFIKASI.....	67
13.4.9 CONTOH	67
13.4.10 TRAIN/TEST dan CROSS VALIDATION	69
13.4.11 REGRESSION ML	70
13.4.12 TUNE ML MODEL	73
13.4.13 VARIABEL KATEGORIKAL DALAM SKLEARN	74
13.4.14 MENGATASI DATA YANG HILANG.....	75
13.5 KESIMPULAN.....	76
BAB XIV UNSUPERVISED MACHINE LEARNING.....	77
14.1 STRUKTUR	77
14.2 TUJUAN	77
14.3 UNSUPERVISED LEARNING	77
14.4 TEKNIK UNSUPERVISED LEARNING	77
14.4.1 PENGELOMPOKAN (CLUSTERING):	77
14.4.2 K-mean clustering	78
14.4.3 t-SNE	80
14.4.4 PRINCIPAL COMPONENT ANALYSIS (PCA)	82
14.4.5 STUDI KASUS	83
14.4.6 VALIDATION OF UNSUPERVISED ML	87
14.5 KESIMPULAN.....	87
BAB XV TIME SERIES DATA	88
15.1 STRUKTUR	88
15.2 TUJUAN	88
15.3 URGENSI TIME-SERIES.....	88
15.4 MENGUBAH DATA TIME-SERIES.....	90
15.5 MANIPULASI DATA TIME-SERIES	92

15.6	MEMBANDINGKAN TINGKAT PERTUMBUHAN DERET WAKTU	93
15.7	MENGUBAH FREKUENSI TIME SERIES	95
15.8	KESIMPULAN.....	98
16.1	STRUKTUR	99
16.2	TUJUAN	99
16.3	PERAMALAN TIME SERIES.....	99
16.4	LANGKAH-LANGKAH PERAMALAN.....	99
16.5	TEKNIK-TEKNIK PERAMALAN TIME SERIES	100
16.5.1	AUTOREGRESSION (AR)	100
16.5.2	MOVING AVERAGE (MA)	100
16.5.3	AUTOREGRESSIVE MOVING AVERAGE (ARMA).....	100
16.5.4	AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA)	101
16.5.5	SEASONAL AUTOREGRESSIVE INTEGRATED MOVING-AVERAGE (SARIMA)	101
16.5.6	SEASONAL AUTOREGRESSIVE INTEGRATED MOVING-AVERAGE DENGAN REGRESOR EKSTERNAL (SARIMAX)	102
16.5.7	VECTOR AUTOREGRESSION MOVING-AVERAGE (VARMA).....	102
16.5.8	HOLT WINTER'S EXPONENTIAL SMOOTHING (HWES).....	102
16.6	PERAMALAN HALAMAN WEB	103
16.7	KESIMPULAN.....	107
BAB XVII STUDI KASUS 1	108	
17.1	PEMBAYARAN PINJAMAN	108
17.2	PROSES	108
17.2.1	PERSIAPAN DATA	108
17.2.2	PRE-PROCESSING	114
17.2.3	LOGISTIC REGRESION	114
17.2.4	RANDOM FOREST	115
17.2.5	EKSTRAKSI FITUR.....	116
17.3	KESIMPULAN.....	117
BAB XVIII STUDI KASUS 1	118	
18.1	MODEL PREDIKSI DENGAN MENGKLASIFIKASIKAN TEKS	118
18.2	PROSES	118
18.2.1	IMPOR DATA	118
18.2.2	ANALISA DATA	118
18.2.3	EKSTRAKSI FITUR.....	120
18.2.4	SPLIT DATA	120
18.2.5	TRAINING NAÏVE BAYES	120
18.2.6	EVALUASI NAÏVE BAYES.....	121
18.2.7	SUPPORT VECTOR MACHINE.....	121
18.2.8	EVALUASI SVM	122
18.2.9	KESIMPULAN	122
BAB XIX STUDI KASUS II	123	
19.1	BUILD A FILM RECOMMENDATION ENGINE.....	123

19.2 KESIMPULAN.....	127
BAB XX STUDI KASUS III	128
20.1 PPREDIKSI PENJUALAN RUMAH MENGGUNAKAN REGRESI.....	128
20.2 KESIMPULAN.....	132

BAB I

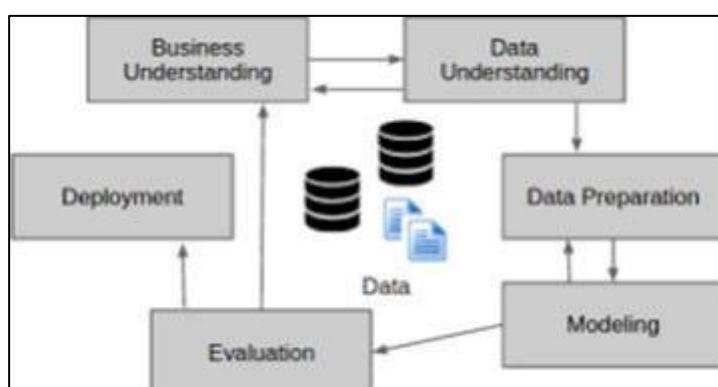
FUNDAMENTAL DATA SCIENCE

1.1 PENDAHULUAN

Saat ini, ilmu data ada di mana-mana. Pertumbuhan dunia digital yang eksplosif membutuhkan para profesional dengan tidak hanya keterampilan yang kuat, tetapi juga kemampuan beradaptasi dan semangat untuk tetap berada di garis depan teknologi. Sebuah studi terbaru menunjukkan bahwa permintaan akan data scientist dan analis diproyeksikan tumbuh sebesar 28 persen pada tahun 2021. Ini di atas kebutuhan pasar saat ini. Menurut Biro Statistik Tenaga Kerja AS, pertumbuhan keterampilan pekerjaan ilmu data akan tumbuh sekitar 28% hingga 2026. Kecuali ada sesuatu yang berubah, kesenjangan keterampilan ini akan terus melebar. Dalam bab pertama ini, Anda akan belajar bagaimana menjadi akrab dengan data, peran Anda sebagai calon data scientist, dan pentingnya bahasa pemrograman Python dalam ilmu data.

1.2 DATA SCIENCE

Ilmu data adalah bidang yang mengekstrak pengetahuan dari data menggunakan matematika, statistik, ilmu komputer, dan pemrograman. Seorang ilmuwan data memiliki keterampilan analitis dan berperan dalam mengungkap wawasan dari data. Ilmu data sangat penting dalam pengambilan keputusan berbasis data. Ilmuwan data mengajukan pertanyaan yang tepat untuk mengungkap wawasan tersembunyi dari data. Mereka bekerja di berbagai domain, mulai dari perawatan kesehatan hingga keuangan, dengan langkah-langkah seperti pengumpulan data, visualisasi, pemodelan, dan evaluasi model. Sebagian besar waktu seorang ilmuwan data dihabiskan untuk menemukan, membersihkan, dan mengorganisir data.



Gambar 1.1 Work cycle of a data scientist

1.3 STRUKTUR

Data dapat dibagi menjadi tiga kategori: terstruktur, tidak terstruktur, dan semi-terstruktur. Data terstruktur adalah data yang terorganisir dalam tabel, seperti data SQL. Data tidak terstruktur mencakup gambar, PDF, audio, dan data lainnya yang memerlukan alat khusus untuk akses. Data semi-terstruktur adalah data terstruktur yang tidak terorganisir, seperti file JSON atau CSV.

1.4 PHYTHON

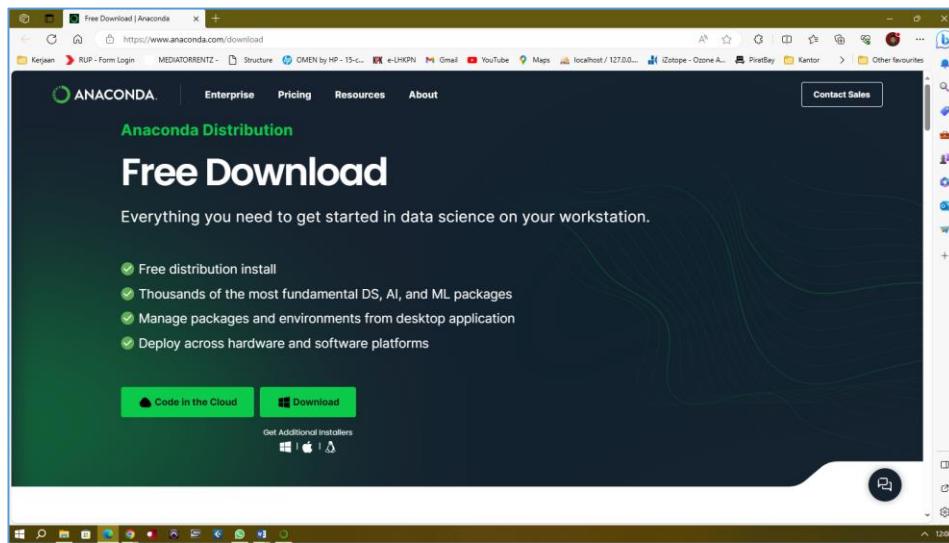
Python adalah bahasa pemrograman yang mendukung berbagai paradigma pemrograman dan sering digambarkan sebagai Swiss Army knife dalam dunia pemrograman. Dalam konteks ilmu data, Pandas adalah perpustakaan yang sangat penting. Python mudah dipelajari dan memiliki banyak perpustakaan analitis yang tersedia secara gratis. Hal ini membuatnya menjadi pilihan utama bagi ilmuwan data di berbagai sektor. Survei dari KDnuggets menunjukkan bahwa Python adalah pilihan yang lebih disukai dalam ilmu data dan pembelajaran mesin.

BAB II

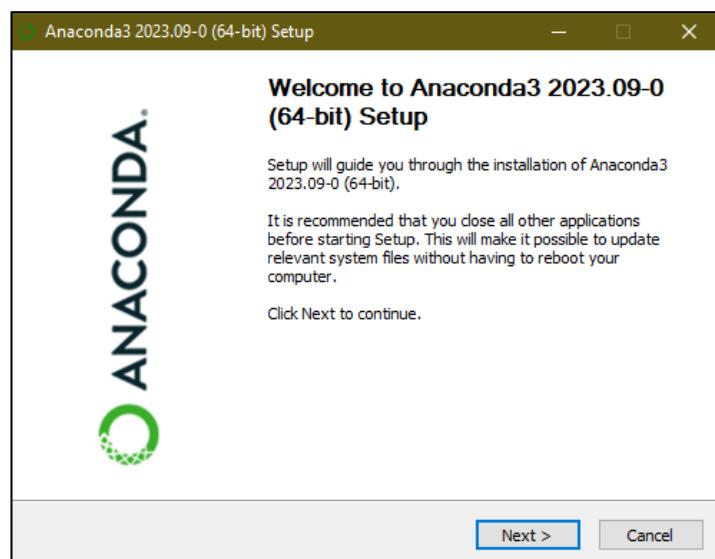
INSTALASI DAN PENGATURAN

2.1 ANCONDA

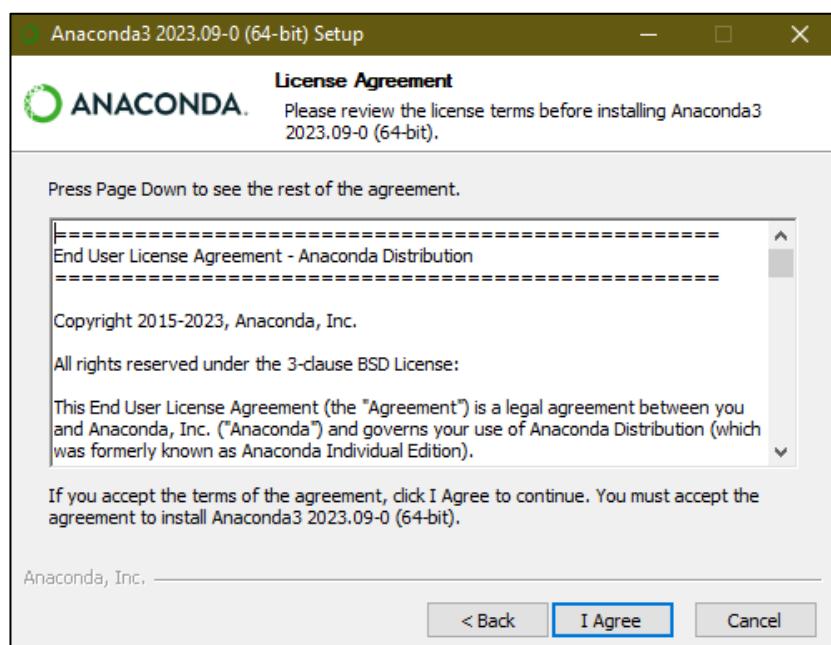
Praktikum ini menggunakan aplikasi *ANACONDA* sebagai terminal agar dapat menggunakan *Jupyter Notebook* sebagai *compiler python*, unduh pada link berikut : <https://www.anaconda.com/download>. Pilih Operating system yang digunakan, praktikum ini menggunakan *windows* sebagai sistem operasi.



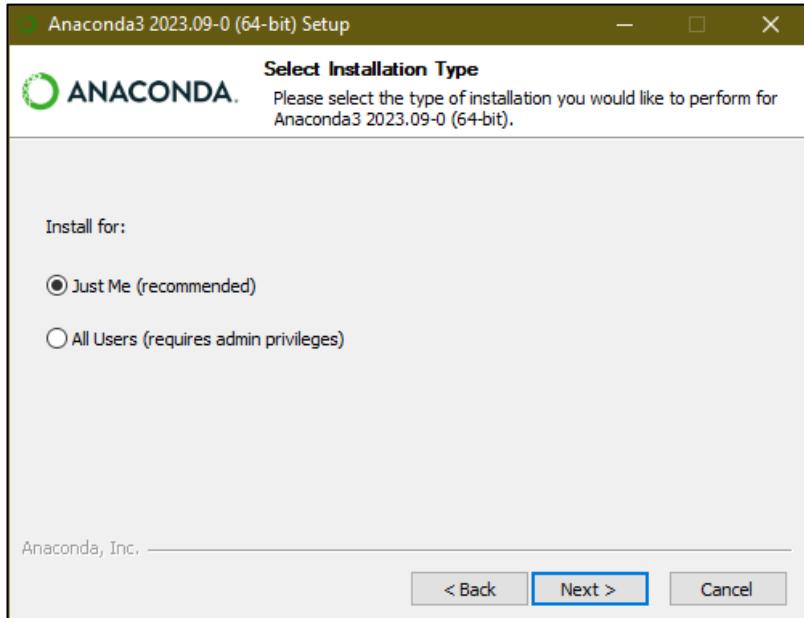
Instal hasil download, lalu pilih “*next*”, ikuti langkah berikutnya.



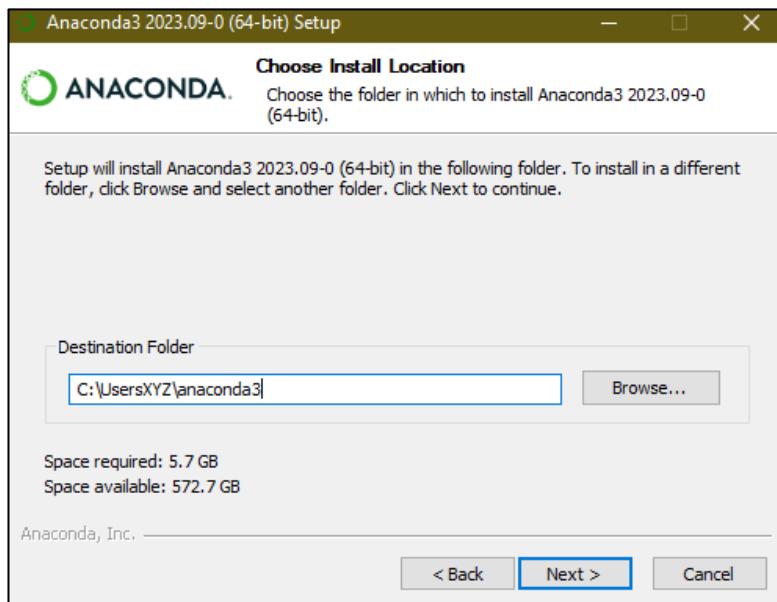
Pilih “*I Agree*” dan tunggu sampai proses instalasi selesai.



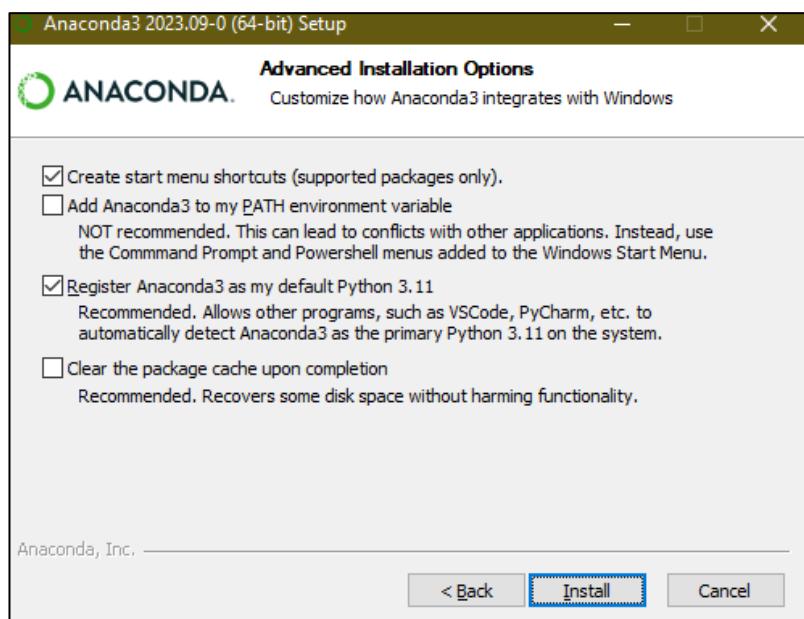
Pilih “*Just Me*” jika hanya ingin menginstal pada akun anda atau pilih “*All User*” jika ingin instalasi dapat digunakan oleh semua pengguna Komputer anda dan tekan “*next*”.



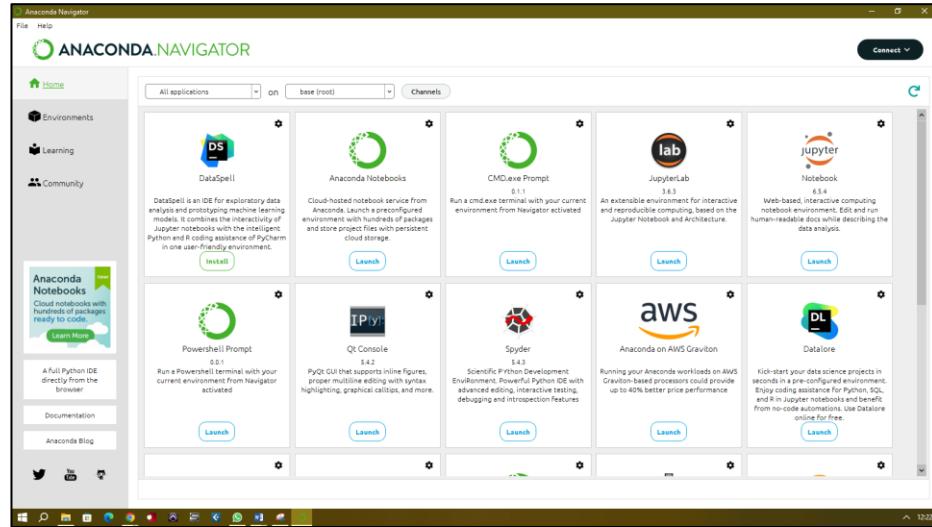
Pilih *directory* yang ingin digunakan atau tidak perlu mengubah *directory* jika ingin menggunakan *default directory* lalu tekan “*next*”.



Pengaturan ini tidak perlu dirubah, sesuaikan saja dengan rekomendasi yang diberikan oleh system lalu tekan “Instal”.

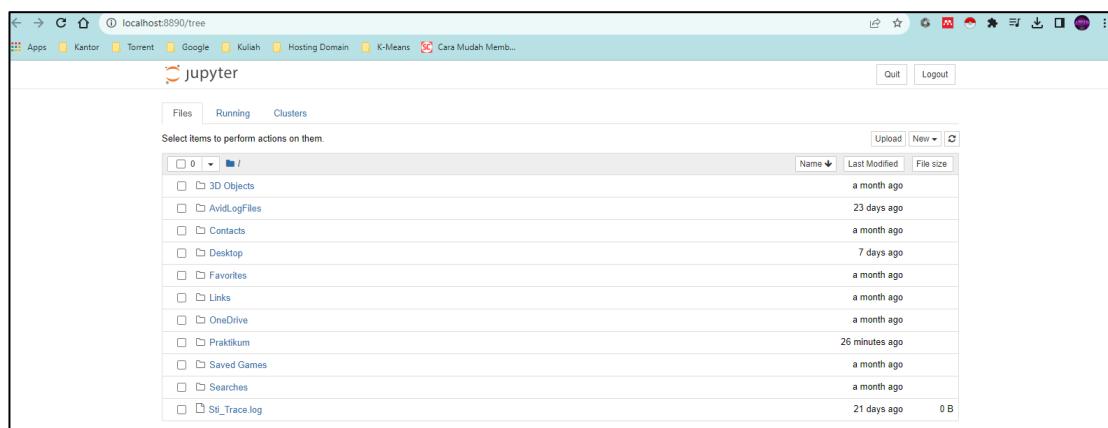


Berikut adalah tampilan *Anaconda* setelah selesai instalasi.



2.2 JUPYTER NOTEBOOK

Praktikum ini menggunakan *Jupyter Notebook* sebagai *compiler python*. Pilihlah *Jupyter Notebook* untuk memulai praktikum. Berikut adalah tampilan dari *Jupyter Notebook*:

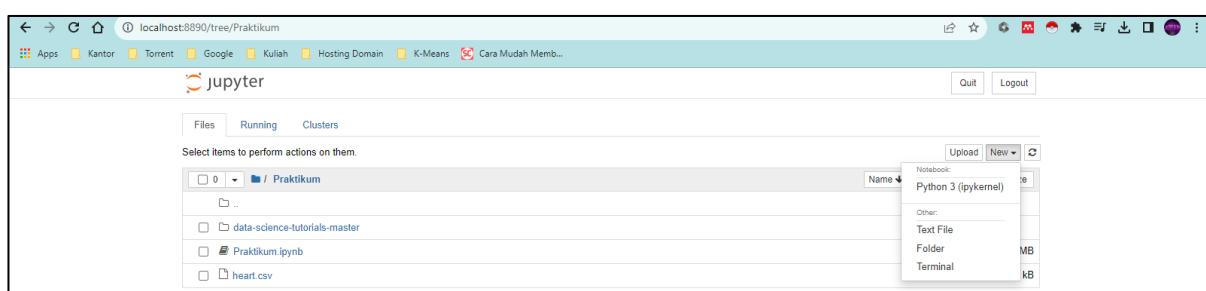


2.3 PENGATURAN

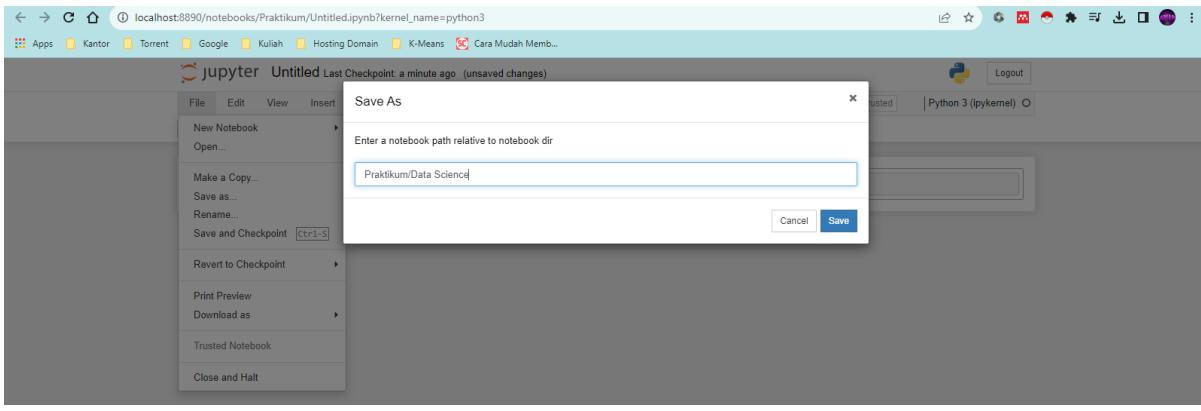
Buatlah Folder dengan nama “PRAKTIKUM” pada directory “C:\Users\NAMA DIREKTORY ANDA”.

Name	Date modified	Type
.anaconda	01/10/2023 12:01	File folder
.conda	01/10/2023 12:34	File folder
.continuum	01/10/2023 12:01	File folder
.ipython	01/10/2023 12:02	File folder
.jupyter	01/10/2023 12:02	File folder
.keras	19/09/2023 12:13	File folder
.QtWebEngineProcess	23/08/2023 14:20	File folder
.Waves Central	23/08/2023 14:20	File folder
3D Objects	23/08/2023 07:49	File folder
AvidLogFiles	08/09/2023 12:41	File folder
Contacts	23/08/2023 07:49	File folder
Desktop	24/09/2023 09:13	File folder
Favourites	23/08/2023 07:49	File folder
Links	23/08/2023 07:49	File folder
OneDrive	28/08/2023 08:36	File folder
Praktikum	01/10/2023 12:02	File folder

Pada halaman *Jupyter Notebook* pilih dan masuk kedalam folder PRAKTIKUM yang sudah dibuat. Lalu Tekan tombol “new” lalu pilih “python 3” yang berada pada pojok kanan atas.



Lalu, pilih file -> save as -> buatlah nama project “Data Science” dan tekan “save”.



Sampai disini kita sudah melakukan instalasi dan pengaturan untuk *compiler python*.

BAB III

LISTS AND DICTIONARIES

3.1 STRUKTUR

Struktur Data adalah cara mengatur dan menyimpan data dalam bahasa pemrograman agar dapat diakses dan dikerjakan dengan efisien. Mereka mendefinisikan hubungan antara data dan operasi yang dapat dilakukan pada data tersebut. Sebagai calon ilmuwan data, Anda akan menggunakan berbagai struktur data dalam pekerjaan sehari-hari, sehingga memahami jenis data adalah keterampilan yang sangat penting. Dalam bab ini, kita akan mempelajari dua struktur data Python yang paling umum digunakan dalam ilmu data ketika bekerja dengan data besar - yaitu list dan dictionary. Kami juga akan membandingkan keduanya dengan struktur data lain yang tampak serupa tetapi memiliki perbedaan mendasar.

3.2 TUJUAN

Tujuan dari pembelajaran ini adalah memahami konsep dasar tentang tipe data list dalam Python. Tujuan lainnya meliputi:

1. Memahami bahwa list adalah tipe data yang dapat digunakan untuk menyimpan banyak nilai dalam berbagai format.
2. Mengetahui bahwa list adalah mutable, yang berarti Anda dapat mengubah nilai dalam daftar.
3. Menyadari bahwa list adalah struktur data terurut, yang berarti elemen-elemen dalam list akan ditampilkan dalam urutan yang sama seperti saat dimasukkan. Mempelajari cara membuat list, mengakses elemen dalam list dengan menggunakan indeks, dan memeriksa tipe data variabel yang berisi list.
4. Memahami pentingnya penggunaan list dalam situasi praktis, seperti menyimpan daftar tinggi keluarga atau informasi lainnya.

3.3 LIST

List adalah tipe data dalam Python yang digunakan untuk menyimpan koleksi nilai dalam berbagai format, bukan hanya satu nilai. List bersifat mutable, artinya isinya dapat diubah. Mereka juga terurut, artinya elemen-elemen akan ditampilkan dalam urutan yang sama seperti saat dimasukkan. List diperlakukan seperti tipe data lain dalam Python dan dapat disimpan dalam variabel. Mereka dibuat dengan tanda kurung siku dan elemen-elemennya dipisahkan oleh koma. Anda dapat membuat list kosong dalam notebook Anda, menyimpannya dalam variabel, dan memeriksa tipe variabel tersebut.

```
In [2]: # creating an empty list in Python
height = []
type(height)

Out[2]: list
```

Gambar 3.1 Creating an empty list in Python

Buatlah daftar yang berisi tinggi anggota keluarga dalam meter seperti yang ditunjukkan dalam gambar.

```
In [3]: # a List containing heights
height_list = [1.76, 1.64, 1.79, 1, 57]
print(height_list)

[1.76, 1.64, 1.79, 1, 57]
```

Gambar 3.2. A list containing heights

Keuntungan dari list adalah kemampuannya untuk menyimpan berbagai jenis nilai dalam satu list, bahkan list dalam list itu sendiri.

```
In [4]: # a list containing str and float
name_height_list = ["Tom",1.76,"Harry",1.64,"Lisa",1.79,"Mona",1.57]
print(name_height_list)

['Tom', 1.76, 'Harry', 1.64, 'Lisa', 1.79, 'Mona', 1.57]
```

Gambar 3.3. A list containing str and float

3.4 OPERASI MANIPULASI DAFTAR YANG BERBEDA.

Anda akan mempelajari cara manipulasi daftar. Misalnya, membuat daftar, mengakses elemennya berdasarkan indeks, dan mencetak nilai-nilai dalam daftar tersebut. Perhatikan bahwa indeks dalam daftar dimulai dari nol (0).

```
In [6]: lang = ['python','c','java']

print (lang[0] + ' is very easy to learn for Data Science')
print (lang[1] + ' is the first language I have learnt')
print (lang[2]+ ' is difficult to learn for Data Science')

python is very easy to learn for Data Science
c is the first language I have learnt
java is difficult to learn for Data Science
```

Gambar 3.4. Begin with number zero (0)

Karena daftar bersifat dapat diubah, kita dapat mengubah nilai yang ada dari setiap elemen; mari kita lakukan ini dengan mengganti bahasa Java dengan bahasa COBOL, seperti gambar berikut:

```
In [1]: lang = ['python','c','java']
print("old list:", lang)
lang[2] = 'cobol'
print("new list:", lang)

old list: ['python', 'c', 'java']
new list: ['python', 'c', 'cobol']
```

Gambar 3.5. Changing java with cobol language

Sekarang Anda ingin mencetak semua elemen dalam daftar satu per satu dengan menggunakan perulangan "for" seperti yang ditunjukkan dalam gambar berikut.

```
In [2]: language_list = ['python','c','cobol']
for language in language_list:
    print("language is: ", language)

language is: python
language is: c
language is: cobol
```

Gambar 3.6. Using 'for' loop

Mari kita periksa berapa banyak elemen yang ada dalam daftar bahasa kita menggunakan metode len() dari daftar (lihat gambar berikut).

```
In [3]: language_list = ['python','c','cobol']
print("elements in the list: ", len(language_list))

elements in the list: 3
```

Gambar 3.7. Using list's len() method

Sekarang Anda ingin menambahkan bahasa atau item baru ke dalam daftar Anda; mari kita lakukan ini menggunakan metode append() dari daftar (lihat gambar berikut).

```
In [4]: language_list = ['python','c','cobol']
language_list.append('java')
print("updated list is:", language_list)

updated list is: ['python', 'c', 'cobol', 'java']
```

Gambar 3.8. Using list's append() method

Anda dapat menambahkan elemen baru pada posisi tertentu dalam list menggunakan metode insert() dengan menyertakan indeksnya.

```
In [5]: language_list = ['python','c','cobol','java']
language_list.insert(2, '.net')
print("modified list is:", language_list)

modified list is: ['python', 'c', '.net', 'cobol', 'java']
```

Gambar 3.9. Adding a new language .net

Anda dapat menghapus elemen dari daftar dengan menggunakan metode remove() berdasarkan nama atau pop() berdasarkan indeks.

```
In [8]: language_list = ['python','c','.net','cobol','java']
# remove element by name
language_list.remove('cobol')
print("updated list:", language_list)
# remove element by index
language_list.pop(3)
print("latest list:", language_list)

updated list: ['python', 'c', '.net', 'java']
latest list: ['python', 'c', '.net']
```

Gambar 3.10. Removing some element from the list

Metode del() menghapus elemen pada indeks tertentu dengan sintaks yang berbeda dari pop() atau remove(). Ini berguna ketika Anda ingin menghapus elemen dengan indeks tertentu dalam daftar.

```
In [11]: number_list = [1,2,3,4,1]
number_list.remove(4)
print("list after remove() example:", number_list)

number_list = [1,2,3,4,1]
number_list.pop(4)
print("list after pop() example:", number_list)

number_list = [1,2,3,4,1]
del(number_list[4])
print("list after del() example:", number_list)

list after remove() example: [1, 2, 3, 1]
list after pop() example: [1, 2, 3, 4]
list after del() example: [1, 2, 3, 4]
```

Gambar 3.11. Applying different methods to remove an element

Anda dapat mengurutkan daftar Anda secara naik atau turun menggunakan metode sort().

```
In [18]: language_list = ['python','c','.net','cobol','java']
language_list.sort()
print("sort in ascending order:", language_list)
languages_list = ['python','c','.net','cobol','java','c#']
language_list.sort(reverse=True)
print("sort in descending order:", language_list)

sort in ascending order: ['.net', 'c', 'cobol', 'java', 'python']
sort in descending order: ['python', 'java', 'cobol', 'c', '.net']
```

Gambar 3.12. Sorting list in ascending or descending order

3.5 PERBEDAAN LIST DAN TUPLE

Tuple adalah tipe data dalam Python yang diinisialisasi dengan tanda kurung () dan bersifat tidak dapat diubah. Tidak ada metode seperti append(), remove(), atau pop() dalam Tuple.

```

tuple_example = ('CS', 'IT', 'EC', 'ME')
print("tuple example: ", tuple_example)
print("data type of the example is", type(tuple_example))

tuple example: ('CS', 'IT', 'EC', 'ME')
data type of the example is <class 'tuple'>

```

Gambar 3.13. *Tuples*

3.6 KAMUS (DICTIONARY)

Dalam Python, kamus adalah koleksi pasangan kunci-nilai. Mereka diinisialisasi dalam kurung kurawal {} dengan kunci dan nilai yang dipisahkan oleh tanda titik dua. Kamus bersifat tidak terurut.

MEMBUAT KAMUS

Membuat kamus untuk menyimpan informasi mobil dengan menggunakan kunci sebagai nama properti dan nilai sebagai nama atau nilai propertinya.

```

In [19]: dict_example = {
    'brand': 'Hyundai',
    'model': 'Creta',
    'type': 'SUV',
    'year': '2017'
}
print("dictionary example: ", dict_example)

dictionary example: {'brand': 'Hyundai', 'model': 'Creta', 'type': 'SUV', 'year': '2017'}

```

Gambar 3.14. Creating a dictionary

BEBERAPA OPERASI DENGAN KAMUS

Setelah membuat kamus, Anda dapat mengakses item dengan menggunakan kunci atau metode get().

```

In [22]: # access the brand value by key
car_brand_by_key = dict_example['brand']
print("car brand by key:", car_brand_by_key)
# access the brand value by get()
print("car brand by method:", dict_example.get('brand'))

car brand by key: Hyundai
car brand by method: Hyundai

```

Gambar 3.15 Using key and get() methods

Anda dapat mengubah nilai dalam kamus dengan merujuk ke nama kunci, contohnya mengubah tahun pembuatan mobil dari 2017 menjadi 2018.

```

In [24]: dict_example['year'] = '2018'
print("updated dict: ", dict_example)

updated dict: {'brand': 'Hyundai', 'model': 'Creta', 'type': 'SUV', 'year': '2018'}

```

Gambar 3.16. Changing value in the dictionary

Anda perlu kunci atau nilai dari kamus. Anda dapat mencetak semua nama kunci atau nilai dengan perulangan "for"

```

In [26]: # printing all keys
for car_property in dict_example:
    print("key in dict:", car_property)

# printing all values
for car_property_value in dict_example.values():
    print("value in dict:", car_property_value)

key in dict: brand
key in dict: model
key in dict: type
key in dict: year
value in dict: Hyundai
value in dict: Creta
value in dict: SUV
value in dict: 2018

```

Gambar 3.17. *For loop*

Jika seorang pemilik bisnis ingin Anda menampilkan detail mobil dalam format pasangan kunci-nilai, Anda dapat melakukannya seperti yang ditunjukkan pada gambar.

```
In [27]: for car_property, car_property_value in dict_example.items():
    print(car_property, car_property_value)

brand Hyundai
model Creta
type SUV
year 2018
```

Gambar 3.18 Displaying car details in a key-value pair

3.7 KESIMPULAN

List dan Dictionary adalah tipe data yang penting untuk mengelola data besar. Pembelajaran bab ini akan membantu Anda lebih percaya diri dalam penggunaannya. Selanjutnya, kita akan mempelajari tentang fungsi dan paket Python.

BAB IV

PAKET, FUNGSI, dan PERULANGAN

4.1 STRUKTUR

Paket, fungsi, dan perulangan meningkatkan modularitas dan penggunaan ulang kode. Kami akan mempelajari fungsi Python bawaan, mengimpor paket, membuat dan menggunakan fungsi, parameter fungsi, variabel global dan lokal, fungsi Lambda, pemahaman tentang fungsi main, serta perulangan while dan for di Python.

4.2 TUJUAN

Setelah mempelajari bab ini, Anda akan dapat menggunakan fungsi dan paket bawaan Python dan menulis fungsi Anda.

4.3 FUNGSI HELP()

Untuk memahami lebih lanjut tentang fungsi Python, Anda dapat menggunakan fungsi help(). Misalnya, untuk fungsi len(), Anda dapat mengetik help() untuk mendapatkan informasi lengkap tentangnya.

```
help(len)
Help on built-in function len in module builtins:

len(obj, /)
    Return the number of items in a container.
```

Gambar 4.1. Using help() function

4.4 IMPORT PAKET

Untuk menggunakan fungsionalitas bawaan, Anda perlu mengimpor paket yang sesuai dengan kata kunci "import", seperti paket math untuk menghitung luas lingkaran.

```
import math

# define area as variable area
area = 0
# define radius as variable r
r = 5.89
# calculate area
area = math.pi * r**2
print("area of the land is: ", area)

area of the land is: 108.98844649760245
```

Gambar 4.2 Math package provided by Python

4.5 MEMANGGIL FUNGSI

Mendefinisikan fungsi dengan "def", contoh mencetak "hello world" dalam fungsi, dan perhatikan indentasi yang diperlukan setelah titik dua.

```
# defining my own function
def my_function():
    print("Hello World")

# calling my function
my_function()

Hello World
```

Gambar 4.4. Create and call a function

4.6 MELEWATI PARAMETER DALAM SUATU FUNGSI

Kita dapat mengirimkan informasi ke dalam fungsi melalui parameter atau argumen.

```
# defining a function to return sum of two numbers
def add_two_numbers(a,b):
    return a + b
# call the function
add_two_numbers(9,8)
```

17

Gambar 4.5. Defining a function to return sum of two numbers

4.7 PARAMETER YANG TIDAK DIKETAHUI DALAM SUATU FUNGSI

Jika Anda tidak tahu jumlah argumen yang akan dikirim ke dalam fungsi, tetapi Anda masih bisa mengirim parameter seperti yang ditunjukkan dalam gambar untuk menjumlahkan tiga angka dengan sum().

```
# Define `add_function()` function to accept any no.of parameters
def add_function(*args):
    return sum(args)
# Calculate the sum of the numbers
add_function(9,4,8)
```

21

Gambar 4.7. Sum() function to add three numbers

```
def my_function():
    #print(my_text)
    my_text = "I am also learning"
    print(my_text)

    # define a Global scope variable
    my_text = "I am learning Python for Data Science"
    my_function()
    print(my_text)
```

I am also learning
I am learning Python for Data Science

Gambar 4.8. Commenting the first print line after function declaration

4.8 FUNGSI LAMDA

Fungsi Lambda dalam Python dikenal sebagai fungsi anonim dan digunakan dengan kata kunci "lambda" untuk mendefinisikan fungsi sederhana, seperti mengalikan dengan 5.

```
def multiply(x):
    return x*5
multiply(2)

10

#same functionality with lambda function
multiply = lambda x: x*5
multiply(2)

10
```

Gambar 4.9. Normal function and lambda function

4.9 WHILE dan LOOP

Perulangan adalah blok kode yang diulang dalam Python, ada dua jenisnya: perulangan while dan perulangan for. Perulangan while mengulang kode selama kondisi tertentu terpenuhi.

```
x = 1
while x < 4:
    print(x)
    x = x + 1
```

Gambar 4.10 While loop code snippet

Dalam kode sebelumnya, x diberi nilai 1 dan perulangan while digunakan untuk menjalankan kode selama x kurang dari 4. Perulangan for menjalankan kode untuk setiap item dalam koleksi.

```
for letter in "Science":  
    print(letter)
```

Gambar 4.11 For loop code snippet

4.10 KESIMPULAN

Anda akan sering menulis fungsi untuk menyelesaikan masalah data. Anda juga akan mengimpor paket dan menulis fungsi untuk berbagai tugas. Pada bab berikutnya, kita akan mempelajari paket dasar pertama untuk komputasi ilmiah di Python.

BAB V

FUNDAMENTAL NUMPY

5.1 STRUKTUR

NumPy merupakan paket fundamental untuk komputasi ilmiah dalam Python. NumPy menyediakan objek array N-dimensi yang kuat dan kemampuan aljabar linear yang berguna. Bab ini mencakup topik-topik berikut:

1. Mengimpor paket NumPy.
2. Keunggulan penggunaan array NumPy dibandingkan dengan daftar.
3. Atribut-atribut array NumPy.
4. Cara membuat array NumPy.
5. Cara mengakses elemen dalam array NumPy.
6. Penggunaan slicing dalam array NumPy.

5.2 TUJUAN

Anda akan memiliki pemahaman yang kuat tentang cara menggunakan array NumPy dalam manipulasi data.

5.3 IMPOR PAKET NUMPY

```
# importing numpy package
import numpy as np
```

Gambar 5.1 Importing a NumPy package

Dalam pernyataan impor di atas, "np" adalah alias yang digunakan untuk merujuk ke NumPy. Alias ini digunakan agar nama paket dapat disingkat dalam penggunaan selanjutnya.

5.4 MENGGUNAKAN ARRAY NUMPY DARIPADA LIST

Daftar memiliki keterbatasan dalam melakukan operasi matematika, sehingga Python menggunakan array NumPy untuk mengatasinya. Untuk melakukannya, kita perlu mengimpor paket NumPy, mengonversi daftar menjadi array NumPy, dan kemudian melakukan operasi yang diperlukan. Array NumPy memungkinkan operasi matematika yang lebih efisien daripada daftar biasa.

```
import numpy as np
distance = [55,60,45]
speed = [6,10,7]
dist = np.array(distance)
spd = np.array(speed)
time= dist/spd
print(time)

[9.16666667 6.           6.42857143]
```

Gambar 5.2 Using NumPy Array

5.5 ATRIBUT ARRAY NUMPY

```
# data type
print("data type of array:", time.dtype)
# no. of dimensions
print("no. of dimensions:", time.ndim)
# size of each dimension
print("size of each dimension:", time.shape)
# total size of array
print("total size of array:", time.size)

data type of array: float64
no. of dimensions: 1
size of each dimension: (3,)
total size of array: 3
```

Gambar 5.3. NumPy attributes

5.6 MEMBUAT ARRAY NUMPY

Anda dapat membuat array NumPy dengan satu, dua, atau tiga dimensi, tergantung pada masalah yang ingin Anda selesaikan. Untuk menghasilkan array dengan angka acak (dalam kasus ini, angka bulat), Anda dapat menggunakan fungsi random dari NumPy.

```
# creating arrays with random values
np.random.seed(0) # seed for reproducibility

x1 = np.random.randint(10, size=6) # One-dimensional array
x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array
x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array

print("x1 ndim: ", x1.ndim)
print("x1 shape:", x1.shape)
print("x1 size: ", x1.size)

print("x2 ndim: ", x2.ndim)
print("x2 shape:", x2.shape)
print("x2 size: ", x2.size)

print("x3 ndim: ", x3.ndim)
print("x3 shape:", x3.shape)
print("x3 size: ", x3.size)

x1 ndim: 1
x1 shape: (6,)
x1 size: 6
x2 ndim: 2
x2 shape: (3, 4)
x2 size: 12
x3 ndim: 3
x3 shape: (3, 4, 5)
x3 size: 60
```

Gambar 5.4. Creating arrays with random values

Membuat array lain dengan menggunakan fungsi arange() dari NumPy, yang menghasilkan nilai-nilai dengan selang tertentu.

```
f = np.arange(0, 20, 5)
print ("sequential array with steps of 5:\n", f)

sequential array with steps of 5:
[ 0  5 10 15]
```

Gambar 5.4. Using arange() function

5.7 MENGAKSES ELEMEN DALAM ARRAY NUMPY

Untuk menganalisis dan memanipulasi sebuah array, Anda perlu mengakses elemen-elemennya. Pada array satu dimensi, nilai (mulai dari nol) dapat diakses dengan menentukan indeks yang diinginkan dalam tanda kurung siku, sama seperti dalam daftar Python.

```
print("1-d array:", x1)
print("second element of first array:", x1[1])
print("last element of first array:", x1[-1])
print("first element of first array:", x1[0])

1-d array: [5 0 3 3 7 9]
second element of first array: 0
last element of first array: 9
first element of first array: 5
```

Gambar 5.5. Accessing elements of the arrays

```
print("2-d array:\n", x2)
print("first elements of 2-d array:\n", x2[0,0])
print("3-d array:\n", x3)
print("first element of 3-d array:\n", x3[0,0,0])

2-d array:
[[3 5 2 4]
 [7 6 8 8]
 [1 6 7 7]]
first elements of 2-d array:
3
3-d array:
[[[8 1 5 9 8]
 [9 4 3 0 3]
 [5 0 2 3 8]
 [1 3 3 3 7]]

 [[0 1 9 9 0]
 [4 7 3 2 7]
 [2 0 0 4 5]
 [5 6 8 4 1]]

 [[4 9 8 1 1]
 [7 9 9 3 6]
 [7 2 0 3 5]
 [9 4 4 6 4]]]
first element of 3-d array:
8
```

Gambar 5.6. Accessing first elements of 2D and 3D arrays

5.8 PEMOTONGAN (SLICING) DALAM ARRAY NUMPY

Kita dapat menggunakan tanda kurung siku dengan notasi slice (dengan tanda titik dua) untuk mengakses subarray dalam array NumPy. Ini berbeda dengan daftar karena slicing array mengembalikan tampilan (view) bukan salinan data array.

```
# create an array
x = np.arange(10)
print("our array:", x)
print("first five elements:", x[:5])
print("elements after index 5:", x[5:])
print("middle sub-array:", x[4:7])
print("every other element:", x[::2])
print("every other element, starting at index 1:", x[1::2])
print("elements in reversed order:", x[::-1])

our array: [0 1 2 3 4 5 6 7 8 9]
first five elements: [0 1 2 3 4]
elements after index 5: [5 6 7 8 9]
middle sub-array: [4 5 6]
every other element: [0 2 4 6 8]
every other element, starting at index 1: [1 3 5 7 9]
elements in reversed order: [9 8 7 6 5 4 3 2 1 0]
```

Gambar 5.7. Creating 1D array

Untuk array multi-dimensi, kita perlu menggunakan beberapa slicing dengan tanda koma, seperti yang ditunjukkan dalam gambar berikut.

```
print("2-d array:\n", x2)
print("two rows, three columns:\n", x2[:, :, :3])

2-d array:
[[3 5 2 4]
 [7 6 8 8]
 [1 6 7 7]]
two rows, three columns:
[[3 5 2]
 [7 6 8]]
```

Gambar 5.8. Using multiple slices with comma for multidimensional array

Untuk memastikan integritas data array, kita dapat membuat salinan array asli menggunakan `copy()` dan kemudian mengubahnya tanpa memengaruhi array asli.

```

# original array
print("original 2-d array:\n", x2)
# creating a 2x2 subarray from the original array
x2_sub = x2[:2, :2]
print("sub-array:\n", x2_sub)
# modifying sub-array
x2_sub[0, 0] = 88
print("modified sub array:\n", x2_sub)
# original array after sub-array changes
print("original array after changes in sub-array:\n", x2)
print("making a copy of the original array")
x2_sub_copy = x2[:2, :2].copy()
print("copy of the orinal array:\n", x2_sub_copy)
# modifying copied array
x2_sub_copy[0, 0] = 42
print("copied array after changes:\n", x2_sub_copy)
print("original array:\n", x2)

```

```

# creating a sample array
x = np.array([1, 2, 3])
# creating a 2-d array
grid = np.array([[9, 8, 7],
                [6, 5, 4]])

# vertically stack the arrays
np.vstack([x, grid])

array([[1, 2, 3],
       [9, 8, 7],
       [6, 5, 4]])

```

```

# horizontally stack the arrays
y = np.array([[99],
              [99]])
np.hstack([grid, y])

array([[ 9,  8,  7, 99],
       [ 6,  5,  4, 99]])

```

Gambar 5.9. Adding different dimension arrays into one

5.9 KESIMPULAN

Dalam bab ini, kita belajar melakukan operasi matematika standar pada elemen-elemen individu atau seluruh array menggunakan NumPy. Ini mencakup berbagai fungsi untuk aljabar linear, operasi statistik, dan operasi matematika khusus lainnya. Untuk tujuan kita, kita hanya perlu tahu tentang array N-dimensi atau ndarray dan berbagai fungsi matematika yang relevan untuk penelitian kita. Sampai jumpa di bab berikutnya di mana kita akan belajar tentang paket Python yang kedua paling penting - Pandas.

BAB VI

PANDAS DAN DATAFRAME

6.1 STRUKTUR

Pandas adalah paket Python populer dalam bidang ilmu data. Ia menawarkan struktur data yang kuat, ekspresif, dan fleksibel yang memudahkan manipulasi dan analisis data, di antara banyak hal lain. Salah satu struktur data yang sangat kuat dan berguna dalam Pandas adalah DataFrame. Perpustakaan Pandas adalah salah satu alat yang paling disukai oleh ilmuwan data untuk melakukan manipulasi dan analisis data, selain matplotlib untuk visualisasi data dan NumPy, perpustakaan dasar untuk komputasi ilmiah dalam Python yang digunakan sebagai dasar Pandas. Struktur mencakup:

1. Mengimpor Pandas
2. Struktur data Pandas
3. .loc[] dan .iloc[]
4. Fungsi DataFrame
5. Penanganan nilai-nilai yang hilang.

6.2 TUJUAN

Anda akan memiliki kemampuan untuk membuat, memanipulasi, dan mengakses informasi yang Anda butuhkan dari data Anda dengan bantuan struktur data Pandas.

6.3 IMPOR PANDAS

Importing pandas in your notebook is quite simple. Pandas is preinstalled with Anaconda

```
# importing Pandas package using alias
import pandas as pd
```

Gambar 6.1. Importing Pandas

6.4 STRUKTUR DATA PANDAS

SERIES

Pandas adalah array satu dimensi dengan label yang dapat menyimpan berbagai jenis data seperti integer, float, dan string. Ini mirip dengan array satu dimensi NumPy. Pandas memberikan label kepada setiap nilai, dan jika label tidak diberikan oleh programmer, maka pandas akan menetapkan label sendiri (0 untuk elemen pertama, 1 untuk elemen kedua, dan seterusnya). Pemberian label memudahkan manipulasi data. Series pandas dapat dibuat dengan pd.Series().

```
# creating an empty Series
x = pd.Series()
print("empty series example: ", x)

empty series example: Series([], dtype: float64)
```

Gambar 6.2. Constructing pandas Series

```
# series example
series1 = pd.Series([10,20,30,50])
print(series1)

0    10
1    20
2    30
3    50
dtype: int64
```

Gambar 6.3. Another example of Series

Outputnya adalah tabel dengan dua kolom: indeks (dimulai dari nol) dan elemen-elemen. Anda bisa mengganti indeks dengan indeks yang Anda tentukan menggunakan parameter "indeks".

```
# re-indexing the default index column
series2 = pd.Series([10,20,30,50], index=['a','b','c','d'])
print(series2)

a    10
b    20
c    30
d    50
dtype: int64
```

Gambar 6.4 Re-indexing the default index column

Manipulasi data dengan Series juga mudah dilakukan. Anda dapat menerapkan perhitungan matematika seperti yang dilakukan di NumPy, seperti yang ditunjukkan dalam gambar berikut.

```
# data manipulation with Series
print("adding 5 to a Series:\n", series2 + 5)
print("filtering series with greater than 30:\n", series2[series2>30])
print("square root of Series elements:\n", np.sqrt(series2))

adding 5 to a Series:
a    15
b    25
c    35
d    55
dtype: int64
filtering series with greater than 30:
d    50
dtype: int64
square root of Series elements:
a    3.162278
b    4.472136
c    5.477226
d    7.071068
dtype: float64
```

Gambar 6.5. Data manipulation with Series

DATAFRAME

Berbeda dengan Series yang hanya memiliki satu indeks, objek DataFrame memiliki satu indeks untuk kolom dan satu indeks untuk baris. Ini memungkinkan fleksibilitas dalam mengakses dan memanipulasi data. Anda dapat membuat DataFrame menggunakan pd.DataFrame(), seperti yang ditunjukkan dalam gambar berikut.

```
# creating an empty dataframe
df = pd.DataFrame()
print("dataframe example:\n", df)

dataframe example:
Empty DataFrame
Columns: []
Index: []
```

Gambar 6.6. Creating an empty DataFrame

Membuat DataFrame dari sebuah list di mana list tersebut berisi nama dan usia seseorang. Kami juga akan mengganti nama kolom DataFrame kami menggunakan parameter columns, seperti yang ditunjukkan dalam gambar berikut.

# a sample list containing name and age		
data = [['Tom',10],['Harry',12],['Jim',13]]		
# creating a dataframe from given list with column names		
df = pd.DataFrame(data,columns=['Name','Age'])		
df		
Name	Age	
0	Tom	10
1	Harry	12
2	Jim	13

Gambar 6.7. A sample list containing name and age

Menambahkan kolom baru ke DataFrame Anda yang akan menyimpan tahun kelahiran seseorang. Anda dapat melakukannya dengan mudah seperti yang ditunjukkan dalam gambar berikut.

# adding a column in existing dataframe			
df['Year'] = 2008			
df			
Name	Age	Year	
0	Tom	10	2008
1	Harry	12	2008
2	Jim	13	2008

Gambar 6.8. Adding a column in existing DataFrame

Selanjutnya, penghapusan kolom juga merupakan tugas yang mudah. Anda dapat menggunakan `.pop()` untuk menghapus sebuah kolom. Lihat gambar berikut.

```
print("original dataframe:\n", df)
del df['Year']
print("dataframe after del:\n", df)
df.pop('Age')
print("dataframe after pop:\n", df)

original dataframe:
    Name  Age  Year
0   Tom   10  2008
1  Harry   12  2008
2   Jim   13  2008
dataframe after del:
    Name  Age
0   Tom   10
1  Harry   12
2   Jim   13
dataframe after pop:
    Name
0   Tom
1  Harry
2   Jim
```

Gambar 6.8. Deleting a column in existing DataFrame [.loc\[\] and .iloc\[\]](#)

Memilih baris atau indeks dalam DataFrame cukup berbeda, tetapi sangat mudah jika Anda tahu cara menggunakan fungsi `.loc[]` dan `.iloc[]`. Untuk memahami keduanya, mari pertama-tama

membuat DataFrame untuk menyimpan harga saham perusahaan, seperti yang ditunjukkan dalam gambar berikut.

# a sample dataframe containing company stock data			
data = pd.DataFrame({'price':[95, 25, 85, 41],			
'ticker':['AXP', 'CSCO', 'DIS', 'MSFT'],			
'company':['American Express', 'Cisco', 'Walt Disney', 'Microsoft'])})			
data			
company price ticker			
0	American Express	95	AXP
1	Cisco	25	CSCO
2	Walt Disney	85	DIS
3	Microsoft	41	MSFT

Gambar 6.9. A sample DataFrame containing company stock data

6.5 FUNGSI DATAFRAME

Menyimpan data dalam DataFrame memiliki berbagai manfaat dan cukup sederhana untuk analisis data. Mari kita lihat beberapa fungsi yang sangat berguna dari DataFrame.

# inspecting top 5 rows of a dataframe			
print("top five data:\n", data.head())			
# inspecting below 5 rows of a dataframe			
print("below 5 data:\n", data.tail())			
top five data:			
company price ticker			
0	American Express	95	AXP
1	Cisco	25	CSCO
2	Walt Disney	85	DIS
3	Microsoft	41	MSFT
below 5 data:			
company price ticker			
0	American Express	95	AXP
1	Cisco	25	CSCO
2	Walt Disney	85	DIS
3	Microsoft	41	MSFT

Gambar 6.10. Useful DataFrame functions

Fungsi .head() dan .tail() berguna saat Anda memiliki ribuan baris dan kolom dalam data Anda dan ingin melihatnya secara cepat. Juga, Anda dapat memeriksa tipe data setiap kolom dalam data Anda menggunakan fungsi tertentu.

# check data type of columns	
data.dtypes	
company	object
price	int64
ticker	object
dtype: object	

Gambar 6.11. Checking data type of columns

Anda dapat menggunakan .describe() untuk mengetahui statistik ini. Dengan .describe(), Anda dapat dengan mudah melihat statistik seperti harga saham tertinggi, harga saham terendah, dan jumlah saham dalam dataset Anda. Ini sangat berguna ketika Anda memiliki dataset yang besar.

# descriptive statistics of the data	
data.describe()	
price	
count	4.000000
mean	61.500000
std	33.798422
min	25.000000
25%	37.000000
50%	63.000000
75%	87.500000
max	95.000000

Gambar 6.12. Using describe() function to know statistics

```
# information of the dataframe
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
company      4 non-null object
price        4 non-null int64
ticker       4 non-null object
dtypes: int64(1), object(2)
memory usage: 176.0+ bytes
```

Gambar 6.13. Checking information of the DataFrame

6.6 MENANGANI DATA HILANG DI DATAFRAME

Anda dapat mendeteksinya dengan mudah menggunakan fungsi isnan().

```
# a sample dataframe
df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f', 'h'],
                  columns=['one', 'two', 'three'])
# creating a data with missing values by reindexing
df2 = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

   one      two      three
a -1.282674  1.081757 -0.559330
b      NaN      NaN      NaN
c  1.009585  0.876217  0.830863
d      NaN      NaN      NaN
e  1.308541 -0.434903 -1.224001
f  1.995670  1.199008 -0.671072
g      NaN      NaN      NaN
h  0.032248 -1.083125 -0.679454
```

Gambar 6.14. A sample DataFrame

```
# checking missing values using isnan()
print(df2.isnull())
missing_values_count = df2.isnull().sum()
print("count of missing values:\n", missing_values_count)

   one      two      three
a  False  False  False
b  True   True   True
c  False  False  False
d  True   True   True
e  False  False  False
f  False  False  False
g  True   True   True
h  False  False  False
count of missing values:
   one      3
   two      3
   three     3
dtype: int64
```

Gambar 6.15. Checking missing values

Jika Anda tidak memiliki petunjuk mengapa ada nilai yang hilang, salah satu cara sederhana untuk mengatasi masalah ini adalah dengan menghapusnya menggunakan fungsi .dropna() dan mengganti nilai dengan .fillna(0).

```
# remove all the rows that contain a missing value
df2 = df2.dropna()
print(df2)

   one      two      three
a -1.282674  1.081757 -0.559330
c  1.009585  0.876217  0.830863
e  1.308541 -0.434903 -1.224001
f  1.995670  1.199008 -0.671072
h  0.032248 -1.083125 -0.679454
```

Gambar 6.16. Removing all rows containing missing value

	one	two	three
a	2.000749	-0.256641	-0.041130
b	0.000000	0.000000	0.000000
c	-0.074203	-1.090353	-0.066285
d	0.000000	0.000000	0.000000
e	1.088535	-1.029808	0.553896
f	1.316821	0.125611	-0.627532
g	0.000000	0.000000	0.000000
h	-0.623504	-1.266855	1.043820

Gambar 6.17. Filling missing values

6.7 KESIMPULAN

Struktur data pandas yang cepat, fleksibel, dan ekspresif dirancang untuk membuat analisis data dunia nyata menjadi lebih mudah; namun, hal ini mungkin tidak segera terjadi bagi mereka yang baru memulai dengan pandas. Ada begitu banyak fungsionalitas yang dibangun ke dalam paket ini sehingga belajar opsi dalam satu kali coba bisa menjadi hal yang membingungkan. Sangat disarankan untuk berlatih dengan studi kasus yang sesuai. Jadi, buka notebook Anda, terapkan pembelajaran dari bab ini, dan jelajahi lebih lanjut. Pada bab berikutnya, kita akan belajar cara berinteraksi dengan berbagai basis data berbeda di Python.

BAB VII

KOMUNIKASI BASIS DATA

7.1 STRUKTUR

Anda akan berinteraksi dengan basis data secara teratur. Untuk tujuan ini, Anda perlu tahu cara melakukan kueri, membangun, dan menulis ke berbagai basis data. Pengetahuan tentang SQL (Structured Query Language) sangat cocok untuk hal ini. SQL digunakan untuk tiga hal: membaca/mengambil data, menulis data dalam basis data, serta memperbarui dan menyisipkan data baru. Python memiliki toolkit-nya sendiri, yaitu SQLAlchemy, yang memberikan cara yang mudah diakses dan intuitif untuk melakukan kueri, membangun, dan menulis ke basis data SQLite, MySQL, dan PostgreSQL (dan banyak lainnya). Kami akan mencakup semua detail basis data yang diperlukan khusus untuk ilmu data di sini. Bab ini mempelajari tentang:

1. SQLAlchemy
2. Memasang Paket SQLAlchemy
3. Menggunakan SQLAlchemy
4. Konfigurasi Mesin SQLAlchemy
5. Membuat Tabel dalam Basis Data
6. Memasukkan Data ke dalam Tabel
7. Memperbarui Rekaman
8. Menggabungkan Dua Tabel

7.2 TUJUAN

Setelah mempelajari bab ini, Anda akan menjadi akrab dengan dasar-dasar basis data relasional dan model relasional. Anda akan belajar bagaimana menghubungkan ke database dan berinteraksi dengannya dengan menulis kueri SQL dasar, baik dalam bentuk SQL mentah maupun dengan SQLAlchemy.

7.3 SQLAlchemy

SQLAlchemy adalah toolkit SQL Python dan Object Relational Mapper (ORM) yang memberikan Anda kekuatan dan fleksibilitas penuh dari SQL. Ini menyediakan cara berinteraksi dengan basis data yang sangat khas Python. Alih-alih berurusan dengan perbedaan antara dialek-dialek tertentu dari SQL tradisional seperti MySQL atau PostgreSQL atau Oracle, Anda dapat memanfaatkan kerangka kerja Pythonic SQLAlchemy untuk menyederhanakan alur kerja Anda dan menginterogasi data Anda dengan lebih efisien.

INSTALASI



The screenshot shows a terminal window titled "Anaconda Prompt". The command line displays two lines of text:
C:\Users\prateek1.gupta>set "KERAS_BACKEND=theano"
(base) C:\Users\prateek1.gupta>conda install -c anaconda sqlalchemy

Gambar 7.1 Installing SQLAlchemy package

■ IMPOR PAKET SQLAlchemy

```
In [1]: import sqlalchemy as db
```

Gambar 7.2 Importing package in the notebook

■ PENGGNAAN SQLAlchemy

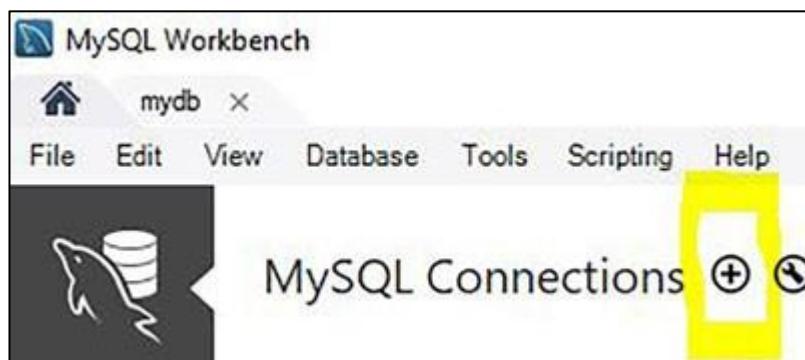
Sebelum menggunakan toolkit kita, harus ada basis data yang ingin Anda hubungkan terlebih dahulu. Kami dapat menggunakan SQLAlchemy untuk terhubung dengan PostgreSQL, MySQL, Oracle, Microsoft SQL, SQLite, dan banyak lainnya. Untuk tujuan pembelajaran kita, kita akan menggunakan basis data MySQL. Anda dapat mengunduh dan menginstal basis data MySQL dari situs web resminya:

<https://dev.mysql.com/downloads/installer/>

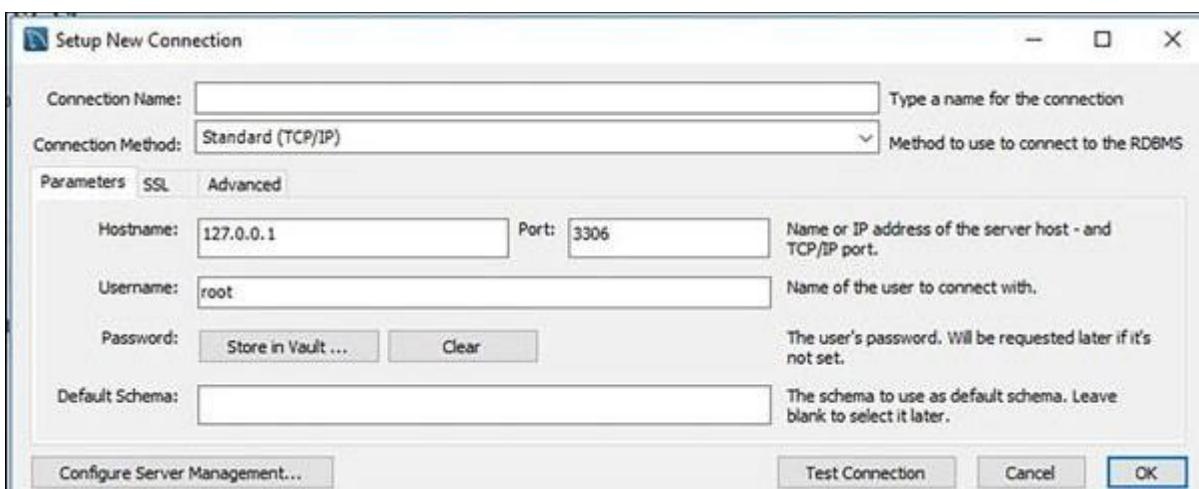
Untuk membuat basis data, tabel, dan lainnya, Anda dapat menginstal Workbench menggunakan tautan berikut:

<https://dev.mysql.com/downloads/workbench/>

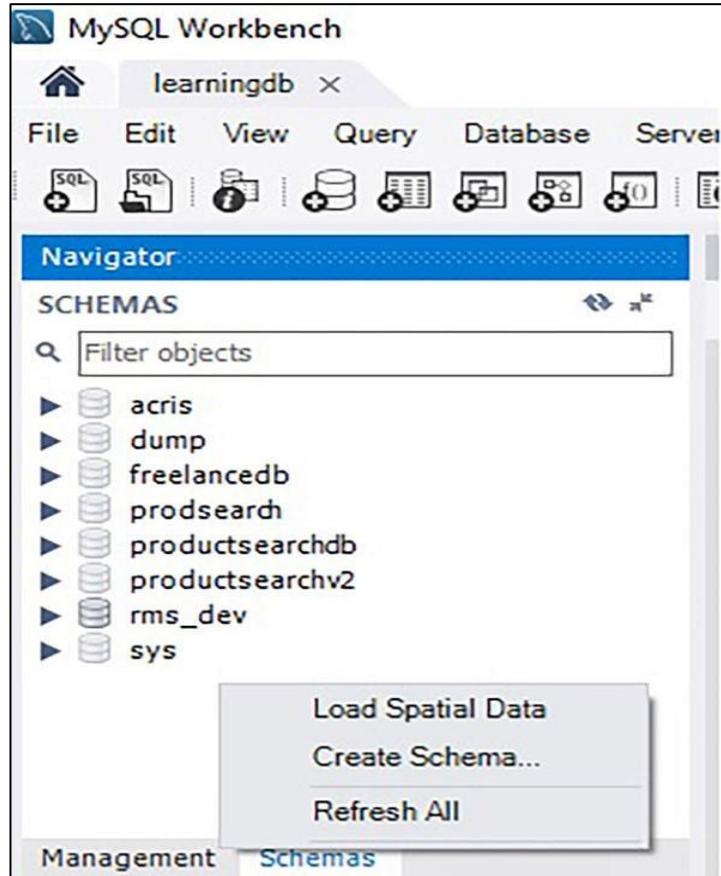
Setelah menginstal MySQL Workbench, Anda perlu pertama-tama membuat koneksi di sana. Untuk ini, buka Workbench dan klik ikon + seperti yang ditunjukkan dalam gambar berikut:



Gambar 7.3. Open MySQL workbench and click on the + icon



Gambar 7.3 Setting up new connection



Gambar 7.4. Creating a Schema

SQLALCHEMY ENGINE CONFIGURATION

Setelah Anda mengetahui informasi basis data Anda, langkah berikutnya adalah berinteraksi dengan basis data tersebut. Untuk ini, SQLAlchemy menggunakan Engine. Membuat Engine dengan SQLAlchemy cukup sederhana. Anda perlu menggunakan `create_engine`. Anda dapat mengimpor API ini dengan mengimpor `sqlalchemy import create_engine`. API `create_engine()` ini menggunakan sintaks berikut untuk menyimpan informasi basis data sebagai parameter:

```
engine = create_engine('database_type://username:password@host:port/database_name')
```

Gambar 7.5. SQLAlchemy `create_engine()` syntax

Di sini, nama dialek mencakup nama identifikasi dari dialek SQLAlchemy, seperti 'mysql' atau 'postgresql'. Nama pengemudi ('drivername') adalah nama dari DBAPI yang akan digunakan untuk terhubung ke basis data dengan huruf kecil semua. Jika tidak dijelaskan, DBAPI default akan diimpor jika tersedia – biasanya ini adalah driver yang paling umum yang tersedia untuk backend tersebut. Anda dapat memeriksa nama DBAPI dengan mengklik tautan berikut :

<https://docs.sqlalchemy.org/en/latest/core/engines.html#mysql>

```
engine = db.create_engine('mysql://root:admin@127.0.0.1:3306/rms_dev')
connection = engine.connect()
```

Gambar 7.6. Creating an engine

MEMBUAT TABEL DI DATABASE

```
CREATE TABLE [IF NOT EXISTS] 'TableName' ('fieldname' dataType [optional parameters]) ENGINE = storage Engine;
```

Gambar 7.7. Creating a table

```

query = "CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))"

connection.execute(query)
print("Table Name:", engine.table_names())
connection.close()

Table Name: ['customers']

```

Gambar 7.8. Printing the table name

INPUT DATA

```

engine = db.create_engine('mysql://root:admin@127.0.0.1:3306/schemaexample')
connection = engine.connect()
sql = "INSERT INTO customers (name, address) VALUES ('Prateek', 'India')"
connection.execute(sql)
connection.close()

```

Gambar 7.9. Inserting data in a table

```

engine = db.create_engine('mysql://root:admin@127.0.0.1:3306/schemaexample')
connection = engine.connect()
sql = "SELECT * from customers"
result = connection.execute(sql)
print("table data:", result.fetchall())
connection.close()

table data: [('Prateek', 'India')]

```

Gambar 7.10. Checking existing records

UPDATE DATA

```

engine = db.create_engine('mysql://root:admin@127.0.0.1:3306/schemaexample')
connection = engine.connect()
sql = "UPDATE customers SET address = 'Singapore' WHERE address = 'India'"
connection.execute(sql)
print("record(s) is updated")
q = "SELECT * from customers"
result = connection.execute(q)
print("table data:", result.fetchall())
connection.close()

record(s) is updated
table data: [('Prateek', 'Singapore')]

```

Gambar 7.11. Updating a record

```

engine = db.create_engine('mysql://root:admin@127.0.0.1:3306/schemaexample')
connection = engine.connect()
sql = "DELETE FROM customers WHERE address = 'Singapore'"
connection.execute(sql)
print("record is deleted!")
connection.close()

record is deleted!

```

Gambar 7.12. Deleting a record

JOIN TWO TABLES

Dalam basis data relasional, mungkin terdapat banyak tabel, dan dalam tabel-tabel tersebut, mungkin terdapat hubungan antara kolom-kolomnya. Dalam kondisi seperti itu, Anda perlu menggabungkan tabel-tabel tersebut. Contoh dunia nyata dari skenario ini adalah dari domain e-commerce di mana data terkait produk ada dalam satu tabel, data yang spesifik untuk pengguna ada dalam tabel lain, dan inventaris ada dalam tabel lainnya; di sini Anda perlu mengambil detail produk berdasarkan pengguna atau inventaris. Penggabungan tabel dapat dilakukan dalam tiga cara – inner join, left join, dan right join. Mari pahami masing-masing penggabungan ini.

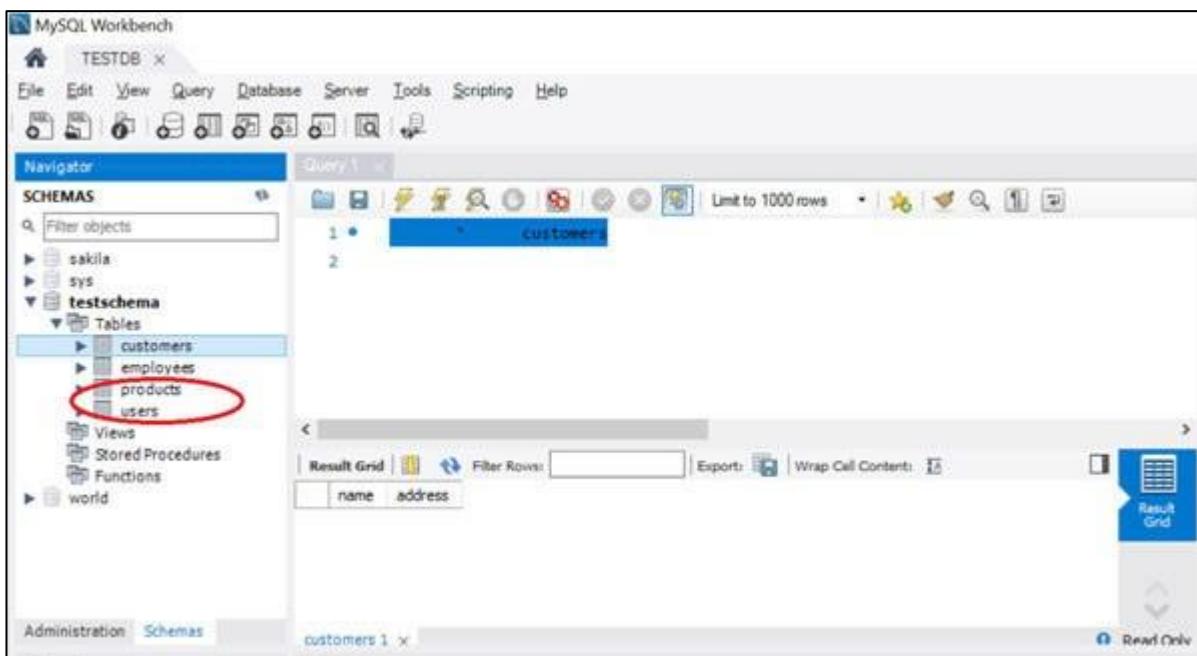
INNER JOIN

Membuat dua tabel - users dan products di dalam basis data kita untuk memahami jenis penggabungan ini terlebih dahulu. Jangan lupa untuk membuat dan kemudian menghubungkan koneksi basis data Anda sebelum menjalankan kode berikut:

```
query = "CREATE TABLE IF NOT EXISTS users (id INT, name VARCHAR(255), prod_id INT)"
connection.execute(query)
sql = "INSERT INTO users (id, name, prod_id) VALUES (1, 'Prateek', 11),(2,'John',12),(3,'Tom',13)"
connection.execute(sql)
query2 = "CREATE TABLE IF NOT EXISTS products (id INT, name VARCHAR(255))"
connection.execute(query2)
sql2 = "INSERT INTO products (id, name) VALUES (11, 'Apple'),(12,'Samsung'),(15,'Vivo')"
connection.execute(sql2)
connection.close()
```

Gambar 7.13. Inner join

Buka MySQL Workbench dan pilih skema yang Anda buat di awal, kemudian perluas skema tersebut. Anda akan melihat tabel baru yang baru saja Anda buat dari sel di atas, seperti yang ditunjukkan pada gambar berikut:



Gambar 7.14. A new table is created

Karena dalam contoh kita, tabel users dan products memiliki kolom product ID yang sama, kita dapat menggabungkan tabel users dan products berdasarkan product ID untuk melihat produk apa yang dibeli oleh pengguna tertentu, seperti yang ditunjukkan pada gambar berikut:

```
engine = db.create_engine('mysql://root:admin@127.0.0.1:3306/schemaexample')
connection = engine.connect()
join_query = "SELECT \
    users.name AS user, \
    products.name AS favorite \
    FROM users \
    INNER JOIN products ON users.prod_id = products.id"
result = connection.execute(join_query)
myresult = result.fetchall()
for bought_product in myresult:
    print(bought_product)

('Prateek', 'Apple')
('John', 'Samsung')
```

Gambar 7.15. Joining tables based on product ID

LEFT JOIN

```
left_join = "SELECT \
    users.name AS user, \
    products.name AS favorite \
    FROM users \
    LEFT JOIN products ON users.prod_id = products.id"
result = connection.execute(left_join)
myresult = result.fetchall()
for bought_product in myresult:
    print(bought_product)

('Prateek', 'Apple')
('John', 'Samsung')
('Tom', None)
```

Gambar 7.15. *Left join*

RIGHT JOIN

```
right_join = "SELECT \
    users.name AS user, \
    products.name AS favorite \
    FROM users \
    RIGHT JOIN products ON users.prod_id = products.id"
result = connection.execute(right_join)
myresult = result.fetchall()
for bought_product in myresult:
    print(bought_product)

('Prateek', 'Apple')
('John', 'Samsung')
(None, 'Vivo')
```

Gambar 7.16. *Right join*

7.4 KESIMPULAN

Kemahiran SQL penting untuk pekerjaan ilmu data. Belajar SQL akan memahami basis ilmu data dan meningkatkan profil profesional Anda. Terus latih kemampuan Python untuk berinteraksi dengan SQL dan pelajari konsep dasar statistik di bab berikutnya.

BAB VIII

STATISTIK DALAM ILMU DATA

8.1 STRUKTUR

Statistik memiliki peran penting dalam ilmu data. Jika digunakan dengan bijak, Anda dapat mengambil pengetahuan dari dunia nyata yang kabur, kompleks, dan sulit. Pemahaman yang jelas tentang statistik dan makna berbagai ukuran statistik penting untuk membedakan antara kebenaran dan penyimpangan. Dalam bab ini, Anda akan mempelajari konsep statistik penting dan alat statistik berbasis Python yang akan membantu Anda memahami data yang difokuskan pada ilmu data:

1. Statistik dalam ilmu data
2. Jenis data/variabel statistik
3. Mean, median, dan modus
4. Dasar probabilitas
5. Distribusi statistik
6. Koefisien korelasi Pearson
7. Fungsi densitas probabilitas
8. Contoh dunia nyata
9. Inferensi statistik dan pengujian hipotesis

8.2 TUJUAN

Setelah mempelajari bab ini, Anda akan dapat menerapkan statistik dengan cara berbasis Python untuk menganalisis data.

8.3 STATISTIK DALAM ILMU DATA

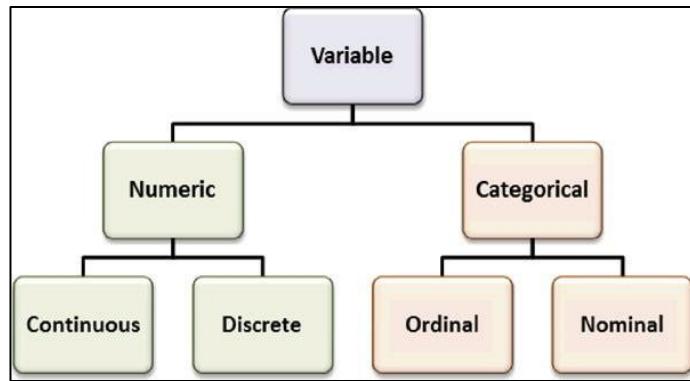
Statistik dalam ilmu data dibagi menjadi dua jenis: statistik deskriptif dan statistik inferensial. Statistik deskriptif digunakan untuk menganalisis data secara eksploratif, memberikan ringkasan data dasar, dan memahami dataset. Ini membantu menyajikan data dengan cara yang lebih bermakna untuk interpretasi yang lebih mudah. Sementara itu, statistik inferensial digunakan untuk membuat kesimpulan tentang populasi berdasarkan sampel. Ini melibatkan estimasi, interval kepercayaan, pengujian hipotesis, dan pertimbangan tentang populasi yang ingin digeneralisasi dari sampel. Dalam statistik, populasi merujuk pada seluruh rangkaian pengamatan yang mungkin ada.

8.4 JENIS DATA/VARIABEL STATISTIK

Data dalam statistik dapat dibagi menjadi dua jenis: data numerik dan data kategori. Data numerik adalah data berupa pengukuran seperti tinggi badan, berat badan, IQ, atau tekanan darah, atau bisa juga berupa hitungan seperti jumlah saham yang dimiliki seseorang atau jumlah gigi anjing. Data kategori adalah data yang menggambarkan karakteristik seperti jenis kelamin, status pernikahan, kota asal, atau jenis film yang disukai. Data kategori dapat memiliki nilai numerik, tetapi nilai-nilai tersebut tidak memiliki makna matematika. Kedua jenis variabel dalam statistik ini dapat dibagi lebih lanjut. Variabel dalam statistik dapat dibagi menjadi empat jenis:

1. Variabel Diskrit: Dapat dihitung dalam jumlah terbatas, seperti jumlah uang di dompet atau jumlah uang di rekening bank.
2. Variabel Kontinu: Tidak dapat dihitung dalam jumlah terbatas, seperti usia seseorang yang bisa memiliki berbagai satuan waktu yang sangat kecil.

3. Variabel Nominal: Memiliki dua atau lebih kategori tanpa urutan intrinsik, seperti jenis properti seperti rumah, kondominium, dll.
4. Variabel Ordinal: Memiliki dua atau lebih kategori yang dapat diurutkan, tetapi tidak memiliki nilai absolut, seperti tingkat kepuasan dari "sangat puas" hingga "tidak puas".



Gambar 8.1. Variables in statistics

8.5 MEAN, MEDIAN, DAN MODUS

MEAN

Mean, atau rata-rata, adalah nilai tengah dari sebuah set data. Anda dapat menggunakan NumPy untuk menghitung mean dengan mudah menggunakan fungsi `np.mean()`. Berikut adalah cara Pythonic untuk menghitung mean:

```

import pandas as pd
import numpy as np
a = np.array([[1, 2], [3, 4]])
print(np.mean(a))
print(np.mean(a, axis=0))
print(np.mean(a, axis=1))

2.5
[2. 3.]
[1.5 3.5]
  
```

Gambar 8.2. Pythonic way to calculate mean

MEDIAN

Median adalah angka yang berada di tengah-tengah daftar angka yang diurutkan. Berikut adalah cara menghitung median:

```

a = np.array([[10, 7, 4], [3, 2, 1]])
print(np.median(a))
print(np.median(a, axis=0))
print(np.median(a, axis=1))

3.5
[6.5 4.5 2.5]
[7. 2.]
  
```

Gambar 8.5. Calculating the median

MODUS

Mode adalah nilai yang muncul paling banyak dalam suatu data.

```

from scipy import stats
a = np.array([[1, 3, 4, 2, 2, 7],
              [5, 2, 2, 1, 4, 1],
              [3, 3, 2, 2, 1, 1]])
m = stats.mode(a)
print(m[0])

[[1 3 2 2 1 1]]

```

Gambar 8.4. Calculating the mode

8.6 DASAR PROBABILITAS

Kehidupan kita penuh dengan ketidakpastian. Probabilitas adalah ukuran seberapa mungkin suatu kejadian terjadi dalam situasi yang tidak pasti. Istilah yang perlu diketahui terkait probabilitas meliputi eksperimen, hasil, kejadian, dan probabilitas itu sendiri.

8.7 DISTRIBUSI STATISTIK

Salah satu hal paling penting yang perlu Anda ketahui dalam mempersiapkan diri dengan statistik yang diperlukan untuk ilmu data adalah distribusi. Distributions membantu visualisasi dalam statistik dan beberapa distribusi penting yang perlu diketahui.

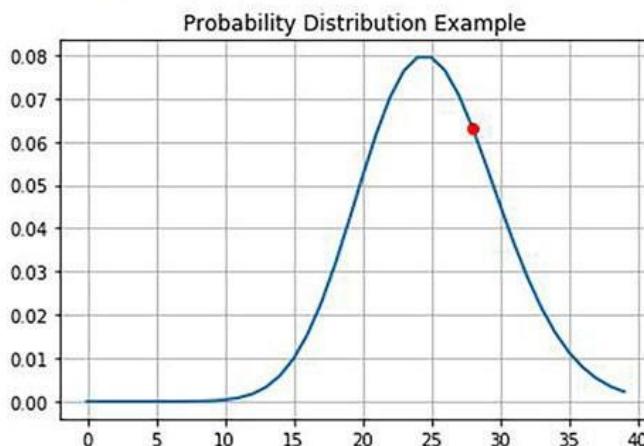
8.8 DISTRIBUSI POISSON

Distribusi Poisson digunakan untuk menghitung jumlah peristiwa dalam interval waktu kontinu. Di Python, kita dapat menghitungnya dengan menggunakan pustaka scipy dan memvisualisasikan sampelnya dengan matplotlib.

```

from scipy.stats import poisson
import matplotlib.pyplot as plt
plt.title('Probability Distribution Example')
arr = []
rv = poisson(25)
for num in range(0,40):
    arr.append(rv.pmf(num))
prob = rv.pmf(28)
plt.grid(True)
plt.plot(arr, linewidth=2.0)
plt.plot([28], [prob], marker='o', markersize=6, color="red")
plt.show()

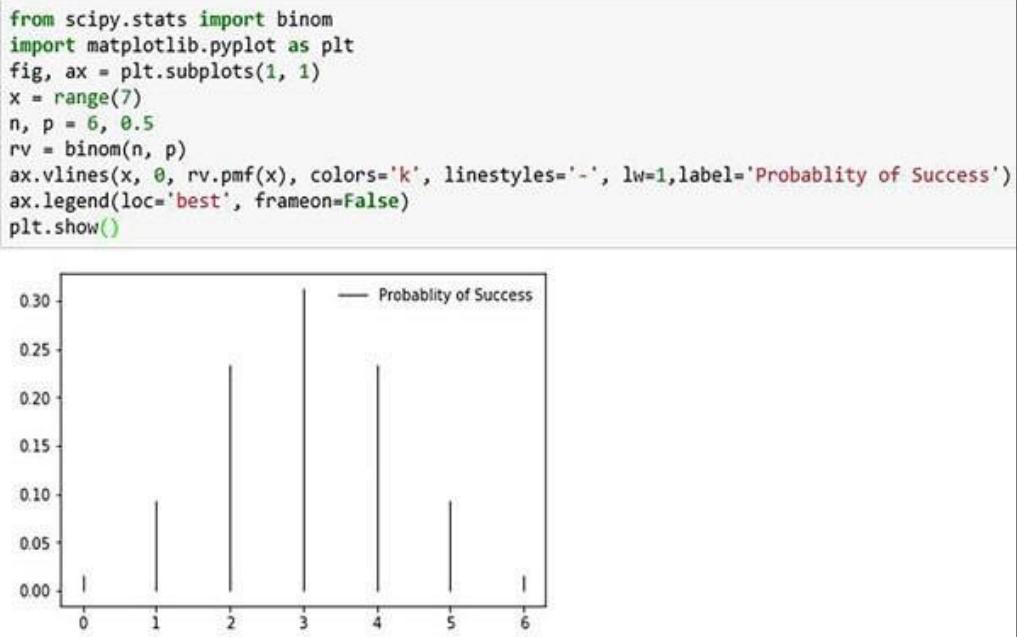
```



Gambar 8.5. Poisson Distribution

8.9 BINOMIAL DISTRIBUTION

Distribusi binomial adalah distribusi dengan hanya dua hasil yang mungkin, seperti sukses atau gagal, dan dapat digambarkan dengan matplotlib dan dibuat dengan scipy.



Gambar 8.5. Binomial Distribution

8.10 NORMAL DISTRIBUTION

Distribusi disebut distribusi normal jika memiliki karakteristik berikut:

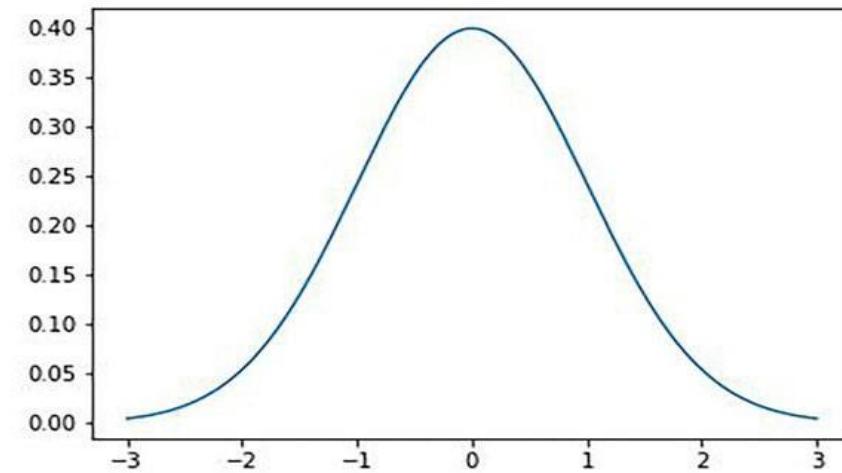
1. Rata-rata, median, dan modus dari distribusi tersebut bersamaan.
2. Bentuk kurva distribusi berbentuk lonceng dan simetris terhadap garis tengah, dengan separuh nilai berada di sebelah kiri pusat dan separuh lainnya di sebelah kanan.

```

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

range = np.arange(-3,3,0.001)
plt.plot(range, norm.pdf(range, 0, 1))
plt.show()

```



Gambar 8.6. Normal Distribution

8.11 KOEFISIEN KORELASI PEARSON

Cara Pythonic untuk menginterpretasikan koefisien korelasi Pearson, di mana `r_row` merupakan koefisien korelasi Pearson dan `p_value` merupakan probabilitas dari sistem yang tidak berkorelasi menghasilkan dataset yang memiliki korelasi Pearson setidaknya sekuat yang dihitung dari dataset ini. Nilai-nilai `p` tidak sepenuhnya dapat diandalkan, tetapi mungkin cukup baik untuk dataset yang lebih besar dari sekitar 500 atau lebih:

```

import scipy
from scipy.stats import pearsonr
x = scipy.array([-0.65499887, 2.34644428, 3.0])
y = scipy.array([-1.46049758, 3.86537321, 21.0])
r_row, p_value = pearsonr(x, y)
print(r_row)
print(p_value)

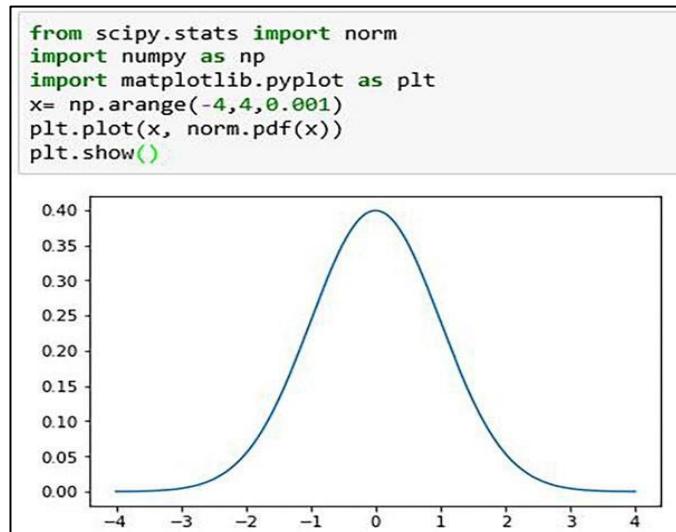
0.7961701483197555
0.41371200873701036

```

Gambar 8.7. Calculating Pearson correlation coefficient

8.12 FUNGSI DENSITAS PROBABILITAS

Fungsi Kepadatan Probabilitas (PDF) digunakan untuk menghitung probabilitas variabel acak berada dalam rentang tertentu. Di Python, kita dapat membuat dan memvisualisasikan PDF menggunakan pustaka seperti `scipy.stats` dan `matplotlib`.



Gambar 8.8. Probability Density Function [Real-world example](#)

8.13 CONTOH

Korelasi Pearson digunakan dalam banyak situasi kehidupan nyata, seperti dalam penelitian ilmiah di China yang ingin mengetahui hubungan antara populasi padi liar yang berbeda secara genetik. Tujuannya adalah untuk mengetahui potensi evolusioner dari padi tersebut. Hasil analisis korelasi Pearson antara kedua kelompok menunjukkan korelasi Pearson Product Moment positif antara 0,783 dan 0,895 untuk populasi padi liar.

8.14 INFERENSI STATISTIK DAN PENGUJIAN HIPOTESIS

Inferensi statistik adalah proses mengambil kesimpulan tentang populasi berdasarkan analisis data dengan menguji asumsi hipotesis nol (H_0) dan alternatif (H_1), dan membatasi kesalahan tipe 1.

```

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# array containing no of total experience
dss_exp = np.array([12, 15, 13, 20, 19, 20, 11, 19, 11, 12, 19, 13,
                    12, 10, 6, 19, 3, 1, 1, 0, 4, 4, 6, 5, 3, 7,
                    12, 7, 9, 8, 12, 11, 11, 18, 19, 18, 19, 3, 6,
                    5, 6, 9, 11, 10, 14, 14, 16, 17, 17, 19, 0, 2,
                    0, 3, 1, 4, 6, 6, 8, 7, 7, 6, 7, 11, 11, 10,
                    11, 10, 13, 13, 15, 18, 20, 19, 1, 10, 8, 16,
                    19, 19, 17, 16, 11, 1, 10, 13, 15, 3, 8, 6, 9,
                    10, 15, 19, 2, 4, 5, 6, 9, 11, 10, 9, 10, 9,
                    15, 16, 18, 13])

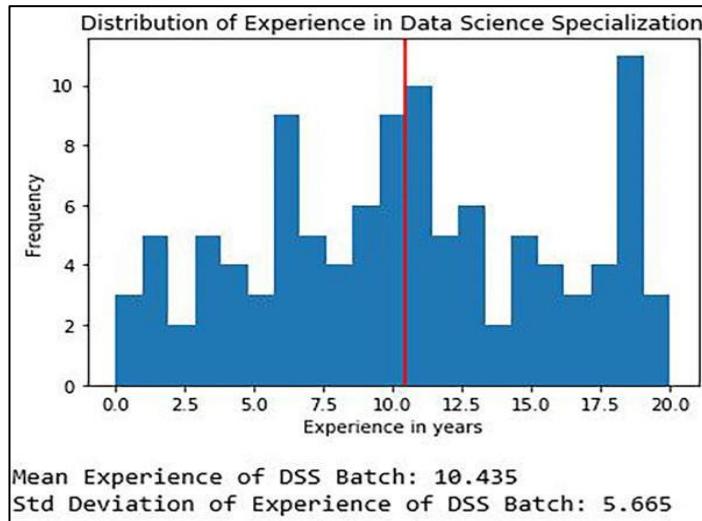
```

Gambar 8.9. a sample numpy array containing number of experience

Membuat histogram untuk melihat distribusi pengalaman menggunakan fungsi hist() dari matplotlib dengan pengaturan jumlah bins.

```
#Understanding the Underlying distribution of Experience
# Plot the distribution of Experience
plt.hist(dss_exp, range = (0,20), bins = 21)
# Add axis Labels
plt.xlabel("Experience in years")
plt.ylabel("Frequency")
plt.title("Distribution of Experience in Data Science Specialization")
# Draws the red vertical line in graph at the average experience
plt.axvline(x=dss_exp.mean(), linewidth=2, color = 'r')
plt.show()
# Statistics of DSS Batch experience
print("Mean Experience of DSS Batch: {:.4f}".format(dss_exp.mean()))
print("Std Deviation of Experience of DSS Batch: {:.4f}".format(dss_exp.std()))
```

Gambar 8.10 Understanding underlying distribution of experience



Gambar 8.11 Distribution of experience in data science specialization

```
# Set the parameters for sampling
n = 10
NUM_TRIALS = 1000
#Estimating DSS Experience from samples
samp = np.random.choice(dss_exp, size = n, replace = True) #Just try for 1 iteration
samp_mean = samp.mean()
samp_sd = samp.std()
print("Samp_mean = {:.4f} Sample_SD = {:.4f}".format(samp_mean, samp_sd))
print("sample values:", samp)

Samp_mean = 10.900 Sample_SD = 5.665
sample values: [10  1 19  9  3 13 19 15 11  9]
```

Gambar 8.12 Estimating experiences

```
#How will the distribution of Sample Mean Look like
np.random.seed(100)
mn_array = np.zeros(NUM_TRIALS)
sd_array = np.zeros(NUM_TRIALS)

# Extract Random Samples and compute mean & standard deviation
for i in range(NUM_TRIALS):
    samp = np.random.choice(dss_exp, size = n, replace = True)
    mn_array[i] = samp.mean()
```

Gambar 8.13. How distribution of sample mean look

Untuk menghitung mean, standard deviation, dan persentil, kita menggunakan fungsi mean(), std(), dan percentile() dari numpy.

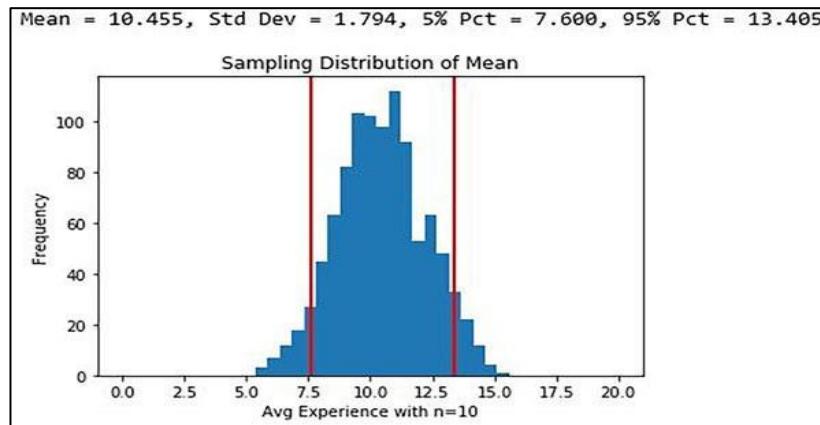
```

mn = mn_array.mean()
sd = mn_array.std()
x5_pct = np.percentile(mn_array, 5.0)
x95_pct = np.percentile(mn_array, 95.0)
print("Mean = {:4.3f}, Std Dev = {:4.3f}, 5% Pct = {:4.3f}, 95% Pct = {:4.3f}".format(mn, sd, x5_pct, x95_pct))
# Plot Sampling distribution of Mean
plt.hist(mn_array, range=(0,20), bins = 41)
# Add axis labels
plt.xlabel("Avg Experience with n={}".format(n))
plt.ylabel("Frequency")
plt.title("Sampling Distribution of Mean")
plt.axvline(x=x5_pct, linewidth=2, color = 'r')
plt.axvline(x=x95_pct, linewidth=2, color = 'r')
plt.show()

```

Gambar 8.14 Computing mean, standard deviation, and percentile

Grafik histogram menunjukkan bahwa pengalaman awal siswa ilmu data tidak mengikuti distribusi normal. Terdapat puncak di sekitar 5 tahun, 10 tahun, dan 19 tahun pengalaman.



Gambar 8.15. Sampling distribution of Mean

Untuk menghitung rentang kepercayaan dari sampel, kita memilih tingkat kepercayaan seperti 90%, 95%, atau 99%. Tingkat kepercayaan ini menentukan seberapa besar probabilitas interval kepercayaan yang dihasilkan akan mengandung nilai parameter sebenarnya. Semakin tinggi tingkat kepercayaan, semakin besar interval kepercayaan, dan semakin besar kepastian kita dalam mengestimasi parameter populasi. Misalnya, interval kepercayaan 95% mencakup 95% dari kurva kepadatan normal, dengan probabilitas hanya sekitar 5% untuk nilai di luar interval ini.

```

# Function to check if the true mean lies within 90% Confidence Interval
def samp_mean_within_ci(mn, l_5pct, u_95pct):
    out = True
    if (mn < l_5pct) | (mn > u_95pct):
        out = False
    return out

# Estimation and Confidence Interval
samp = np.random.choice(dss_exp, size = n, replace = True)
samp_mean = samp.mean()
samp_sd = samp.std()
# divided by sqrt(n) is done so as to compensate for the reduction in std. dev due to sample size of n
sd_ci = samp_sd/np.sqrt(n)
# Lower 90% confidence interval (This is approximate version to build intution)
samp_lower_5pct = samp_mean - 1.645 * sd_ci
# Upper 90% confidence interval (This is approximate version to build intution)
samp_upper_95pct = samp_mean + 1.645 * sd_ci
print("Pop Mean: {:4.3f} | Sample: L_5PCT = {:4.3f} | M = samp_mean = {:4.3f} | H_95PCT = {:4.3f}".format(dss_exp.mean(), 
# Checking if the population mean lies within 90% Confidence Interval (CI)
mn_within_ci_flag = samp_mean_within_ci(dss_exp.mean(), samp_lower_5pct, samp_upper_95pct)
print("True mean lies with the 90% confidence Intervel = {}".format(mn_within_ci_flag))
<

```

Gambar 8.16. Function to select confidence interval

Dalam menguji hipotesis, penting untuk memahami dua jenis kesalahan konseptual yang mungkin terjadi: kesalahan tipe 1 dan kesalahan tipe 2. Selain itu, kita harus menentukan batasan parametrik, misalnya seberapa besar kesalahan tipe 1 yang diperbolehkan.

```

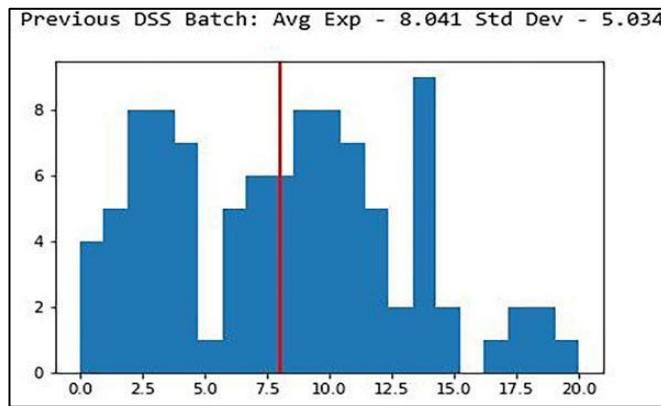
# Previous Batch Data for working experience
dss_exp_prev = np.array([1, 14, 6, 7, 10, 10, 19, 15, 19, 15,
                       2, 2, 14, 14, 14, 3, 0, 4, 11, 7,
                       1, 2, 0, 1, 2, 2, 2, 1, 1, 2,
                       4, 4, 3, 3, 3, 3, 4, 3, 3, 7,
                       8, 6, 6, 6, 7, 8, 8, 8, 8, 7,
                       8, 0, 0, 7, 6, 9, 10, 9, 9, 11,
                       11, 9, 10, 10, 11, 10, 11, 9, 9, 9,
                       12, 14, 13, 14, 18, 14, 11, 10, 17, 20,
                       18, 5, 13, 4, 2, 4, 3, 12, 12, 14,
                       12, 12, 10, 14, 4, 11, 9])
avg_exp_prev = dss_exp_prev.mean()
std_exp_prev = dss_exp_prev.std()
print("Previous DSS Batch: Avg Exp - {:.4f} Std Dev - {:.4f}".format(avg_exp_prev, std_exp_prev))

plt.hist(dss_exp_prev, range=(0,20), bins = 21)
plt.axvline(x=dss_exp_prev.mean(), linewidth=2, color = 'r')
plt.show()

Previous DSS Batch: Avg Exp - 8.041 Std Dev - 5.034

```

Gambar 8.17. Performing hypothesis testing



Gambar 8.18. plot of hypothesis testing example

```

np.random.seed(100)
n = 20

dss_mean = dss_exp.mean()
dss_sd = dss_exp.std()
print("Current DSS Batch : Population Mean - {:.4f}".format(dss_mean))

dss_prev_samp = np.random.choice(dss_exp_prev, size = n, replace = True)
dss_prev_samp_mean = dss_prev_samp.mean()
print("Previous DSS Batch Sample Mean: {:.4f}".format(dss_prev_samp_mean))

Current DSS Batch : Population Mean - 10.435
Previous DSS Batch Sample Mean: 8.250

from scipy import stats
t_statistic = (dss_prev_samp_mean - dss_mean)/(dss_sd/np.sqrt(n))
p_val = 2 * stats.t.cdf(t_statistic, df= (n-1))
print("T-Statistic : {:.4f}, p-Value = {:.4f}".format(t_statistic,p_val))

T-Statistic : -1.72, p-Value = 0.10

# For 2-tailed hypothesis testing
from scipy import stats
dss_exp_prev_samp = np.random.choice(dss_exp_prev, size = 20, replace = True)
dss_exp_samp = np.random.choice(dss_exp, size = 20, replace = True)
stats.ttest_ind(dss_exp_prev_samp, dss_exp_samp)

Ttest_indResult(statistic=-0.24857316405070548, pvalue=0.80502950101657478)

```

Gambar 8.19. Hypothesis testing

Uji t membandingkan rata-rata dan memberi tahu seberapa signifikannya perbedaan. Nilai t besar menunjukkan perbedaan yang signifikan. Nilai p adalah probabilitas hasil dari data sampel terjadi secara kebetulan. Nilai p rendah bagus, menunjukkan hasil tidak terjadi secara kebetulan. Kita tidak dapat menolak hipotesis nol jika nilai p besar.

8.15 KESIMPULAN

Kita telah mempelajari beberapa konsep inti statistik dalam bab ini. Statistik penting dalam analisis data dan membantu dalam mengekstraksi pengetahuan dari data. Selanjutnya, kita akan belajar cara mengimpor berbagai bentuk data dan bekerja dengan data.

BAB IX

PYTHON

9.1 STRUKTUR

Data importing adalah langkah pertama sebelum melakukan analisis data. Data dapat hadir dalam berbagai bentuk, seperti .txt, .csv, .excel, JSON, dll., dan mengimpor atau membaca data tersebut berbeda-beda, tetapi cukup sederhana dalam gaya Pythonic. Saat mengimpor data eksternal, Anda perlu memeriksa beberapa hal, seperti apakah terdapat baris header dalam data atau tidak, apakah terdapat nilai yang hilang, tipe data setiap atribut, dll. Dalam bab ini, dengan bantuan Pandas I/O API, Anda akan belajar tidak hanya cara membaca data, tetapi juga cara menulis data ke berbagai format file, diantaranya:

1. Impor text data
2. Impor CSV data
3. Impor Excel data
4. Impor JSON data
5. Impor pickled data
6. Import a compressed data Objective

9.2 TUJUAN

Setelah mempelajari bab ini, Anda akan menjadi ahli dalam mengimpor, membaca, dan menyempurnakan berbagai bentuk data.

9.3 IMPORTING TEXT DATA

Download dataset dibawah ini:

<https://drive.google.com/file/d/1AdBiQi6FH-4P03osgKmd0pR0DE1ipbV4/view?usp=sharing>

The screenshot shows a Jupyter Notebook cell. The code cell contains the following Python code:

```
import pandas as pd
cpi_data = pd.read_table('E:/pg/docs/BPB/data/cpi_us.txt')
cpi_data.head()
```

The output cell displays the first five rows of a DataFrame named 'cpi_data' with the following structure:

	series_id	year	period	value	footnote_codes
0	APU0000701111	1995	M01	0.238	
1	APU0000701111	1995	M02	0.242	
2	APU0000701111	1995	M03	0.242	
3	APU0000701111	1995	M04	0.236	
4	APU0000701111	1995	M05	0.244	

Gambar 9.1. Importing text data

Fungsi `pd.read_table()` mengimpor semua data ke dalam sebuah variabel. Jika Anda memeriksa tipe variabel ini, Anda akan melihat bahwa ini adalah DataFrame pandas.

The screenshot shows a Jupyter Notebook cell with the following output:

```
type(cpi_data)
```

The output is:

```
pandas.core.frame.DataFrame
```

Gambar 9.2. Importing in DataFrame

Kita dapat memeriksa data menggunakan fungsi `head()`. Fungsi `read_table()` pandas memiliki fungsionalitas bawaan untuk menyaring kolom yang kosong, yang dapat Anda gunakan dengan melewatkannya argumen yang dipisahkan oleh koma. Dengan menggunakan argumen

usecols seperti yang ditunjukkan dalam gambar berikut, kita dapat menyaring kolom yang tidak diinginkan:

series_id year period value				
0	APU0000701111	1995	M01	0.238
1	APU0000701111	1995	M02	0.242
2	APU0000701111	1995	M03	0.242
3	APU0000701111	1995	M04	0.236
4	APU0000701111	1995	M05	0.244

Gambar 9.3. Using cols argument

9.4 IMPOR CSV

CSV adalah format yang umum digunakan untuk menyimpan data. Contohnya, saya menggunakan data kejahatan Chicago untuk analisis:

<https://drive.google.com/file/d/1S6-A5Gr82XyhCeH9qxiBtHdY8M020vP3/view?usp=sharing>

crime_data = pd.read_csv('E:\pg\docs\BPB\data\Crimes_-_2001_to_present.csv') crime_data.head()												
ID	Case Number	Date	Block	IUCR	Primary Type	Description	Location Description	Arrest	Domestic
0	10000092	HY189866	03/18/2015 07:44:00 PM	047XX W OHIO ST	041A	BATTERY	AGGRAVATED: HANDGUN	STREET	False	False
1	10000094	HY190059	03/18/2015 11:00:00 PM	066XX S MARSHFIELD AVE	4625	OTHER OFFENSE	PAROLE VIOLATION	STREET	True	False
2	10000095	HY190052	03/18/2015 10:45:00 PM	044XX S LAKE PARK AVE	0486	BATTERY	DOMESTIC BATTERY SIMPLE	APARTMENT	False	True
3	10000096	HY190054	03/18/2015 10:30:00 PM	051XX S MICHIGAN AVE	0460	BATTERY	SIMPLE	APARTMENT	False	False
4	10000097	HY189976	03/18/2015 09:00:00 PM	047XX W ADAMS ST	031A	ROBBERY	ARMED: HANDGUN	SIDEWALK	False	False

Gambar 9.4. Importing CSV data

9.5 IMPOR EXCEL DATA

Excel adalah format dataset umum dengan lembaran berbeda. Gunakan fungsi `read_excel()` dari pandas dengan `sheet_name` untuk membaca data dari lembaran tertentu.

https://docs.google.com/spreadsheets/d/1-gXj7ZXPYRm_yE_KNAgWfjvw9xJ0R1l/edit?usp=sharing&ouid=113242821086051483753&rtpof=true&sd=true

order_data = pd.read_excel('E:\pg\docs\BPB\data\Sample - Superstore.xls', sheet_name='Orders') order_data.head()													
Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID
0	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798
1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454
2	CA-2016-138688	2016-06-12	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240

Gambar 9.5. Importing Excel data

9.6 IMPOR JSON DATA

Untuk data berformat JSON, Anda dapat menggunakan fungsi `read_json()` dari pandas dengan argumen `orient`.

https://drive.google.com/file/d/10HCnCFpgYVz6LiT30h-Bz5IAh6hMFkR1/view?usp=drive_link

```
glossary_data = pd.read_json('E:/pg/docs/BPB/data/glossary.json', orient='table')
glossary_data.head()

glossary
GlossDiv {"title": "S", "GlossList": {"GlossEntry": {"I...
title           example glossary
```

Gambar 9.6. Importing JSON data

9.7 PICKLED DATA

Dalam Python, Anda dapat menggunakan modul **pickle** untuk membaca objek yang telah disimpan ke dalam format "pickled". Proses "pickling" mengonversi objek menjadi byte stream sehingga dapat disimpan di disk dan diambil kembali nanti. Ini berguna dalam pembelajaran mesin untuk menyimpan model yang telah dilatih.

https://drive.google.com/file/d/1jN4dRvIP_1KNSpfAXaBP6RJDYeORWi09/view?usp=sharing

```
import pandas as pd
unpickled_data = pd.read_pickle("E:/pg/docs/BPB/data/mnist.pkl")
print("data type::", type(unpickled_data))
for index, digit in enumerate(unpickled_data):
    print(index, ":", digit)

data type:: <class 'tuple'>
0 : (array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
```

Gambar 9.7. Importing pickled data

9.8 DATA TERKOMPRESI

Data dalam bentuk terkompresi adalah data yang dikompresi dalam format ZIP. Anda dapat menggunakan modul `zipfile` dalam Python untuk membuka file ZIP dan membaca datanya.

https://drive.google.com/file/d/1DTChuc4it615Zlwxs_8IpqOadE6UQo_o/view?usp=sharing

```
import zipfile
Dataset = "africa_soil_train_data.zip"
with zipfile.ZipFile("E:/pg/bpb/BPB-Publications/Datasets/" + Dataset, "r") as z:
    z.extractall("E:/pg/bpb/BPB-Publications/Datasets")
```

Gambar 9.8. Importing a compressed data

9.9 KESIMPULAN

Dalam data science, impor data merupakan langkah pertama yang penting. Dalam bab ini, kita telah mempelajari berbagai format impor data. Pandas `read_csv()` sangat berguna karena banyak dataset tersedia dalam format CSV. Semakin sering Anda berlatih, semakin Anda akan memahami proses impor data. Selanjutnya, kita akan mempelajari proses pembersihan data di bab berikutnya.

BAB X

PEMBERSIHAN DATA

10.1 STRUKTUR

Akurasi model machine learning sangat bergantung pada kualitas data, sehingga pembersihan data adalah tugas penting dalam pekerjaan seorang ilmuwan data. Dalam bab ini, Anda akan mempelajari cara membersihkan data melalui studi kasus dan penerapan konsep yang telah dipelajari sebelumnya.

- Mengenal data
- Menganalisis nilai-nilai yang hilang
- Menghapus nilai-nilai yang hilang
- Cara menyesuaikan dan normalisasi data
- Cara memproses tanggal
- Cara menerapkan pengkodean karakter
- Membersihkan data yang inkonsisten

10.2 TUJUAN

Dalam bab ini, Anda akan belajar langkah-langkah konkret untuk membersihkan data agar menjadi lebih terstruktur, akurat, dan siap untuk digunakan dalam analisis data lebih lanjut.

10.3 MENGENAL DATA

Langkah pertama dalam pembersihan data adalah memahami masalah bisnis dan mengakses data yang diberikan dalam format ZIP. Gunakan fungsi Pandas .zipfile() untuk membuka dan membaca data dari lokasi file yang tepat.

```
# unzipping nfl zip data
import zipfile
Dataset = "NFL Play by Play 2009-2017 (v4).csv.zip"
with zipfile.ZipFile("E:/pg/bpb/BPB-Publications/Datasets/" + Dataset, "r") as z:
    z.extractall("E:/pg/bpb/BPB-Publications/Datasets")

# unzipping building permit zip data
import zipfile
Dataset2 = "Building_Permits.csv.zip"
with zipfile.ZipFile("E:/pg/bpb/BPB-Publications/Datasets/" + Dataset2, "r") as z:
    z.extractall("E:/pg/bpb/BPB-Publications/Datasets")

# import required modules
import pandas as pd
import numpy as np

# reading NFL data
nfl_data = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/NFL Play by Play 2009-2017 (v4).csv", low_memory=False)
building_permits = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/Building_Permits.csv", low_memory=False)
```

Gambar 10.1 Unzipping a ZIP file

Setelah membaca data, kita dapat melihatnya untuk mendapatkan gambaran tentang atribut-atributnya. Kita dapat menggunakan fungsi .sample() untuk melihat data secara acak.

# Looking up the data														
nfl_data.sample(5)														
Date	GameID	Drive	qtr	down	time	TimeUnder	TimeSecs	PlayTimeDiff	SideofField	...	yacEPA	Home_WP_pre	...	
219237	2013-12-15	2013121507	20	4	NaN	03:07		4	187.0	0.0	NO	...	NaN	0.955040
113588	2011-11-13	2011111302	20	4	1.0	11:16		12	676.0	1.0	HOU	...	NaN	0.010379
214092	2013-12-05	2013120500	13	3	2.0	14:20		15	1760.0	19.0	HOU	...	0.228558	0.783278
299928	2015-11-26	2015112601	17	4	NaN	04:17		5	257.0	4.0	DAL	...	NaN	NaN
277934	2015-09-27	2015092703	16	3	1.0	11:45		12	1605.0	5.0	TB	...	NaN	0.482456

Gambar 10.2. Viewing the data

building_permits.sample(5)												
	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Constr
106029	201511162641	8	otc alterations permit	11/16/2015	1744	006	1237	NaN	06th	Av	...	
98590	2015091116784	8	otc alterations permit	09/11/2015	0690	116	1	NaN	Daniel Burnham	Ct	...	
45517	201404213721	8	otc alterations permit	04/21/2014	1081	048	2549	NaN	Post	St	...	
136805	201609157776	8	otc alterations permit	09/15/2016	6534	010A	454	NaN	Fair Oaks	St	...	
74940	201502057581	8	otc alterations permit	02/05/2015	3538	040	63	NaN	Noe	St	...	

5 rows x 43 columns

Gambar 10.3. Applying .sample() function in building_permits

Untuk menghitung nilai-nilai null, pandas memiliki fungsi .isnull(). Karena nfl_data memiliki 102 kolom, kita akan menganalisis sepuluh kolom pertama yang berisi nilai-nilai yang hilang.

```
# getting the number of missing values per column
missing_values_count = nfl_data.isnull().sum()
# Looking at first 10 columns missing values in nfl dataset
missing_values_count[0:10]
```

Date	0
GameID	0
Drive	0
qtr	0
down	61154
time	224
TimeUnder	0
TimeSecs	224
PlayTimeDiff	444
SideofField	528
dtype:	int64

Gambar 10.4. Analyzing missing values

Kami ingin menghitung persentase nilai yang hilang dalam dataset, dan untuk itu kami menggunakan kombinasi fungsi .prod() dari NumPy dan fungsi shape dari pandas.

```
# how many total missing values do we have in nfl_data
total_cells = np.prod(nfl_data.shape)
total_missing = missing_values_count.sum()
# percent of data that is missing
(total_missing/total_cells) * 100
```

24.87214126835169

Gambar 10.5. Calculating percentage of values missing

10.4 MENGHAPUS NILAI HILANG

Menghapus nilai yang hilang adalah langkah yang perlu dilakukan jika data memiliki banyak nilai yang hilang. Pandas menyediakan fungsi **dropna()** untuk melakukan ini. Penting untuk berhati-hati saat menggunakan fungsi ini, dan jika Anda tidak memberikan parameter apa pun, semua data dalam baris atau kolom akan dihapus.

```
# remove all columns with at least one missing value
columns_with_na_dropped = nfl_data.dropna(axis=1)
columns_with_na_dropped.head()

# checking how much data did we lose?
print("Columns in original dataset:", nfl_data.shape[1])
print("Columns with missing values dropped:", columns_with_na_dropped.shape[1])

Columns in original dataset: 102
Columns with missing values dropped: 41
```

Gambar 10.6. Dropping missing values

10.5 MENGISI NILAI KOSONG

Kita dapat menggantikan nilai NaN dengan nilai tertentu menggunakan fungsi `fillna()`.

Pada dataset `nfl_data`, kita akan mengganti nilai NaN dengan 0.

```
# get small subset of the NFL dataset
subset_nfl_data = nfl_data.loc[:, 'EPA':'Season'].head()
subset_nfl_data
# replace all NA's with 0
subset_nfl_data.fillna(0)
```

	EPA	airEPA	yacEPA	Home_WP_pre	Away_WP_pre	Home_WP_post	Away_WP_post	Win_Prob	WPA	airWPA	yacWPA	Season
0	2.014474	0.000000	0.000000	0.485675	0.514325	0.546433	0.453567	0.485675	0.060758	0.000000	0.000000	2009
1	0.077907	-1.068169	1.146076	0.546433	0.453567	0.551088	0.448912	0.546433	0.004655	-0.032244	0.036899	2009
2	-1.402760	0.000000	0.000000	0.551088	0.448912	0.510793	0.489207	0.551088	-0.040295	0.000000	0.000000	2009
3	-1.712583	3.318841	-5.031425	0.510793	0.489207	0.461217	0.538783	0.510793	-0.049576	0.106663	-0.156239	2009
4	2.097796	0.000000	0.000000	0.461217	0.538783	0.558929	0.441071	0.461217	0.097712	0.000000	0.000000	2009

Gambar 10.7. Automatically filling missing values

10.6 NORMALISASI DATA

Algoritma machine learning sering memerlukan penskalaan data sebelum analisis. Penskalaan memastikan bahwa atribut numerik berada dalam rentang tertentu agar sesuai dengan machine learning.

```
import pandas as pd
import numpy as np

# for Box-Cox Transformation
from scipy import stats

# for min_max scaling
from mlxtend.preprocessing import minmax_scaling

# for visualization
import seaborn as sns
import matplotlib.pyplot as plt

# reading kickstarters project data
kickstarters_2017 = pd.read_csv("E:/pg/bpb/8PB-Publications/Datasets/ks-projects-201801.csv")
# set seed for reproducibility
np.random.seed(0)
kickstarters_2017.head()
```

Gambar 10.8. min max scaling example

ID	name	category	main_category	currency	deadline	goal	launched	pledged	state	backers	country	usd pledged
0	The Songs of Adelaide & Abdullah	Poetry	Publishing	GBP	2015-10-09	1000.0	2015-08-11 12:12:28	0.0	failed	0	GB	0.0
1	Greeting From Earth: ZGAC Arts Capsule For ET	Narrative Film	Film & Video	USD	2017-11-01	30000.0	2017-09-02 04:43:57	2421.0	failed	15	US	100.0
2	Where is Hank?	Narrative Film	Film & Video	USD	2013-02-26	45000.0	2013-01-12 00:20:50	220.0	failed	3	US	220.0
3	ToshCapital Rekordz Needs Help to Complete Album	Music	Music	USD	2012-04-16	5000.0	2012-03-17 03:24:11	1.0	failed	1	US	1.0
4	Community Film Project: The Art of Neighborhoods	Film & Video	Film & Video	USD	2015-08-29	19500.0	2015-07-04 08:35:03	1283.0	canceled	14	US	1283.0

Gambar 10.9. output of the min max scaling example

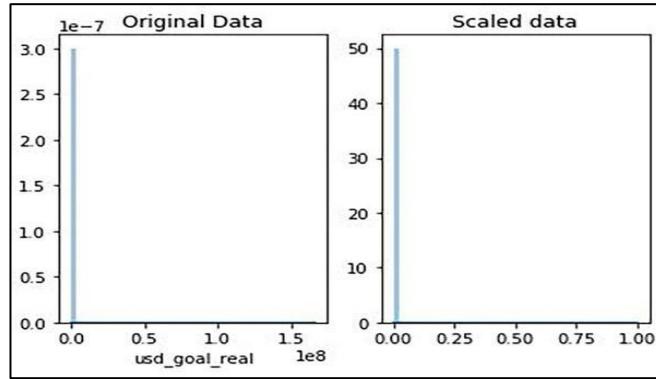
Dengan penskalaan variabel, Anda dapat membandingkan berbagai variable.

```
# select the usd_goal_real column
usd_goal = kickstarters_2017.usd_goal_real

# scale the goals from 0 to 1
scaled_data = minmax_scaling(usd_goal, columns = [0])

# plot the original & scaled data together to compare
fig, ax=plt.subplots(1,2)
sns.distplot(kickstarters_2017.usd_goal_real, ax=ax[0])
ax[0].set_title("Original Data")
sns.distplot(scaled_data, ax=ax[1])
ax[1].set_title("Scaled data")
plt.show()
```

Gambar 10.10. Scaling the goals of each campaign



Gambar 10.11. Plots that are displayed

Metode yang kita gunakan untuk normalisasi disebut Transformasi Box-Cox. Dalam contoh data Kickstarter ini, kita akan mengnormalisasi jumlah uang yang dijanjikan untuk setiap kampanye:

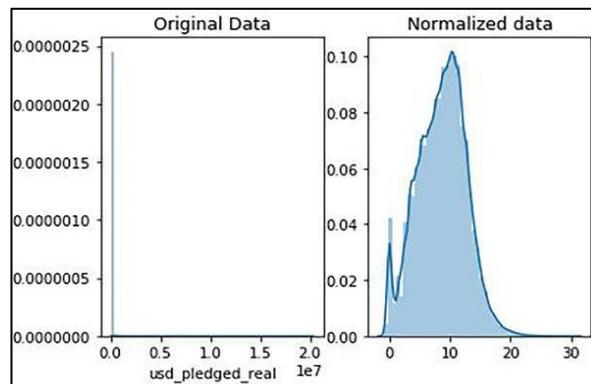
```
# get the index of all positive pledges (Box-Cox only takes positive values)
index_of_positive_pledges = kickstarters_2017.usd_pledged_real > 0

# get only positive pledges (using their indexes)
positive_pledges = kickstarters_2017.usd_pledged_real.loc[index_of_positive_pledges]

# normalize the pledges (w/ Box-Cox)
normalized_pledges = stats.boxcox(positive_pledges)[0]

# plot both together to compare
fig, ax=plt.subplots(1,2)
sns.distplot(positive_pledges, ax=ax[0])
ax[0].set_title("Original Data")
sns.distplot(normalized_pledges, ax=ax[1])
ax[1].set_title("Normalized data")
plt.show()
```

Gambar 10.12. Box-Cox Transformation



Gambar 10.13 Result of Box-Cox Transformation

10.7 PARSE DATES

Anda akan mempelajari cara mengurai data tanggal dalam dataset. Ini penting ketika Anda perlu menangani data dengan kolom tanggal dan melakukan operasi berdasarkan tanggal, seperti pengambilan data transaksional untuk bulan tertentu atau pengecekan tanggal dalam sebulan.

```
import pandas as pd
import numpy as np
import seaborn as sns
import datetime

# read in our data
earthquakes = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/database.csv")
landslides = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/catalog.csv")
volcanos = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/database.csv")

# set seed for reproducibility
np.random.seed(0)
```

Gambar 10.14. Importing required modules

	<code>id</code>	<code>date</code>	<code>time</code>	<code>continent_code</code>	<code>country_name</code>	<code>count</code>
0	34	3/2/07	Night		NaN	United States
1	42	3/22/07	NaN		NaN	United States
2	56	4/6/07	NaN		NaN	United States
3	59	4/14/07	NaN		NaN	Canada
4	61	4/15/07	NaN		NaN	United States

Gambar 10.15. Checking landslides DataFrame using .head() function

```
landslides.info()  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1693 entries, 0 to 1692  
Data columns (total 23 columns):  
id                1693 non-null int64  
date              1690 non-null object  
time              629 non-null object  
continent code    164 non-null object
```

Gambar 10.16. Verifying data types of each column using .info() function

Untuk mengubah tipe data kolom tanggal yang aneh menjadi tipe data tanggal, kita akan menggunakan fungsi pandas `to_datetime()`.

```
# create a new column, date_parsed, with the parsed dates
landslides['date_parsed'] = pd.to_datetime(landslides['date'], format = "%m/%d/%y")
# print the first few rows
landslides['date_parsed'].head()

0    2007-03-02
1    2007-03-22
2    2007-04-06
3    2007-04-14
4    2007-04-15
Name: date_parsed, dtype: datetime64[ns]
```

Gambar 10.17. Parsing date value

10.8 PENGODEAN KARAKTER

Karakter encodings adalah aturan untuk mengonversi data biner menjadi teks. Dalam Python 3, Anda akan menangani string dan bytes. UTF-8 adalah encoding umum, tetapi beberapa data mungkin tidak sesuai, menyebabkan kesalahan saat membaca teks.

```
# read in the file with the encoding detected by chardet
kickstarter_2016 = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/ks-projects-201612.csv",
                                encoding='Windows-1252', low_memory=False)

# Look at the first few lines
kickstarter_2016.head()
```

Gambar 10.18. reading csv data with correct encoding

10.9 CLEANING INCONSISTENT DATA

Penyusunan data ganda dalam dataset bisa berupa entri yang serupa, namun memiliki perbedaan spasi atau huruf besar-kecil, seperti "Karachi" dan "Karachi." Masalah semacam ini perlu diatasi dengan menghapus ketidak konsistensian tersebut.

# read in our dat	suicide_attacks = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/PakistanSuicideAttacks Ver 11 (30-November-2017).csv", encoding='Windows-1252')	suicide_attacks.head()														
S#	Date	Islamic Date	Blast Day Type	Holiday Type	Time	City	Latitude	Longitude	Province	...	Targeted Sect if any	Killed Min	Killed Max	Injured Min	Injured Max	N Su B
0	Sunday-November 19-1995	25 Jumaada al-Thaany 1416 A.H	Holiday	Weekend	NaN	Islamabad	33.7180	73.0718	Capital	...	None	14.0	15.0	NaN	60	
1	Monday-November 6-2000	10 SH'aaban 1421 A.H	Working Day	NaN	NaN	Karachi	24.9918	66.9911	Sindh	...	None	NaN	3.0	NaN	3	

Gambar 10.19. Cleaning inconsistent data

```
# get all the unique values in the 'City' column
cities = suicide_attacks['City'].unique()

# sort them alphabetically and then take a closer look
cities.sort()
cities

array(['ATTOCK', 'Attock ', 'Bajaur Agency', 'Bannu', 'Bhakkar ', 'Buner',
       'Chakwal ', 'Chaman', 'Charsadda', 'Charsadda ', 'D. I Khan',
       'D.G Khan', 'D.G Khan ', 'D.I Khan', 'D.I Khan ', 'Dara Adam Khel',
       'Dara Adam khel', 'Fateh Jang', 'Ghallanai, Mohmand Agency ',
       'Gujrat', 'Hangu', 'Haripur', 'Hayatabad', 'Islamabad',
       'Islamabad ', 'Jacobabad', 'KURRAM AGENCY', 'Karachi', 'Karachi '],
```

Gambar 10.20. Cleaning inconsistent data in City column

Untuk membersihkan data yang tidak konsisten, Anda dapat mengubah semua huruf menjadi huruf kecil dan menghapus spasi putih di awal dan akhir sel, yang akan membantu mengatasi sebagian besar masalah ketidaksesuaian dalam data teks.

```
# convert to lower case
suicide_attacks['City'] = suicide_attacks['City'].str.lower()
# remove trailing white spaces
suicide_attacks['City'] = suicide_attacks['City'].str.strip()
```

Gambar 10.21. Fixing inconsistencies using str module functions

10.10 KESIMPULAN

Anda akan memiliki pemahaman praktis tentang proses pembersihan data. Dengan berlatih menggunakan teknik-teknik yang diajarkan dalam bab ini pada berbagai dataset, Anda akan memperoleh keterampilan yang kompetitif dalam peran ilmu data. Tetap berlatih dan eksplorasi lebih banyak teknik untuk terus memajukan diri dalam ilmu data. Bab berikutnya akan membahas visualisasi secara rinci.

BAB XI

VISUALISASI DATA

11.1 STRUKTUR

Data adalah alat yang kuat. Untuk memahami data besar, kita perlu visualisasi. Ini adalah keterampilan inti dalam ilmu data. Dalam bab ini, kita akan mempelajari berbagai jenis grafik dengan Python. Bab ini mempelajari tentang:

1. Bar chart
2. Line chart
3. Histograms
4. Scatter plot
5. Stacked plot
6. Box plot Objective

11.2 TUJUAN

Anda akan memperoleh pengetahuan dan keterampilan yang luas dalam bidang visualisasi data menggunakan

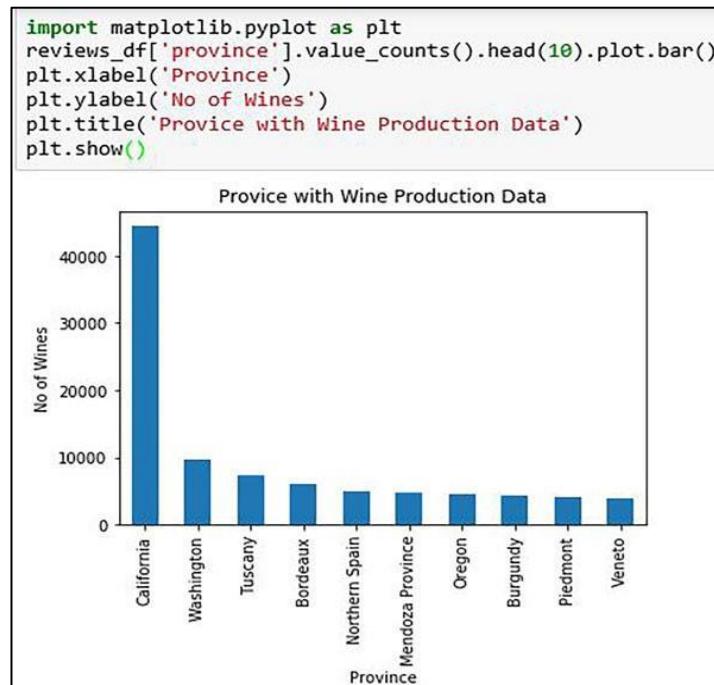
11.3 BAR CHART

Grafik batang adalah metode visualisasi sederhana yang digunakan untuk membandingkan kelompok atau kategori dengan nilai numerik. Dalam bab ini, Anda akan belajar cara membuat grafik batang menggunakan dataset Tinjauan Poin Anggur, yang membandingkan wilayah produsen anggur dengan jumlah label anggur yang dihasilkan.

```
import pandas as pd
reviews_df = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/winemag-data_first150k.csv", index_col=0)
reviews_df.head(5)
```

	country	description	designation	points	price	province	region_1	region_2	variety	winery
0	US	This tremendous 100% varietal wine hails from ...	Martha's Vineyard	96	235.0	California	Napa Valley	Napa	Cabernet Sauvignon	Heitz
1	Spain	Ripe aromas of fig, blackberry and cassis are ...	Carodorum Selección Especial Reserva	96	110.0	Northern Spain	Toro	NaN	Tinta de Toro	Bodega Carmen Rodriguez
2	US	Mac Watson honors the memory of a wine once ma...	Special Selected Late Harvest	96	90.0	California	Knights Valley	Sonoma	Sauvignon Blanc	Macaulay
3	US	This spent 20 months in 30% new French oak, an...	Reserve	96	65.0	Oregon	Willamette Valley	Willamette Valley	Pinot Noir	Ponzi

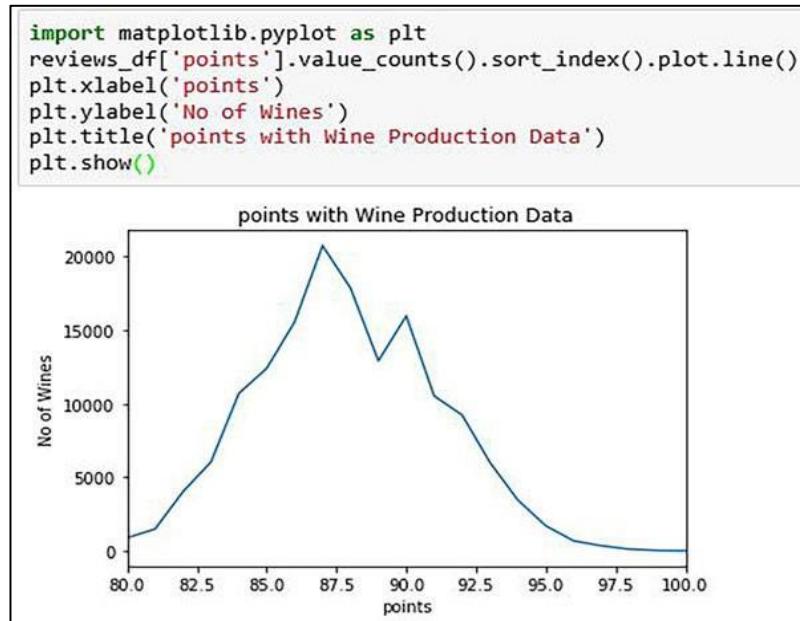
Gambar 11.1. Loading the wine review dataset into a DataFrame



Gambar 11.2. Provinces with Wine Production Data

11.4 LINE CHART

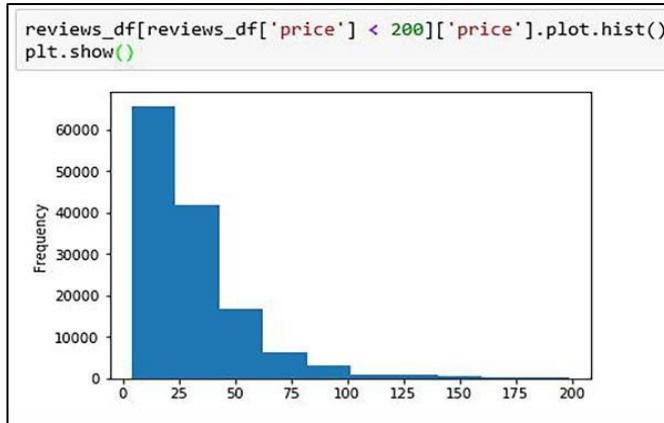
Grafik garis digunakan untuk menunjukkan perubahan skor ulasan anggur dari waktu ke waktu



Gambar 11.3. Wine Review Points using plot.line() function

11.5 HISTOGRAMS

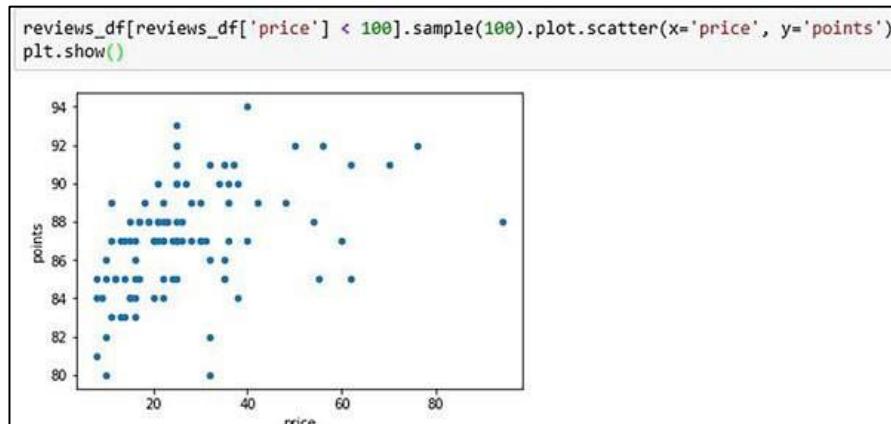
Histogram adalah grafik batang yang membagi data ke dalam interval-interval yang sama dan menunjukkan jumlah data dalam setiap interval tersebut. Histogram kurang cocok untuk data yang cenderung condong ke salah satu sisi.



Gambar 11.4. Number of Wines priced below \$200

11.6 SCATTER PLOT

Scatter plot adalah grafik dua dimensi yang memetakan dua variabel numerik ke dalam ruang dua dimensi. Ini digunakan untuk melihat hubungan antara dua variabel numerik. Dalam dataset anggur kita, kita akan memeriksa hubungan antara harga dan melakukan scatter plot dengan harga di bawah \$100.



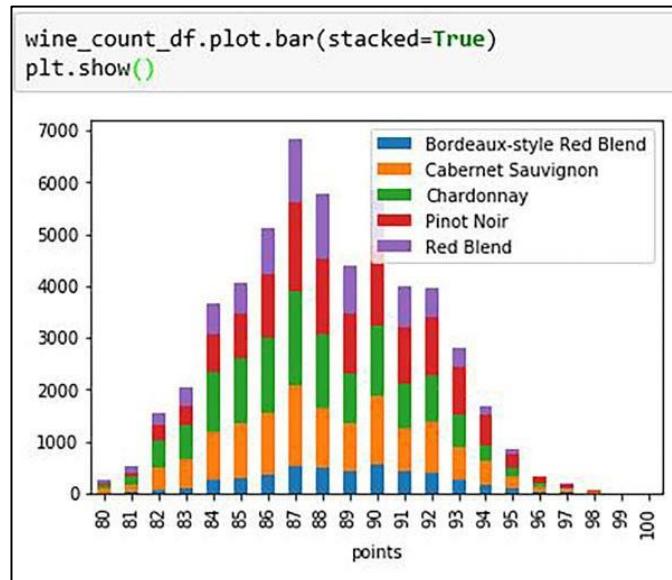
Gambar 11.5. Plotting relationship using scatter() function

11.7 STACKED PLOT

Grafik bertumpuk adalah jenis grafik yang memplot variabel satu di atas yang lain. Ini mirip dengan grafik batang atau grafik garis, tetapi dibagi menjadi komponennya sehingga perbandingan serta totalnya dapat terlihat.

points	Bordeaux-style Red Blend	Cabernet Sauvignon	Chardonnay	Pinot Noir	Red Blend
80	5.0	87.0	68.0	36.0	72.0
81	18.0	159.0	150.0	83.0	107.0
82	72.0	435.0	517.0	295.0	223.0
83	95.0	570.0	669.0	346.0	364.0
84	268.0	923.0	1146.0	733.0	602.0

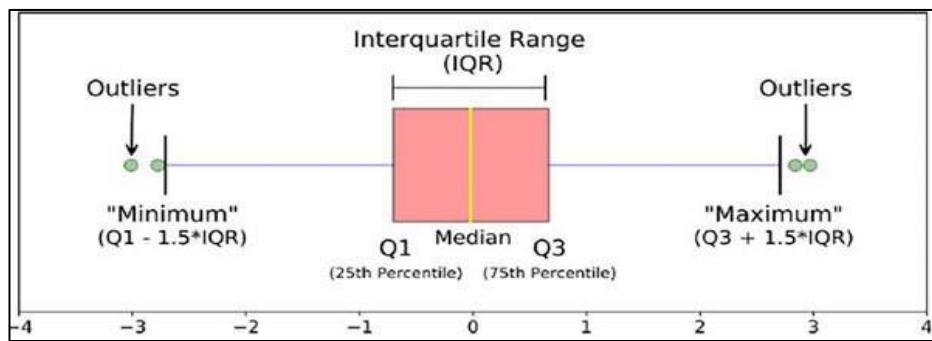
Gambar 11.6. Stacked plot



Gambar 11.7. Plotting a stacked plot

11.8 BOX PLOT

Box plot adalah visualisasi statistik yang berguna untuk melihat distribusi data. Kotak di dalamnya menunjukkan kuartil pertama (25%) hingga ketiga (75%), dengan garis di tengahnya sebagai median. Garis di luar kotak, atau "whiskers," menunjukkan rentang data.



Gambar 11.8. Box plot

Median (Q2/50th percentile) adalah nilai tengah dalam dataset. Kuartil pertama (Q1/25th percentile) adalah nilai tengah antara nilai terkecil dan median dataset. Kuartil ketiga (Q3/75th percentile) adalah nilai tengah antara median dan nilai tertinggi dataset. Rentang interkuartil adalah perbedaan antara kuartil ke-25 dan ke-75. Whisker (ditunjukkan dalam biru) adalah garis yang menghubungkan kotak dengan data yang berada di luar rentang interkuartil, menunjukkan sebaran data. Data yang berada di luar batas-batas adalah data yang kurang dari $Q1 - 1.5 * IQR$ atau lebih dari $Q3 + 1.5 * IQR$. Mari lihat cara menggunakan box plot pada dataset dunia nyata -

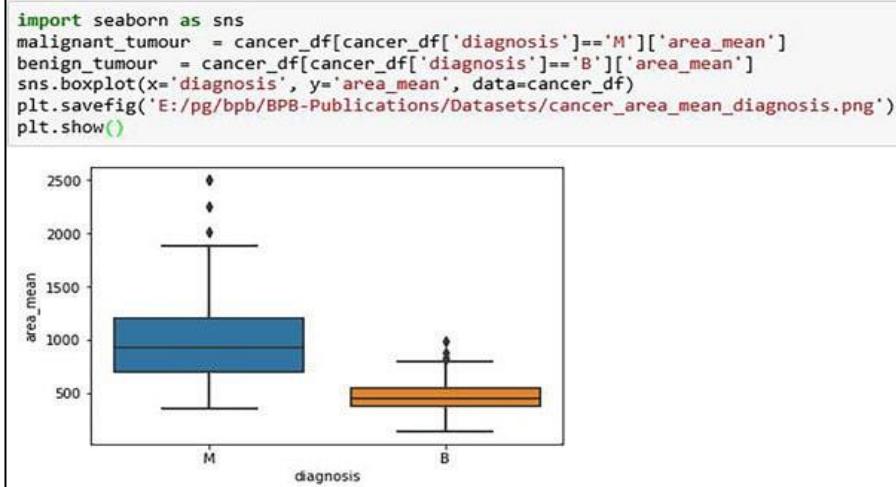
Diagnosis Kanker Payudara, yang dapat Anda unduh dari repositori kami dan baca seperti yang ditunjukkan pada gambar berikut.

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

5 rows × 33 columns

Gambar 11.9. Utilizing a Box plot

Tugas selanjutnya adalah menganalisis hubungan antara tumor ganas atau jinak (fitur kategorikal) dan area_mean (fitur kontinu).



Gambar 11.10. Plotting box plot using function from seaborn library

11.9 KESIMPULAN

Visualisasi data adalah keterampilan inti dalam ilmu data. Ini penting untuk memahami dataset dan membangun model yang berguna. Dengan menguasai keterampilan visualisasi, Anda dapat mengungkap wawasan berharga dari data Anda.

BAB XII

DATA PRE-PROCESSING

12.1 STRUKTUR

Dalam bab ini, Anda akan belajar tentang langkah-langkah pra-pemrosesan data yang penting untuk mempersiapkan data sebelum digunakan dalam proyek machine learning. Ini melibatkan pembersihan data, rekayasa fitur, dan visualisasi, serta dapat membantu Anda memilih algoritma yang tepat untuk masalah data Anda. Langkah-langkah pre-processing:

1. Studi Kasus
2. Mengimpor dataset.
3. Analisis data eksploratori.
4. Pembersihan dan pra-pemrosesan data.
5. Rekayasa fitur.

12.2 TUJUAN

Setelah mempelajari bab ini, Anda akan memiliki keterampilan untuk mempersiapkan data Anda sehingga siap digunakan dalam proyek machine learning.

12.3 STUDI KASUS

Diabetes adalah suatu kondisi kesehatan yang memengaruhi cara tubuh Anda mengubah makanan menjadi energi. Sebagian besar makanan yang Anda makan dipecah menjadi gula (juga disebut glukosa) dan dilepaskan ke aliran darah Anda. Ketika gula darah Anda naik, ini memberi sinyal pada pankreas Anda untuk melepaskan insulin. Tanpa penanganan yang cermat dan berkelanjutan, diabetes dapat menyebabkan penumpukan gula dalam darah, yang dapat meningkatkan risiko komplikasi berbahaya, termasuk stroke dan penyakit jantung. Sehingga saya memutuskan untuk memprediksi menggunakan Machine Learning dengan Python. Dalam analisis kali ini, kita akan mencari informasi tentang Pima Indians dan Penyakit Diabetes yang ada disana menggunakan data yang didapat dari *Kaggle*.

12.4 IMPORT LIBRARY

Pada langkah pertama ini saya telah mengimpor perpustakaan paling umum yang digunakan dalam python untuk pembelajaran mesin seperti Pandas, Seaborn, Matplotlib dan lainnya.

```
# Import libraries  
import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
from collections import Counter  
import os
```

Gambar 12.1 *Import Library*

12.5 IMPORT DATASET

```
# Import dataset  
df = pd.read_csv("C:/Users/SESUAIKAN DENGAN NAMA PC/Praktikum/diabetes.csv")  
# Get familiar with dataset structure  
df.info()
```

Gambar 12.2 *Import Dataset*

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   Pregnancies      768 non-null    int64  
 1   Glucose          768 non-null    int64  
 2   BloodPressure    768 non-null    int64  
 3   SkinThickness    768 non-null    int64  
 4   Insulin          768 non-null    int64  
 5   BMI              768 non-null    float64 
 6   DiabetesPedigreeFunction 768 non-null    float64 
 7   Age              768 non-null    int64  
 8   Outcome          768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 kB

```

Gambar 12.3. Menampilkan Atribut Dataset

```

# Show top 5 rows
df.head()

```

Gambar 12.4. Menampilkan Data Teratas

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6		0.627	50
1	1	85	66	29	0	26.6		0.351	31
2	8	183	64	0	0	23.3		0.672	32
3	1	89	66	23	94	28.1		0.167	21
4	0	137	40	35	168	43.1		2.288	33

Gambar 12.5. Hasil Data Teratas

```

# Explore missing values
df.isnull().sum()

```

Gambar 12.5. Mencari Nilai Hilang

Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0
dtype:	int64

Gambar 12.6. Hasil Pencarian Data Hilang

Analisa mendeteksi tidak ada nilai yang hilang dalam dataset namun fitur seperti Glukosa, Tekanan Darah, Insulin, Ketebalan Kulit memiliki nilai 0 yang tidak mungkin. Kita harus mengganti nilai 0 dengan nilai rata-rata atau median kolom tertentu.

```

df['Glucose'] = df['Glucose'].replace(0, df['Glucose'].mean())
# Correcting missing values in blood pressure
df['BloodPressure'] = df['BloodPressure'].replace(0, df['BloodPressure'].mean())
# There are 35 records with 0 BloodPressure in dataset
# Correcting missing values in BMI
df['BMI'] = df['BMI'].replace(0, df['BMI'].median())

```

```
# Correct missing values in Insulin and SkinThickness
df['SkinThickness'] = df['SkinThickness'].replace(0, df['SkinThickness'].median())
df['Insulin'] = df['Insulin'].replace(0, df['Insulin'].median())
# Review dataset statistics
df.describe()
```

Gambar 12.7. Mengkoreksi Data Hilang

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	121.681605	72.254807	27.334635	94.652344	32.450911	0.471876	33.240885	0.348958
std	3.369578	30.436016	12.115932	9.229014	105.547598	6.875366	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.750000	64.000000	23.000000	30.500000	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	31.250000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Gambar 12.8. Hasil Setelah Koreksi

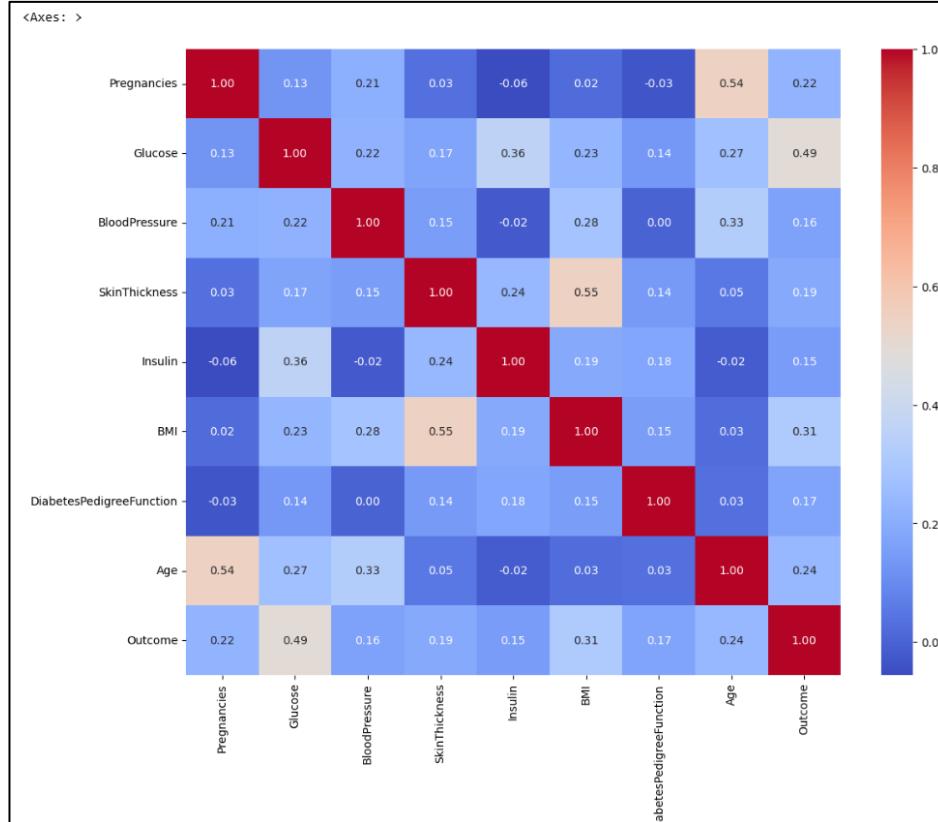
12.6 ANALISIS DATA

KORELASI

Korelasi adalah satu atau lebih variabel yang berhubungan satu sama lain. Ini juga membantu untuk menemukan pentingnya fitur dan membersihkan dataset sebelum memulai Pemodelan.

```
plt.figure(figsize=(13,10))
sns.heatmap(df.corr(), annot=True, fmt = ".2f", cmap = "coolwarm")
```

Gambar 12.9. Korelasi Data



Gambar 12.10. Heatmap Korelasi Data

Menurut pengamatan, fitur seperti Kehamilan, Glukosa, BMI, dan Usia lebih berkorelasi dengan Hasil. Pada langkah berikutnya, saya memamerkan representasi detail dari fitur-fitur ini.

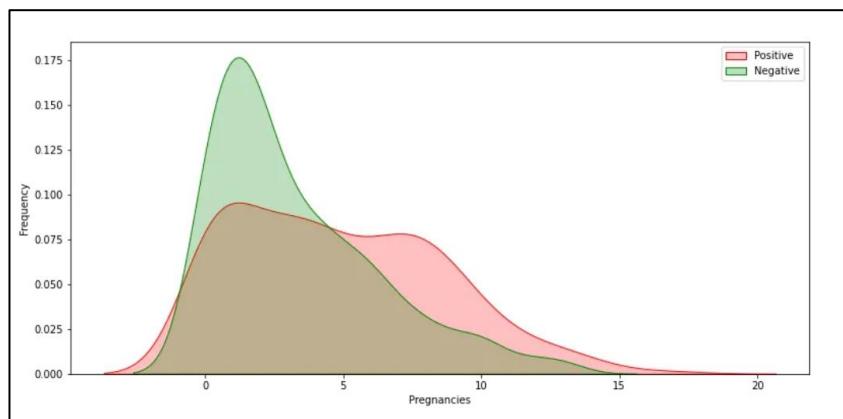
EKSPLORASI VARIABEL I

Wanita dengan diabetes dapat dan memang memiliki kehamilan yang sehat dan bayi yang sehat. Mengelola diabetes dapat membantu mengurangi risiko komplikasi. Diabetes yang tidak

diobati meningkatkan risiko komplikasi kehamilan, seperti tekanan darah tinggi, depresi, kelahiran prematur, cacat lahir dan keguguran.

```
# Explore Pregnancies vs Outcome  
plt.figure(figsize=(13,6))  
g = sns.kdeplot(df["Pregnancies"][df["Outcome"] == 1],  
                 color="Red", shade = True)  
g = sns.kdeplot(df["Pregnancies"][df["Outcome"] == 0],  
                 ax=g, color="Green", shade= True)  
g.set_xlabel("Pregnancies")  
g.set_ylabel("Frequency")  
g.legend(["Positive","Negative"])
```

Gambar 12.11. Eksplorasi Variebel 1



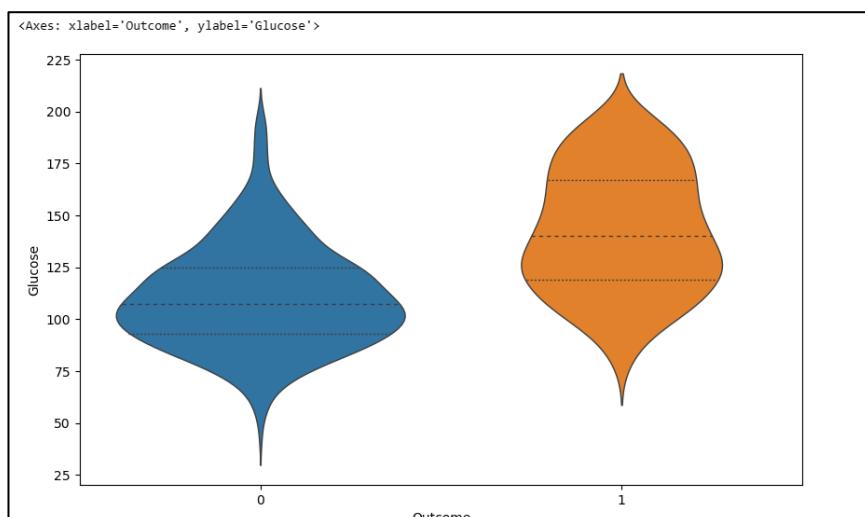
Gambar 12.12. Plot Ekspolrasi Variabel 1

Hasil memiliki nilai 1 dan 0 di mana 1 menunjukkan bahwa orang tersebut menderita diabetes dan 0 menunjukkan orang tersebut tidak menderita diabetes. Ini adalah kolom label saya di himpunan data.

EKSPLORASI VARIABEL 2

```
# Explore Glucose vs Outcome  
plt.figure(figsize=(10,6))  
sns.violinplot(data=df, x="Outcome", y="Glucose",  
                split=True, inner="quart", linewidth=1)
```

Gambar 12.13. Eksplorasi Variabel 2



Gambar 12.14. Quart Plot Variabel 2

Kemungkinan diabetes secara bertahap meningkat dengan tingkat glukosa.

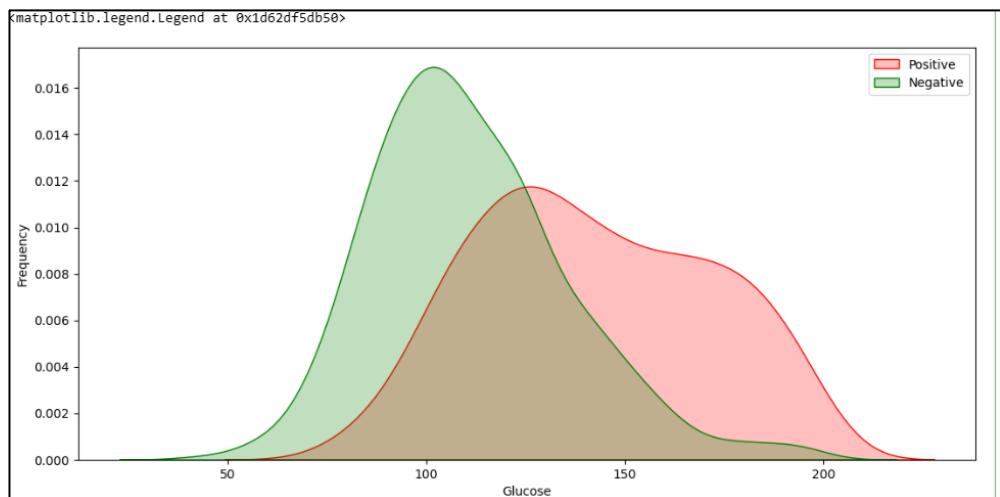
```
# Explore Glucose vs Outcome
```

```

plt.figure(figsize=(13,6))
g = sns.kdeplot(df["Glucose"][df["Outcome"] == 1], color="Red", shade=True)
g = sns.kdeplot(df["Glucose"][df["Outcome"] == 0], ax=g, color="Green", shade=True)
g.set_xlabel("Glucose")
g.set_ylabel("Frequency")
g.legend(["Positive","Negative"])

```

Gambar 12.15. Frekuensi Variabel 2



Gambar 12.15. Plot Frekuensi Variabel 2

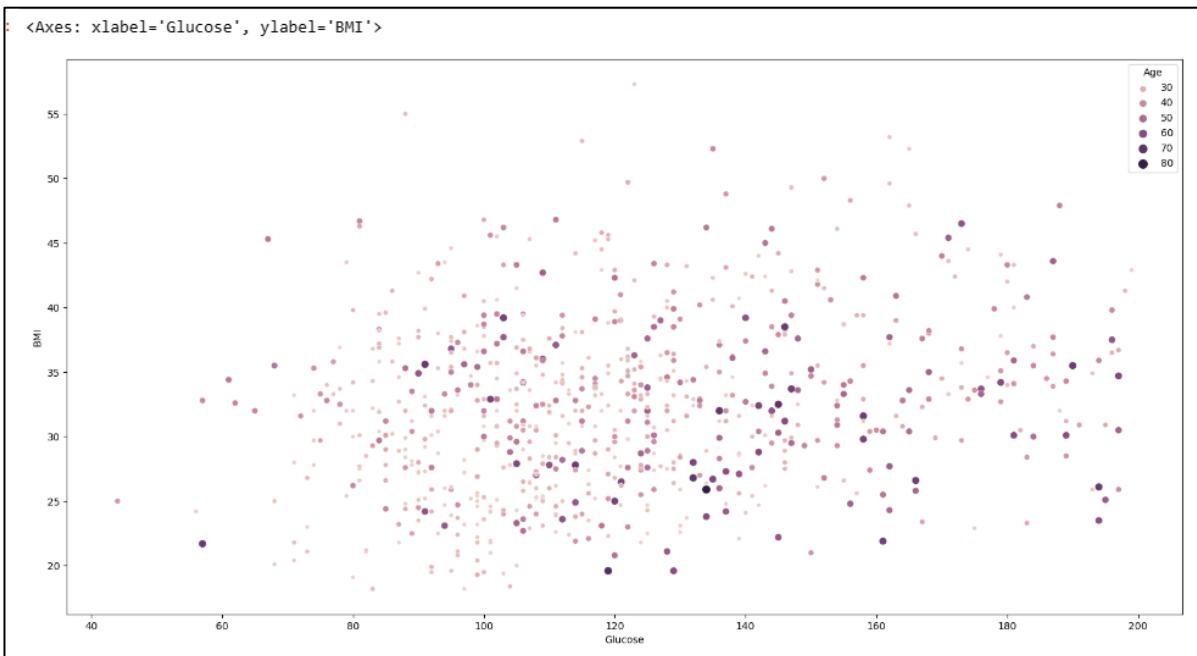
EKSPLORASI VARIABEL 2,3,4

```

# Glucose vs BMI vs Age
plt.figure(figsize=(20,10))
sns.scatterplot(data=df, x="Glucose", y="BMI", hue="Age", size="Age")

```

Gambar 12.16. Eksplorasi Variabel 2,3,4



Gambar 12.17. Scatter Plot Variabel 2,3,4

12.7 DATA PREPROCESSING

FEATURE ENGINEERING

```

def detect_outliers(df,n,features):
    outlier_indices = []
    """
    Detect outliers from given list of features. It returns a list of the indices

```

```

according to the observations containing more than n outliers according
to the Tukey method
"""

# iterate over features(columns)
for col in features:
    Q1 = np.percentile(df[col], 25)
    Q3 = np.percentile(df[col],75)
    IQR = Q3 - Q1
    # outlier step
    outlier_step = 1.5 * IQR
    # Determine a list of indices of outliers for feature col
    outlier_list_col = df[(df[col] < Q1 - outlier_step) | (df[col] > Q3 + outlier_step )].index
    # append the found outlier indices for col to the list of outlier indices
    outlier_indices.extend(outlier_list_col)
    # select observations containing more than 2 outliers
    outlier_indices = Counter(outlier_indices)
    multiple_outliers = list( k for k, v in outlier_indices.items() if v > n )
return multiple_outliers

# detect outliers from numeric features
outliers_to_drop = detect_outliers(df, 2 ,["Pregnancies", 'Glucose', 'BloodPressure', 'BMI',
'DiabetesPedigreeFunction', 'SkinThickness', 'Insulin', 'Age'])

```

Gambar 12.18. Feature Engineering

Sampai tahap ini, eksplorasi dataset melakukan koreksi nilai yang hilang dan visualisasi data. Selanjutnya, memulai rekayasa fitur. Rekayasa fitur berguna untuk meningkatkan kinerja algoritma pembelajaran mesin dan sering dianggap sebagai pembelajaran mesin terapan. Memilih fitur penting dan mengurangi ukuran set fitur membuat komputasi dalam pembelajaran mesin dan algoritma analitik data lebih layak.

■ OUTLIER DETECTION

Pada bagian ini akan menghapus semua catatan yang diuraikan dalam dataset. Outlier berdampak pada akurasi Model. Saya menggunakan *Metode Tukey* yang digunakan untuk deteksi penculan.

```

def detect_outliers(df,n,features):
outlier_indices = []
"""

Detect outliers from given list of features. It returns a list of the indices
according to the observations containing more than n outliers according
to the Tukey method

"""

# iterate over features(columns)
for col in features:
    Q1 = np.percentile(df[col], 25)
    Q3 = np.percentile(df[col],75)
    IQR = Q3 - Q1

```

```

# outlier step
outlier_step = 1.5 * IQR

# Determine a list of indices of outliers for feature col
outlier_list_col = df[(df[col] < Q1 - outlier_step) | (df[col] > Q3 + outlier_step )].index

# append the found outlier indices for col to the list of outlier indices
outlier_indices.extend(outlier_list_col)

# select observations containing more than 2 outliers
outlier_indices = Counter(outlier_indices)

multiple_outliers = list( k for k, v in outlier_indices.items() if v > n )

return multiple_outliers

# detect outliers from numeric features
outliers_to_drop = detect_outliers(df, 2 ,["Pregnancies", 'Glucose', 'BloodPressure', 'BMI',
'DiabetesPedigreeFunction', 'SkinThickness', 'Insulin', 'Age'])

```

Gambar 12.20. Turkey Method

Di sini menemukan pencilan dari semua fitur seperti Kehamilan, Glukosa, Tekanan Darah, BMI, DiabetesPedigreeFunction, SkinThickness, Insulin, dan Usia.

```
df.drop(df.loc[outliers_to_drop].index, inplace=True)
```

Gambar 12.21. Menghapus Pencilan

Kode tersebut telah berhasil menghapus semua pencilan dari dataset sekarang. Langkah selanjutnya adalah membagi dataset dalam melatih dan menguji dan melanjutkan pemodelan.

12.8 KESIMPULAN

Feature engineering dan visualisasi merupakan keterampilan yang sangat berpengaruh dalam analisis data.

BAB XIII

SUPERVISED MACHINE LEARNING

13.1 STRUKTUR

Bab ini membawa Anda ke dunia machine learning (ML), yang mengajarkan mesin dan komputer untuk belajar dari data yang ada dan membuat prediksi pada data baru tanpa pemrograman eksplisit. Anda akan mempelajari berbagai jenis ML, fokus pada supervised learning, dan cara menggunakan Python dengan scikit-learn untuk membangun model prediksi. Anda juga akan belajar mengatur parameter model dan mengukur kinerjanya pada data baru. Semua ini akan diilustrasikan dengan dataset dunia nyata. Bab ini menjelaskan tentang:

1. Pengenalan tentang Machine Learning (ML)
2. Daftar Algoritma Machine Learning Umum
3. Dasar-dasar Supervised Machine Learning
4. Menyelesaikan Masalah Klasifikasi dalam ML
5. Menyelesaikan Masalah Regresi dalam ML
6. Cara Menyetel Model ML Anda
7. Bagaimana Mengatasi Variabel Kategorikal dalam Sklearn
8. Teknik Lanjutan untuk Mengatasi Data yang Hilang

13.2 TUJUAN

Anda akan memiliki kemampuan yang lebih mendalam dalam mengatasi berbagai masalah dalam Supervised Machine Learning. Anda akan memahami istilah-istilah dasar, prinsip dasar pembelajaran mesin, berbagai algoritma umum dalam machine learning, serta teknik dasar dalam Supervised Machine Learning seperti klasifikasi dan regresi.

13.3 DAFTAR ISTILAH

- **Training** adalah subset dari dataset yang digunakan untuk melatih sebuah model.
- **Validasi** adalah subset dari dataset yang berbeda dari dataset pelatihan yang digunakan untuk menyesuaikan hyperparameter.
- **Testing** adalah subset dari dataset yang digunakan untuk menguji model Anda setelah model melalui tahap awal oleh dataset validasi.
- **Hyperparameter** adalah parameter yang nilainya ditetapkan sebelum melatih model pembelajaran mesin atau deep learning. Berbagai model memerlukan hyperparameter yang berbeda dan beberapa tidak memerlukan.
- **Feature** adalah variabel dari sebuah model yang sistem ML pelajari dengan sendirinya.
- **Feature** adalah variabel input individu yang bertindak sebagai input dalam sistem Anda.
- **Label** adalah keluaran akhir.

13.4 MACHINE LEARNING (ML)

Algoritma machine learning dibagi menjadi empat kategori berdasarkan tujuannya:

■ UNSUPERVISED LEARNING

Unsupervised ML digunakan saat komputer dilatih dengan data yang tidak berlabel, yang berarti data pelatihan tidak memiliki target atau kita tidak memberi tahu sistem harus mencapai apa. Sistem ini akan memahami sendiri dari data yang diberikan.

SEMI-SUPERVISED LEARNING

Semi-supervised ML adalah metode di antara supervised dan unsupervised learning yang cocok saat label data mahal.

REINFORCEMENT LEARNING

Reinforcement ML memungkinkan mesin/agen perangkat lunak belajar perilaku ideal dalam konteks tertentu. Hal ini digunakan dalam aplikasi seperti permainan papan komputer, robotik, dan mobil otonom.

DAFTAR ALGORITMA

Berikut adalah daftar algoritma Machine Learning yang perlu diketahui:

1. Regresi Linear
2. Regresi Logistik
3. Pohon Keputusan
4. SVM (Support Vector Machine)
5. Naive Bayes
6. k-Nearest Neighbors (k-NN)
7. K-Means
8. Random Forest
9. Algoritma Reduksi Dimensi
10. Algoritma Peningkatan Gradien
11. GBM (Gradient Boosting Machine)
12. XGBoost
13. LightGBM
14. CatBoost

SUPERVISED LEARNING

Algoritma pembelajaran mesin terawasi (Supervised ML) mencoba memodelkan hubungan dan ketergantungan antara keluaran prediksi target dan fitur-fitur input sehingga kita dapat memprediksi nilai keluaran untuk data baru berdasarkan hubungan tersebut yang dipelajari dari kumpulan data asli. Algoritma pembelajaran mesin terawasi dibagi menjadi dua kategori berdasarkan dua jenis masalah:

1. Masalah klasifikasi (Classification problem) dapat didefinisikan sebagai masalah yang menghasilkan variabel keluaran yang hanya berada dalam kategori-kategori tertentu.
2. Masalah regresi (Regression problem) terjadi ketika variabel keluaran adalah nilai nyata, seperti "dolar", "Rupee", atau bisa juga "berat".
- 3.

SUPERVISED ML FUNDAMENTALS

Supervised ML melibatkan pelatihan komputer dengan data input dan output yang benar. Ini digunakan baik untuk masalah klasifikasi dengan output berupa label atau masalah regresi dengan output berupa angka kontinu.

KLASIFIKASI

Multi-label classification adalah ketika ada lebih dari satu label atau output yang mungkin untuk prediksi. Ini berguna untuk segmentasi pelanggan, kategorisasi audio dan gambar, serta analisis teks untuk mengekstrak sentimen pelanggan. Ini adalah beberapa algoritma Machine Learning yang umum digunakan:

1. Logistic Regression: Digunakan untuk masalah klasifikasi di mana variabel target adalah biner.
2. Decision Tree Classifier: Menggunakan serangkaian pertanyaan dan kondisi untuk mengelompokkan data ke dalam kelas yang berbeda.
3. K-Nearest Neighbor Classifier: Menggunakan sejumlah tetangga terdekat dalam data pelatihan untuk menentukan kelas dari data baru.
4. Linear Discriminant Analysis (LDA): Digunakan sebagai teknik reduksi dimensi dan juga sebagai algoritma klasifikasi.
5. Gaussian Naive Bayes Classifier: Digunakan ketika fitur memiliki nilai kontinu dan mengikuti distribusi Gaussian.
6. Support Vector Classifier (SVM): Menggunakan hiperplane untuk memisahkan dua kelas.

MEMECAHKAN MASALAH KLASIFIKASI

Untuk menyelesaikan masalah dalam Machine Learning berbasis supervisi, Anda memerlukan data yang sudah dilabeli. Data tersebut bisa diperoleh dari data historis yang sudah dilabeli, eksperimen seperti A/B testing, atau melalui crowdsourcing. Tujuannya adalah belajar dari data tersebut dan membuat prediksi pada data baru berdasarkan pembelajaran sebelumnya. Dalam contoh berikutnya, kita akan menggunakan perpustakaan scikit-learn atau sklearn di Python untuk memecahkan masalah klasifikasi. Terdapat juga perpustakaan lain seperti TensorFlow dan Keras yang umum digunakan dalam Machine Learning.

CONTOH

```
from sklearn import datasets
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')

#Load the iris dataset
iris = datasets.load_iris()
```

Gambar 13.1. Loading dataset

```
print(type(iris))
<class 'sklearn.utils.Bunch'>
```

Gambar 13.2. Checking type of dataset

```
print(iris.keys())
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names'])
```

Gambar 13.3. Printing keys with .keys() function

```
#check rows(samples) and columns(features) in iris data
iris.data.shape
(150, 4)
```

Gambar 13.4. Diagnosis of the data key

```
#check target variables
iris.target_names
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

Gambar 13.5. Values associated with target variable

```

In [24]: iris['target'].shape
Out[24]: (150,)

In [25]: # applying knn algorithm
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(iris['data'],iris['target'])

Out[25]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_neighbors=6, p=2,
weights='uniform')

In [26]: # Predict the labels for the training data X: y_pred
y_pred = knn.predict(iris['data'])

In [27]: # Predict and print the label for the new data
import numpy as np
X_new = np.array([[5, 2, 4, 0.2], [ 4.7, 3, 1.3, 0.2 ]])
new_prediction = knn.predict(X_new)
print("Prediction: {}".format(new_prediction))

Prediction: [1 0]

```

Gambar 13.6.

Selanjutnya, kita akan menyimpan panjang/lebar tanaman iris dan spesiesnya dalam variabel terpisah dan mengonversi data iris ke dalam Pandas DataFrame.

```

X = iris.data
y = iris.target
#converting data in Pandas Dataframe
iris_df = pd.DataFrame(X, columns=iris.feature_names)
#check first five rows of iris dataframe
print(iris_df.head())

```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

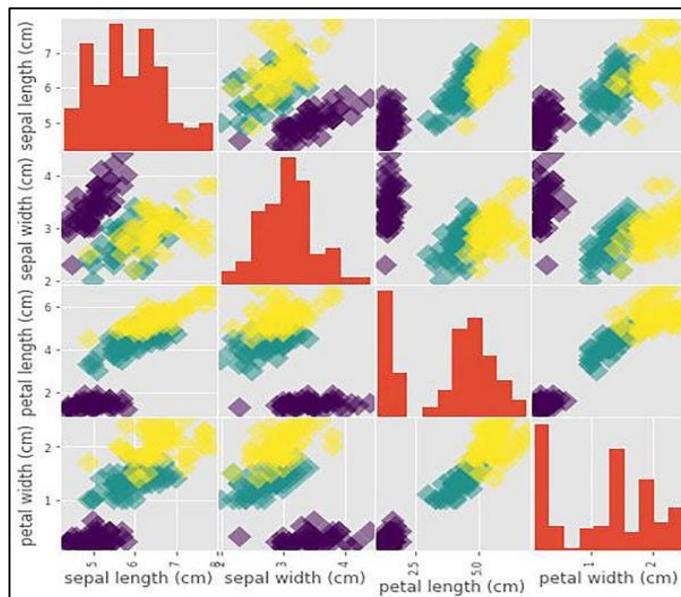
Gambar 13.7. Converting iris data

```

#plotting histogram of features
_= pd.plotting.scatter_matrix(iris_df, c=y, figsize=[8,8], s=150, marker='D')
plt.show()

```

Gambar 13.8. Plotting histogram of features



Gambar 13.9. Scatter plot matrix of Iris dataframe

Selanjutnya, kita akan membagi dataset menjadi dua bagian: 80% untuk pelatihan model dan 20% untuk validasi.

```

validation_size = 0.20
seed = 7
from sklearn import model_selection
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, y,
test_size=validation_size,
random_state=seed)

```

Gambar 13.10. Splitting a dataset

TRAIN/TEST dan CROSS VALIDATION

Overfitting terjadi ketika model terlalu cocok dengan data pelatihan, sedangkan underfitting terjadi ketika model terlalu sederhana. Train/test split dan cross-validation membantu menghindari overfitting. Akurasi digunakan untuk mengevaluasi performa model.

```
seed = 7
scoring = 'accuracy'
```

Gambar 13.11. Using scoring variable

Untuk mengevaluasi berbagai algoritma klasifikasi, seperti Regresi Logistik, Analisis Diskriminan Linear, K-Nearest Neighbors, Decision Trees Classifier, Gaussian Naive Bayes, dan Support Vector Classifier, digunakan loop for.

```
#Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

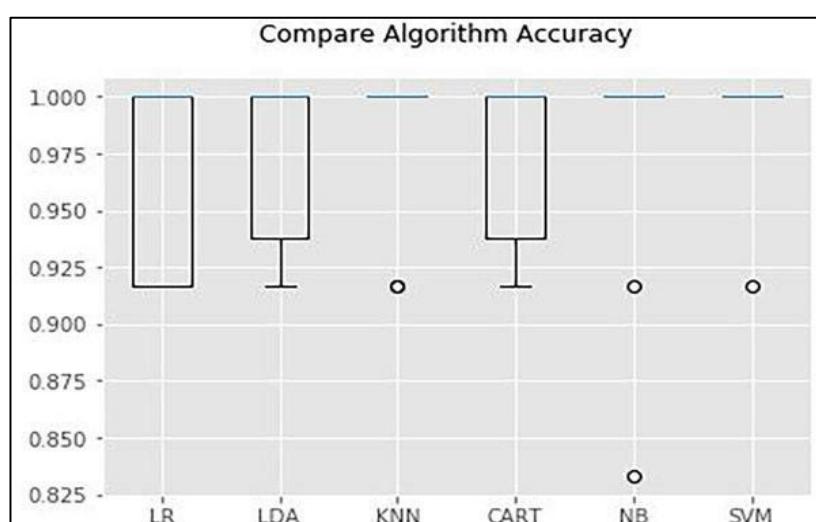
LR: 0.966667 (0.040825)
LDA: 0.975000 (0.038188)
KNN: 0.983333 (0.033333)
CART: 0.975000 (0.038188)
NB: 0.975000 (0.053359)
SVM: 0.991667 (0.025000)
```

Gambar 13.12. Code cell

SVM memiliki akurasi tertinggi sekitar 99%. Evaluasi model ini dapat membantu dalam memilih algoritma yang paling cocok untuk masalah yang dihadapi.

```
fig = plt.figure()
fig.suptitle('Compare Algorithm Accuracy')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

Gambar 13.13. Code snippet for accuracy comparison



Gambar 13.14. Plot of accuracy comparison

Jalankan model SVM pada data validasi, peroleh skor akurasi akhir, matriks kebingungan, dan laporan klasifikasi untuk evaluasi model.

```
#import required metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

svm = SVC()
svm.fit(X_train, Y_train)
predictions = svm.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

0.9333333333333333
[[ 7  0  0]
 [ 0 10  2]
 [ 0  0 11]]
      precision    recall   f1-score   support
          0       1.00     1.00     1.00       7
          1       1.00     0.83     0.91      12
          2       0.85     1.00     0.92      11
avg / total       0.94     0.93     0.93      30
```

Gambar 13.15. Generating accuracy score and report

Latih model dengan .fit(), lakukan prediksi, dan simpan model dengan pickle. Hasil akurasi, matriks kebingungan, dan laporan klasifikasi digunakan untuk mengevaluasi model.

```
# save the model to disk
import pickle
filename = 'finalized_model.sav'
pickle.dump(svm, open(filename, 'wb'))
# Load the model from disk for next time you open this notebook
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_validation, Y_validation)
print(result)

0.9333333333333333
```

Gambar 13.16. Pickle library serializes ML algorithms

REGRESSION ML

```
gapminder_df = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/regression/gm_2008_region.csv")

gapminder_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 139 entries, 0 to 138
Data columns (total 10 columns):
population      139 non-null float64
fertility       139 non-null float64
HIV             139 non-null float64
CO2             139 non-null float64
BMI_male        139 non-null float64
GDP             139 non-null float64
BMI_female      139 non-null float64
life            139 non-null float64
child_mortality 139 non-null float64
Region          139 non-null object
```

Gambar 13.17. Inspecting columns and data types

```
# Create arrays for features and target variable
y = gapminder_df['life'].values
X = gapminder_df['fertility'].values
```

Gambar 13.18. Creating arrays for features and target variable

```
# Print the dimensions of X and y before reshaping
print("Dimensions of target variable before reshaping: {}".format(y.shape))
print("Dimensions of feature variable before reshaping: {}".format(X.shape))

Dimensions of target variable before reshaping: (139,)
Dimensions of feature variable before reshaping: (139,
```

Gambar 13.19. Reshaping the variables

```

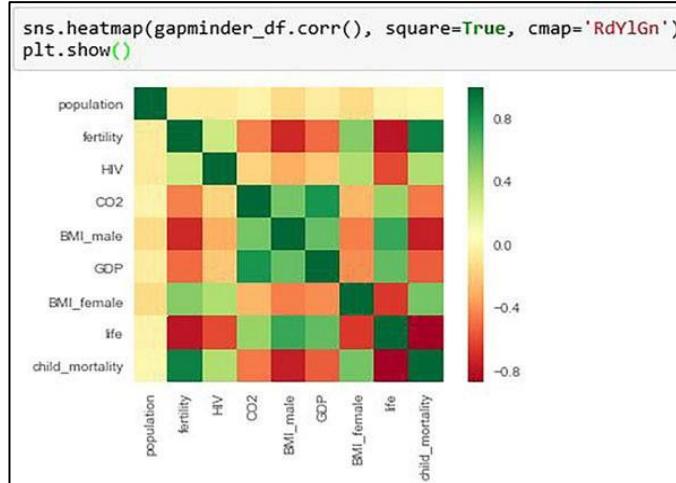
# Reshape X and y
y = y.reshape(-1,1)
X = X.reshape(-1,1)
# Print the dimensions of X and y after reshaping
print("Dimensions of target after reshaping: {}".format(y.shape))
print("Dimensions of feature variable after reshaping: {}".format(X.shape))

Dimensions of target after reshaping: (139, 1)
Dimensions of feature variable after reshaping: (139, 1)

```

Gambar 13.20. Changing the dimension

Untuk memeriksa korelasi antar fitur dalam dataframe kita, gunakan fungsi heatmap. Alih-alih matplotlib, kita akan menggunakan seaborn karena menghasilkan plot.



Gambar 13.21. Heat map

Mari terapkan Regresi Linear pada dataset kita tanpa membaginya terlebih dahulu.

```

# Import LinearRegression
from sklearn.linear_model import LinearRegression
# Create the regressor: reg
reg = LinearRegression()
# Create the prediction space
prediction_space = np.linspace(min(X), max(X)).reshape(-1,1)
# Fit the model to the data
reg.fit(X, y)
# Compute predictions over the prediction space: y_pred
y_pred = reg.predict(prediction_space)
# Print R^2
print(reg.score(X, y))
# Plot regression Line
plt.plot(prediction_space, y_pred, color='black', linewidth=3)
plt.show()

0.6192442167740035

```

Gambar 13.22. Applying Linear Regression

Anda akan membagi dataset GapMinder menjadi set pelatihan dan pengujian, lalu melatih dan memprediksi fitur-fitur regresi linear serta menghitung metrik R^2 dan RMSE.

```

# Import necessary modules
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
# Create training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)
# Create the regressor: reg_all
reg_all = LinearRegression()
# Fit the regressor to the training data
reg_all.fit(X_train, y_train)
# Predict on the test data: y_pred
y_pred = reg_all.predict(X_test)
# Compute and print R^2 and RMSE
print("R^2: {}".format(reg_all.score(X_test, y_test)))
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error: {}".format(rmse))

R^2: 0.7298987360907494
Root Mean Squared Error: 4.194027914110243

```

Gambar 13.23. Evaluating predictions on regression ML problems

Kinerja model sangat bergantung pada cara data dibagi. Ini memungkinkan model untuk belajar dari sebagian besar data yang tersedia. Validasi silang memastikan bahwa model diuji pada berbagai bagian data pelatihan untuk evaluasi yang lebih akurat.

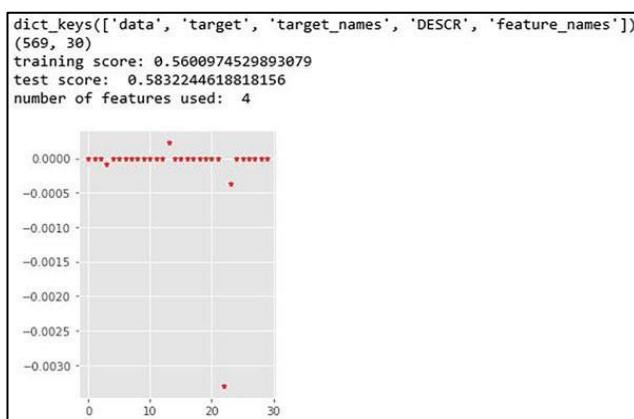
```
# Import the necessary modules
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
# Create a linear regression object: reg
reg = LinearRegression()
# Compute 5-fold cross-validation scores: cv_scores
cv_scores = cross_val_score(reg, X, y, cv=5)
# Print the 5-fold cross-validation scores
print(cv_scores)
# Print the average 5-fold cross-validation score
print("Average 5-Fold CV Score: {}".format(np.mean(cv_scores)))
[0.71001079 0.75007717 0.55271526 0.547501 0.52410561]
Average 5-Fold CV Score: 0.6168819644425119
```

Gambar 13.24. Cross validation code snippet

Dalam validasi silang, kami menggunakan metrik R^2 untuk mengevaluasi model regresi. Regularisasi adalah teknik untuk mencegah overfitting dalam model. Ada dua jenis umumnya: Lasso dan Ridge. Lasso dapat digunakan untuk seleksi fitur, sedangkan Ridge cocok untuk fitur berkorelasi tinggi.

```
from sklearn.linear_model import Lasso
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
print(cancer.keys())
print(cancer.data.shape)
cancer_df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
X = cancer.data
Y = cancer.target
X_train,X_test,y_train,y_test=train_test_split(X,Y, test_size=0.3, random_state=31)
lasso = Lasso()
lasso.fit(X_train,y_train)
train_score=lasso.score(X_train,y_train)
test_score=lasso.score(X_test,y_test)
coeff_used = np.sum(lasso.coef_!=0)
print("training score: ", train_score )
print("test score: ", test_score)
print("number of features used: ", coeff_used)
plt.xlabel('Coefficient Index',fontsize=16)
plt.ylabel('Coefficient Magnitude',fontsize=16)
plt.legend(fontsize=13,loc=4)
plt.subplot(1,2,2)
plt.plot(lasso.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Lasso; $\alpha = 1$',zorder=7)
plt.tight_layout()
plt.show()
```

Gambar 13.25 Applying Lasso package



Gambar 13.26. Output of running Lasso package

Dalam dataset kami, setelah menerapkan regresi Lasso, hanya 4 fitur yang digunakan, dan skor pelatihan serta pengujian rendah. Ini menunjukkan underfitting.

```
from sklearn.datasets import load_boston
from sklearn.linear_model import Ridge
boston=load_boston()
boston_df=pd.DataFrame(boston.data,columns=boston.feature_names)
boston_df['Price']=boston.target
newX=boston_df.drop('Price',axis=1)
newY=boston_df['Price']
```

Gambar 13.27. Applying Ridge Regression

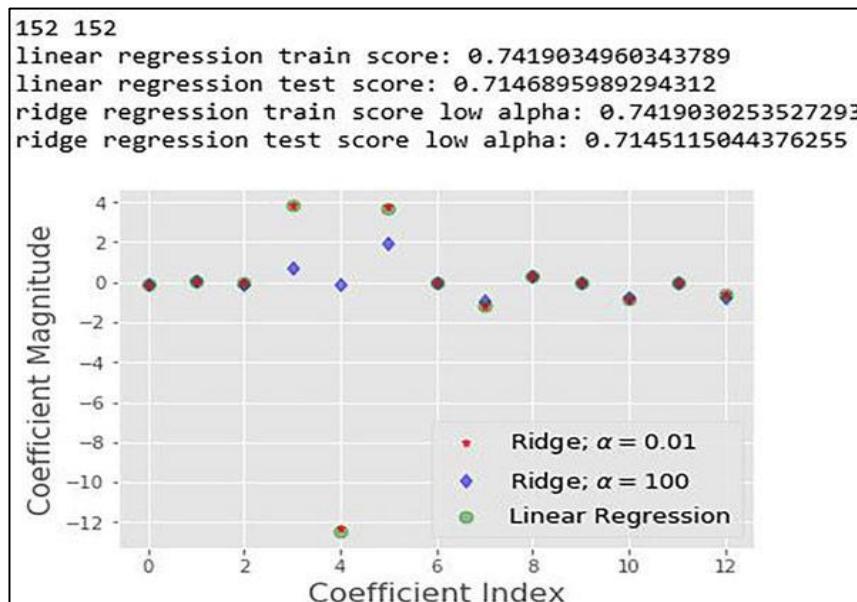
Di sini, axis=1 berarti kita menerapkan logika secara baris. Kami telah memisahkan kolom target

- Harga dari dataframe dan menyimpannya sebagai kolom target:

```
X_train,X_test,y_train,y_test=train_test_split(newX,newY,test_size=0.3,random_state=3)
print(len(X_test), len(y_test))
lr = LinearRegression()
lr.fit(X_train, y_train)
rr = Ridge(alpha=0.01)
rr.fit(X_train, y_train)
train_score=lr.score(X_train, y_train)
test_score=lr.score(X_test, y_test)
Ridge_train_score = rr.score(X_train,y_train)
Ridge_test_score = rr.score(X_test, y_test)
print("linear regression train score:", train_score)
print("linear regression test score:", test_score)
print("ridge regression train score low alpha:", Ridge_train_score)
print("ridge regression test score low alpha:", Ridge_test_score)
plt.plot(rr.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge; $\alpha$ = 0.01',zorder=7)
plt.plot(lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regression')
plt.xlabel('Coefficient Index',fontsize=16)
plt.ylabel('Coefficient Magnitude',fontsize=16)
plt.legend(fontsize=13,loc=4)
plt.show()
```

Gambar 13.28. Separating a column and storing as target column

Dalam grafik di atas, sumbu X mewakili indeks koefisien, yang mengacu pada fitur-fitur dalam dataset.



Gambar 13.29. Plot of coefficient index vs magnitude

TUNE ML MODEL

Anda perlu menyetel model machine learning untuk meningkatkan akurasi dengan memilih hyperparameter yang tepat. Ini dapat dilakukan dengan mencoba berbagai nilai untuk hyperparameter seperti alpha dalam Lasso/Ridge regression atau n_neighbors dalam K-NN, dan kemudian memilih yang memberikan hasil terbaik. Anda dapat menggunakan perpustakaan GridSearchCV untuk melakukan pencarian hyperparameter secara otomatis dengan mengidentifikasi hyperparameter dan nilai-nilainya.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

df = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/diabetes.csv")
print(df.columns)

y=df['diabetes']
X=df.drop('diabetes',axis=1)

#Setup the hyperparameter grid
c_space = np.logspace(-5, 8, 15)
param_grid = {'C': c_space}

logreg = LogisticRegression()
logreg_cv = GridSearchCV(logreg, param_grid, cv=5)
logreg_cv.fit(X, y)

print("Tuned Logistic Regression Parameters: {}".format(logreg_cv.best_params_))
print("Best score is {}".format(logreg_cv.best_score_))
```

Gambar 13.30. Setting up hyperparameter grid and performing gridsearch cross-validation

```

# Import necessary modules
from scipy.stats import randint
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV

param_dist = {"max_depth": [3, None],
              "max_features": randint(1, 9),
              "min_samples_leaf": randint(1, 9),
              "criterion": ["gini", "entropy"]}

tree = DecisionTreeClassifier()
tree_cv = RandomizedSearchCV(tree, param_dist, cv=5)

tree_cv.fit(X, y)

print("Tuned Decision Tree Parameters: {}".format(tree_cv.best_params_))
print("Best score is {}".format(tree_cv.best_score_))

Tuned Decision Tree Parameters: {'criterion': 'entropy', 'max_depth': 3, 'max_features': 7, 'min_samples_leaf': 4}
Best score is 0.7447916666666666

```

Gambar 13.31. Decision Tree Classifier example

VARIABEL KATEGORIKAL DALAM SKLEARN

Anda dapat mengatasi masalah variabel kategorikal dalam scikit-learn dengan mengonversinya menjadi variabel biner menggunakan fungsi `get_dummies()` dari Pandas.

```

# handling categorical variable 'Region' by binarizing it(creating dummy variables)
# Create dummy variables: df_region
df_region = pd.get_dummies(df)

# Print the columns of df_region
print(df_region.columns)

# Drop 'Region_America' from df_region
df_region = pd.get_dummies(df, drop_first=True)

# Print the new columns of df_region
print(df_region.columns)

```

Gambar 13.32. Converting non-numeric variable in desired format

Dalam contoh di atas, `pd.get_dummies(df)` mengonversi variabel kategorikal dalam dataframe menjadi variabel biner. Setelah Anda menjalankan sel sebelumnya, Anda akan melihat bahwa kolom Region telah diubah menjadi kolom dengan nama-nama region yang sesuai.

```

Index(['population', 'fertility', 'HIV', 'CO2', 'BMI_male', 'GDP',
       'BMI_female', 'life', 'child_mortality', 'Region_America',
       'Region_East Asia & Pacific', 'Region_Europe & Central Asia',
       'Region_Middle East & North Africa', 'Region_South Asia',
       'Region_Sub-Saharan Africa'],
      dtype='object')

```

Gambar 13.33. Region DataFrame columns

```

from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Ridge

ridge = Ridge(alpha=0.5, normalize=True)
y=df_region['life'].values
X=df_region.drop('life', axis=1).values

# Perform 5-fold cross-validation: ridge_cv
ridge_cv = cross_val_score(ridge, X, y, cv=5)
print(ridge_cv)

[0.86808336 0.80623545 0.84004203 0.7754344 0.87503712]

```

Gambar 13.34. Regression technique on GapMinder dataset

MENGATASI DATA YANG HILANG

```
df = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/pima-indians-diabetes.data.csv', header = None)
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 0    768 non-null int64
 1    768 non-null int64
 2    768 non-null int64
 3    768 non-null int64
 4    768 non-null int64
 5    768 non-null float64
 6    768 non-null float64
 7    768 non-null int64
 8    768 non-null int64
dtypes: float64(2), int64(7)
```

Gambar 13.35. Handling missing data

```
missing_values_count = df.isnull().sum()
print("count of missing values:\n", missing_values_count)

count of missing values:
 0    0
 1    0
 2    0
 3    0
 4    0
 5    0
 6    0
 7    0
 8    0
```

Gambar 13.36. Counting missing values

	0	1	2	3	4	5	6	7	8
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Gambar 13.37. Performing statistical analysis

Pertama-tama, mari gantikan nilai nol pada beberapa kolom dengan nilai yang sebenarnya yang hilang - NaN, dan kemudian kita akan mengatasi NaN:

```
# mark some columns zero values as missing or NaN
import numpy as np
df[[1,2,3,4,5]] = df[[1,2,3,4,5]].replace(0, np.NaN)
print(df.isnull().sum())

 0    0
 1    5
 2   35
 3  227
 4  374
 5   11
 6    0
 7    0
 8    0
```

Gambar 13.38. Replacing zero with actual missing value

Ketika nilai-nilai hilang dalam dataset, hal ini dapat menyebabkan kesalahan pada algoritma machine learning seperti LDA. Mari kita isi nilai-nilai tersebut sebelum menerapkan LDA.

```
from sklearn.preprocessing import Imputer
# fill missing values with mean column values
values = df.values
imputer = Imputer()
transformed_values = imputer.fit_transform(values)
# count the number of NaN values in each column
print(np.isnan(transformed_values).sum())

0
```

Gambar 13.39. Imputing missing values

```

# evaluate an LDA model on the dataset using k-fold cross validation
model = LinearDiscriminantAnalysis()
kfold = KFold(n_splits=3, random_state=7)
result = cross_val_score(model, transformed_values, y, cv=kfold, scoring='accuracy')
print(result.mean())

0.7669270833333334

```

Gambar 13.40. Applying LDA algorithm

Contoh lain tentang pengisian nilai dengan pipa (pipeline) menggunakan algoritma lain yang dikenal sebagai SVM Classifier.

```

from sklearn.preprocessing import Imputer
from sklearn.svm import SVC
imp = Imputer(missing_values='NaN', strategy='most_frequent', axis=0)

# Instantiate the SVC classifier: clf
clf = SVC()

# Setup the pipeline with the required steps: steps
steps = [('imputation', imp),
          ('SVM', clf)]

```

Gambar 13.41. Example of imputing

Variabel "steps" berisi daftar tupel dengan langkah pengisian nilai dan algoritma klasifikasi. Ini adalah konsep pipa (pipeline) untuk imputasi dan klasifikasi.

```

from sklearn.preprocessing import Imputer
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Setup the pipeline steps: steps
steps = [('imputation', Imputer(missing_values='NaN', strategy='most_frequent', axis=0)),
          ('SVM', SVC())]

# Create the pipeline: pipeline
pipeline = Pipeline(steps)

# Create training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Fit the pipeline to the train set
pipeline.fit(X_train, y_train)

# Predict the labels of the test set
y_pred = pipeline.predict(X_test)

# Compute metrics
print(classification_report(y_test, y_pred))

```

Gambar 13.42. Using for classification

	precision	recall	f1-score	support
0.0	0.65	1.00	0.79	151
1.0	0.00	0.00	0.00	80
avg / total	0.43	0.65	0.52	231

Gambar 13.43. classification report result See, how

13.5 KESIMPULAN

Anda telah mempelajari dasar-dasar serta beberapa teknik lanjutan dari algoritma machine learning berbasis pengawasan. Anda juga telah menyelesaikan masalah berbasis pengawasan dunia nyata. Namun, masih banyak yang bisa dieksplorasi dan dipelajari dalam bidang ini. Hal itu hanya bisa terwujud jika Anda mencoba berbagai pendekatan dan algoritma sendiri; jadi cobalah sebanyak yang Anda bisa. Sampai saat itu, kejarlah impian Anda, nikmatilah hari yang luar biasa, manfaatkan setiap detik. Sampai jumpa nanti di bab berikutnya dari buku ini, di mana Anda akan mempelajari tentang machine learning tanpa pengawasan.

BAB XIV

UNSUPERVISED MACHINE LEARNING

14.1 STRUKTUR

Unsupervised learning berbeda dari metode klasifikasi dan regresi karena mengolah data masukan yang tidak berlabel, menemukan struktur data secara mandiri. Ini melibatkan berbagai teknik seperti pengelompokan dan reduksi dimensi. Bab ini membahas dasar-dasar unsupervised learning dan implementasi praktisnya dengan scikit-learn dan scipy. Anda akan mempelajari:

1. Unsupervised Learning
2. Teknik
3. K-Means Klastering
4. Principal Component Analysis (PCA)
5. Studi kasus
6. Validasi Unsupervised ML

14.2 TUJUAN

Setelah mempelajari dan berlatih bab ini, Anda akan menguasai pembelajaran tanpa pengawasan dan dapat mengelompokkan, mentransformasi, visualisasi, dan mengekstrak wawasan dari dataset yang tidak berlabel.

14.3 UNSUPERVISED LEARNING

Pembelajaran tanpa pengawasan memproses data tanpa label dan mengekstrak fitur dari distribusi probabilitas data. Ini berguna ketika kita tidak tahu kelas data sebelumnya, data terstruktur sulit didapat, dan untuk generasi konten. Kemajuan dalam komputasi awan dan deep learning memungkinkan penerapan efisien.

14.4 TEKNIK UNSUPERVISED LEARNING

PENGELOMPOKAN (CLUSTERING):

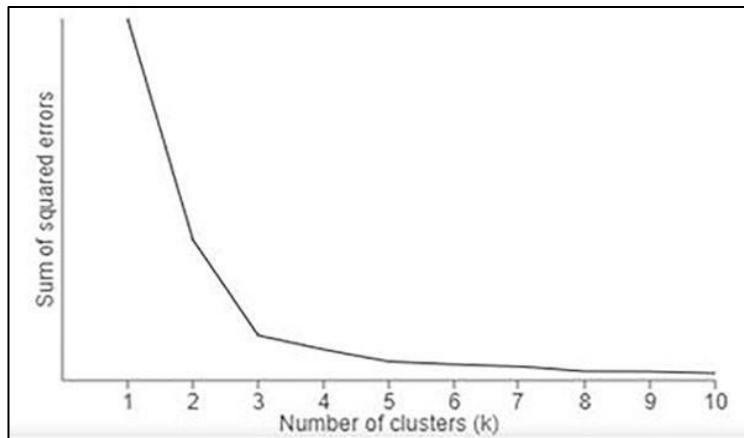
Pengelompokan adalah proses mengelompokkan entitas yang mirip. Ini membantu mengidentifikasi pola dalam data dan mengurangi dimensi data dengan algoritma seperti K-means dan hierarkis.

1. **Deteksi Anomali (Anomaly Detection):** Dapat secara otomatis menemukan titik data yang tidak biasa dalam dataset Anda. Ini berguna untuk mengidentifikasi transaksi curang, menemukan komponen perangkat keras yang rusak, atau mengidentifikasi outlier yang disebabkan oleh kesalahan manusia saat memasukkan data.
2. **Pencarian Asosiasi (Association Mining):** Mengidentifikasi set item yang sering muncul bersama dalam dataset Anda. Pemilik toko sering menggunakan analisis keranjang, karena memungkinkan analisis untuk menemukan barang yang sering dibeli bersamaan, dan mengembangkan strategi pemasaran dan penjualan yang lebih efektif.
3. **Model Variabel Tersembunyi (Latent Variable Models):** Umumnya digunakan untuk pra-pemrosesan data, seperti mengurangi jumlah fitur dalam dataset (reduksi dimensi) atau memecah dataset menjadi beberapa komponen.

Some applications of unsupervised machine learning techniques include the following:

K-mean clustering

Dalam K-means clustering, K menentukan jumlah cluster. Centroid ditempatkan secara acak, dan data ditugaskan ke cluster terdekat. Metode Elbow digunakan untuk memilih jumlah cluster optimal dengan memplot persentase variasi yang dijelaskan berdasarkan nilai K.



Gambar 14.1. *Elbow method plot*

Lihat plot sebelumnya yang mirip dengan siku manusia. Kami akan menerapkan K-means pada dataset poker dan mengevaluasi cluster dengan metode Elbow. Untuk kode contoh, muat notebook pembelajaran tanpa pengawasan yang disediakan dalam bundel kode.

```
# read training and test data from the url link and save the file to your working directory
url = "http://archive.ics.uci.edu/ml/machine-learning-databases/poker/poker-hand-training-true.data"
urllib.request.urlretrieve(url, "E:/pg/bpb/BPB-Publications/Datasets/unsupervised/poker_train.csv")
url2 = "http://archive.ics.uci.edu/ml/machine-learning-databases/poker/poker-hand-testing.data"
urllib.request.urlretrieve(url2, "E:/pg/bpb/BPB-Publications/Datasets/unsupervised/poker_test.csv")

# read the data in and add column names
data_train = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/unsupervised/poker_train.csv", header=None,
                        names=['S1', 'C1', 'S2', 'C2', 'S3', 'C3', 'S4', 'C4', 'S5', 'C5', 'CLASS'])
data_test = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/unsupervised/poker_test.csv", header=None,
                        names=['S1', 'C1', 'S2', 'C2', 'S3', 'C3', 'S4', 'C4', 'S5', 'C5', 'CLASS'])
```

Gambar 14.2. *Performing k-means with elbow method*

Kode di sel sebelumnya membaca data pelatihan dan pengujian dari URL menggunakan `urllib.request.urlretrieve()`. Kami menyimpannya dalam variabel `url` dan `url2`. Kemudian, kami membaca file CSV yang telah disimpan dalam direktori lokal ke dalam Pandas Dataframe untuk pemrosesan lebih lanjut dan memilih subset dari dataset pelatihan.

```
# subset clustering variables
cluster=data_train[['S1', 'C1', 'S2', 'C2', 'S3', 'C3', 'S4', 'C4', 'S5', 'C5']]
```

Gambar 14.3. *Subset clustering variables*

Untuk mengelompokkan data dengan baik, standarisasi fitur-fitur diperlukan. Fungsi preprocessing.scale() digunakan untuk menskalakannya sehingga variabel memiliki kontribusi yang setara. Ini ditunjukkan dalam tangkapan layar berikut:

```
from sklearn import preprocessing
# standardize clustering variables to have mean=0 and sd=1 so that card suit and
# rank are on the same scale as to have the variables equally contribute to the analysis
clustervar = cluster.copy() # create a copy
clustervar['S1']=preprocessing.scale(clustervar['S1'].astype('float64'))
clustervar['C1']=preprocessing.scale(clustervar['C1'].astype('float64'))
clustervar['S2']=preprocessing.scale(clustervar['S2'].astype('float64'))
clustervar['C2']=preprocessing.scale(clustervar['C2'].astype('float64'))
clustervar['S3']=preprocessing.scale(clustervar['S3'].astype('float64'))
clustervar['C3']=preprocessing.scale(clustervar['C3'].astype('float64'))
clustervar['S4']=preprocessing.scale(clustervar['S4'].astype('float64'))
clustervar['C4']=preprocessing.scale(clustervar['C4'].astype('float64'))
clustervar['S5']=preprocessing.scale(clustervar['S5'].astype('float64'))
clustervar['C5']=preprocessing.scale(clustervar['C5'].astype('float64'))

# The data has been already split data into train and test sets
clus_train = clustervar
```

Gambar 14.4. *Scaling variables*

Anda tidak perlu melakukan perhitungan jarak antara setiap pasangan dua koleksi input secara manual. Fungsi `scipy.spatial.distance` dan pustaka `cdist`-nya dapat digunakan untuk itu. Setelah menghitung jarak, langkah selanjutnya adalah melatih model pada set pelatihan untuk setiap cluster, menghasilkan penugasan cluster yang diprediksi, dan menghitung rata-rata jarak dengan menjumlahkannya dan membaginya dengan jumlah data.

```
from sklearn.cluster import KMeans

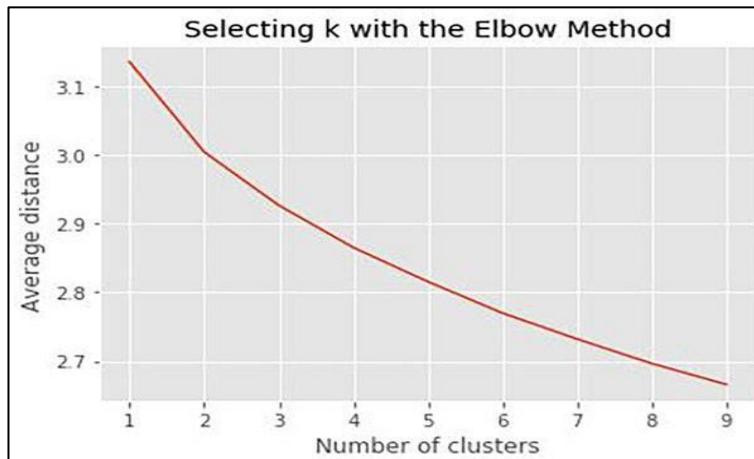
# k-means cluster analysis for 1-10 clusters due to the 10 possible class outcomes for poker hands
from scipy.spatial.distance import cdist
clusters=range(1,20)
meandist=[]

# Loop through each cluster and fit the model to the train set
# generate the predicted cluster assigment and append the mean distance my taking the sum divided
for k in clusters:
    model=KMeans(n_clusters=k)
    model.fit(clus_train)
    clusassign=model.predict(clus_train)
    meandist.append(sum(np.min(cdist(clus_train, model.cluster_centers_, 'euclidean'), axis=1))
    / clus_train.shape[0])

"""
Plot average distance from observations from the cluster centroid
to use the Elbow Method to identify number of clusters to choose
"""
plt.plot(clusters, meandist)
plt.xlabel('Number of clusters')
plt.ylabel('Average distance')
plt.title('Selecting k with the Elbow Method') # pick the fewest number of clusters that reduces the
<
Text(0.5,1,'Selecting k with the Elbow Method')
```

Gambar 14.5. *Finding value of k with Elbow method*

Setelah menjalankan sel sebelumnya, Anda akan melihat bahwa jumlah cluster yang tepat adalah 3 (lihat sumbu X dalam plot). Setelah nilai tersebut, peningkatan performa model tidak signifikan. Jumlah cluster sama dengan nilai pada sudut siku (plot sering terlihat seperti siku).



Gambar 14.6. *Selecting k with the elbow method* [Hierarchical clustering](#)

Berbeda dengan pengelompokan K-means, pengelompokan hierarkis dimulai dengan menugaskan semua titik data ke dalam cluster mereka masing-masing. Seperti namanya, ini membangun hirarki, dan dalam langkah berikutnya, menggabungkan dua titik data terdekat dan menggabungkannya menjadi satu cluster. Berikut adalah langkah-langkah untuk menerapkan teknik ini:

1. Mulai dengan N cluster, tetapkan setiap titik data ke cluster mereka.
2. Temukan pasangan cluster terdekat menggunakan jarak Euclidean dan gabungkan mereka menjadi satu cluster.
3. Hitung jarak antara dua cluster terdekat dan gabungkan hingga semua item terkelompok dalam satu cluster.

```

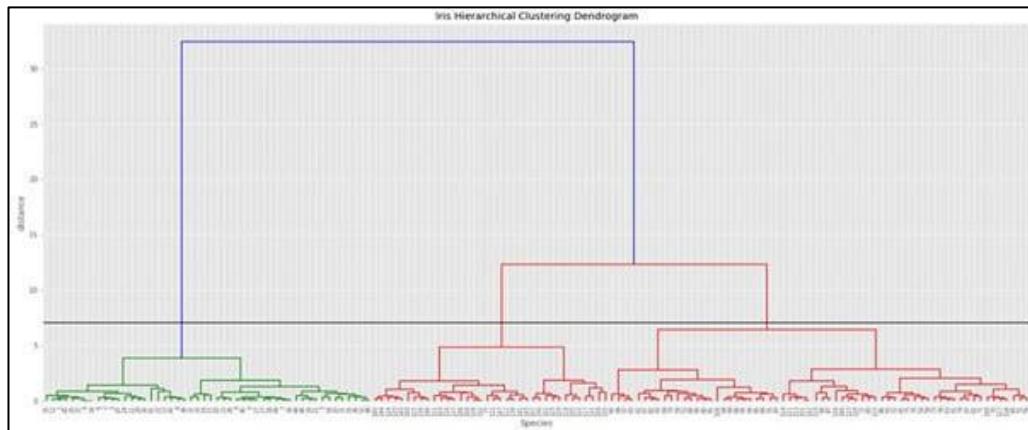
# calculate full dendrogram
from scipy.cluster.hierarchy import dendrogram, linkage
# generate the linkage matrix
Z = linkage(iris, 'ward')
# set cut-off to 50
max_d = 7.08           # max_d as in max_distance

plt.figure(figsize=(25, 10))
plt.title('Iris Hierarchical Clustering Dendrogram')
plt.xlabel('Species')
plt.ylabel('distance')
dendrogram(
    Z,
    truncate_mode='lastp',  # show only the last p merged clusters
    p=150,                 # Try changing values of p
    leaf_rotation=90.,      # rotates the x axis labels
    leaf_font_size=8.,       # font size for the x axis labels
)
plt.axhline(y=max_d, c='k')
plt.show()

```

Gambar 14.7. *Using hierarchical clustering on Iris dataset*

Di sini, kami menggunakan fungsi linkage() dengan argumen ward untuk pengelompokan hierarkis sampel iris, dan dendrogram() untuk visualisasi hasilnya. Argumen ward meminimalkan variasi antara kelompok. Setelah menjalankan sel sebelumnya, akan ditampilkan plot yang sesuai.



Gambar 14.8. *dendrogram plot example*

t-SNE

t-SNE adalah teknik untuk memetakan data berdimensi tinggi ke ruang 2D atau 3D untuk visualisasi. Anda dapat menggunakannya dengan sklearn untuk dataset seperti MNIST, tetapi perhatikan bahwa t-SNE dapat memakan banyak memori.

```

from sklearn.datasets import fetch_mldata
mnist = fetch_mldata("MNIST original")
X = mnist.data / 255.0
y = mnist.target
print(X.shape, y.shape)

(70000, 784) (70000,)

```

Gambar 14.9. *Using fetch_mldata function*

Jika ada masalah saat memuat data, unduh dari sumber lain seperti Google atau GitHub. Import numpy dan pandas, lalu konversi data pelatihan ke dataframe. Ekstrak variabel target dari dataframe tersebut.

```

#convert the matrix and vector to a Pandas DataFrame
feat_cols = [ 'pixel'+str(i) for i in range(X.shape[1]) ]

df = pd.DataFrame(X,columns=feat_cols)
df['label'] = y
df['label'] = df['label'].apply(lambda i: str(i))

X, y = None, None

print('Size of the dataframe: {}'.format(df.shape))
Size of the dataframe: (70000, 785)

```

Gambar 14.10. *Extracting target variable*

Selanjutnya, kita akan mengambil subset acak dari digit-digits tersebut untuk memastikan randomisasi dalam analisis kita.

```

rndperm = np.random.permutation(df.shape[0])

```

Gambar 14.11. *Random selecting the permutations*

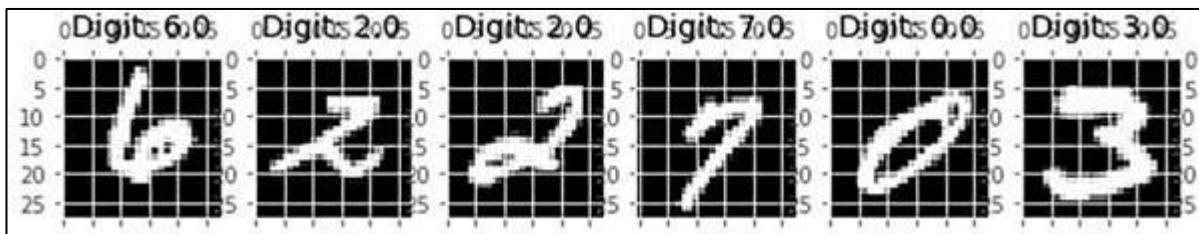
Sekarang kita memiliki dataframe dan vektor randomisasi. Mari kita lihat gambar-gambar ini dengan menghasilkan 30 plot gambar secara acak. Pastikan untuk mengimpor matplotlib.pyplot as plt sebelum menjalankan kode ini.

```

plt.gray()
fig = plt.figure( figsize=(16,7) )
for i in range(0,30):
    ax = fig.add_subplot(3,10,i+1, title='Digit: ' + str(df.loc[rndperm[i],'label']))
    ax.matshow(df.loc[rndperm[i],feat_cols].values.reshape((28,28)).astype(float))
plt.show()

```

Gambar 14.12. *code snippet for plotting the numbers*



Gambar 14.13. *result of the plotting code snippet*

Jika Anda tidak melihat gambar asli sebagai output, tambahkan titik koma setelah plt.show(). Kami akan menggunakan 7.000 sampel pertama untuk algoritma ini.

```

import time
from sklearn.manifold import TSNE

n_sne = 7000

time_start = time.time()
tsne = TSNE(n_components=2, verbose=1, perplexity=40, n_iter=300)
tsne_results = tsne.fit_transform(df.loc[:n_sne,feat_cols].values)

print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))

```

Gambar 14.14. *use of TSNE*

Kita dapat memvisualisasikan dua dimensi ini dengan membuat scatter plot dan mewarnai setiap sampel sesuai dengan labelnya. Jalankan salah satu perintah berikut dalam prompt Anaconda:

1. conda install -c conda-forge ggplot
2. conda install -c conda-forge/label/gcc7 ggplot
3. conda install -c conda-forge/label/cf201901 ggplot

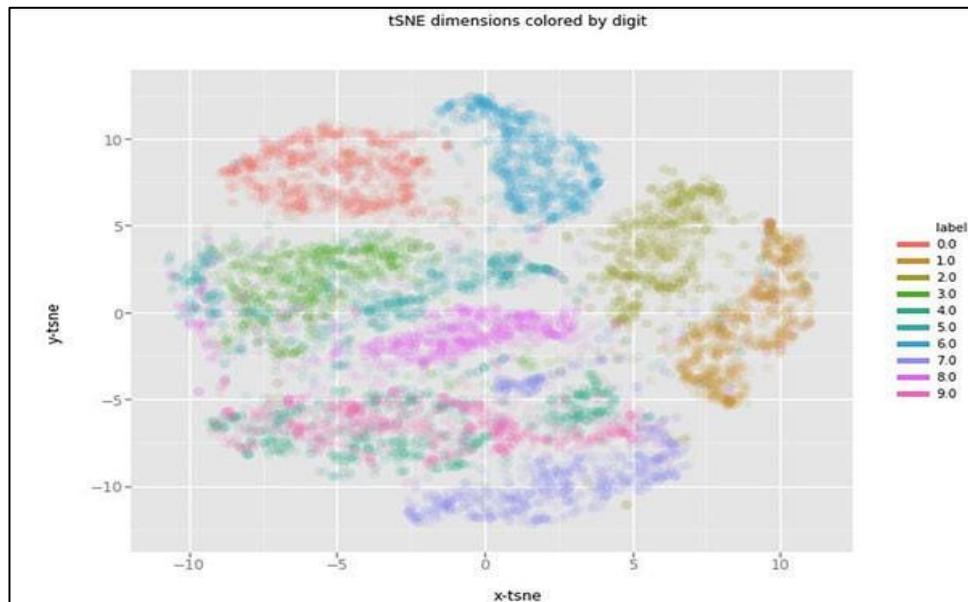
```

from ggplot import *
df_tsne = df.loc[rndperm[:n_tsne], :].copy()
df_tsne['x-tsne'] = tsne_results[:, 0]
df_tsne['y-tsne'] = tsne_results[:, 1]

tsne_plot = ggplot(df_tsne, aes(x='x-tsne', y='y-tsne', color='label')) \
    + geom_point(size=70, alpha=0.1) \
    + gtitle("tSNE dimensions colored by digit")
tsne_plot

```

Gambar 14.16. *ggplot code snippet*



Gambar 14.17. *tSNE dimensions colored by digit*

Kita bisa melihat bahwa digit-digit ini sangat jelas terkelompok dalam kelompok-kelompok kecil mereka (lihat warna label, di mana setiap warna mengindikasikan kelompok yang berbeda). Visualisasi yang sama tidak dapat dilakukan tanpa t-SNE jika kita menyertakan dimensi yang lebih tinggi.

PRINCIPAL COMPONENT ANALYSIS (PCA)

Salah satu tugas umum dalam pembelajaran tanpa pengawasan adalah reduksi dimensi, yang dapat membantu visualisasi dan mengatasi multicollinearity. PCA adalah metode yang sering digunakan untuk reduksi dimensi, memutar data tanpa kehilangan informasi.

```

student_data_mat = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/unsupervised/PCA/student-mat.csv", delimiter=";")
student_data_por = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/unsupervised/PCA/student-por.csv", delimiter=";")
student_data = pd.merge(student_data_mat, student_data_por, how="outer")
student_data.head()

```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	... famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3	
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5	6	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5	5	6
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8	10
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14	15
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10	10

5 rows x 33 columns

Gambar 14.18. *Loading example datasets*

Dataset ini berisi prestasi siswa di dua sekolah Portugal. Atributnya termasuk nilai siswa, demografi, sosial, dan fitur sekolah. Ada dua dataset untuk mata pelajaran yang berbeda: Matematika dan Bahasa Portugis. Kolom-kolom tertentu akan dianggap variabel kategoris dan akan ditangani.

```

student_data.isnull().values.any()
False

col_str = student_data.columns[student_data.dtypes == object]

from sklearn.preprocessing import LabelEncoder
lenc = LabelEncoder()
student_data[col_str] = student_data[col_str].apply(lenc.fit_transform)

```

Gambar 14.19. *Handling categorical columns*

print(student_data[["G1","G2","G3"]].corr())			
	G1	G2	G3
G1	1.000000	0.858739	0.809142
G2	0.858739	1.000000	0.910743
G3	0.809142	0.910743	1.000000

Gambar 14.20. *Using .corr() function*

Dari sel output sebelumnya, dapat dengan mudah dilihat bahwa G1 dan G2 memiliki korelasi yang tinggi dengan G3, jadi kita dapat menghapus G1 dan G2 untuk analisis lebih lanjut:

```

# Since, G1,G2,G3 have very high correlation, we can drop G1,G2
student_data.drop(axis = 1,labels= ["G1","G2"])

```

Gambar 14.21. *Dropping G1 and G2*

Langkah berikutnya adalah memisahkan target dan sampel dari dataset, lalu menerapkan PCA menggunakan sklearn.decomposition. Variabel target adalah G3. Kami menggunakan cumsum() dari numpy untuk menghitung jumlah kumulatif elemen.

```

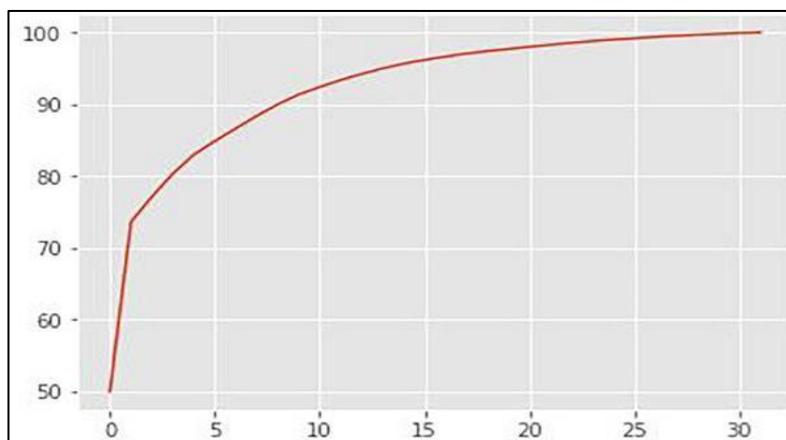
label = student_data["G3"].values
predictors = student_data[student_data.columns[:-1]].values

from sklearn.decomposition import PCA
pca = PCA(n_components=len(student_data.columns)-1)
pca.fit(predictors)
variance_ratio = pca.explained_variance_ratio_
variance_ratio_cum_sum=np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100)
print(variance_ratio_cum_sum)
plt.plot(variance_ratio_cum_sum)
plt.show()

```

Gambar 14.22. *PCA code snippet example*

PCA menggantikan variabel asli dengan komponen utama yang memiliki varian tertinggi dalam urutan penurunan.



Gambar 14.23. *Plot of the code snippet*

■ STUDI KASUS

Sekarang, Anda harus telah memahami pengetahuan dasar tentang teknik regresi. Saatnya menerapkan pengetahuan Anda pada masalah nyata. Di sini, Anda akan bekerja pada dataset visi

komputer MNIST, yang terdiri dari gambar digit berukuran 28 x 28 piksel. Mari impor data pelatihan terlebih dahulu:

```
train = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/regression/MNIST/train.csv")
print(train.shape)
train.head()

(42000, 785)

label pixel0 pixel1 pixel2 pixel3 pixel4 pixel5 pixel6 pixel7 pixel8 ... pixel774 pixel775 pixel776
0 1 0 0 0 0 0 0 0 0 ... 0 0 0
1 0 0 0 0 0 0 0 0 0 ... 0 0 0
```

Gambar 14.24. *Importing train data*

Dataset MNIST terdiri dari 42.000 baris dan 785 kolom. Ada 784 kolom yang mewakili piksel gambar, dan satu kolom tambahan berisi label kelas untuk menentukan apakah setiap baris menggambarkan angka 1 atau 9. Setiap komponen baris berisi nilai antara satu dan nol, yang menggambarkan intensitas masing-masing piksel.

```
# save the labels to a Pandas series target
target = train['label']
# Drop the label feature
train = train.drop("label",axis=1)
```

Gambar 14.25. *target column handling*

Kita akan menghitung vektor-vektor eigen dan nilai-nilai eigen matriks kovarian.

```
# Standardizing MNIST dataset features by removing the mean and scaling to unit variance
from sklearn.preprocessing import StandardScaler
train_X = train.values
train_X_std = StandardScaler().fit_transform(train_X)

# Calculating Eigenvectors and Eigenvalues of Covariance matrix
covariance_matrix = np.cov(train_X_std.T)
eigen_values, eigen_vectors = np.linalg.eig(covariance_matrix)

# Creating a list of (eigenvalue, eigenvector)
eigen_pairs = [ (np.abs(eigen_values[i]),eigen_vectors[:,i]) for i in range(len(eigen_values))]

# Sorting the eigenvalue, eigenvector pair from high to Low
eigen_pairs.sort(key = lambda x: x[0], reverse= True)

# Calculating Individual and Cumulative explained variance
total_eigen_values = sum(eigen_values)
individual_exp_var = [(i/total_eigen_values)*100 for i in sorted(eigen_values, reverse=True)]
cumulative_exp_var = np.cumsum(individual_exp_var)
```

Gambar 14.26. *Calculating eigenvectors and eigenvalues of covariance matrix*

Kita akan gunakan plotly untuk membuat grafik interaktif.

```
import plotly.offline as py
py.init_notebook_mode(connected=True)
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
import plotly.tools as tls
import seaborn as sns
```

Gambar 14.27. *Installing plotly libraries*

Selanjutnya, kita akan membuat plot sebar sederhana menggunakan Plotly.

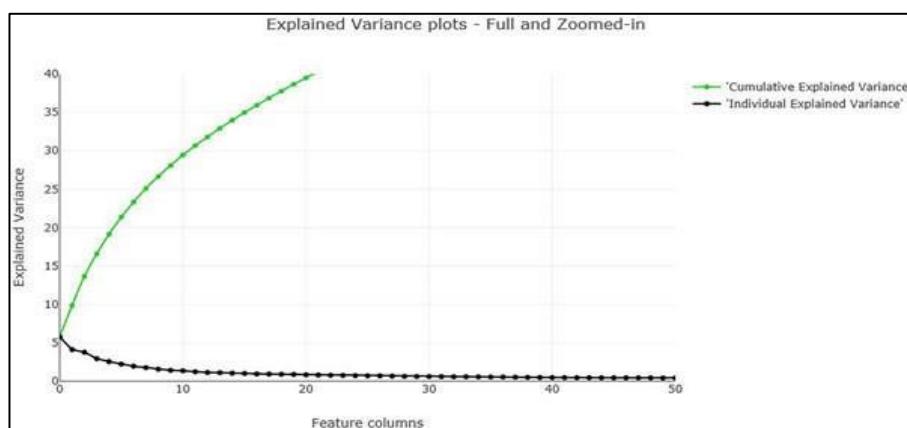
```

cumulative_plot = go.Scatter(
    x=list(range(784)),
    y= cumulative_exp_var,
    mode='lines+markers',
    name="'Cumulative Explained Variance'",
    line=dict(
        shape='spline',
        color = 'limegreen'
    )
)
individual_plot = go.Scatter(
    x=list(range(784)),
    y= individual_exp_var,
    mode='lines+markers',
    name="'Individual Explained Variance'",
    line=dict(
        shape='linear',
        color = 'black'
    )
)
fig = tls.make_subplots(insets=[{'cell': (1,1), 'l': 0.7, 'b': 0.5}],
print_grid=True)

fig.append_trace(cumulative_plot, 1, 1)
fig.append_trace(individual_plot,1,1)
fig.layout.title = 'Explained Variance plots - Full and Zoomed-in'
fig.layout.xaxis = dict(range=[0, 50], title = 'Feature columns')
fig.layout.yaxis = dict(range=[0, 40], title = 'Explained Variance')
iplot(fig)

```

Gambar 14.28. scatter plot code snippet



Gambar 14.29. Explained variance plots

Dari 784 fitur, sekitar 90% Varians yang Dijelaskan dapat dijelaskan dengan menggunakan lebih dari 200 fitur. Kita akan menggunakan PCA untuk mengekstrak 30 nilai eigen teratas dan membandingkan 5 nilai eigen teratas dengan yang lebih kecil.

```

# Invoke SKlearn's PCA method
n_components = 30
pca = PCA(n_components=n_components).fit(train.values)
eigenvalues = pca.components_.reshape(n_components, 28, 28)
# Extracting the PCA components ( eigenvalues )
eigenvalues = pca.components_

```

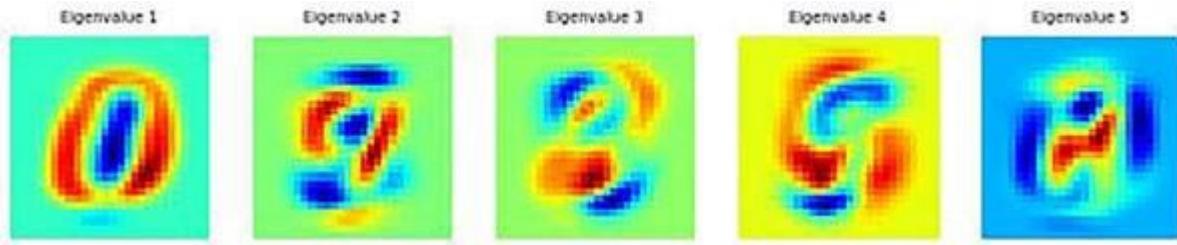
Gambar 14.30. checking eigenvalues

```

n_row = 4
n_col = 7
# Plot the first 8 eigenvalues
plt.figure(figsize=(13,12))
for i in list(range(n_row * n_col)):
    offset =0
    plt.subplot(n_row, n_col, i + 1)
    plt.imshow(eigenvalues[i].reshape(28,28), cmap='jet')
    title_text = 'Eigenvalue ' + str(i + 1)
    plt.title(title_text, size=6.5)
    plt.xticks(())
    plt.yticks(())
plt.show()

```

Gambar 14.31. code snippet for plotting eigen values



Gambar 14.32. *plot of different eigen values*

Sekarang, kita akan menerapkan algoritma PCA menggunakan toolkit sklearn:

```
# Delete our earlier created train_X object
del train_X
# Taking only the first N rows to speed things up
X= train[:6000].values
del train
# Standardising the values
X_std = StandardScaler().fit_transform(X)

# Call the PCA method with 5 components.
pca = PCA(n_components=5)
pca.fit(X_std)
X_5d = pca.transform(X_std)

# Restrict the target values also for speed up
Target = target[:6000]
```

Gambar 14.33. *Implementing PCA algorithm*

KMeans clustering untuk mencoba memisahkan data dalam ruang fitur baru setelah menggunakan PCA.

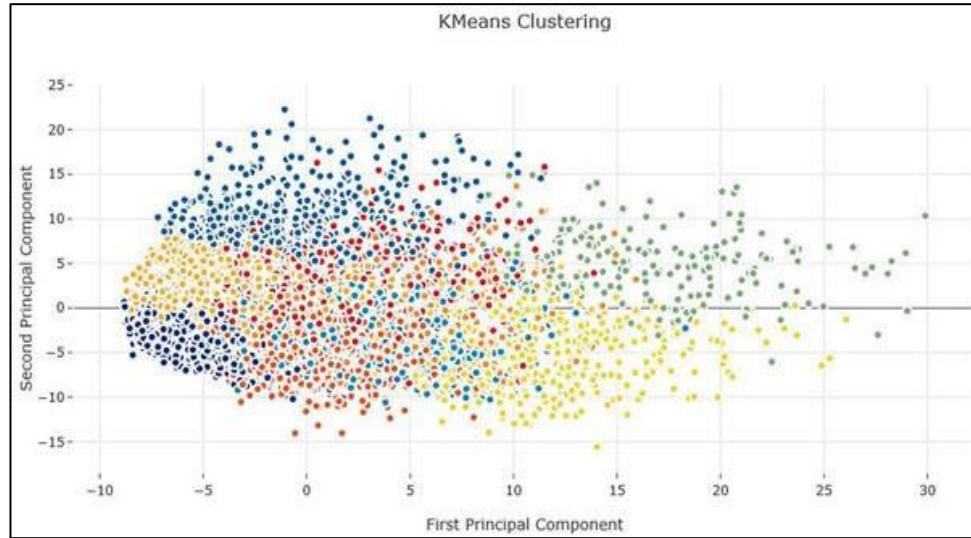
```
from sklearn.cluster import KMeans
# Set a KMeans clustering with 9 components
kmeans = KMeans(n_clusters=9)
# Compute cluster centers and predict cluster indices
X_clustered = kmeans.fit_predict(X_5d)

trace_Kmeans = go.Scatter(x=X_5d[:, 0], y= X_5d[:, 1], mode="markers",
                           showLegend=False,
                           marker=dict(
                               size=8,
                               color = X_clustered,
                               colorscale = 'Portland',
                               showscale=False,
                               line = dict(
                                   width = 2,
                                   color = 'rgb(255, 255, 255)'
                               )
                           ))
```

Gambar 14.34. *KMeans clustering code snippet*

```
layout = go.Layout(
    title= 'KMeans Clustering',
    hovermode= 'closest',
    xaxis= dict(
        title= 'First Principal Component',
        ticklen= 5,
        zeroline= False,
        gridwidth= 2,
    ),
    yaxis=dict(
        title= 'Second Principal Component',
        ticklen= 5,
        gridwidth= 2,
    ),
    showlegend= True
)
data = [trace_Kmeans]
fig1 = dict(data=data, layout= layout)
# fig1.append_trace(contour_list)
py.iplot(fig1, filename="svm")
```

Gambar 14.35. *clustering plot code snippet*



Gambar 14.36. *K-means clustering*

Secara visual, cluster yang dihasilkan oleh algoritma K-Means memberikan pemisahan yang lebih jelas di antara cluster, dibandingkan dengan hanya menambahkan label kelas ke dalam proyeksi PCA. Ini tidak mengherankan karena PCA dimaksudkan sebagai metode tanpa pengawasan, dan oleh karena itu tidak dioptimalkan untuk memisahkan label kelas yang berbeda.

VALIDATION OF UNSUPERVISED ML

Validasi pembelajaran tanpa pengawasan tergantung pada jenis algoritma yang digunakan. Beberapa teknik dapat diuji dengan metode seperti cross-validation, sementara yang lain menggunakan metrik internal atau eksternal untuk mengevaluasi hasilnya. Selain itu, validasi juga bisa dilakukan dalam konteks alur kerja yang lebih kompleks dengan mengukur akurasi atau performa ekstrinsik.

14.5 KESIMPULAN

Dalam bab ini, Anda telah mempelajari konsep dasar pembelajaran tanpa pengawasan beserta kasus penggunaan praktis teknik reduksi dimensionalitas. Sangat disarankan untuk menerapkan pembelajaran dari bab ini, bersama dengan teknik reduksi dimensionalitas berbasis pengawasan lainnya seperti LDA, dan membandingkan hasilnya. Seperti yang selalu dikatakan, praktikkan lebih banyak lagi pada berbagai dataset, dan Anda akan menemukan wawasan baru dalam setiap latihan. Pada bab berikutnya, Anda akan belajar cara mengatasi data deret waktu.

BAB XV

TIME SERIES DATA

15.1 STRUKTUR

Ini bab tentang data deret waktu. Data deret waktu adalah serangkaian poin data yang diurutkan berdasarkan waktu. Anda akan belajar cara bekerja dengan data deret waktu, menganalisis tren bisnis, meramalkan pendapatan, dan lainnya. Bab ini berisi tentang:

1. Urgensi Time-Series
2. Menangani tanggal dan waktu
3. Manipulasi data time-series
4. Membandingkan tingkat pertumbuhan deret waktu
5. Mengubah frekuensi deret waktu

15.2 TUJUAN

Setelah mempelajari bab ini, Anda akan dapat memanipulasi dan memvisualisasikan data deret waktu untuk mengekstrak statistik yang bermakna dan karakteristik lain dari data tersebut.

15.3 URGENSI TIME-SERIES

Deret waktu adalah kumpulan data yang diurutkan berdasarkan waktu. Dalam bab ini, Anda akan mempelajari pentingnya deret waktu, cara menangani tanggal dan waktu menggunakan Pandas, serta bagaimana melakukan transformasi dan manipulasi data deret waktu. Anda juga akan belajar tentang perbandingan tingkat pertumbuhan deret waktu dan cara mengubah frekuensi deret waktu. Bab ini akan membantu Anda memahami dan bekerja dengan data deret waktu secara efektif.

```
import pandas as pd
from datetime import datetime # for manually creating dates
```

Gambar 15.1. *Importing basic libraries*

Membuat sebuah dataframe Pandas dan memeriksa tipe datanya

```
# creating pandas timestamp
time_stamp = pd.Timestamp(datetime(2019,1,1))

# using a date string as datetime object
pd.Timestamp(datetime(2019,1,1)) == time_stamp

True

time_stamp
Timestamp('2019-01-01 00:00:00')
```

Gambar 15.2. *Creating pandas dataframe and checking datatype*

Timestamp Pandas memiliki atribut seperti `year`, `month`, `day`, dan lainnya yang dapat digunakan untuk mengakses informasi waktu dengan mudah.

time_stamp.year
2019
time_stamp.month
1
time_stamp.day
1

Gambar 15.3. *Accessing time-specific information*

Time series adalah kumpulan data waktu yang digunakan untuk analisis tren dan peramalan. Pandas memiliki fitur untuk penanganan tanggal dan waktu, serta modul statsmodels untuk analisis time series. Data tanggal dapat digunakan sebagai indeks Time Series dalam Pandas, dan dapat diubah frekuensinya dengan resampling.

```
period = pd.Period('2019-01')
print("period:: ", period)

#convert period to daily from month
print(period.asfreq('D'))

#convert period to timestamp back
print(period.to_timestamp().to_period('M'))

#basic date arithmetic operation
print(period + 3)

period:: 2019-01
2019-01-31
2019-01
2019-04
```

Gambar 15.4. *Handling time periods*

Selanjutnya, buatlah rangkaian waktu dengan urutan tanggal menggunakan fungsi date_range() dari pandas. Anda juga dapat mengonversi indeks menjadi indeks periode, seperti yang ditunjukkan pada sel berikutnya. Perhatikan tipe data pada hasilnya.

```
index = pd.date_range(start='2018-1-1', periods=12, freq='M')
index
DatetimeIndex(['2018-01-31', '2018-02-28', '2018-03-31', '2018-04-30',
               '2018-05-31', '2018-06-30', '2018-07-31', '2018-08-31',
               '2018-09-30', '2018-10-31', '2018-11-30', '2018-12-31'],
              dtype='datetime64[ns]', freq='M')

index[0]
Timestamp('2018-01-31 00:00:00', freq='M')

index.to_period()
PeriodIndex(['2018-01', '2018-02', '2018-03', '2018-04', '2018-05', '2018-06',
            '2018-07', '2018-08', '2018-09', '2018-10', '2018-11', '2018-12'],
            dtype='period[M]', freq='M')
```

Gambar 15.5. *How to use date_range()*

Dapat membuat rangkaian waktu dengan mudah. Misalnya, membuat rangkaian waktu pertama seperti ini:

```
pd.DataFrame({'date' : index}).info()
print("=====")
import numpy as np
my_data = np.random.rand(12, 2)
pd.DataFrame(data = my_data, index = index).info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 1 columns):
date    12 non-null datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 176.0 bytes
=====
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 12 entries, 2018-01-31 to 2018-12-31
Freq: M
Data columns (total 2 columns):
0    12 non-null float64
1    12 non-null float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

Gambar 15.6. *Creating a time series data with DataFrame*

Dalam output sebelumnya, setiap tanggal dalam pd.DatetimeIndex adalah pd.Timestamp, yang memiliki atribut yang memungkinkan Anda mengakses informasi tentang tanggal tersebut.

```

# Create the range of dates here
seven_days = pd.date_range('2019-1-1', periods=7)

# Iterate over the dates and print the number and name of the weekday
for day in seven_days:
    print(day.dayofweek, day.weekday_name)

1 Tuesday
2 Wednesday
3 Thursday
4 Friday
5 Saturday
6 Sunday
0 Monday

```

Gambar 15.7. Creating a week of data to obtain result

15.4 MENGUBAH DATA TIME-SERIES

Transformasi data adalah proses mengubah data deret waktu agar lebih baik dan dapat diolah dengan baik. Ini penting untuk menganalisis dan memahami data deret waktu dengan lebih baik. Sebagai contoh, Anda mungkin perlu mengonversi kolom tanggal dari tipe objek menjadi tipe datetime64, atau Anda mungkin ingin menghasilkan data baru dari data deret waktu yang ada. Dalam kasus ini, kita akan menggunakan data harga saham Google untuk mendemonstrasikan berbagai transformasi data.

```

google_df = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/timeseries/stock_data/google.csv")
google_df.head()

```

	Date	Close
0	2014-01-02	556.00
1	2014-01-03	551.95
2	2014-01-04	NaN
3	2014-01-05	NaN
4	2014-01-06	558.10

Gambar 15.8. Understanding importance of transformation

```

google_df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1094 entries, 0 to 1093
Data columns (total 2 columns):
Date      1094 non-null object
Close     756 non-null float64
dtypes: float64(1), object(1)
memory usage: 17.2+ KB

```

Gambar 15.9. Datatype is a string

Karena banyak algoritma machine learning tidak menerima input berupa string, Anda harus mengonversi tipe data kolom menjadi tipe data yang benar. Anda dapat mengonversi tipe data string menjadi dateTime64[ns].

```

google_df.Date = pd.to_datetime(google_df.Date)
google_df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1094 entries, 0 to 1093
Data columns (total 2 columns):
Date      1094 non-null datetime64[ns]
Close     756 non-null float64
dtypes: datetime64[ns](1), float64(1)

```

Gambar 15.10. Converting string datatype to dateTime64[ns]

Sekarang, kolom Tanggal kita adalah tipe data yang salah, dan kita dapat mengaturnya sebagai indeks seperti yang ditunjukkan dalam tangkapan layar berikut:

```

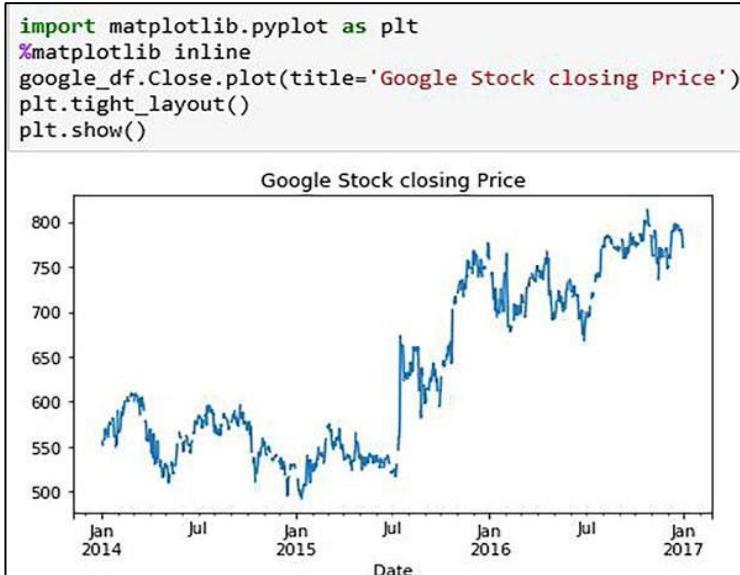
google_df.set_index('Date', inplace=True)
google_df.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1094 entries, 2014-01-02 to 2016-12-30
Data columns (total 1 columns):
Close    756 non-null float64
dtypes: float64(1)

```

Gambar 15.11. *How to set date as index*

Untuk mengubah tipe data string menjadi datetime64, gunakan metode `pd.to_datetime()` pada kolom tanggal.



Gambar 15.12. *Visualizing stock price data*

Anda mungkin telah memperhatikan bahwa tidak ada frekuensi pada indeks datetime kita; frekuensi harian dalam kalender dapat ditetapkan seperti yang ditunjukkan dalam tangkapan layar berikut:

```

google_df.asfreq('D').info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1094 entries, 2014-01-02 to 2016-12-30
Freq: D
Data columns (total 1 columns):
Close    756 non-null float64
dtypes: float64(1)

```

Gambar 15.13. *Setting calendar day frequency*

Setelah transformasi ini, mari periksa data baru kita, karena mungkin ada beberapa nilai null yang ditambahkan.

Close	
Date	
2014-01-02	556.00
2014-01-03	551.95
2014-01-04	NaN
2014-01-05	NaN
2014-01-06	558.10

Gambar 15.14. *Head of dataset*

15.5 MANIPULASI DATA TIME-SERIES

Manipulasi data deret waktu melibatkan pergeseran nilai-nilai dalam waktu, perhitungan perbedaan nilai untuk periode tertentu, atau perhitungan persentase perubahan dalam beberapa periode. Pandas memiliki metode bawaan untuk melakukan manipulasi semacam itu.

```
google_df = pd.read_csv("E:/pg/bpb/BPB-Publications/Datasets/timeseries/stock_data/google.csv",
                        parse_dates=['Date'],
                        index_col='Date')
google_df.head()
```

Date	Close
2014-01-02	556.00
2014-01-03	551.95
2014-01-04	NaN
2014-01-05	NaN
2014-01-06	558.10

Gambar 15.15. *Reloading Google stock price data with additional parameters*

Pandas melakukan parsing tanggal secara otomatis. Shift() menggeser satu periode ke masa depan secara default.

```
google_df['shifted'] = google_df.Close.shift()
google_df.head()
```

Date	Close	shifted
2014-01-02	556.00	NaN
2014-01-03	551.95	556.00
2014-01-04	NaN	551.95
2014-01-05	NaN	NaN
2014-01-06	558.10	NaN

Gambar 15.16. *Using shift() method*

Pandas menyediakan metode shift() untuk menggeser data deret waktu ke masa depan secara default, serta lagged() untuk menggesernya ke masa lalu. Selain itu, Anda dapat menghitung perubahan finansial satu periode atau pengembalian keuangan dengan menggunakan metode div() bersama dengan operasi aritmatika.

```
google_df['change'] = google_df.Close.div(google_df.shifted)
google_df.head()
```

Date	Close	shifted	change
2014-01-02	556.00	NaN	NaN
2014-01-03	551.95	556.00	0.992716
2014-01-04	NaN	551.95	NaN
2014-01-05	NaN	NaN	NaN
2014-01-06	558.10	NaN	NaN

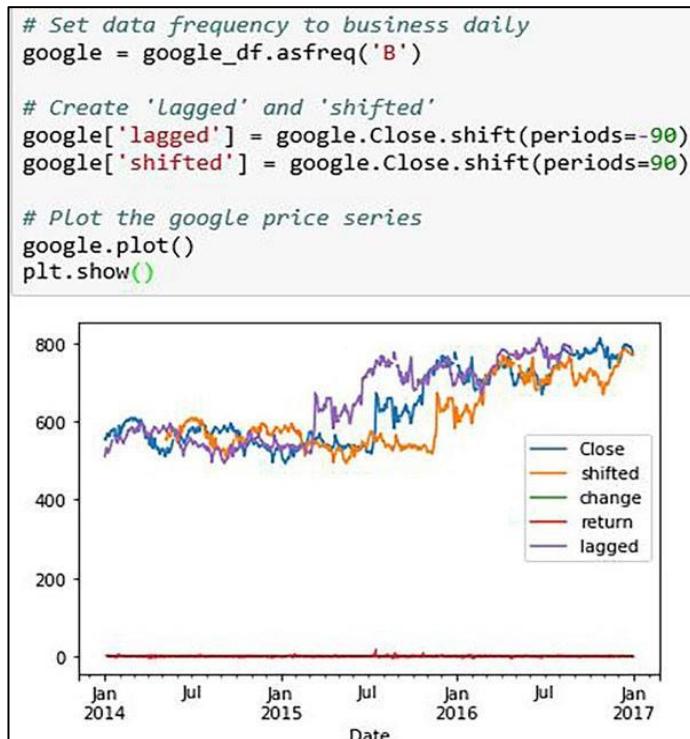
Gambar 15.17. *Calculating one period finance using div()*

```
google_df['return'] = google_df.change.sub(1).mul(100)
google_df.head()
```

Date	Close	shifted	change	return
2014-01-02	556.00	NaN	NaN	NaN
2014-01-03	551.95	556.00	0.992716	-0.728417
2014-01-04	NaN	551.95	NaN	NaN
2014-01-05	NaN	NaN	NaN	NaN
2014-01-06	558.10	NaN	NaN	NaN

Gambar 15.18. *Peek of the DataFrame*

Anda juga dapat menghitung perbedaan nilai antara dua periode berturut-turut menggunakan metode diff().



Gambar 15.19. *Visually comparing time series*

15.6 MEMBANDINGKAN TINGKAT PERTUMBUHAN DERET WAKTU

Membandingkan pertumbuhan deret waktu adalah tugas umum dalam analisis deret waktu. Sebagai contoh, membandingkan kinerja saham. Salah satu cara untuk membandingkan ini adalah dengan mengnormalisasi deret harga agar dimulai dari 100.



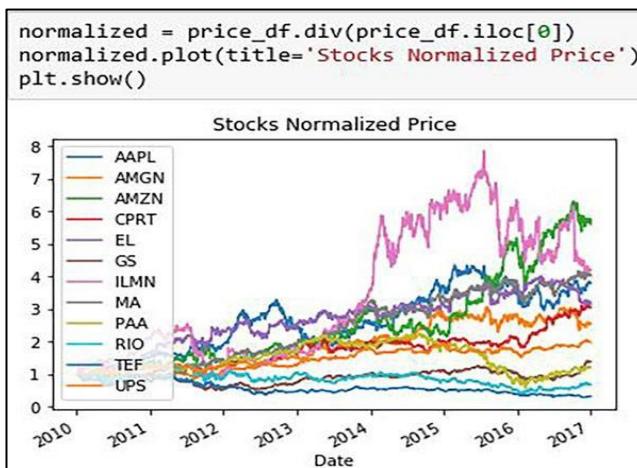
Gambar 15.20. *Comparing time-series growth*

Anda dapat mengnormalisasi beberapa deret dengan mudah.

Date	AAPL	AMGN	AMZN	CPRT	EL	GS	ILMN	MA	PAA	RIO	TEF	UPS
2010-01-04	30.57	57.72	133.90	4.55	24.27	173.08	30.55	25.68	27.00	56.03	28.55	58.18
2010-01-05	30.63	57.22	134.69	4.55	24.18	176.14	30.35	25.61	27.30	56.90	28.53	58.28
2010-01-06	30.14	56.79	132.25	4.53	24.25	174.26	32.22	25.56	27.29	58.64	28.23	57.85

Gambar 15.21. *Normalizing stock price of different companies*

Pada bagian ini, kami mempelajari bagaimana menggambarkan data time-series menggunakan Pandas. Ini termasuk pengenalan data time-series, cara menghandle tanggal dan waktu, dan berbagai metode transformasi dan manipulasi data time-series.



Gambar 15.22. *Using div() method*

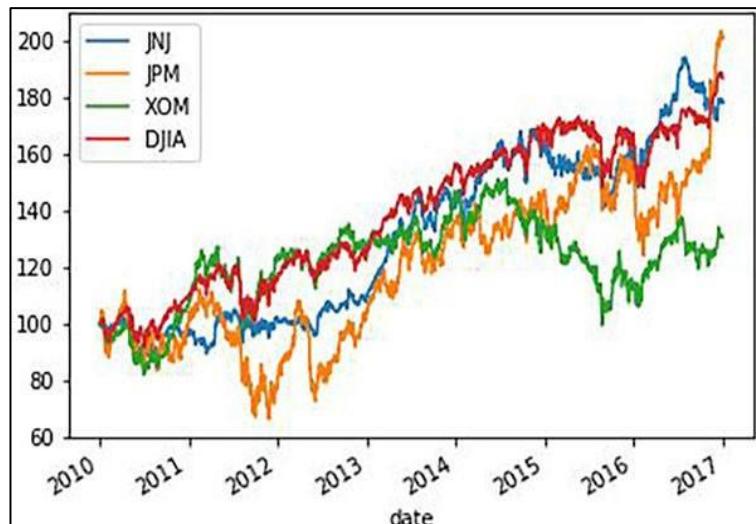
Dalam menormalisasi harga saham, Anda bisa membandingkan kinerja berbagai saham terhadap benchmark.

```
# Import stock prices and index here
stocks = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/timeseries/stock_data/nyse.csv',
                     parse_dates=['date'], index_col='date')
dow_jones = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/timeseries/stock_data/dow_jones.csv',
                       parse_dates=['date'], index_col='date')

# Concatenate data and inspect result
data = pd.concat([stocks, dow_jones], axis=1)
print(data.info())

# Normalize and plot your data
data.div(data.iloc[0]).mul(100).plot()
plt.show()
```

Gambar 15.23. *Comparing various stocks data*



Gambar 15.24. *Plot of the stocks comparison*

Selanjutnya, kita akan belajar bagaimana membandingkan kinerja Microsoft (MSFT) dan Apple (AAPL) dengan dataset S&P 500 selama 10 tahun terakhir.

```
# Create tickers
tickers = ['MSFT', 'AAPL']

# Import stock data here
stocks = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/timeseries/stock_data/msft_aapl.csv',
                     parse_dates=['date'], index_col='date')

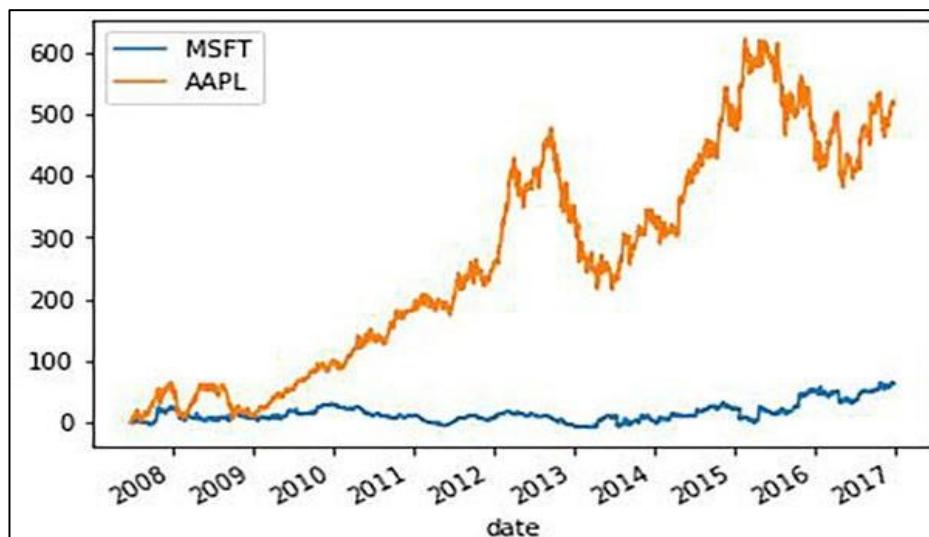
# Import index here
sp500 = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/timeseries/stock_data/sp500.csv',
                     parse_dates=['date'], index_col='date')

# Concatenate stocks and index here
data = pd.concat([stocks, sp500], axis=1).dropna()

# Normalize data
normalized = data.div(data.iloc[0]).mul(100)

# Subtract the normalized index from the normalized stock prices, and plot the result
normalized[tickers].sub(normalized['SP500'], axis=0).plot()
plt.show()
```

Gambar 15.25. Comparing performances of Apple and Microsoft stock data



Gambar 15.26. Plot of the comparison

15.7 MENGUBAH FREKUENSI TIME SERIES

Perubahan frekuensi juga memengaruhi data. Upsampling memerlukan penanganan nilai yang hilang, sedangkan downsampling memerlukan penggabungan data. Dalam contoh ini, kami akan mengubah frekuensi data dari kuartalan menjadi bulanan.

```
dates = pd.date_range(start='2018', periods=4, freq='Q')
my_data = range(1,5)
quaterly = pd.Series(data=my_data, index=dates)
quaterly

2018-03-31    1
2018-06-30    2
2018-09-30    3
2018-12-31    4
Freq: Q-DEC, dtype: int64

# upsampling quaterly to Month
monthly = quaterly.asfreq('M')
monthly

2018-03-31    1.0
2018-04-30    NaN
2018-05-31    NaN
2018-06-30    2.0
2018-07-31    NaN
2018-08-31    NaN
2018-09-30    3.0
2018-10-31    NaN
2018-11-30    NaN
2018-12-31    4.0
Freq: M, dtype: float64
```

Gambar 15.27. Using monthly frequency for upsampling and downsampling

	baseline	ffill	bfill
2018-03-31	1.0	1	1
2018-04-30	NaN	0	2
2018-05-31	NaN	0	2
2018-06-30	2.0	2	2
2018-07-31	NaN	0	3

Gambar 15.28. Handling missing values

Dalam kasus ini, kami akan mengeksplorasi metode interpolate() pada Pandas untuk mengisi nilai-nilai yang hilang dalam dataset yang baru.

```
# Import & inspect data
data_df = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/timeseries/stock_data/debt_unemployment.csv',
                      parse_dates=['date'],
                      index_col='date')
data_df.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 89 entries, 2010-01-01 to 2017-05-01
Data columns (total 2 columns):
Debt/GDP      89 non-null float64
Unemployment  89 non-null float64
dtypes: float64(2)
```

Gambar 15.29. Understanding interpolate() method

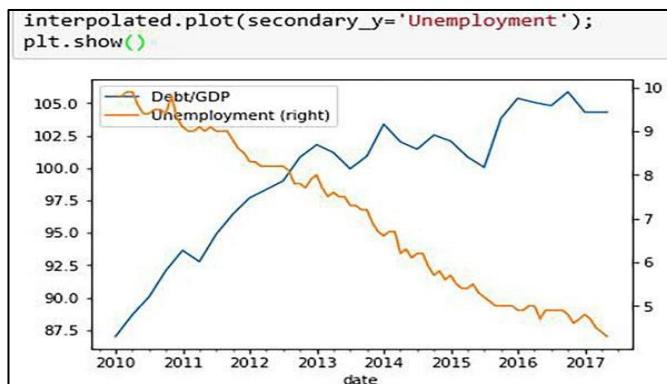
Sekarang kita akan melakukan interpolasi pada rasio utang/PDB dan membandingkannya dengan tingkat pengangguran, seperti yang ditunjukkan dalam tangkapan layar berikut:

```
interpolated = data_df.interpolate()
interpolated.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 89 entries, 2010-01-01 to 2017-05-01
Data columns (total 2 columns):
Debt/GDP      89 non-null float64
Unemployment  89 non-null float64
dtypes: float64(2)
```

Gambar 15.30. Interpolating dept/GDP

Kita dapat visualisasikan ini seperti yang ditunjukkan dalam tangkapan layar berikut:



Gambar 15.31. Plot of the GDP and unemployment data

Dalam plot sebelumnya, Debt/GDP (biru) meningkat, sedangkan tingkat pengangguran (coklat) menurun. Kita telah mempelajari upsampling, pengisian logika, dan interpolasi. Sekarang, kita akan fokus pada downsampling.

```

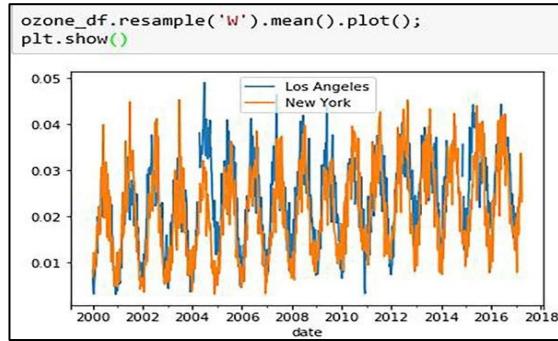
ozone_df = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/timeseries/air_quality_data/ozone_nyla.csv',
                      parse_dates=['date'], index_col='date')
ozone_df.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 6291 entries, 2000-01-01 to 2017-03-31
Data columns (total 2 columns):
Los Angeles    5488 non-null float64
New York       6167 non-null float64
dtypes: float64(2)

```

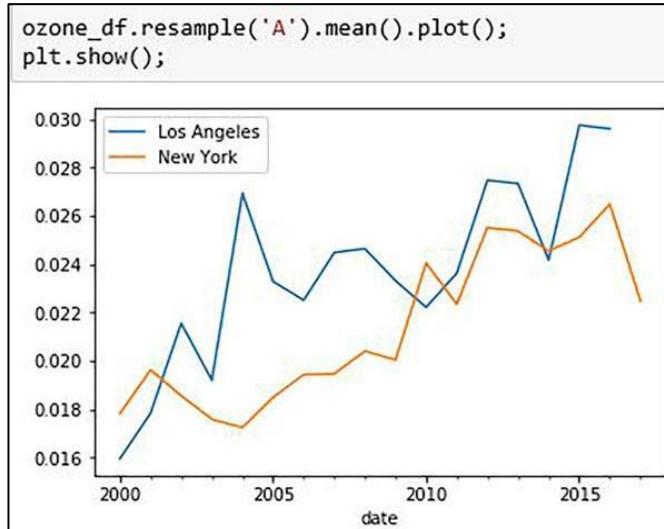
Gambar 15.33. To understand how to do downsampling

Pertama, kita menghitung dan memplot tren rata-rata ozon bulanan, seperti yang ditunjukkan dalam tangkapan layar berikut:



Gambar 15.34. Calculating and plotting monthly average ozone trend

Selanjutnya, kita menghitung dan memplot tren ozon rata-rata tahunan, seperti yang ditunjukkan dalam tangkapan layar berikut:



Gambar 15.35. Calculating and plotting annual average ozone trend

Anda dapat dengan mudah membandingkan seri harga saham frekuensi tinggi dengan seri waktu ekonomi frekuensi rendah. Sebagai contoh pertama, mari bandingkan tingkat pertumbuhan PDB triwulanan dengan tingkat pengembalian kuartalan pada indeks Dow Jones Industrial (yang diambil sampel ulang) dari 30 saham besar AS.

```

# Import and inspect gdp_growth
gdp_growth = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/timeseries/stock_data/gdp_growth.csv',
                         parse_dates=['date'], index_col='date')
gdp_growth.info()

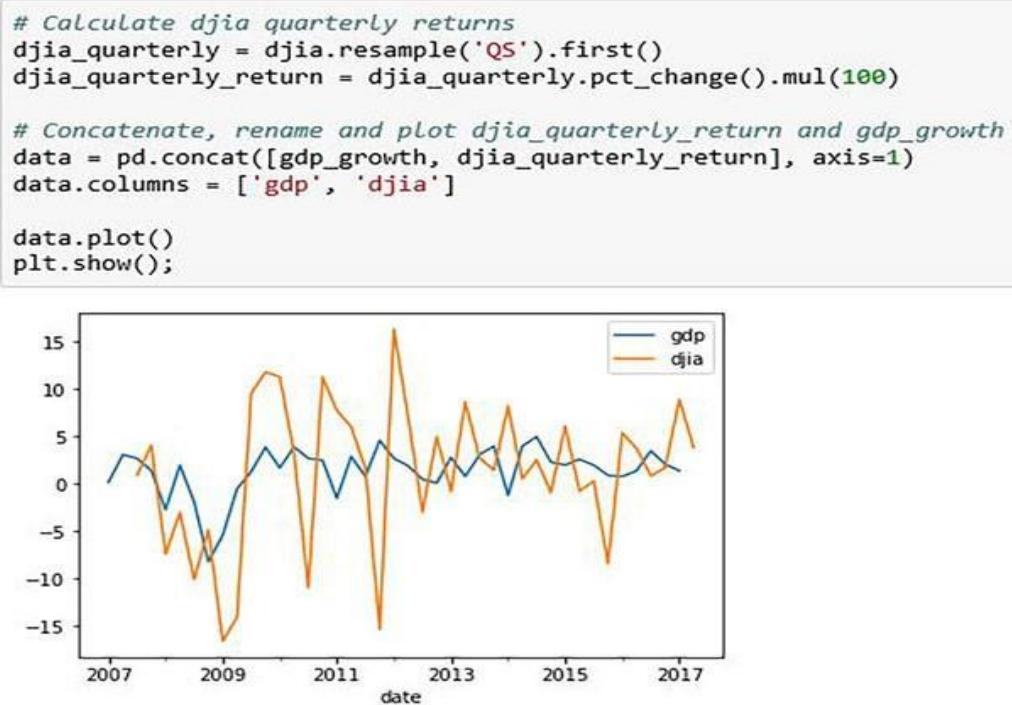
# Import and inspect djia
djia = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/timeseries/stock_data/djia.csv',
                   parse_dates=['date'], index_col='date')
djia.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 41 entries, 2007-01-01 to 2017-01-01
Data columns (total 1 columns):
gdp_growth    41 non-null float64
dtypes: float64(1)
memory usage: 656.0 bytes
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2610 entries, 2007-06-29 to 2017-06-29
Data columns (total 1 columns):
djia          2519 non-null float64
dtypes: float64(1)
memory usage: 40.8 KB

```

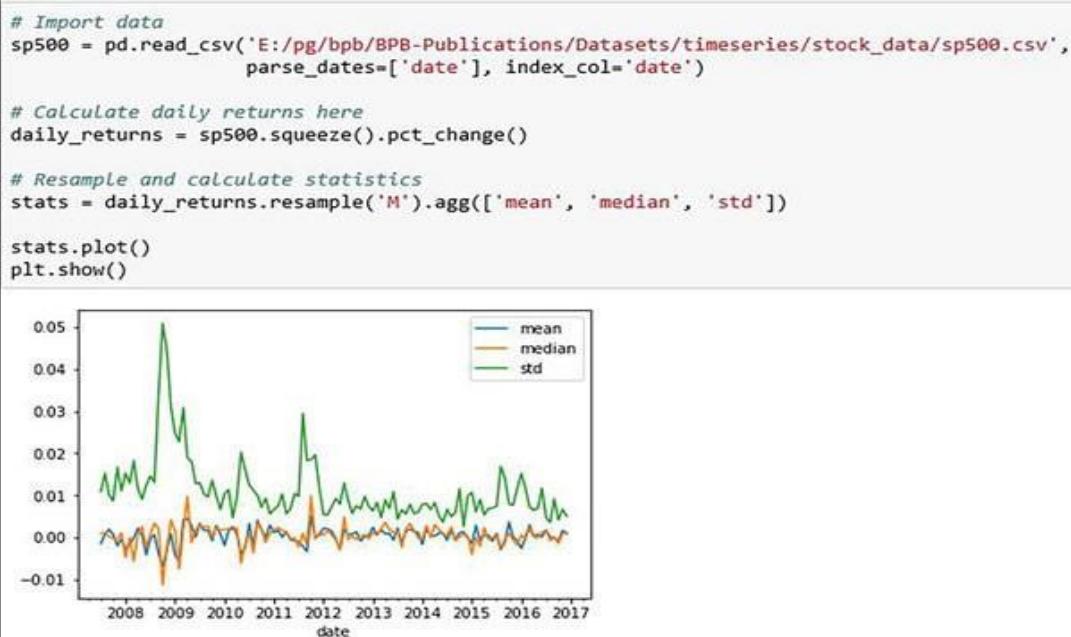
Gambar 15.36. Loading gdp growth and DJI datasets

Selanjutnya, kita akan menghitung tingkat pengembalian per kuartal dan memplotnya dengan pertumbuhan PDB.



Gambar 15.37. Calculating quarterly return and plotting with GDP growth

Rata-rata bulanan, median, dan deviasi standar dari pengembalian harian S&P500 telah berkembang selama 10 tahun terakhir dengan menggunakan metode resample().



Gambar 15.38. Aggregating mean, median, and standard deviation with resample() method

15.8 KESIMPULAN

Pembelajaran dari bab ini pasti akan membantu Anda ketika Anda bekerja dengan prediksi harga saham, prediksi cuaca, atau data penjualan.

BAB XVI

METODE TIME SERIES

16.1 STRUKTUR

Dalam bab ini, Anda akan memahami berbagai metode peramalan time series menggunakan perpustakaan statsmodels. Contoh kode yang disediakan dalam bab ini akan membantu Anda memahami dan mengaplikasikan metode-metode ini pada data time series Anda. Semua contoh yang dibahas dapat ditemukan dalam notebook Time Series Methods.ipynb. Bab ini mempelajari:

1. Peramalan time series
2. Langkah-langkah dasar dalam peramalan
3. Teknik-teknik peramalan time series
4. Peramalan lalu lintas masa depan ke halaman web

16.2 TUJUAN

Setelah mempelajari bab ini, Anda akan familiar dengan berbagai metode peramalan time series dan dapat mengaplikasikan teknik-teknik tersebut untuk meramalkan masalah time series apa pun.

16.3 PERAMALAN TIME SERIES

Peramalan time series adalah proses memprediksi nilai-nilai masa depan berdasarkan data historis dengan memanfaatkan dimensi waktu. Dalam peramalan ini, masa depan dianggap tidak dapat diprediksi sepenuhnya dan harus diestimasi berdasarkan apa yang telah terjadi sebelumnya. Peramalan time series dapat digunakan dalam berbagai konteks, seperti prediksi harga saham harian, penjualan produk harian di toko, jumlah penumpang harian di stasiun kereta, dan banyak lagi.

16.4 LANGKAH-LANGKAH PERAMALAN

Dua statistikawan terkenal, Dr. Hyndman telah merangkum 5 langkah dasar dalam peramalan, yang meliputi:

1. Pertimbangan bisnis: Memikirkan dengan cermat siapa yang memerlukan peramalan dan bagaimana peramalan tersebut akan digunakan.
2. Pengumpulan informasi: Mengumpulkan data historis untuk dianalisis dan dimodelkan. Ini juga mencakup akses ke para ahli domain dan pengumpulan informasi yang dapat membantu dalam menginterpretasikan data historis dan peramalan yang akan dibuat.
3. Eksplorasi data: Menggunakan alat sederhana seperti grafik dan statistik ringkas untuk lebih memahami data. Meninjau plot dan merangkum serta mencatat struktur temporal yang jelas, seperti tren, musiman, anomali seperti data yang hilang, korupsi, dan penciran, serta struktur lain yang dapat memengaruhi peramalan.
4. Pemilihan dan penyesuaian model: Mengevaluasi dua, tiga, atau sekelompok model berbagai jenis pada masalah yang dihadapi. Model-model ini dikonfigurasi dan disesuaikan dengan data historis.
5. Penggunaan dan evaluasi peramalan: Model digunakan untuk membuat peramalan, dan kinerja peramalan tersebut dievaluasi serta kemampuan model diestimasi.

16.5 TEKNIK-TEKNIK PERAMALAN TIME SERIES

Pustaka statsmodels memiliki banyak metode untuk peramalan time series. Berikut beberapa teknik umum yang akan kita bahas dalam bab ini:

AUTOREGRESSION (AR)

Autoregresi adalah model time series yang menggunakan data dari langkah waktu sebelumnya untuk memprediksi nilai pada langkah waktu berikutnya.

```
# Autoregression(AR) example
from statsmodels.tsa.ar_model import AR
from random import random
# create a sample dataset
data = [a + random() for a in range(1, 100)]
# fit model
model = AR(data)
model_fit = model.fit()
# make prediction
prediction = model_fit.predict(len(data), len(data))
prediction

array([100.6504561])
```

Gambar 16.1. *Example of using Autoregression model*

Pada sel kode sebelumnya, kita melakukan pemodelan likelihood maksimum tanpa syarat dari proses AR menggunakan `statsmodels.tsa.ar_model.AR.fit()`, dan kemudian kita mengembalikan prediksi dalam-sample dan out-of-sample menggunakan metode `predict()`. Metode `predict()` mengambil argumen pertama sebagai awal peramalan dan argumen kedua sebagai akhir peramalan. Dalam contoh kita, model autoregresi telah memprediksi nilai sebesar 100.65 untuk dataset sampel.

MOVING AVERAGE (MA)

Algoritma perataan digunakan untuk memprediksi data waktu yang tidak memiliki tren. Dalam model ini, kita mengambil rerata aritmatika dari beberapa pengamatan terakhir untuk memprediksi nilai berikutnya.

```
# Moving Average(MA) example
from statsmodels.tsa.arima_model import ARMA
from random import random
# create a sample dataset
data = [a + random() for a in range(1, 100)]
# fit model
model = ARMA(data, order=(0, 1))
model_fit = model.fit(disp=False)
# make prediction
ma_predict = model_fit.predict(len(data), len(data))
ma_predict

array([75.32920306])
```

Gambar 16.2. *Order of MA model in order argument*

Pada sel kode sebelumnya, kami telah memasang model MA dengan maksimum likelihood yang tepat melalui filter Kalman menggunakan metode statsmodels.tsa.arima_model.fit() dengan parameter disp sebagai false, dan kemudian kami mengembalikan prediksi dalam sampel dan luar sampel menggunakan metode predict().

AUTOREGRESSIVE MOVING AVERAGE (ARMA)

Pada sel kode sebelumnya, kami telah menerapkan model ARMA menggunakan metode exact maximum likelihood melalui filter Kalman dengan metode statsmodels.tsa.arima_model.fit(), dan kemudian kami menggunakan metode predict() untuk

menghasilkan prediksi dalam sampel dan luar sampel. Dalam contoh dataset kami, prediksi moving average adalah 0.58.

```
# ARMA example
from statsmodels.tsa.arima_model import ARMA
from random import random
# create a sample dataset
data = [random() for x in range(1, 100)]
# fit model
model = ARMA(data, order=(2, 1))
model_fit = model.fit(disp=False)
# make prediction
arma_pred = model_fit.predict(len(data), len(data))
arma_pred

array([0.58318755])
```

Gambar 16.3. ARMA model example

AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA)

Metode ARIMA menggabungkan Autoregression, Moving Average, dan differencing untuk membuat data stationary. Model ini dikenal sebagai ARIMA(p, d, q), dengan tiga parameter yang memperhitungkan musiman, tren, dan noise dalam data.

```
# ARIMA example
from statsmodels.tsa.arima_model import ARIMA
from random import random
# create a sample dataset
data = [x + random() for x in range(1, 100)]
# fit model
model = ARIMA(data, order=(1, 1, 1))
model_fit = model.fit(disp=False)
# make prediction
arima_pred = model_fit.predict(len(data), len(data), typ='levels')
arima_pred

array([100.57016203])
```

Gambar 16.4. ARIMA model example

SEASONAL AUTOREGRESSIVE INTEGRATED MOVING-AVERAGE (SARIMA)

SARIMA (Seasonal Autoregressive Integrated Moving-Average) adalah ekstensi dari ARIMA yang mendukung pemodelan komponen musiman dalam rangkaian waktu. SARIMA menggabungkan model ARIMA dengan kemampuan untuk melakukan autoregression, differencing, dan moving average pada tingkat musiman. Metode ini cocok untuk data time series univariat dengan tren dan/atau komponen musiman. Perbedaan besar antara model ARIMA dan model SARIMA adalah penambahan komponen error musiman ke dalam model.

```
# SARIMA example
from statsmodels.tsa.statespace.sarimax import SARIMAX
from random import random
# create a sample dataset
data = [x + random() for x in range(1, 100)]
# fit model
model = SARIMAX(data, order=(1, 1, 1), seasonal_order=(1, 1, 1, 1))
model_fit = model.fit(disp=False)
# make prediction
sarima_pred = model_fit.predict(len(data), len(data))
sarima_pred

array([100.5407515])
```

Gambar 16.5. SARIMA model

SEASONAL AUTOREGRESSIVE INTEGRATED MOVING-AVERAGE DENGAN REGRESOR EKSTERNAL (SARIMAX)

Perluasan dari model SARIMA dengan tambahan pemodelan variabel eksogen. Ini berguna untuk memperbaiki proyeksi dalam situasi seperti rantai pasokan makanan dengan banyak faktor yang memengaruhi permintaan harian.

```
# SARIMAX example
from statsmodels.tsa.statespace.sarimax import SARIMAX
from random import random
# create datasets
data1 = [x + random() for x in range(1, 100)]
data2 = [x + random() for x in range(101, 200)]
# fit model
model = SARIMAX(data1, exog=data2, order=(1, 1, 1), seasonal_order=(0, 0, 0, 0))
model_fit = model.fit(disp=False)
# make prediction
exog2 = [200 + random()]
sarimax_pred = model_fit.predict(len(data1), len(data1), exog=[exog2])
sarimax_pred
array([100.09306379])
```

Gambar 16.6. *SARIMAX method*

VECTOR AUTOREGRESSION MOVING-AVERAGE (VARMA)

Metode VARMA memodelkan langkah selanjutnya dalam setiap time-series menggunakan model ARMA. Ini adalah generalisasi dari ARMA untuk time-series multivariat. Metode ini cocok untuk time-series multivariat tanpa tren dan komponen musiman.

```
# VARMA example
from statsmodels.tsa.statespace.varmax import VARMAX
from random import random
# create dataset with dependency
data = list()
for i in range(100):
    v1 = random()
    v2 = v1 + random()
    row = [v1, v2]
    data.append(row)
# fit model
model = VARMAX(data, order=(1, 1))
model_fit = model.fit(disp=False)
# make prediction
varma_pred = model_fit.forecast()
varma_pred
array([[0.58299814, 1.10249435]])
```

Gambar 16.7. *VARMA method*

HOLT WINTER'S EXPONENTIAL SMOOTHING (HWES)

Metode Holt Winter's Exponential Smoothing, juga disebut metode Triple Exponential Smoothing, memodelkan langkah waktu berikutnya sebagai fungsi linier tertimbang secara eksponensial dari pengamatan pada langkah-langkah waktu sebelumnya, dengan mempertimbangkan tren dan musiman. Metode ini cocok untuk time-series univariat dengan tren dan/atau komponen musiman.

```
# HWES example
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from random import random
# create dataset
data = [x + random() for x in range(1, 100)]
# fit model
model = ExponentialSmoothing(data)
model_fit = model.fit()
# make prediction
hwes_pred = model_fit.predict(len(data), len(data))
hwes_pred
array([99.90409631])
```

Gambar 16.8. *HWES method*

16.6 PERAMALAN HALAMAN WEB

Sekarang saatnya untuk menerapkan pembelajaran dari bab ini dan bab sebelumnya untuk menyelesaikan masalah time-series yang sesungguhnya. Pada latihan berikut, tujuan Anda adalah meramalkan lalu lintas masa depan ke halaman Wikipedia. Anda dapat mengunduh dataset yang diperlukan untuk latihan ini dari repositori kami. Mari mulai analisis dengan memuat dataset tersebut:

	Page	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08	2015-07-09	...	2016-12-22	2016-12-23
0	2NE1_zh.wikipedia.org_all-access_spider	18.0	11.0	5.0	13.0	14.0	9.0	9.0	22.0	26.0	...	32.0	63.0
1	2PM_zh.wikipedia.org_all-access_spider	11.0	14.0	15.0	18.0	11.0	13.0	22.0	11.0	10.0	...	17.0	42.0
2	3C_zh.wikipedia.org_all-access_spider	1.0	0.0	1.0	1.0	0.0	4.0	0.0	3.0	4.0	...	3.0	1.0
3	4minute_zh.wikipedia.org_all-access_spider	35.0	13.0	10.0	94.0	4.0	26.0	14.0	9.0	11.0	...	32.0	10.0
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_s...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	48.0	9.0

5 rows x 551 columns

Gambar 16.9. Loading the dataset

Kita akan menggunakan regulasi yang lebih sederhana.

```
def get_language(page):
    res = re.search('[a-z][a-z].wikipedia.org',page)
    if res:
        return res[0][0:2]
    return 'na'

train['lang'] = train.Page.map(get_language)
from collections import Counter
Counter(train.lang)

Counter({'de': 18547,
         'en': 24108,
         'es': 14069,
         'fr': 17802,
         'ja': 20431,
         'na': 17855,
         'ru': 15022,
         'zh': 17229})
```

Gambar 16.10. Searching language code using regular expression

```
lang_sets = {}
lang_sets['de'] = train[train.lang=='de'].iloc[:,0:-1]
lang_sets['en'] = train[train.lang=='en'].iloc[:,0:-1]
lang_sets['es'] = train[train.lang=='es'].iloc[:,0:-1]
lang_sets['fr'] = train[train.lang=='fr'].iloc[:,0:-1]
lang_sets['ja'] = train[train.lang=='ja'].iloc[:,0:-1]
lang_sets['na'] = train[train.lang=='na'].iloc[:,0:-1]
lang_sets['ru'] = train[train.lang=='ru'].iloc[:,0:-1]
lang_sets['zh'] = train[train.lang=='zh'].iloc[:,0:-1]

sums = {}
for key in lang_sets:
    sums[key] = lang_sets[key].iloc[:,1:].sum(axis=0) / lang_sets[key].shape[0]
```

Gambar 16.11. Creating dataframes to hold each language

Gambarkan semua set yang berbeda pada plot yang sama untuk mengetahui bagaimana jumlah total tampilan berubah seiring waktu:

```

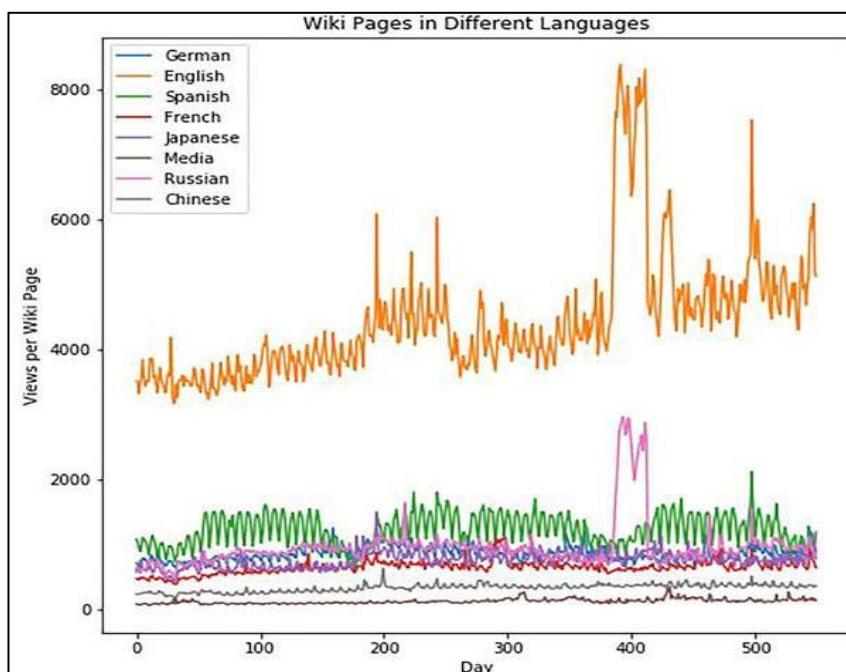
days = [r for r in range(sums['en'].shape[0])]
fig = plt.figure(1, figsize=[10,10])
plt.ylabel('Views per Wiki Page')
plt.xlabel('Day')
plt.title('Wiki Pages in Different Languages')
labels={ 'en': 'English', 'ja': 'Japanese', 'de': 'German',
        'na': 'Media', 'fr': 'French', 'zh': 'Chinese',
        'ru': 'Russian', 'es': 'Spanish'
    }

for key in sums:
    plt.plot(days,sums[key],label = labels[key] )

plt.legend()
plt.show()

```

Gambar 16.12. Plotting different sets on same plot



Gambar 16.13. Wiki pages in different languages

Dari plot tersebut, dapat disimpulkan bahwa bahasa Inggris memiliki jumlah tampilan per halaman yang tinggi. Terdapat lonjakan di bahasa Inggris dan Rusia sekitar hari ke-400, serta pola aneh di bahasa Inggris sekitar hari ke-200. Data bahasa Spanyol menunjukkan struktur periodik. Selanjutnya, kami akan analisis frekuensi terkuat dalam sinyal periodik menggunakan FFT.

```

from scipy.fftpack import fft
def plot_with_fft(key):

    fig = plt.figure(1, figsize=[15,5])
    plt.ylabel('Views per Page')
    plt.xlabel('Day')
    plt.title(labels[key])
    plt.plot(days,sums[key],label = labels[key] )

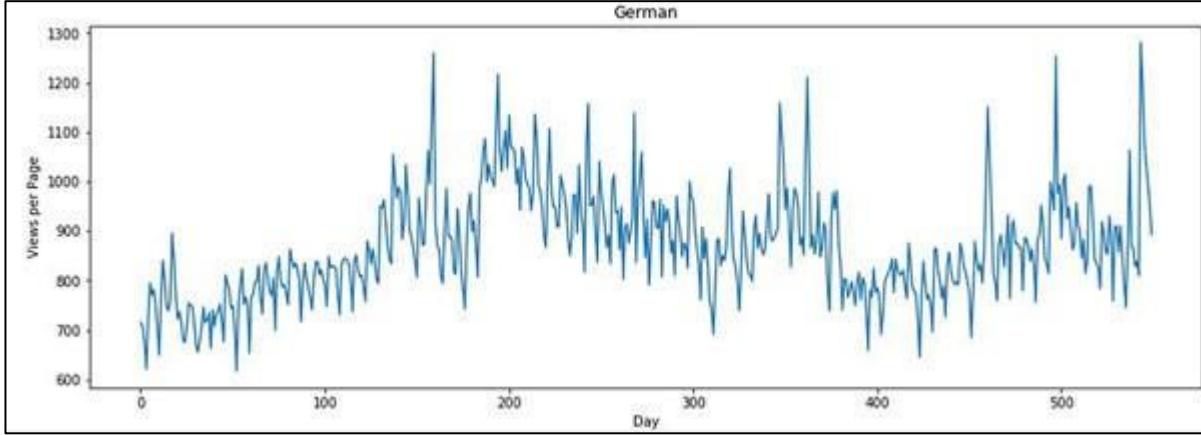
    fig = plt.figure(2, figsize=[15,5])
    fft_complex = fft(sums[key])
    fft_mag = [np.sqrt(np.real(x)*np.real(x)+np.imag(x)*np.imag(x)) for x in fft_complex]
    fft_xvals = [day / days[-1] for day in days]
    npts = len(fft_xvals) // 2 + 1
    fft_mag = fft_mag[:npts]
    fft_xvals = fft_xvals[:npts]

    plt.ylabel('FFT Magnitude')
    plt.xlabel(r"Frequency [days]$^{-1}$")
    plt.title('Fourier Transform')
    plt.plot(fft_xvals[1:],fft_mag[1:],label = labels[key] )
    plt.axvline(x=1./7,color='red',alpha=0.3)
    plt.axvline(x=2./7,color='red',alpha=0.3)
    plt.axvline(x=3./7,color='red',alpha=0.3)

    plt.show()
for key in sums:
    plot_with_fft(key)

```

Gambar 16.14. One snippet for FFT magnitude



Gambar 16.15. *German plot*

Dari plot, data bahasa Spanyol memiliki fitur periodik yang kuat, sementara bahasa Rusia dan media tidak menunjukkan pola jelas. Puncak pada FFT terutama pada periode 1 dan 1/2 minggu. Model mungkin sulit memprediksi lonjakan tiba-tiba tanpa informasi tambahan tentang peristiwa khusus. Selanjutnya, kita akan melihat halaman paling populer dalam setiap bahasa untuk memahami tren yang signifikan.

```
# For each Language get highest few pages
nPages = 5
top_pages = {}
for key in lang_sets:
    print(key)
    sum_set = pd.DataFrame(lang_sets[key][['Page']])
    sum_set['total'] = lang_sets[key].sum(axis=1)
    sum_set = sum_set.sort_values('total', ascending=False)
    print(sum_set.head(10))
    top_pages[key] = sum_set.index[0]
print('\n\n')
```

Gambar 16.16. *Most popular pages code snippet*

de	Page	total
139119	Wikipedia:Hauptseite_de.wikipedia.org_all-access...	1.603934e+09
116196	Wikipedia:Hauptseite_de.wikipedia.org_mobile-w...	1.112689e+09
67049	Wikipedia:Hauptseite_de.wikipedia.org_desktop_...	4.269924e+08
140151	Spezial:Suche_de.wikipedia.org_all-access_all...	2.234259e+08
66736	Spezial:Suche_de.wikipedia.org_desktop_all-agents	2.196368e+08
140147	Spezial:Anmelden_de.wikipedia.org_all-access_a...	4.029181e+07
138800	Special:Search_de.wikipedia.org_all-access_all...	3.988154e+07
68104	Spezial:Anmelden_de.wikipedia.org_desktop_all...	3.535523e+07
68511	Special:MyPage/toolserverhelperleinconfig.js_d...	3.258496e+07

Gambar 16.17. *Popular pages of German language*

Sebelumnya, kita telah melihat bahwa paket statsmodels menyertakan sejumlah alat untuk melakukan analisis deret waktu. Di sini, saya akan menunjukkan autocorrelation dan partial autocorrelation untuk halaman paling banyak dilihat dalam setiap bahasa. Kedua metode ini menunjukkan korelasi sinyal dengan versi yang tertunda dari dirinya sendiri. Pada setiap lag, partial autocorrelation mencoba menunjukkan korelasi pada lag tersebut setelah menghapus korelasi pada lag-lag yang lebih pendek:

```

from statsmodels.tsa.stattools import pacf
from statsmodels.tsa.stattools import acf
for key in top_pages:
    fig = plt.figure(1,figsize=[10,5])
    ax1 = fig.add_subplot(121)
    ax2 = fig.add_subplot(122)
    cols = train.columns[1:-1]
    data = np.array(train.loc[top_pages[key],cols])
    data_diff = [data[i] - data[i-1] for i in range(1,len(data))]
    autocorr = acf(data_diff)
    pac = pacf(data_diff)

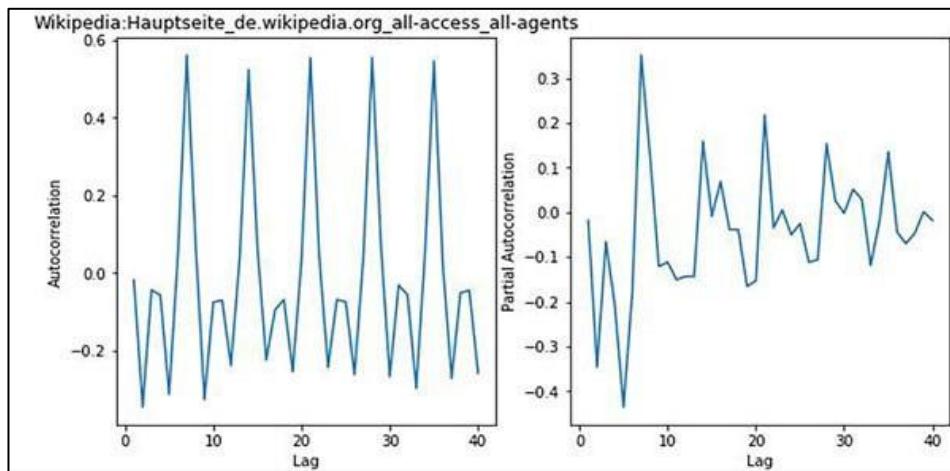
    x = [x for x in range(len(pac))]
    ax1.plot(x[1:],autocorr[1:])

    ax2.plot(x[1:],pac[1:])
    ax1.set_xlabel('Lag')
    ax1.set_ylabel('Autocorrelation')
    ax1.set_title(train.loc[top_pages[key],'Page'])

    ax2.set_xlabel('Lag')
    ax2.set_ylabel('Partial Autocorrelation')
    plt.show()

```

Gambar 16.18. *autocorrelation and partial autocorrelation*



Gambar 16.19. *Partial autocorrelation*

Mari terapkan salah satu metode peramalan statistik klasik - model ARIMA - untuk satu set halaman kecil, dan kemudian kita akan melihat wawasan yang kita dapatkan dari plot ini:

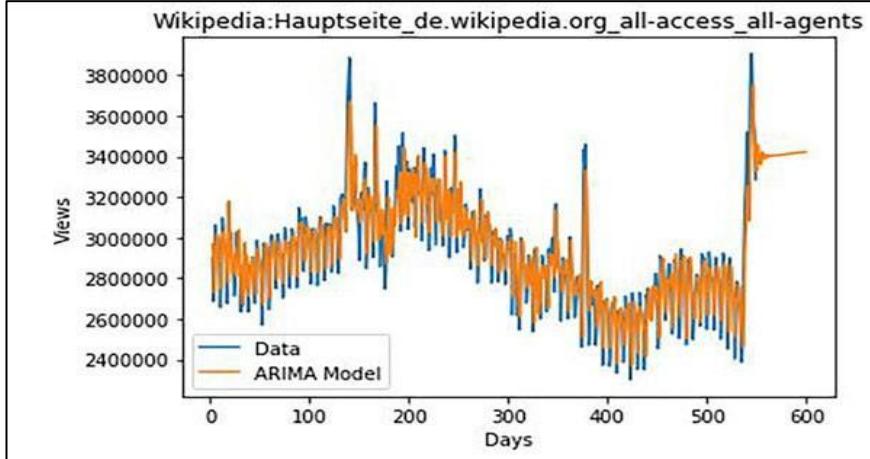
```

cols = train.columns[1:-1]
for key in top_pages:
    data = np.array(train.loc[top_pages[key],cols], 'f')
    result = None
    with warnings.catch_warnings():
        warnings.filterwarnings('ignore')
        try:
            arima = ARIMA(data,[2,1,4])
            result = arima.fit(disp=False)
        except:
            try:
                arima = ARIMA(data,[2,1,2])
                result = arima.fit(disp=False)
            except:
                print(train.loc[top_pages[key],'Page'])
                print('\tARIMA failed')
    pred = result.predict(2,599,typ='levels')
    x = [i for i in range(600)]
    i=0

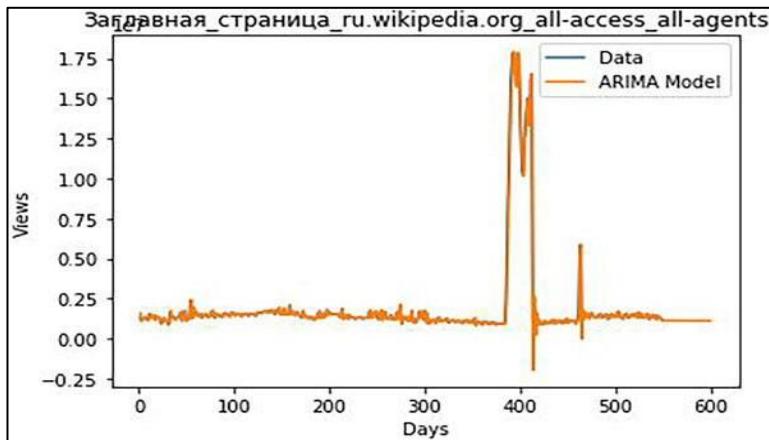
    plt.plot(x[2:len(data)],data[2:] ,label='Data')
    plt.plot(x[2:],pred,label='ARIMA Model')
    plt.title(train.loc[top_pages[key],'Page'])
    plt.xlabel('Days')
    plt.ylabel('Views')
    plt.legend()
    plt.show()

```

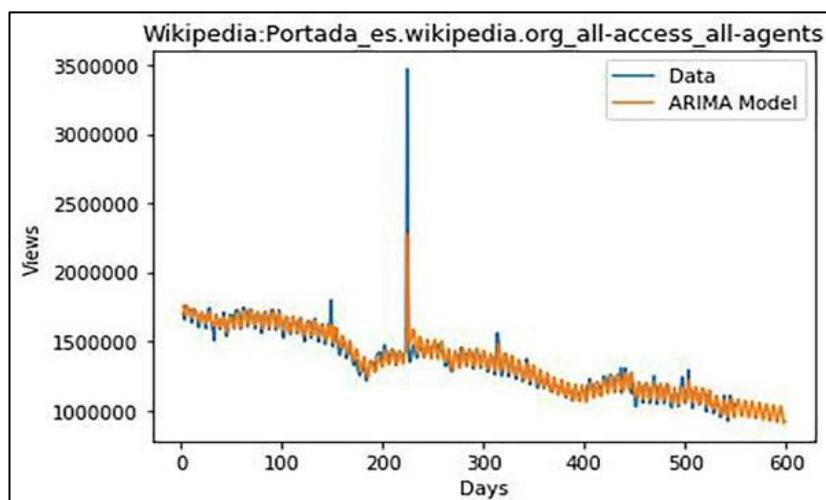
Gambar 16.20. *ARIMA model applied*



Gambar 16.21. ARIMA model denoted in orange line



Gambar 16.22. ARIMA model vs Data plot example



Gambar 16.23. ARIMA model vs Data plot example

Anda akan memahami bahwa model ARIMA, dalam beberapa kasus, mampu memprediksi substruktur mingguan sinyal dengan efektif.

16.7 KESIMPULAN

Metode peramalan time-series adalah yang paling dapat diandalkan ketika data mewakili periode waktu yang luas. Informasi tentang kondisi dapat diekstraksi dengan mengukur data pada berbagai interval waktu - misalnya, per jam, harian, bulanan, triwulan, tahunan, atau pada interval waktu lainnya. Peramalan paling kuat ketika berdasarkan pada sejumlah besar pengamatan untuk periode waktu yang lebih lama untuk mengukur pola dalam kondisi. Untuk mendapatkan lebih banyak keyakinan, mulailah bekerja pada dataset saham atau cuaca baru apa pun, dan terapkan pembelajaran bab ini dalam peramalan. Di bab berikutnya, Anda akan melalui berbagai contoh studi kasus.

BAB XVII

STUDI KASUS 1

17.1 PEMBAYARAN PINJAMAN

1. Tujuan Anda: Dalam studi pertama kami, Anda bekerja untuk klien asuransi dan membantu mereka mengimplementasikan model machine learning untuk memprediksi probabilitas apakah seorang pemohon akan mampu atau tidak membayar kembali pinjaman.
2. Klien Anda: Klien Anda adalah penyedia keuangan konsumen internasional yang beroperasi di 10 negara.
3. Tentang dataset: Klien telah memberikan dataset mereka yang memiliki data statis untuk semua aplikasi. Satu baris mewakili satu pinjaman dalam data mereka, yang dapat Anda unduh dari repositori kami.
4. Model ML dasar kami: Dalam contoh ini, saya akan menerapkan algoritma regresi logistik dan hutan acak.

Karena tujuan dari kompetisi ini adalah menggunakan data aplikasi pinjaman historis untuk memprediksi apakah seorang pemohon akan mampu atau tidak membayar kembali pinjaman, ini adalah tugas klasifikasi terawasi standar. Mari impor semua perpustakaan dasar yang diperlukan dan baca dataset:

17.2 PROSES

PERSIAPAN DATA

```
# numpy and pandas for data manipulation
import numpy as np
import pandas as pd

# sklearn preprocessing for dealing with categorical variables
from sklearn.preprocessing import LabelEncoder

# Suppress warnings
import warnings
warnings.filterwarnings('ignore')

# matplotlib and seaborn for plotting
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

# Load and explore training data
train_df = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/Case Studies/case_study_1/application_train.csv')
print('Training data shape: ', train_df.shape)
train_df.head()
```

Gambar 17.1. Importing basic libraries and the dataset

Training data shape: (307511, 122)						
SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	
0	100002	1	Cash loans	M	N	Y
1	100003	0	Cash loans	F	N	N
2	100004	0	Revolving loans	M	Y	Y
3	100006	0	Cash loans	F	N	Y
4	100007	0	Cash loans	M	N	Y

5 rows x 122 columns

Gambar 17.2. Peek of the imported dataset

Data pelatihan terdiri dari 307.511 observasi dan 122 fitur, termasuk TARGET yang akan diprediksi. Data pelatihan berisi informasi tentang setiap pinjaman, dan TARGET mengindikasikan apakah pinjaman tersebut "0: telah dibayar kembali" atau "1: tidak dibayar kembali." Kami juga memiliki dataset pengujian terpisah. Analisis Data Eksploratori kami menunjukkan bahwa ini adalah masalah kelas yang tidak seimbang, dengan lebih banyak pinjaman yang sudah dibayar kembali tepat waktu.

```

# Load and explore testing data
test_df = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/Case Studies/case_study_1/application_test.csv')
print('Testing data shape: ', test_df.shape)
test_df.head()

Testing data shape: (48744, 121)

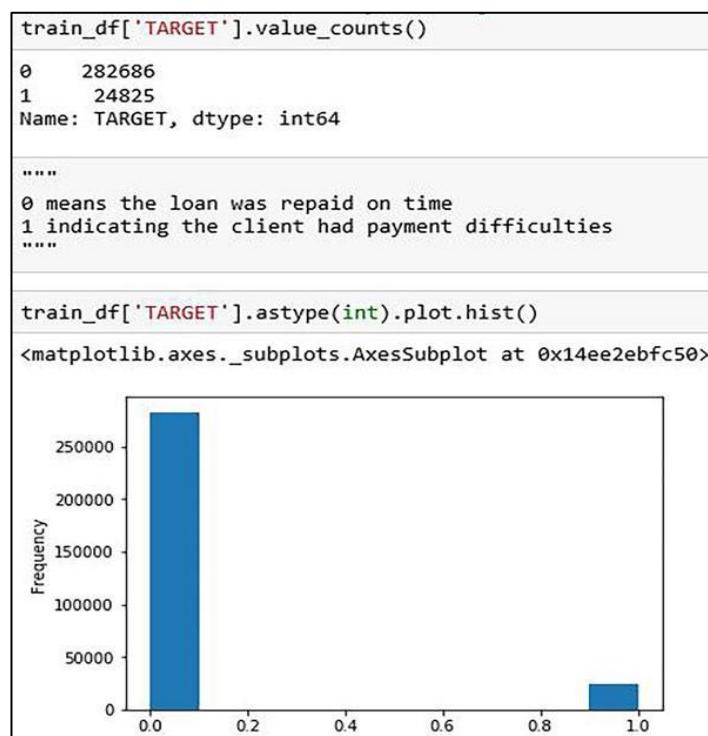
   SK_ID_CURR NAME_CONTRACT_TYPE CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY CNT_CHILDREN AMT_INCOME_TOTAL
0    100001      Cash loans          F           N            Y             0        135000.0
1    100005      Cash loans          M           N            Y             0        99000.0
2    100013      Cash loans          M           Y            Y             0        202500.0
3    100028      Cash loans          F           N            Y             2        315000.0
4    100038      Cash loans          M           Y            N             1        180000.0

5 rows × 121 columns

```

Gambar 17.3. *Loading Testing dataset*

Selanjutnya, kami akan melakukan Analisis Data Eksploratori (EDA) untuk menemukan tren dan pola dalam data pinjaman. Target prediksi kami adalah menentukan apakah pinjaman akan dibayar tepat waktu ("0") atau klien mengalami kesulitan pembayaran ("1"). Dari grafik berikut, terlihat bahwa ini adalah masalah dengan ketidakseimbangan kelas, di mana lebih banyak pinjaman yang sudah dibayar tepat waktu dibandingkan dengan yang gagal bayar.



Gambar 17.4. *Distribution of the target*

Kita akan memeriksa nilai-nilai yang hilang dalam dataset menggunakan fungsi.

```

# check number and percentage of missing values in each column
def missing_values_table(df):
    mis_val = df.isnull().sum() # Total missing values
    mis_val_percent = 100 * df.isnull().sum() / len(df) # Percentage of missing values
    mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1) # Make a table with the results
    mis_val_table.columns = mis_val_table.rename(
    columns = {0 : 'Missing Values', 1 : '% of Total Values'}) # Rename the columns

    # Sort the table by percentage of missing descending
    mis_val_table['% of Total Values'].sort_values(
        ascending=False).round(1)

    # Print some summary information
    print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
    "There are " + str(mis_val_table['% of Total Values'].shape[0]) +
    " columns that have missing values.")

    # Return the dataframe with missing information
    return mis_val_table['% of Total Values']

```

Gambar 17.5. *Examining missing values*

Missing Values % of Total Values		
COMMONAREA_MEDI	214865	69.9
COMMONAREA_AVG	214865	69.9
COMMONAREA_MODE	214865	69.9
NONLIVINGAPARTMENTS_MEDI	213514	69.4
NONLIVINGAPARTMENTS_MODE	213514	69.4

Gambar 17.6. *Missing values statistics*

# data types of each type of column	
float64	65
int64	41
object	16
dtype:	int64

# Number of unique classes in each object column	
NAME_CONTRACT_TYPE	2
CODE_GENDER	3
FLAG_OWN_CAR	2
FLAG_OWN_REALTY	2
NAME_TYPE_SUITE	7
NAME_INCOME_TYPE	8
NAME_EDUCATION_TYPE	5
NAME_FAMILY_STATUS	6
NAME_HOUSING_TYPE	6
OCCUPATION_TYPE	18
WEEKDAY_APPR_PROCESS_START	7
ORGANIZATION_TYPE	58
FONDKAPREMONT_MODE	4
HOUSETYPE_MODE	3
WALLSMATERIAL_MODE	7
EMERGENCYSTATE_MODE	2

Gambar 17.7. *Checking unique entries*

Kodekan variabel kategorikal menggunakan metode label encoding atau one-hot encoding tergantung pada jumlah nilai uniknya.

```
le = LabelEncoder()
le_count = 0

for col in train_df:
    if train_df[col].dtype == 'object':
        # If 2 or fewer unique categories
        if len(list(train_df[col].unique())) <= 2:
            # Train on the training data
            le.fit(train_df[col])
            # Transform both training and testing data
            train_df[col] = le.transform(train_df[col])
            test_df[col] = le.transform(test_df[col])
            # Keep track of how many columns were label encoded
            le_count += 1

print('%d columns were label encoded.' % le_count)
3 columns were label encoded.
```

Gambar 17.8. *Handling categorical columns*

```
# one-hot encoding of categorical variables
train_df = pd.get_dummies(train_df)
test_df = pd.get_dummies(test_df)
print('Training Features shape: ', train_df.shape)
print('Testing Features shape: ', test_df.shape)

Training Features shape: (307511, 243)
Testing Features shape: (48744, 239)
```

Gambar 17.9. *Shape of the data*

Setelah menggunakan one-hot encoding, kita perlu menyelaraskan data pelatihan dan data pengujian agar memiliki kolom yang sama.

```
# seperate target variable
train_labels = train_df['TARGET']
# combine the training and testing data, keep only columns present in both dataframes
train_df, test_df = train_df.align(test_df, join = 'inner', axis = 1)
# add the target back in
train_df['TARGET'] = train_labels

print('Training Features shape: ', train_df.shape)
print('Testing Features shape: ', test_df.shape)

Training Features shape: (307511, 240)
Testing Features shape: (48744, 239)
```

Gambar 17.10. *Preparing train and test data*

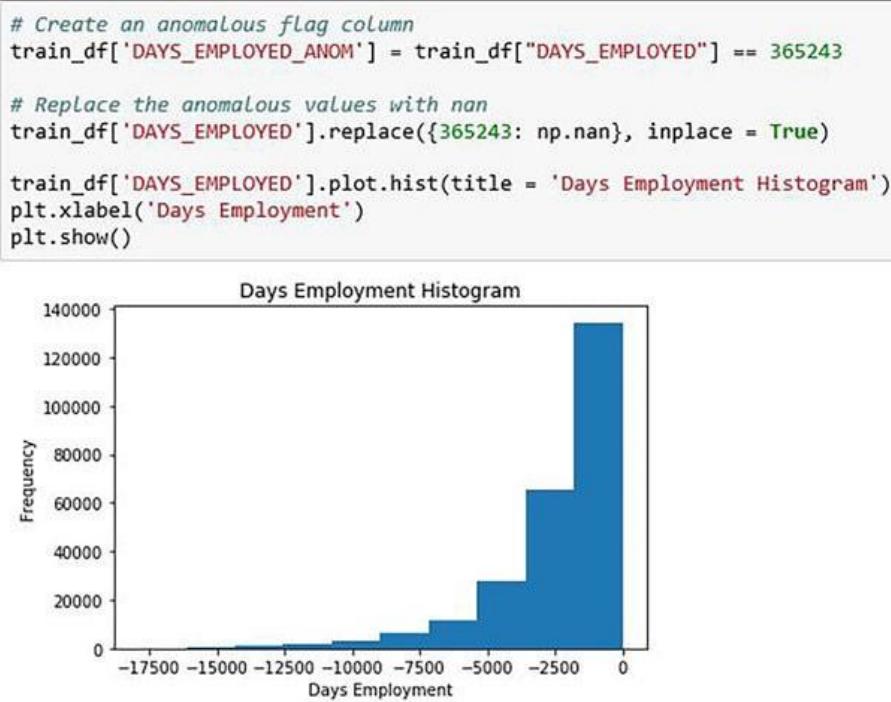
Gunakan `describe()` pada `DAYS_EMPLOYED`.

```
anomalous_clients = train_df[train_df['DAYS_EMPLOYED'] == 365243]
non_anomalous_clients = train_df[train_df['DAYS_EMPLOYED'] != 365243]
print('The non-anomalies default on %0.2f%% of loans' % (100 * non_anomalous_clients['TARGET'].mean()))
print('The anomalies default on %0.2f%% of loans' % (100 * anomalous_clients['TARGET'].mean()))
print('There are %d anomalous days of employment' % len(anomalous_clients))

The non-anomalies default on 8.66% of loans
The anomalies default on 5.40% of loans
There are 55374 anomalous days of employment
```

Gambar 17.11. *Using .describe() method*

Data-data yang aneh akan diubah menjadi NaN dan kita akan membuat kolom baru untuk menandai data-data aneh tersebut.



Gambar 17.12. *Days Employment Histogram*

Perlu diingat bahwa apa pun yang kita lakukan pada data pelatihan, juga harus kita lakukan pada data pengujian. Silakan ulangi langkah sebelumnya pada dataset pengujian. Setelah kita mengatasi variabel kategorikal dan data yang aneh, kita akan mencari korelasi antara fitur-fitur dan target. Kita dapat menghitung koefisien korelasi Pearson antara setiap variabel dan target menggunakan metode .corr pada dataframe:

```

# Find correlations with the target and sort
correlations = train_df.corr()['TARGET'].sort_values()
# Display correlations
print('Most Positive Correlations:\n', correlations.tail())
print('\nMost Negative Correlations:\n', correlations.head())

Most Positive Correlations:
REGION_RATING_CLIENT           0.058899
REGION_RATING_CLIENT_W_CITY     0.060893
DAYS_EMPLOYED                     0.074958
DAYS_BIRTH                        0.078239
TARGET                           1.000000
Name: TARGET, dtype: float64

Most Negative Correlations:
EXT_SOURCE_3                      -0.178919
EXT_SOURCE_2                      -0.160472
EXT_SOURCE_1                      -0.155317
NAME_EDUCATION_TYPE_Higher education -0.056593
CODE_GENDER_F                      -0.054704
Name: TARGET, dtype: float64

```

Gambar 17.13. Using .corr dataframe method

Korelasi antara usia klien (DAYS_BIRTH) dan kemungkinan keterlambatan pembayaran pinjaman adalah negatif setelah diambil nilai absolutnya.

```

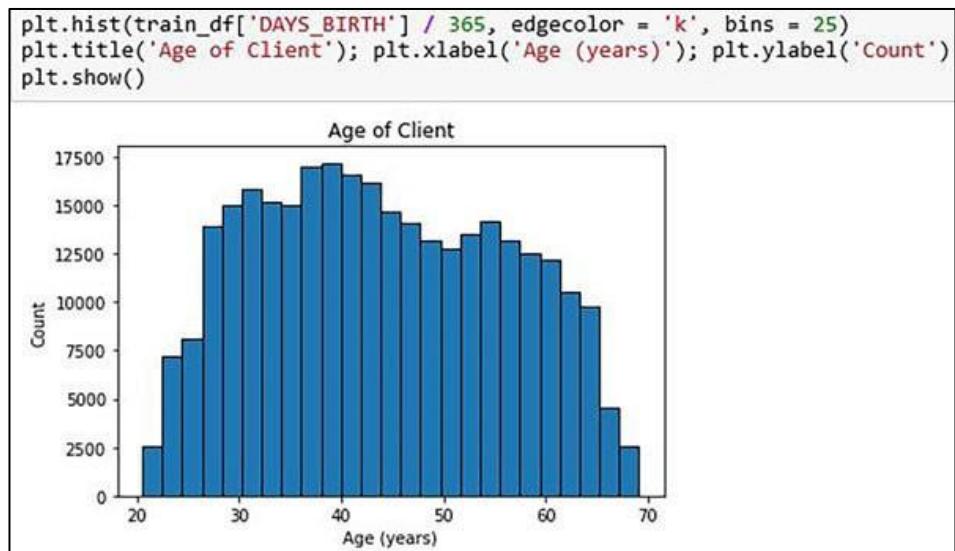
# Find the correlation of the positive days since birth and target
train_df['DAYS_BIRTH'] = abs(train_df['DAYS_BIRTH'])
train_df['DAYS_BIRTH'].corr(train_df['TARGET'])

-0.07823930830982712

```

Gambar 17.14. Finding correlation of positive days since birth and target

Membuat histogram usia dalam tahun.



Gambar 17.15. Age of client

Selain itu, kita akan coba dua metode sederhana untuk membuat fitur baru: fitur polinomial dan fitur berdasarkan pengetahuan domain.

```

# Make a new dataframe for polynomial features
poly_features = train_df[['EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'DAYS_BIRTH', 'TARGET']]
poly_features_test = test_df[['EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'DAYS_BIRTH']]

# imputer for handling missing values
from sklearn.preprocessing import Imputer
imputer = Imputer(strategy = 'median')

poly_target = poly_features['TARGET']

poly_features = poly_features.drop(columns = ['TARGET'])

# Need to impute missing values
poly_features = imputer.fit_transform(poly_features)
poly_features_test = imputer.transform(poly_features_test)

from sklearn.preprocessing import PolynomialFeatures
poly_transformer = PolynomialFeatures(degree = 3)

```

Gambar 17.16. New dataframe for polynomial features

```

# Train the polynomial features
poly_transformer.fit(poly_features)
# Transform the features
poly_features = poly_transformer.transform(poly_features)
poly_features_test = poly_transformer.transform(poly_features_test)
print('Polynomial Features shape: ', poly_features.shape)

Polynomial Features shape: (307511, 35)

```

Gambar 17.17. *Training and transforming polynomial features*

Kode di atas akan membuat sejumlah besar fitur baru. Untuk mendapatkan namanya, Anda harus menggunakan metode `get_feature_names()` dari fitur polinomial. Dalam metode ini, berikan nama-nama fitur input, dan itu akan menampilkan output seperti berikut:

```

# get the names
poly_transformer.get_feature_names(input_features = ['EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'DAYS_BIRTH'])

['1',
'EXT_SOURCE_1',
'EXT_SOURCE_2',
'EXT_SOURCE_3',
'DAYS_BIRTH',
'EXT_SOURCE_1^2',
'EXT_SOURCE_1 EXT_SOURCE_2',
'EXT_SOURCE_1 EXT_SOURCE_3',
'EXT_SOURCE_1 DAYS_BIRTH',
'EXT_SOURCE_2^2',
'EXT_SOURCE_2 EXT_SOURCE_3',
'EXT_SOURCE_2 DAYS_BIRTH',
'EXT_SOURCE_3^2',
'EXT_SOURCE_3 DAYS_BIRTH',
'DAYS_BIRTH^2']

```

Gambar 17.18. *Getting names of new features*

Ada 35 fitur dengan fitur individu yang dinaikkan ke pangkat hingga derajat 3 dan istilah interaksi.

```

# Create a dataframe of the features
poly_features = pd.DataFrame(poly_features,
                             columns = poly_transformer.get_feature_names(['EXT_SOURCE_1', 'EXT_SOURCE_2',
                                                                           'EXT_SOURCE_3', 'DAYS_BIRTH']))

# Add in the target
poly_features['TARGET'] = poly_target

# Find the correlations with the target
poly_corrs = poly_features.corr()['TARGET'].sort_values()

# Display most negative and most positive
print(poly_corrs.head())
print(poly_corrs.tail())

EXT_SOURCE_2 EXT_SOURCE_3      -0.193939
EXT_SOURCE_1 EXT_SOURCE_2 EXT_SOURCE_3     -0.189605
EXT_SOURCE_2 EXT_SOURCE_3 DAYS_BIRTH      -0.181283
EXT_SOURCE_2^2 EXT_SOURCE_3      -0.176428
EXT_SOURCE_2 EXT_SOURCE_3^2      -0.172282
Name: TARGET, dtype: float64
DAYS_BIRTH      -0.078239
DAYS_BIRTH^2     -0.076672
DAYS_BIRTH^3     -0.074273
TARGET          1.000000
1                 NaN
Name: TARGET, dtype: float64

```

Gambar 17.19. *Checking correlation between target and input features*

Variabel baru memiliki korelasi tinggi dengan target. Kita akan mencoba menggunakan data dengan dan tanpa fitur-fitur baru ini dalam pembuatan model untuk melihat apakah mereka meningkatkan kinerja model.

```

# Put test features into dataframe
poly_features_test = pd.DataFrame(poly_features_test,
                                    columns = poly_transformer.get_feature_names(['EXT_SOURCE_1', 'EXT_SOURCE_2',
                                                                 'EXT_SOURCE_3', 'DAYS_BIRTH']))

# Merge polynomial features into training dataframe
poly_features['SK_ID_CURR'] = train_df['SK_ID_CURR']
app_train_poly = train_df.merge(poly_features, on = 'SK_ID_CURR', how = 'left')

# Merge polynomial features into testing dataframe
poly_features_test['SK_ID_CURR'] = test_df['SK_ID_CURR']
app_test_poly = test_df.merge(poly_features_test, on = 'SK_ID_CURR', how = 'left')

# Align the dataframes
app_train_poly, app_test_poly = app_train_poly.align(app_test_poly, join = 'inner', axis = 1)

# Print out the new shapes
print('Training data with polynomial features shape: ', app_train_poly.shape)
print('Testing data with polynomial features shape: ', app_test_poly.shape)

Training data with polynomial features shape: (307511, 275)
Testing data with polynomial features shape: (48744, 275)

```

Gambar 17.20. Preparing data with new features

Mari lakukan rekayasa fitur berdasarkan pengetahuan domain.

```

# Domain Knowledge Features
app_train_domain = train_df.copy()
app_test_domain = test_df.copy()

app_train_domain['CREDIT_INCOME_PERCENT'] = app_train_domain['AMT_CREDIT'] / app_train_domain['AMT_INCOME_TOTAL']
app_train_domain['ANNUITY_INCOME_PERCENT'] = app_train_domain['AMT_ANNUITY'] / app_train_domain['AMT_INCOME_TOTAL']
app_train_domain['CREDIT_TERM'] = app_train_domain['AMT_ANNUITY'] / app_train_domain['AMT_CREDIT']
app_train_domain['DAYS_EMPLOYED_PERCENT'] = app_train_domain['DAYS_EMPLOYED'] / app_train_domain['DAYS_BIRTH']

#repeat for test
app_test_domain['CREDIT_INCOME_PERCENT'] = app_test_domain['AMT_CREDIT'] / app_test_domain['AMT_INCOME_TOTAL']
app_test_domain['ANNUITY_INCOME_PERCENT'] = app_test_domain['AMT_ANNUITY'] / app_test_domain['AMT_INCOME_TOTAL']
app_test_domain['CREDIT_TERM'] = app_test_domain['AMT_ANNUITY'] / app_test_domain['AMT_CREDIT']
app_test_domain['DAYS_EMPLOYED_PERCENT'] = app_test_domain['DAYS_EMPLOYED'] / app_test_domain['DAYS_BIRTH']

```

Gambar 17.20. Domain knowledge features

PRE-PROCESSING

Kita akan membuat model dasar dengan menggunakan semua fitur setelah mengkodekan variabel kategorikal.

```

# get a baseline
from sklearn.preprocessing import MinMaxScaler, Imputer
# Drop the target from the training data
if 'TARGET' in train_df:
    train = train_df.drop(columns = ['TARGET'])
else:
    train = train_df.copy()
# Feature names
features = list(train.columns)
# Copy of the testing data
test = test_df.copy()
# Median imputation of missing values
imputer = Imputer(strategy = 'median')
# Scale each feature to 0-1
scaler = MinMaxScaler(feature_range = (0, 1))
# Fit on the training data
imputer.fit(train)
# Transform both training and testing data
train = imputer.transform(train)
test = imputer.transform(test_df)
# Repeat with the scaler
scaler.fit(train)
train = scaler.transform(train)
test = scaler.transform(test)
print('Training data shape: ', train.shape)
print('Testing data shape: ', test.shape)

Training data shape: (307511, 240)
Testing data shape: (48744, 240)

```

Gambar 17.22. Code for pre-processing

LOGISTIC REGRESION

Sekarang kita membuat model dan melatihnya dengan menggunakan metode .fit().

```

from sklearn.linear_model import LogisticRegression
# Make the model with the specified regularization parameter
log_reg = LogisticRegression(C = 0.0001)
# Train on the training data
log_reg.fit(train, train_labels)

LogisticRegression(C=0.0001, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1, max_iter=100,
                   multi_class='ovr', n_jobs=1, penalty='l2', random_state=None,
                   solver='liblinear', tol=0.0001, verbose=0, warm_start=False)

```

Gambar 17.23. Using .fit() method

Setelah melatih model, kita dapat menggunakannya untuk memprediksi probabilitas tidak membayar pinjaman dengan menggunakan metode predict_proba().

```
# Make predictions
# Make sure to select the second column only
log_reg_pred = log_reg.predict_proba(test)[:, 1]
```

Gambar 17.24. *Making predictions*

Sekarang kita siapkan format pengiriman CSV dengan dua kolom: SK_ID_CURR dan TARGET.

# Submission dataframe		
submit = test_df[['SK_ID_CURR']]		
submit['TARGET'] = log_reg_pred		
submit.head()		
SK_ID_CURR TARGET		
0	100001	0.087750
1	100005	0.163957
2	100013	0.110238
3	100028	0.076575
4	100038	0.154924

Gambar 17.25. *Submitting dataframe*

Kemudian kita menyimpannya dalam file CSV dengan menggunakan metode .to_csv() dari dataframe seperti yang ditunjukkan dalam tangkapan layar berikut:

```
# Save the submission to a csv file
submit.to_csv('E:/pg/bpb/BPB-Publications/Datasets/Case Studies/case_study_1/log_reg_baseline.csv', index = False)
```

Gambar 17.26. *Saving in a CSV file*

RANDOM FOREST

Sekarang coba model kedua - Random Forest - pada data pelatihan yang sama untuk melihat bagaimana hal itu memengaruhi kinerja:

```
from sklearn.ensemble import RandomForestClassifier
# Make the random forest classifier
random_forest = RandomForestClassifier(n_estimators = 100, random_state = 50, verbose = 1, n_jobs = -1)
```

Gambar 17.27. *Random Forest Classifier model*

Menginisialisasi model Random Forest Classifier dengan parameter kustom.

```
# Train on the training data
random_forest.fit(train, train_labels)
# Extract feature importances
feature_importance_values = random_forest.feature_importances_
feature_importances = pd.DataFrame({'feature': features, 'importance': feature_importance_values})
# Make predictions on the test data
predictions = random_forest.predict_proba(test)[:, 1]

[Parallel(n_jobs=-1)]: Done 42 tasks      | elapsed:  32.6s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  1.2min finished
[Parallel(n_jobs=4)]: Done 42 tasks      | elapsed:   0.3s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   0.7s finished

# Make a submission dataframe
submit = test_df[['SK_ID_CURR']]
submit['TARGET'] = predictions
# Save the submission dataframe
submit.to_csv('E:/pg/bpb/BPB-Publications/Datasets/Case Studies/case_study_1/random_forest_baseline.csv', index = False)
```

Gambar 17.28. *Modifying the parameters*

Membuat prediksi dengan menggunakan fitur-fitur yang telah dibuat sebelumnya untuk membandingkan kinerja model dengan dan tanpa fitur-fitur tersebut.

```

poly_features_names = list(app_train_poly.columns)
# Impute the polynomial features
imputer = Imputer(strategy = 'median')

poly_features = imputer.fit_transform(app_train_poly)
poly_features_test = imputer.transform(app_test_poly)

# Scale the polynomial features
scaler = MinMaxScaler(feature_range = (0, 1))

poly_features = scaler.fit_transform(poly_features)
poly_features_test = scaler.transform(poly_features_test)

random_forest_poly = RandomForestClassifier(n_estimators = 100, random_state = 50, verbose = 1, n_jobs = -1)

```

Gambar 17.29. *Imputing and training with RFC*

```

# Train on the training data
random_forest_poly.fit(poly_features, train_labels)

# Make predictions on the test data
predictions = random_forest_poly.predict_proba(poly_features_test)[:, 1]

[Parallel(n_jobs=-1)]: Done 42 tasks      | elapsed: 47.0s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 1.8min finished
[Parallel(n_jobs=4)]: Done 42 tasks      | elapsed: 0.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed: 0.4s finished

# Make a submission dataframe
submit = test_df[['SK_ID_CURR']]
submit['TARGET'] = predictions

# Save the submission dataframe
submit.to_csv('E:/pg/bpb/BPB-Publications/Datasets/Case Studies/case_study_1/random_forest_baseline_engineered.csv')

```

Gambar 17.30. *Making prediction and submission*

EKSTRAKSI FITUR

Memeriksa fitur-fitur domain seperti yang kita lakukan dengan model logistik sebelumnya.

```

app_train_domain = app_train_domain.drop(columns = 'TARGET')
domain_features_names = list(app_train_domain.columns)
# Impute the domainnominal features
imputer = Imputer(strategy = 'median')
domain_features = imputer.fit_transform(app_train_domain)
domain_features_test = imputer.transform(app_test_domain)
# Scale the domainnominal features
scaler = MinMaxScaler(feature_range = (0, 1))
domain_features = scaler.fit_transform(domain_features)
domain_features_test = scaler.transform(domain_features_test)
random_forest_domain = RandomForestClassifier(n_estimators = 100, random_state = 50, verbose = 1, n_jobs = -1)
# Train on the training data
random_forest_domain.fit(domain_features, train_labels)
# Extract feature importances
feature_importance_values_domain = random_forest_domain.feature_importances_
feature_importances_domain = pd.DataFrame({'feature': domain_features_names, 'importance': feature_importance_values_domain})
# Make predictions on the test data
predictions = random_forest_domain.predict_proba(domain_features_test)[:, 1]

[Parallel(n_jobs=-1)]: Done 42 tasks      | elapsed: 31.8s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 1.2min finished
[Parallel(n_jobs=4)]: Done 42 tasks      | elapsed: 0.3s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed: 0.7s finished

```

Gambar 17.31. *Checking domain features*

```

# Make a submission dataframe
submit = test_df[['SK_ID_CURR']]
submit['TARGET'] = predictions
# Save the submission dataframe
submit.to_csv('E:/pg/bpb/BPB-Publications/Datasets/Case Studies/case_study_1/random_forest_baseline_domain.csv', index = False)

```

Gambar 17.32. *Saving submission dataframe*

Hitung ROC AUC untuk setiap model dan periksa pentingnya fitur dari Random Forest untuk melihat variabel mana yang paling relevan.

```

def plot_feature_importances(df):
    # Sort features according to importance
    df = df.sort_values('importance', ascending = False).reset_index()

    # Normalize the feature importances to add up to one
    df['importance_normalized'] = df['importance'] / df['importance'].sum()

    # Make a horizontal bar chart of feature importances
    plt.figure(figsize = (8, 4))
    ax = plt.subplot()

    # Need to reverse the index to plot most important on top
    ax.barh(list(reversed(list(df.index[:15]))),
            df['importance_normalized'].head(15),
            align = 'center', edgecolor = 'k')

    # Set the yticks and labels
    ax.set_yticks(list(reversed(list(df.index[:15]))))
    ax.set_yticklabels(df['feature'].head(15))

    # Plot labeling
    plt.xlabel('Normalized Importance'); plt.title('Feature Importances')
    plt.show()
    return df

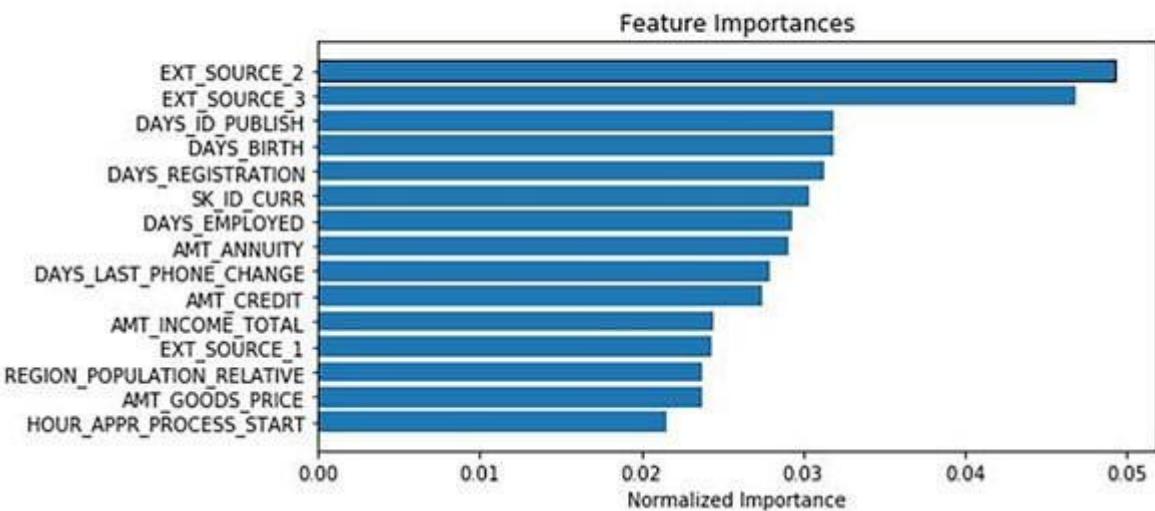
```

Gambar 17.33. *Plotting feature-importance*

```

# Show the feature importances for the default features
feature_importances_sorted = plot_feature_importances(feature_importances)

```



Gambar 17.34. *Feature importances*

17.3 KESIMPULAN

Dalam latihan ini, Anda telah membuat model dasar untuk menyelesaikan masalah pembelajaran mesin berkelanjutan yang sebenarnya. Anda telah mencoba dengan pengklasifikasi LR dan Random Forest, tetapi model lain menunggu Anda untuk mengembangkan model dasar ini dan melihat bagaimana cara meningkatkan akurasi model. Cobalah menerapkan model yang berbeda dan jangan lupa untuk memeriksa kinerja model Anda menggunakan metrik ROC AUC! Pada bab berikutnya, kita akan bekerja pada studi kasus lain tentang deteksi pesan spam atau ham.

BAB XVIII

STUDI KASUS 2

18.1 MODEL PREDIKSI DENGAN MENGKLASIFIKASIKAN TEKS

Bangunlah model prediksi yang dapat mengklasifikasikan dengan akurat pesan teks mana yang merupakan spam atau bukan. Data yang digunakan adalah SMS Spam Collection, yang berisi SMS berlabel ham (legal) atau spam. Model yang akan digunakan adalah Multinomial Naive Bayes dan Support Vector Machines (SVM).

18.2 PROSES

IMPOR DATA

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from sklearn import feature_extraction, model_selection, naive_bayes, metrics, svm
from IPython.display import Image
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

Gambar 18.1. *Importing required libraries*

Dalam studi kasus ini, kita akan menggunakan algoritma Naive Bayes dan Support Vector Machine.

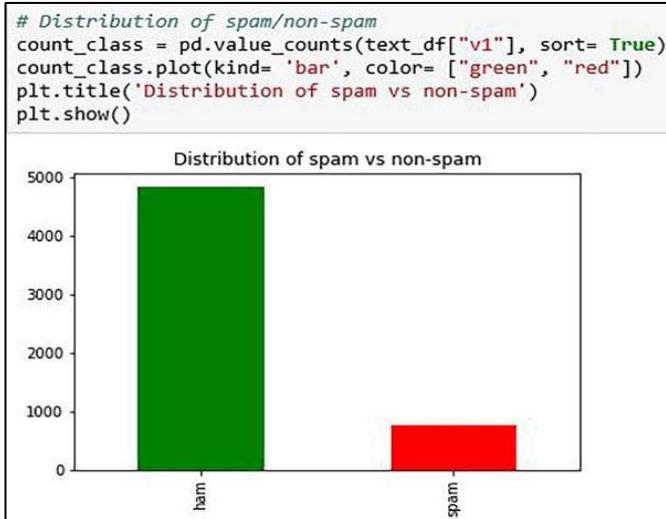
```
text_df = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/Case Studies/case_study_2/spam.csv', encoding='latin-1')
text_df.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wky comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Gambar 18.2. *Loading the spam dataset*

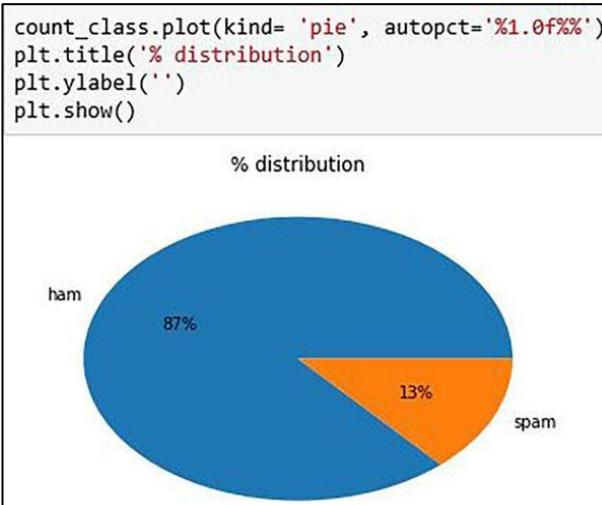
ANALISA DATA

Kita akan memeriksa distribusi pesan spam vs. non-spam dengan grafik.



Gambar 18.3. *Plotting of bar graph*

Mari cek distribusi pesan spam dan non-spam dalam bentuk grafik.



Gambar 18.4. *Plotting of pie chart*

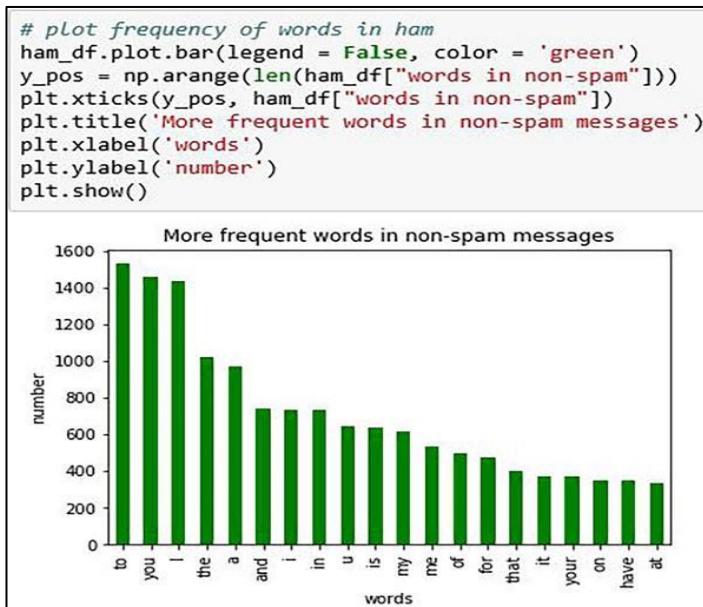
Dalam langkah ini, kita akan menghitung frekuensi setiap kata dalam teks spam dan non-spam menggunakan kamus.

```
# find frequencies of words in the spam and non-spam messages
ham_count = Counter(" ".join(text_df[text_df['v1']=='ham']['v2']).split()).most_common(20)
ham_df = pd.DataFrame.from_dict(ham_count)
ham_df = ham_df.rename(columns={0: "words in non-spam", 1 : "count"})

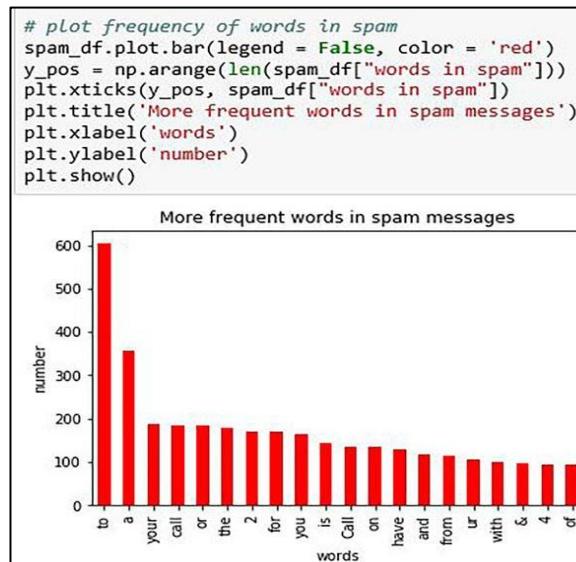
spam_count = Counter(" ".join(text_df[text_df['v1']=='spam']['v2']).split()).most_common(20)
spam_df = pd.DataFrame.from_dict(spam_count)
spam_df = spam_df.rename(columns={0: "words in spam", 1 : "count"})
```

Gambar 18.5. *Using collections.Counter()*

Kami menghitung frekuensi kata dalam pesan non-spam.



Gambar 18.6. *Plotting most frequently appearing words in non-spam messages*



Gambar 18.7. *Plotting most frequently appearing words in spam messages*

EKSTRAKSI FITUR

Mari kita hapus kata-kata stop dari teks pesan dan konversi teks ke dalam matriks token dengan menggunakan CountVectorizer.

```
# remove the stop words and create new features
f = feature_extraction.text.CountVectorizer(stop_words = 'english')
X = f.fit_transform(text_df["v2"])
np.shape(X)
(5572, 8404)
```

Gambar 18.8. *Removing stop words and creating new features*

SPLIT DATA

Kita akan membagi dataset menjadi pelatihan dan pengujian.

```
text_df["v1"] = text_df["v1"].map({'spam':1,'ham':0})
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, text_df['v1'], test_size=0.33, random_state=42)
print([np.shape(X_train), np.shape(X_test)])
[(3733, 8404), (1839, 8404)]
```

Gambar 18.9. *Splitting data into training and test set*

TRAINING NAÏVE BAYES

Kita akan melatih model Naive Bayes yang berbeda dengan mengubah parameter regularisasi dan mengevaluasi akurasi, recall, dan presisi model dengan set pengujian.

```
list_alpha = np.arange(1/100000, 20, 0.11)
score_train = np.zeros(len(list_alpha))
score_test = np.zeros(len(list_alpha))
recall_test = np.zeros(len(list_alpha))
precision_test= np.zeros(len(list_alpha))
count = 0
for alpha in list_alpha:
    bayes = naive_bayes.MultinomialNB(alpha=alpha)
    bayes.fit(X_train, y_train)
    score_train[count] = bayes.score(X_train, y_train)
    score_test[count]= bayes.score(X_test, y_test)
    recall_test[count] = metrics.recall_score(y_test, bayes.predict(X_test))
    precision_test[count] = metrics.precision_score(y_test, bayes.predict(X_test))
    count = count + 1
```

Gambar 18.10. *Training and Evaluating the performance of the model*

Pada sel kode sebelumnya, kami menentukan parameter dan melatih model Naive Bayes. Kami menghitung recall dan presisi untuk mengukur kinerja model. Selanjutnya, kami menganalisis kinerja model menggunakan berbagai metrik.

```
# Let's see some Learning models and their metrics
matrix = np.matrix(np.c_[list_alpha, score_train, score_test, recall_test, precision_test])
models = pd.DataFrame(data = matrix, columns =
                      ['alpha', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision'])
models.head()
```

alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0 0.00001	0.998661	0.974443	0.920635	0.895753
1 0.11001	0.997857	0.976074	0.936508	0.893939
2 0.22001	0.997857	0.977162	0.936508	0.900763
3 0.33001	0.997589	0.977162	0.936508	0.900763
4 0.44001	0.997053	0.977162	0.936508	0.900763

Gambar 18.11. *Calculating performance*

Selanjutnya, kita akan memilih model dengan presisi uji terbaik.

```
best_index = models['Test Precision'].idxmax()
models.iloc[best_index, :]

alpha              15.730010
Train Accuracy     0.979641
Test Accuracy      0.969549
Test Recall        0.777778
Test Precision     1.000000
Name: 143, dtype: float64
```

Gambar 18.12. *Selecting model with best test precision*

Tidak ada model lain dengan presisi 100%.

models[models['Test Precision']==1].head()					
	alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
143	15.73001	0.979641	0.969549	0.777778	1.0
144	15.84001	0.979641	0.969549	0.777778	1.0
145	15.95001	0.979641	0.969549	0.777778	1.0
146	16.06001	0.979373	0.969549	0.777778	1.0
147	16.17001	0.979373	0.969549	0.777778	1.0

Gambar 18.13. *Checking other models with 100% precision*

Dalam kasus seperti ini, lebih baik memilih model yang memiliki akurasi uji yang lebih tinggi.

```
best_index = models[models['Test Precision']==1]['Test Accuracy'].idxmax()
bayes = naive_bayes.MultinomialNB(alpha=list_alpha[best_index])
bayes.fit(X_train, y_train)
models.iloc[best_index, :]

alpha          15.730010
Train Accuracy    0.979641
Test Accuracy     0.969549
Test Recall       0.777778
Test Precision    1.000000
Name: 143, dtype: float64
```

Gambar 18.14. *Using model with more test accuracy*

EVALUASI NAÏVE BAYES

Mari hasilkan Matriks Konfusi untuk model Naive Bayes:

```
# Confusion matrix with naive bayes classifier
m_confusion_test = metrics.confusion_matrix(y_test, bayes.predict(X_test))
pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
              index = ['Actual 0', 'Actual 1'])

Predicted 0 Predicted 1
Actual 0      1587        0
Actual 1       56      196
```

Gambar 18.15. *Generating confusion matrix*

SUPPORT VECTOR MACHINE

Langsung ke poin: kita akan menggunakan model SVM (Support Vector Machine) untuk analisis berikutnya.

```
# repeat same steps with Support Vector Machine
list_C = np.arange(500, 2000, 100)
score_train = np.zeros(len(list_C))
score_test = np.zeros(len(list_C))
recall_test = np.zeros(len(list_C))
precision_test= np.zeros(len(list_C))
count = 0
for C in list_C:
    svc = svm.SVC(C=C)
    svc.fit(X_train, y_train)
    score_train[count] = svc.score(X_train, y_train)
    score_test[count]= svc.score(X_test, y_test)
    recall_test[count] = metrics.recall_score(y_test, svc.predict(X_test))
    precision_test[count] = metrics.precision_score(y_test, svc.predict(X_test))
    count = count + 1
```

Gambar 18.16. *Using Support Vector Machine*

```

matrix = np.matrix(np.c_[list_C, score_train, score_test, recall_test, precision_test])
models = pd.DataFrame(data = matrix, columns =
['C', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision'])
models.head()

   C  Train Accuracy  Test Accuracy  Test Recall  Test Precision
0  500.0        0.994910     0.982599    0.873016       1.0
1  600.0        0.995982     0.982599    0.873016       1.0
2  700.0        0.996785     0.982599    0.873016       1.0
3  800.0        0.997053     0.983143    0.876984       1.0
4  900.0        0.997589     0.983143    0.876984       1.0

best_index = models['Test Precision'].idxmax()
models.iloc[best_index, :]

C              500.000000
Train Accuracy      0.994910
Test Accuracy       0.982599
Test Recall         0.873016
Test Precision      1.000000
Name: 0, dtype: float64

```

Gambar 18.17. Preparing dataframe with matrix data

```

models[models['Test Precision']==1].head()

   C  Train Accuracy  Test Accuracy  Test Recall  Test Precision
0  500.0        0.994910     0.982599    0.873016       1.0
1  600.0        0.995982     0.982599    0.873016       1.0
2  700.0        0.996785     0.982599    0.873016       1.0
3  800.0        0.997053     0.983143    0.876984       1.0
4  900.0        0.997589     0.983143    0.876984       1.0

best_index = models[models['Test Precision']==1]['Test Accuracy'].idxmax()
svc = svm.SVC(C=list_C[best_index])
svc.fit(X_train, y_train)
models.iloc[best_index, :]

C              800.000000
Train Accuracy      0.997053
Test Accuracy       0.983143
Test Recall         0.876984
Test Precision      1.000000
Name: 3, dtype: float64

```

Gambar 18.18. Training with SVM model

EVALUASI SVM

```

m_confusion_test = metrics.confusion_matrix(y_test, svc.predict(X_test))
pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
             index = ['Actual 0', 'Actual 1'])

   Predicted 0  Predicted 1
Actual 0      1587          0
Actual 1       31        221

```

Gambar 18.19. Confusion matrix with SVM model

Anda sekarang dapat menggunakan model SVM Anda untuk mengklasifikasikan teks sebagai spam atau non-spam.

```

# predicting a new text using our svm model
Y = ["A loan for £950 is approved for you if you receive this SMS. 1 min verification & cash in 1 hr at www.example.co.uk"]
f = feature_extraction.text.CountVectorizer(stop_words = 'english')
f.fit(text_df["v2"])
X = f.transform(Y)
res=svc.predict(X)
if res==1:
    print('This text is spam')
else:
    print('This text is not a spam')

This text is spam

```

Gambar 18.20. Predicting new text using SVM model

KESIMPULAN

Anda akan menemukan bahwa mengklasifikasikan email atau pesan bukanlah tugas yang sulit. Naive Bayes dan Support Vector Machines adalah dua algoritma yang paling sering digunakan dalam masalah klasifikasi spam vs. non-spam.

BAB XIX

STUDI KASUS 3

19.1 BUILD A FILM RECOMMENDATION ENGINE

Anda ingin membuat mesin rekomendasi film menggunakan dataset TMDB yang berisi informasi tentang berbagai film dan seri TV. Anda akan menggunakan dua metode, yaitu berdasarkan popularitas dan berdasarkan konten film, untuk memberikan rekomendasi kepada pengguna. Sebelum memulai, Anda akan menjelajahi dataset untuk memahami karakteristiknya dengan lebih baik.

```
credits = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/Case Studies/case_study_3/tmdb_5000_credits.csv')
credits.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 4 columns):
movie_id    4803 non-null int64
title       4803 non-null object
cast        4803 non-null object
crew        4803 non-null object
dtypes: int64(1), object(3)
memory usage: 150.2+ KB
```

Gambar 19.1. *Loading the datasets*

```
movies = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/Case Studies/case_study_3/tmdb_5000_movies.csv')
movies.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
budget           4803 non-null int64
genres           4803 non-null object
homepage         1712 non-null object
id               4803 non-null int64
keywords         4803 non-null object
original_language 4803 non-null object
original_title   4803 non-null object
overview         4800 non-null object
popularity       4803 non-null float64
production_companies 4803 non-null object
production_countries 4803 non-null object
release_date     4802 non-null object
revenue          4803 non-null int64
runtime          4801 non-null float64
spoken_languages 4803 non-null object
status           4803 non-null object
tagline          3959 non-null object
title            4803 non-null object
vote_average     4803 non-null float64
vote_count       4803 non-null int64
dtypes: float64(3), int64(4), object(13)
```

Gambar 19.2. *Exploring the datasets*

Hitung rata-rata suara di seluruh dataset film.

```
# calculate mean vote
C = movies['vote_average'].mean()
C
```

6.092171559442011

Gambar 19.3. *Calculating the mean vote*

Tentukan nilai minimum suara yang diperlukan untuk masuk ke dalam grafik menggunakan persentil ke-90.

```
# calculate minimum votes required to be listed in the chart
m = movies['vote_count'].quantile(0.9)
m
```

1838.4000000000015

Gambar 19.4. *Calculating the required minimum votes*

Now, we can filter out the movies that qualify for the chart:

```
# filter out the movies that qualify for the chart
q_movies = movies.copy().loc[movies['vote_count'] >= m]
q_movies.shape

(481, 23)
```

Gambar 19.5. Filtering out movies qualified for the chart

Kita perlu menghitung metrik untuk setiap film yang memenuhi syarat. Kami akan membuat sebuah fungsi untuk melakukan perhitungan ini dan juga akan menambahkan fitur baru yang disebut skor dengan mengaplikasikan fungsi weighted_rating() pada DataFrame film yang memenuhi syarat. Ini adalah langkah awal menuju pembuatan perekомendasi dasar pertama kami. Untuk menulis fungsi weighted_rating(), silakan lihat panduan di tautan berikut: [link ke panduan IMDB].

<https://help.imdb.com/article/imdb/track-movies-tv/faq-for-imdbratings/G67Y87TFYYP6TWAV#>

```
# calculate our metric for each qualified movie
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)

# Define a new feature 'score' and calculate its value with `weighted_rating()`
q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
#Sort movies based on score calculated above
q_movies = q_movies.sort_values('score', ascending=False)
#Print the top 5 movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head()
```

	title	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239

Gambar 19.6. metric calculation for qualified movies

```
# plot 5 popular movies
popular_movies = movies.sort_values('popularity', ascending=False)
plt.figure(figsize=(12,4))
plt.barh(popular_movies['title'].head(), popular_movies['popularity'].head(), align='center',
         color='yellow')
plt.gca().invert_yaxis()
plt.xlabel("Popularity")
plt.title("Popular Movies")
Text(0.5,1,'Popular Movies')
```

Title	Popularity
Deadpool	~520
Guardians of the Galaxy	~480
Interstellar	~750
Mad Max: Fury Road	~450
Minions	~850

Gambar 19.7. Preparing visual graph

Kami akan menggunakan TfIdfVectorizer dari scikit-learn untuk mengelola teks-teks tersebut.

```

from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(stop_words='english')
# handle missing values
movies['overview'] = movies['overview'].fillna('')
tfidf_matrix = tfidf.fit_transform(movies['overview'])
tfidf_matrix.shape

(4803, 20978)

```

Gambar 19.8. *train movies data with TF-IDF Vectorizer*

Dengan matriks ini, kita dapat menghitung skor kesamaan menggunakan metode kesamaan kosinus.

```

from sklearn.metrics.pairwise import linear_kernel
# compute the cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

```

Gambar 19.9. *Using sklearn's linear_kernel()*

Kita butuh cara untuk mengidentifikasi indeks film berdasarkan judulnya.

```

# construct a reverse map of indices and movie titles
indices = pd.Series(movies.index, index=movies['title']).drop_duplicates()

```

Gambar 19.10. *Constructing reverse map of indices and movie titles*

Selanjutnya, kita akan mendefinisikan fungsi rekomendasi yang akan menghasilkan 10 film paling mirip berdasarkan kesamaan kosinus dengan film yang dipilih.

```

# define our recommendation function
def get_recommendations(title, cosine_sim=cosine_sim):
    idx = indices.get_loc(title)
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:11]
    movie_indices = [i[0] for i in sim_scores]

    return movies['title'].iloc[movie_indices]

```

Gambar 19.11. *defining recommendation function*

```

# test our function
get_recommendations('Spectre')

1343    Never Say Never Again
4071    From Russia with Love
3162    Thunderball
1717    Safe Haven
11      Quantum of Solace
4339    Dr. No
29      Skyfall
1880    Dance Flick
3336    Diamonds Are Forever
1743    Octopussy
Name: title, dtype: object

```

Gambar 19.12. *Testing the function*

Kita perlu mengonversi data dari format string menjadi struktur yang dapat digunakan.

```

# parse the stringified features into their corresponding python objects
from ast import literal_eval
features = ['cast', 'crew', 'keywords', 'genres']
for feature in features:
    movies[feature] = movies[feature].apply(literal_eval)

```

Gambar 19.13. *converting data into usable structure*

Selanjutnya, kita akan menulis fungsi-fungsi yang akan membantu kita mengekstrak informasi yang diperlukan dari setiap fitur:

```

# Get the director's name from the crew feature
def get_director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan

# Returns the list top 3 elements or entire list
def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
        if len(names) > 3:
            names = names[:3]
        return names
    return []

# Define new director, cast, genres and keywords features that are in a suitable form
movies['director'] = movies['crew'].apply(get_director)
features = ['cast', 'keywords', 'genres']
for feature in features:
    movies[feature] = movies[feature].apply(get_list)

```

Gambar 19.14. Writing function to extract information from features

	title	cast	director	keywords	genres
0	Avatar	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	James Cameron	[culture clash, future, space war]	[Action, Adventure, Fantasy]
1	Pirates of the Caribbean: At World's End	[Johnny Depp, Orlando Bloom, Keira Knightley]	Gore Verbinski	[ocean, drug abuse, exotic island]	[Adventure, Fantasy, Action]
2	Spectre	[Daniel Craig, Christoph Waltz, Léa Seydoux]	Sam Mendes	[spy, based on novel, secret agent]	[Action, Adventure, Crime]
3	The Dark Knight Rises	[Christian Bale, Michael Caine, Gary Oldman]	Christopher Nolan	[dc comics, crime fighter, terrorist]	[Action, Crime, Drama]
4	John Carter	[Taylor Kitsch, Lynn Collins, Samantha Morton]	Andrew Stanton	[based on novel, mars, medallion]	[Action, Adventure, Science Fiction]

Gambar 19.15. peek of the data with new features

Konversi nama dan kata kunci menjadi huruf kecil dan hilangkan spasi di antara mereka.

```

# Function to convert all strings to Lower case and strip names of spaces
def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
        else:
            return ''

# Apply clean_data function to our features.
features = ['cast', 'keywords', 'director', 'genres']
for feature in features:
    movies[feature] = movies[feature].apply(clean_data)

```

Gambar 19.16. Converting names and keyword instances into lowercase and stripping spaces

```

def create_soup(x):
    return ' '.join(x['keywords']) + ' ' + ' '.join(x['cast']) + ' ' + x['director'] + ' ' + ' '.join(x['genres'])
movies['soup'] = movies.apply(create_soup, axis=1)

```

Gambar 19.17. creating meta data soup

Langkah selanjutnya, kita akan menggunakan CountVectorizer() dari sklearn untuk menghilangkan kata-kata stop dan mengubah kolom "soup" yang baru saja kita buat.

```

from sklearn.feature_extraction.text import CountVectorizer
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(movies['soup'])

# compute the Cosine Similarity matrix based on the count_matrix
from sklearn.metrics.pairwise import cosine_similarity
cosine_sim2 = cosine_similarity(count_matrix, count_matrix)

```

Gambar 19.18. use of Count Vectorizer and similarity check

```
# test our get_recommendations() function with our new arguement
get_recommendations('Spectre', cosine_sim2)

29           Skyfall
11      Quantum of Solace
1084    The Glimmer Man
1234      The Art of War
2156       Nancy Drew
4638  Amidst the Devil's Wings
62        The Legend of Tarzan
3373   The Other Side of Heaven
4            John Carter
72        Suicide Squad
Name: title, dtype: object
```

Gambar 19.19. *testing recommendation function*

Mesin rekomendasi ini telah berhasil meningkatkan rekomendasi film berdasarkan metadata, seperti aktor, sutradara, dan kata kunci plot. Anda juga dapat meningkatkan mesin ini lebih lanjut dengan pertimbangan bahasa film. Mesin rekomendasi dapat diperluas untuk berbagai masalah rekomendasi lainnya, seperti rekomendasi produk atau kategori produk.

19.2 KESIMPULAN

Sistem rekomendasi banyak digunakan dalam hampir setiap e-commerce dan layanan media over the top. Bagi pengguna, sistem-sistem ini meningkatkan pengalaman dengan membantu memilih pilihan terbaik dibandingkan dengan opsi lain, serta meningkatkan kemungkinan perusahaan untuk mendapatkan lebih banyak pendapatan. Setelah menyelesaikan latihan ini, Anda memiliki pengalaman langsung dalam membangun mesin rekomendasi film. Sekarang jangan berhenti di sini, terapkan pengetahuan Anda untuk membangun mesin rekomendasi produk atau mesin rekomendasi lagu, dan rasakan nilai pekerjaan seorang Ilmuwan Data bagi masyarakat ini.

BAB XX

STUDI KASUS 4

20.1 PPREDIKSI PENJUALAN RUMAH MENGGUNAKAN REGRESI

Dalam studi kasus ini, kita akan memprediksi penjualan rumah di King County, Washington, Amerika Serikat, menggunakan regresi.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

# create evaluation metrics
evaluation = pd.DataFrame({'Model': [],
                            'Details':[],
                            'Mean Squared Error (MSE)':[],
                            'R-squared (training)':[],
                            'Adjusted R-squared (training)':[],
                            'R-squared (test)':[],
                            'Adjusted R-squared (test)':[]})
```

Gambar 20.1. Importing all basic libraries

```
# read and explore data
df = pd.read_csv('E:/pg/bpb/BPB-Publications/Datasets/Case Studies/case_study_4/kc_house_data.csv')
df.head()

   id      date  price  bedrooms  bathrooms  sqft_living  sqft_lot  floors  waterfront  view ... grade  sqft_above  sqft_basement  yr_built
0  7129300520 20141013T000000 221900.0       3     1.0       1180     5650     1.0        0     0 ...    7     1180          0     1955
1  6414100192 20141209T000000 538000.0       3     2.25      2570     7242     2.0        0     0 ...    7     2170        400     1951
2  5631500400 20150225T000000 180000.0       2     1.00       770    10000     1.0        0     0 ...    6     770          0     1933
3  2487200875 20141209T000000 604000.0       4     3.00      1960     5000     1.0        0     0 ...    7     1050        910     1965
4  1954400510 20150218T000000 510000.0       3     2.00      1680     8080     1.0        0     0 ...    8     1680          0     1987

5 rows × 21 columns
```

Gambar 20.2. peek of the housing data

Dalam kasus ini, kita menggunakan regresi linear untuk memprediksi harga rumah di King County, Washington State, USA, dengan menggunakan data penjualan rumah selama periode dari Mei 2014 hingga Mei 2015. Kami akan menguji model regresi linear sederhana dengan menggunakan luas area rumah (dalam kaki persegi) sebagai fitur utama. Dalam proses ini, kami membagi dataset menjadi data pelatihan dan data uji, melatih model, dan mengukur akurasinya menggunakan Mean Squared Error (MSE).

```
%capture
train_data,test_data = train_test_split(df,train_size = 0.8,random_state=3)

lr = linear_model.LinearRegression()
X_train = np.array(train_data['sqft_living'], dtype=pd.Series).reshape(-1,1)
y_train = np.array(train_data['price'], dtype=pd.Series)
lr.fit(X_train,y_train)

X_test = np.array(test_data['sqft_living'], dtype=pd.Series).reshape(-1,1)
y_test = np.array(test_data['price'], dtype=pd.Series)

pred = lr.predict(X_test)
msesm = format(np.sqrt(metrics.mean_squared_error(y_test,pred)), '.3f')
rtrsm = format(lr.score(X_train, y_train), '.3f')
rtesm = format(lr.score(X_test, y_test), '.3f')

print ("Average Price for Test Data: {:.3f}".format(y_test.mean()))
print('Intercept: {}'.format(lr.intercept_))
print('Coefficient: {}'.format(lr.coef_))

r = evaluation.shape[0]
evaluation.loc[r] = ['Simple Model', '-', msesm,rtrsm,'-',rtesm, '-']
evaluation
```

Gambar 20.3. training and prediction on housing data

	Adjusted R-squared (test)	Adjusted R-squared (training)	Details	Mean Squared Error (MSE)	Model	R-squared (test)	R-squared (training)
0	Simple Model	-	- 254289.149	0.492	-	0.496	-
1	Simple Model	-	- 254289.149	0.492	-	0.496	-
2	Simple Model	-	- 254289.149	0.492	-	0.496	-

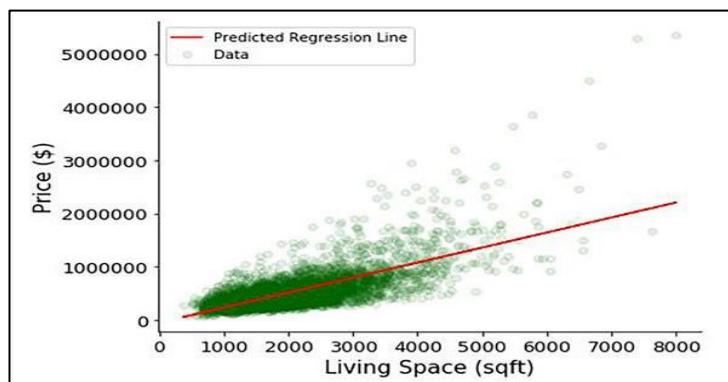
Gambar 20.4. Average price for test data

Grafik hasil regresi sederhana.

```
plt.figure(figsize=(6.5,5))
plt.scatter(X_test,y_test,color='darkgreen',label="Data", alpha=.1)
plt.plot(X_test,lr.predict(X_test),color="red",label="Predicted Regression Line")
plt.xlabel("Living Space (sqft)", fontsize=15)
plt.ylabel("Price ($)", fontsize=15)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
plt.legend()

plt.gca().spines['right'].set_visible(False)
plt.gca().spines['top'].set_visible(False)
```

Gambar 20.5. plot the house price vs space



Gambar 20.6. Predicted regression line data

Korelasi antara fitur-fitur dalam dataset adalah hal yang perlu diperhatikan dalam regresi berganda.

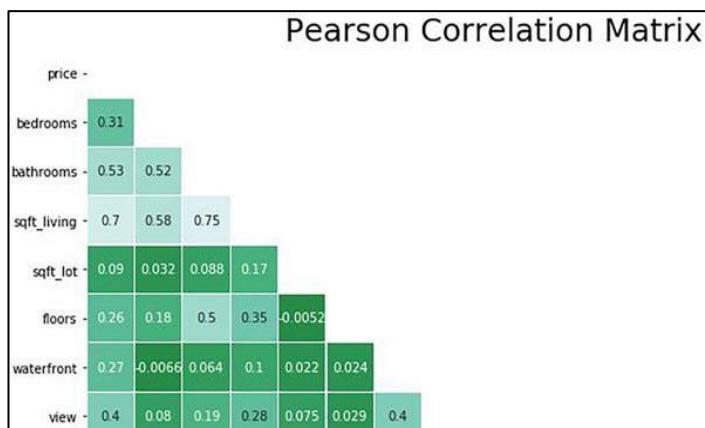
```
features = ['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors',
           'waterfront', 'view', 'condition', 'grade', 'sqft_above', 'sqft_basement',
           'yr_built', 'yr_renovated', 'zipcode', 'sqft_living15', 'sqft_lot15']

mask = np.zeros_like(df[features].corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(16, 12))
plt.title('Pearson Correlation Matrix', fontsize=25)

sns.heatmap(df[features].corr(), linewidths=0.25, vmax=1.0, square=True, cmap="BuGn_r",
            linecolor='w', annot=True, mask=mask, cbar_kws={"shrink": .75});
```

Gambar 20.7. Correlation matrix



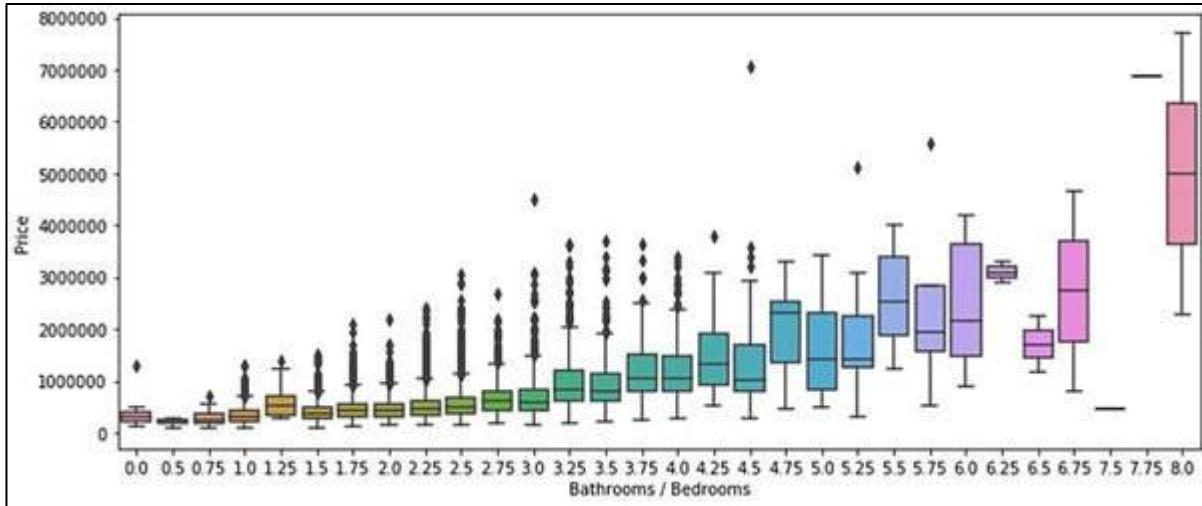
Gambar 20.8. Pearson Correlation Matrix

Dalam analisis korelasi dan perbandingan fitur-fitur menggunakan boxplot.

```
f, axes = plt.subplots(1, 2, figsize=(15,5))
sns.boxplot(x=train_data['bedrooms'],y=train_data['price'], ax=axes[0])
sns.boxplot(x=train_data['floors'],y=train_data['price'], ax=axes[1])
axes[0].set(xlabel='Bedrooms', ylabel='Price')
axes[1].yaxis.set_label_position("right")
axes[1].yaxis.tick_right()
axes[1].set(xlabel='Floors', ylabel='Price')

f, axe = plt.subplots(1, 1, figsize=(12,5))
sns.boxplot(x=train_data['bathrooms'],y=train_data['price'], ax=axe)
axe.set(xlabel='Bathrooms / Bedrooms', ylabel='Price');
```

Gambar 20.9. *creating box plot*



Gambar 20.10. *Graphical representation of bathrooms and bedrooms*

Dalam langkah ini, kita akan mencoba membuat model regresi yang lebih kompleks dengan menggunakan enam fitur dari dataset untuk memprediksi harga rumah.

```
features1 = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'zipcode']
complex_model_1 = linear_model.LinearRegression()
complex_model_1.fit(train_data[features1],train_data['price'])

print('Intercept: {}'.format(complex_model_1.intercept_))
print('Coefficients: {}'.format(complex_model_1.coef_))

pred1 = complex_model_1.predict(test_data[features1])
msecm1 = format(np.sqrt(metrics.mean_squared_error(y_test,pred1)), '.3f')
rtrcm1 = format(complex_model_1.score(train_data[features1],train_data['price']), '.3f')
artrcm1 = format(adjustedR2(complex_model_1.score(train_data[features1],train_data['price']),train_data.shape[0],len(features1)), '.3f')
rtecm1 = format(complex_model_1.score(test_data[features1],test_data['price']), '.3f')
ar tecm1 = format(adjustedR2(complex_model_1.score(test_data[features1],test_data['price']),test_data.shape[0],len(features1)), '.3f'

r = evaluation.shape[0]
evaluation.loc[r] = ['Complex Model-1', '-', msecm1,rtrcm1,artrcm1,rtecm1,ar tecm1]
evaluation.sort_values(by = 'R-squared (test)', ascending=False)
```

Gambar 20.11. *creating a complex model*

Adjusted R-squared (test)	Adjusted R-squared (training)	Details	Mean Squared Error (MSE)	Model	R-squared (test)	R-squared (training)
3	Complex Model-1	-	248514.011	0.514	0.514	0.519
0	Simple Model	-	254289.149	0.492	-	0.496
1	Simple Model	-	254289.149	0.492	-	0.496
2	Simple Model	-	254289.149	0.492	-	0.496

Gambar 20.12. *prediction on complex data*

Dalam output tersebut, kita melihat lebih banyak boxplot yang digunakan untuk membandingkan fitur-fitur tambahan terhadap harga rumah. Dari boxplot tersebut, kita dapat melihat pola hubungan antara fitur-fitur ini dan harga rumah, yang dapat membantu kita memutuskan apakah akan memasukkan fitur-fitur tersebut ke dalam model.

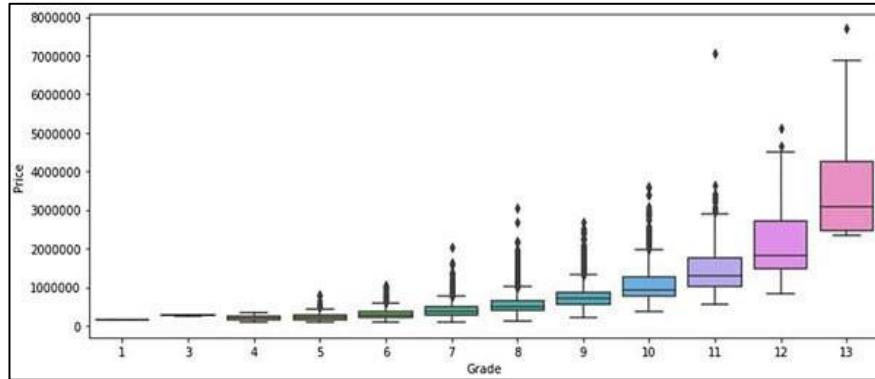
```

f, axes = plt.subplots(1, 2, figsize=(15,5))
sns.boxplot(x=train_data['waterfront'],y=train_data['price'], ax=axes[0])
sns.boxplot(x=train_data['view'],y=train_data['price'], ax=axes[1])
axes[0].set(xlabel='Waterfront', ylabel='Price')
axes[1].yaxis.set_label_position("right")
axes[1].yaxis.tick_right()
axes[1].set(xlabel='View', ylabel='Price')

f, axe = plt.subplots(1, 1,figsize=(12,18))
sns.boxplot(x=train_data['grade'],y=train_data['price'], ax=axe)
axe.set(xlabel='Grade', ylabel='Price');

```

Gambar 20.13. *plot the box plot*



Gambar 20.14. *box plot of house price vs grade*

```

features2 = ['bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','view',
'grade','yr_built','zipcode']
complex_model_2 = linear_model.LinearRegression()
complex_model_2.fit(train_data[features2],train_data['price'])

print('Intercept: {}'.format(complex_model_2.intercept_))
print('Coefficients: {}'.format(complex_model_2.coef_))

pred2 = complex_model_2.predict(test_data[features2])
msecm2 = format(np.sqrt(metrics.mean_squared_error(y_test,pred2)),'.3f')
rtrcm2 = format(complex_model_2.score(train_data[features2],train_data['price']),'.3f')
artrcm2 = format(adjR2(complex_model_2.score(train_data[features2],train_data['price']),train_data.shape[0],len(features2)),'.3f')
rtecm2 = format(complex_model_2.score(test_data[features2],test_data['price']),'.3f')
ar tecm2 = format(adjR2(complex_model_2.score(test_data[features2],test_data['price']),test_data.shape[0],len(features2)),'.3f')

r = evaluation.shape[0]
evaluation.loc[r] = ['Complex Model-2', '-', msecm2,rtrcm2,artrcm2,rtecm2,ar tecm2]
evaluation.sort_values(by = 'R-squared (test)', ascending=False)

```

Gambar 20.15. *adding more features and making prediction*

```

Intercept: 13559209.611222725
Coefficients: [-3.80981692e+04 5.03031727e+04 1.71370475e+02 -2.68019419e-01
 2.21944912e+04 5.53865017e+05 4.70338164e+04 1.23642184e+05
 -3.88306990e+03 -6.82180496e+01]

```

	Adjusted R-squared (test)	Adjusted R-squared (training)	Details	Mean Squared Error (MSE)	Model	R-squared (test)	R-squared (training)
4	Complex Model-2	-	210486.689	0.651	0.650	0.655	0.654
3	Complex Model-1	-	248514.011	0.514	0.514	0.519	0.518
0	Simple Model	-	254289.149	0.492	-	0.496	-
1	Simple Model	-	254289.149	0.492	-	0.496	-
2	Simple Model	-	254289.149	0.492	-	0.496	-

Gambar 20.16. *result of the prediction on more data*

Penambahan fitur dalam model kompleks 2 mengurangi log regresi menjadi 210486.689. Model linear cocok untuk data berdistribusi kuadrat. Transformasi polinomial dapat memberikan hasil yang lebih baik.

```

polyfeat = PolynomialFeatures(degree=2)
X_trainpoly = polyfeat.fit_transform(train_data[features2])
X_testpoly = polyfeat.fit_transform(test_data[features2])
poly = linear_model.LinearRegression().fit(X_trainpoly, train_data['price'])

predp = poly.predict(X_testpoly)
msepoly1 = format(np.sqrt(metrics.mean_squared_error(test_data['price'], pred)), '.3f')
rtrpoly1 = format(poly.score(X_trainpoly, train_data['price']), '.3f')
rtepoly1 = format(poly.score(X_testpoly, test_data['price']), '.3f')

polyfeat = PolynomialFeatures(degree=3)
X_trainpoly = polyfeat.fit_transform(train_data[features2])
X_testpoly = polyfeat.fit_transform(test_data[features2])
poly = linear_model.LinearRegression().fit(X_trainpoly, train_data['price'])

predp = poly.predict(X_testpoly)
msepoly2 = format(np.sqrt(metrics.mean_squared_error(test_data['price'], pred)), '.3f')
rtrpoly2 = format(poly.score(X_trainpoly, train_data['price']), '.3f')
rtepoly2 = format(poly.score(X_testpoly, test_data['price']), '.3f')

r = evaluation.shape[0]
evaluation.loc[r] = ['Polynomial Regression', 'degree=2', msepoly1, rtrpoly1, '--', rtepoly1, '--']
evaluation.loc[r+1] = ['Polynomial Regression', 'degree=3', msepoly2, rtrpoly2, '--', rtepoly2, '--']
evaluation.sort_values(by = 'R-squared (test)', ascending=False)

```

Gambar 20.17. *Applying polynomial transformation*

Dengan menambahkan fitur-fitur polinomial, kami berhasil mengurangi log regresi model kami menjadi 210486.689.

	Adjusted R-squared (test)	Adjusted R-squared (training)	Details	Mean Squared Error (MSE)	Model	R-squared (test)	R-squared (training)
6	Polynomial Regression		degree=3	254289.149		0.723	-
5	Polynomial Regression		degree=2	254289.149		0.716	-
4	Complex Model-2		-	210486.689	0.651 0.650	0.655	0.654
3	Complex Model-1		-	248514.011	0.514 0.514	0.519	0.518
0	Simple Model		-	254289.149	0.492	0.496	-
1	Simple Model		-	254289.149	0.492	0.496	-
2	Simple Model		-	254289.149	0.492	0.496	-

Gambar 20.18. *comparison of all predictions*

Pastikan untuk memulai dengan model sederhana dan kemudian tingkatkan kompleksitasnya dengan menambahkan fitur-fitur tambahan. Selalu periksa berbagai metrik evaluasi untuk mendapatkan model yang akurat.

20.2 KESIMPULAN

Model-model seperti ini sangat membantu dalam penjualan dan dapat membantu para perwakilan penjualan untuk fokus pada fitur-fitur penting dalam suatu properti yang akan dijual. Kini Anda telah memiliki pengalaman langsung dalam menangani masalah terkait penjualan.