

ANALISIS DAN VISUALISASI DATA MENGUNAKAN PYTHON



MOCH. ANDYKA SAPUTRA

LEMBAR KESEPAKATAN

PEMBUATAN BUKU TUTORIAL

- | | |
|---------------------|---|
| 1. Nama Mahasiswa | : Moch. Andyka Saputra |
| 2. NIM | : 190411100070 |
| 3. Program Studi | : Teknik Informatika |
| 4. Jenis | : Menyusun Buku Tutorial |
| 5. Topik/Judul Buku | : Analisis dan Visualisasi Data
Menggunakan Python |

Bangkalan, 20 Juni 2022

Menyetujui,
Dosen Pembimbing

Penyusun,
Mahasiswa

Ika Oktavia Suzanti S.Kom., M.Cs
NIP. 19881018 201504 2 004

Moch. Andyka Saputra
NIM. 190411100070

LEMBAR PENGESAHAN

Telah diperiksa dan diuji oleh :

Pada tanggal :

Dengan Nilai :

Koordinator Kerja Praktek

Dosen Pembimbing

Fika Hastarita Rachman, S.T., M.Eng

NIP. 19830305 200604 2 002

Ika Oktavia Suzanti S.Kom., M.Cs

NIP. 19881018 201504 2 004

KATA PENGANTAR

Segala puji bagi Allah SWT yang Maha Pengasih lagi Maha Penyayang, yang telah memberikan rahmat, taufik, serta hidayah-Nya sehingga penulis dapat menyelesaikan pembuatan Buku Tutorial KP (Kerja Praktek) dengan judul “Analisis dan Visualisasi Data Menggunakan Python” dengan baik tanpa suatu halangan apapun.

Dapat terselesaikannya buku tutorial ini tidak luput dari kerjasama dan bantuan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan terimakasih sebesar-besarnya kepada :

1. Kedua Orang Tua dan Keluarga yang saya cintai, terimakasih atas dukungan serta do'anya.
2. Ibu Fika Hastarita Rachman, S.T., M.Eng selaku Koordinator KP (Kerja Praktek) di Program Studi Teknik Informatika Fakultas Teknik Universitas Trunojoyo Madura.
3. Ibu Ika Suzanti S.Kom., M.SC. selaku dosen pembimbing yang senantiasa memberikan arahan, masukan dan juga ilmu selama Kerja Praktek.
4. Bapak dan Ibu Dosen Program Studi Teknik Informatika Fakultas Teknik Universitas Tunojoyo Madura.
5. Serta teman-teman yang telah memberikan semangat dan dukungan dalam penyelesaian Buku Tutorial Kerja Praktek ini.

Penulis berharap agar buku ini bermanfaat bagi pembaca untuk menambah ilmu pengetahuan. Disamping itu, penulis menyadari bahwa masih banyak kekurangan dalam penulisan buku tutorial ini, baik dari segi bahasa, susunan kalimat, dan juga isi. Penulis berharap saran dan kritik dari pembaca guna membantu penulis agar lebih baik lagi untuk kedepannya.

DAFTAR ISI

LEMBAR KESEPAKATAN	1
LEMBAR PENGESAHAN	2
KATA PENGANTAR	3
DAFTAR ISI	4
DAFTAR GAMBAR	6
BAB 1 Pengenalan dan Penyetelan Instalasi.....	11
1.1 Python.....	11
1.2 Jupyter Notebook.....	18
1.3 Library Python.....	27
BAB 2 Analisis dan Visualisasi Data	35
2.1 Data, Analisis dan Visualisasi	35
2.2 Memilih Visual yang Efektif	37
2.3 Langkah-Langkah Analisis Data	41
2.4 Membedah Model Visual	42
2.5 Jenis Analisis Data.....	45
2.6 Kebutuhan Eksplorasi Analisis Data	47
BAB 3 Konsep Dasar Python	48
3.1 Import Library.....	48
3.2 Membaca Format File	49
3.3 Series Objek Pandas	50
3.4 DataFrame	53
3.5 Tipe Data	58
3.6 Pengoperasian Data	60
3.7 Pivot Table	64
3.8 Loc dan Iloc.....	65
BAB 4 Mengelola Data	68
4.1 Persiapan Data	68
4.2 Distribusi Data.....	79
4.3 Deskripsi Data.....	82
4.4 Korelasi Data	83
4.5 Visualisasi Data	85
BAB 5 Eksplorasi Data Tip	88

5.1	Pemahaman Bisnis	88
5.2	Pemahaman Data	88
5.3	Persiapan Data	88
5.4	Eksplorasi dan Penyajian Data	94
5.5	Penarikan Kesimpulan dan Informasi	101
BAB 6	Prediksi Harga Rumah	103
6.1	Pemahaman Bisnis	103
6.2	Pemahaman Data	103
6.3	Persiapan Data	104
6.4	Eksplorasi dan Penyajian Data	107
6.5	Pemodelan Regresi	109
Daftar Pustaka		113

DAFTAR GAMBAR

Gambar 1. 1 Logo Python	11
Gambar 1. 2 Bahasa Pemrograman Terpopuler	13
Gambar 1. 3 Install Phyton.....	15
Gambar 1. 4 Setup Python.....	15
Gambar 1. 5 Progress Install	16
Gambar 1. 6 Instalasi Sukses	16
Gambar 1. 7 Membuka CMD	17
Gambar 1. 8 Cek Python di CMD.....	17
Gambar 1. 9 Cek Python di Shell.....	18
Gambar 1. 10 Pilihan Anaconda	21
Gambar 1. 11 Welcome Anaconda	22
Gambar 1. 12 Persetujuan Anaconda	22
Gambar 1. 13 Pilih Instalasi Anaconda	23
Gambar 1. 14 Lokasi Anaconda.....	23
Gambar 1. 15 Opsi Anaconda.....	24
Gambar 1. 16 Proses Anaconda.....	24
Gambar 1. 17 Rekomendasi Anaconda	25
Gambar 1. 18 Selesai Anaconda	25
Gambar 1. 19 Anaconda Navigator.....	26
Gambar 1. 20 Lingkungan Anaconda	27
Gambar 1. 21 Tampilan JN	27
Gambar 1. 22 Lingkungan JN	27
Gambar 1. 23 Series	29
Gambar 1. 24 DataFrame	29
Gambar 1. 25 Install Pandas.....	30
Gambar 1. 26 Array 1D	31
Gambar 1. 27 Array 2D	31
Gambar 1. 28 Install Numpy	32
Gambar 1. 29 Install Matplotlib	33
Gambar 1. 30 Install Scikit-Learn.....	34

Gambar 2. 1 Data, Informasi, Pengetahuan.....	36
Gambar 2. 2 Tahapan Data	37
Gambar 2. 3 Grafik 1.....	38
Gambar 2. 4 Grafik 2.....	39
Gambar 2. 5 Peramalan Geafik Garis	42
Gambar 2. 6 Bar Bertumpuk Vertikal	43
Gambar 2. 7 Bar Bertumpuk Horizontal	44
Gambar 3. 1 Proses Membuka Jupyter Notebook	48
Gambar 3. 2 Tampilan Jupyter Notebook	48
Gambar 3. 3 Pilih Python 3	49
Gambar 3. 4 Berhasil Membuka Python 3 di Jupyter Notebook.....	49
Gambar 3. 5 Series Pandas.....	51
Gambar 3. 6 Output code program array	51
Gambar 3. 7 Menampilkan value array	52
Gambar 3. 8 Menampilkan Index	52
Gambar 3. 9 Mengakses value dengan index.....	52
Gambar 3. 10 Hasil output code program penggunaan series.....	53
Gambar 3. 11 Menampilkan nilai secara random.....	53
Gambar 3. 12 Dataframe Pandas	54
Gambar 3. 13 Membuat series baru area	55
Gambar 3. 14 Membuat series baru populasi	55
Gambar 3. 15 Membangun objek dua dimensi tunggal.....	55
Gambar 3. 16 Menampilkan kota sesuai index	56
Gambar 3. 17 Menampilkan kolom dataframe	56
Gambar 3. 18 Menambah objek Series tunggal.....	57
Gambar 3. 19 Menambah List dictionary menjadi DataFrame	57
Gambar 3. 20 Values NaN	57
Gambar 3. 21 Dictionary objek Series.....	57
Gambar 3. 22 Menambah Dataframe	58
Gambar 3. 23 dtypes.....	58
Gambar 3. 24 Mengubah tipe seri.....	59

Gambar 3. 25 Ubah tipe data kolom	59
Gambar 3. 26 Mengubah nilai menjadi tipe numerik	60
Gambar 3. 27 Penggunaan parameter error	60
Gambar 3. 28 Membuat Series dan DataFrame dengan nilai random	61
Gambar 3. 29 Menerapkan ufunc NumPy di Series	62
Gambar 3. 30 Menerapkan ufunc NumPy di Dataframe	62
Gambar 3. 31 Menghitung kepadatan populasi	63
Gambar 3. 32 Penjumlahan dengan cara pencocokan index.....	63
Gambar 3. 33 Memodifikasi nilai pengisian dengan metode objek	64
Gambar 3. 34 Membuat dataframe baru	64
Gambar 3. 35 Menggabungkan & menjumlahkan di pivot table	65
Gambar 3. 36 Menggunakan Indeks eksplisit dan implicit	66
Gambar 3. 37 Penggunaan atribut loc	66
Gambar 3. 38 Penggunaan atribut iloc.....	67
Gambar 4. 1 Data Internal.....	69
Gambar 4. 2 Data Eksternal.....	70
Gambar 4. 3 Data Hilang	73
Gambar 4. 4 Menangani Data Hilang.....	73
Gambar 4. 5 Menambah Data Hilang	73
Gambar 4. 6 Not Null	74
Gambar 4. 7 Filter Missing Data	75
Gambar 4. 8 Dropna Data.....	75
Gambar 4. 9 All Dropna Data.....	75
Gambar 4. 10 Missing Random Data	76
Gambar 4. 11 Rubah Data Missing ke Nol.....	76
Gambar 4. 12 Rubah Data Missing Rentang	76
Gambar 4. 13 Tetapkan Perubahan Data	77
Gambar 4. 14 Duplikasi Data	78
Gambar 4. 15 Menampilkan Duplikasi Data	78
Gambar 4. 16 drop_duplicates.....	78
Gambar 4. 17 Filter Duplikasi Data	79
Gambar 4. 18 Distribusi Pada Data	79

Gambar 4. 19 box-and-whisker	80
Gambar 4. 20 Boxplot	81
Gambar 4. 21 Distribusi pada Data	81
Gambar 4. 22 Deskripsi Data	82
Gambar 4. 23 Korelasi Scatter Plot.....	83
Gambar 4. 24 Korelasi Peta Panas	84
Gambar 4. 25 Korelasi Data.....	85
Gambar 4. 26 Visualisasi Matplotlib	86
Gambar 4. 27 Visualisasi Seaborn.....	87
Gambar 5. 1 Membaca Data Tip	89
Gambar 5. 2 Data Tip Baru	89
Gambar 5. 3 Tipe Data Tip.....	90
Gambar 5. 4 Distribusi Data Tip	90
Gambar 5. 5 Grafik Distribusi Data Tip	91
Gambar 5. 6 Data Tip Kosong	91
Gambar 5. 7 Data Tip Duplikat.....	92
Gambar 5. 8 Data Tip Hapus Duplikat	92
Gambar 5. 9 Deskripsi Data Tip	92
Gambar 5. 10 Data Tip Unik	93
Gambar 5. 11 Boxplot Data Tip.....	93
Gambar 5. 12 Korelasi Data Tip.....	94
Gambar 5. 13 Data Tip Jenkel	94
Gambar 5. 14 Data Tips.....	95
Gambar 5. 15 Data Tips 1	95
Gambar 5. 16 Data Tips 2.....	96
Gambar 5. 17 Hari Tip Besar	96
Gambar 5. 18 Pivot Data Tip.....	97
Gambar 5. 19 Hari Tip Besar 1	97
Gambar 5. 20 Tip Perokok	98
Gambar 5. 21 Hari Tip Perokok	98
Gambar 5. 22 Pivot Data Tip 1.....	99
Gambar 5. 23 Tip Perokok 1	99

Gambar 5. 24 Tip Jenkel.....	100
Gambar 5. 25 Tip Perokok	100
Gambar 5. 26 Data Tip Meja	101
Gambar 6. 1 Data Fitur Boston	105
Gambar 6. 2 Distribusi Boston	105
Gambar 6. 3 Data Boston	106
Gambar 6. 4 Persiapan Data Boston	106
Gambar 6. 5 Deskripsi Data Boston.....	107
Gambar 6. 6 Korelasi Data Boston	107
Gambar 6. 7 Pairplot Boston	108
Gambar 6. 8 Fitur Price.....	109
Gambar 6. 9 Regresi.....	110
Gambar 6. 10 Hasil Split Boston	110
Gambar 6. 11 Regresi Linear Boston.....	111
Gambar 6. 12 Regresi Plot.....	112

Pengenalan dan Penyetelan Instalasi

1.1 Python

Pengertian Bahasa Pemrograman Python



Gambar 1. 1 Logo Python

Apa yang orang awam pikirkan Ketika mendengar kata python? Sejenis ular bukan, tetapi kali ini python adalah salah satu bahasa pemrograman. Python merupakan salah satu bahasa pemrograman tingkat tinggi, bahasa Python cukup populer karena mudah untuk dibaca dan dituliskan oleh manusia. Bahasa pemrograman Python pertama kali dirilis oleh Guido van Rossum yakni pada tahun 1991, yang mana sebelumnya sudah dikembangkan sejak tahun 1989.

Python dapat melakukan eksekusi sejumlah instruksi multiguna secara langsung dengan metode orientasi objek (*Object Oriented Programming*) serta menggunakan semantik dinamis untuk memberikan tingkat keterbacaan syntax. Python merupakan bahasa yang kemampuannya untuk menggabungkan kapabilitas dan sintaksis kode yang sangat jelas dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Bahasa Python sendiri tergolong bahasa pemrograman yang

mempunyai level tinggi, serta dirancang sedemikian rupa agar mudah dipelajari dan dipahami. Python sendiri menampilkan fitur-fitur yang menarik sehingga bagus untuk anda pelajari. Pertama, Python mempunyai tata bahasa dan script yang sangat mudah untuk dipelajari. Python juga mempunyai sistem pengelolaan data dan memori otomatis. Selain itu modul pada bahasa pemrograman Python selalu diupdate. Ditambah lagi, bahasa Python ini juga memiliki banyak fasilitas pendukung. Python banyak diaplikasikan pada berbagai sistem operasi.

Kelebihan dan Kekurangan Python

Kelebihan pada Python yaitu Python bisa dengan mudah dipelajari bahkan untuk pengembang pemula, dikarenakan kodenya mudah dibaca dan bisa menjalankan banyak fungsi kompleks dengan mudah, karena banyaknya sejumlah besar *library* atau pustaka tersedia untuk Python. Sehingga pengembangan program bisa dilakukan dengan cepat dan juga menggunakan kode yang lebih sedikit. Bahkan tim kecil bisa menangani bahasa Python secara efektif. Hal ini memungkinkan membuat program dengan skala yang paling rumit dengan mudah, dan juga Python mempunyai sistem pengelolaan memori yang otomatis, *garbage collection*, layaknya Java.

Kekurangan pada Python yaitu cukup lambat untuk dijalankan. Bahkan Python terbilang buruk dalam pengembangan platform mobile (Android/iOS). Sehingga Python bukanlah menjadi pilihan yang baik untuk tugas-tugas intensif memori. Hampir mustahil untuk membuat game 3 dimensi grafis tinggi menggunakan Python. Selain itu Python mempunyai keterbatasan dengan akses basis data. Python tidak baik jika diperuntukan dalam pekerjaan multi-prosesor / multi-core.

Adapun persyaratan Spesifikasi hardware minimum yang bisa kita gunakan untuk menjalankan software python ini antara lain:

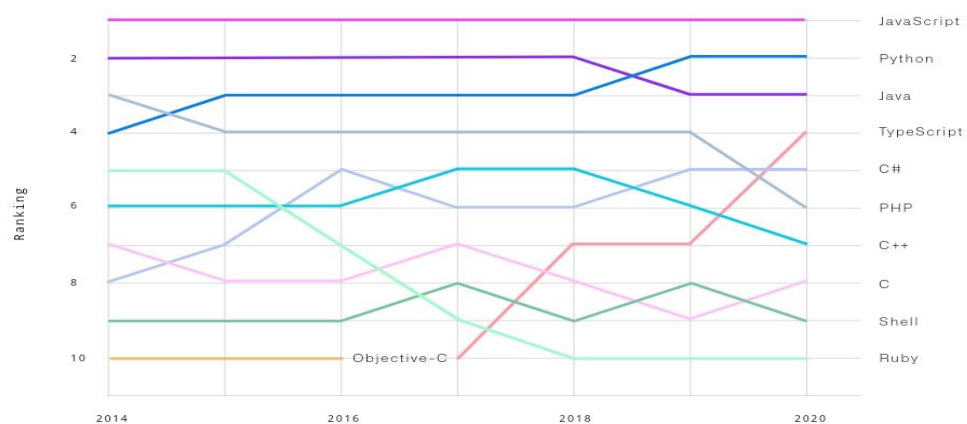
- 1.) Resolusi layar 1024 x 768 atau lebih tinggi
- 2.) Prosesor Intel Celeron (Rekomendasi Core i3 ke atas)
- 3.) RAM 1GB (Rekomendasi 2GB)
- 4.) Sistem operasi Windows, Linux, atau MacOS

Sejarah Perkembangan Python

Python dibuat dan dikembangkan oleh Guido Van Rossum, yaitu seorang programmer yang berasal dari Belanda, pembuatannya berlangsung di kota Amsterdam, Belanda pada tahun 1990. Pada tahun 1995 Python dikembangkan lagi agar lebih kompatibel oleh Guido Van Rossum. Selanjutnya pada awal tahun 2000, terdapat pembaharuan versi Python hingga mencapai Versi 3 sampai saat ini. Pemilihan nama Python sendiri diambil dari sebuah acara televisi yang lumayan terkenal yang bernama Mothy Python Flying Circus yang merupakan acara sirkus favorit dari Guido van Rossum.

Mengapa Harus Python

Bahasa pemrograman Python sangat populer, menurut data dari github.com pada tahun 2020 Python menempati urutan ke-2 sebagai bahasa pemrograman terpopuler. Python juga dapat digunakan di server untuk membuat aplikasi web, data science, data analysis, machine learning, dll.



Gambar 1. 2 Bahasa Pemrograman Terpopuler

Dapat diketahui menurut data tersebut bahwa Python dikenal sebagai salah satu jenis bahasa pemrograman nomor dua yang paling populer. Python juga banyak digunakan oleh para programmer untuk membangun sebuah software, sama seperti bahasa pemrograman lainnya. Tetapi kini kita belajar Python untuk analisis dan visualisasi data. Masih sedikit orang yang tahu bahwa kita bisa menggunakan bahasa programming Python untuk analisis data. Adapun penggunaan yang dimaksud di sini tentunya mengolah data dengan Python. Bahkan, kita juga bisa sekaligus melakukan visualisasi data dengan Python. Tentunya pekerjaan kita akan semakin mudah dan menarik

dengan adanya Python untuk mengolah data. Jadi tidak ada salahnya untuk kita mulai belajar Python untuk analisis data dari sekarang.

Pada saat ini kita sedang berada pada era big data atau revolusi industri 4.0 yang mana big data merupakan istilah yang merujuk pada kumpulan data yang sangat besar jumlahnya. Selain itu, big data juga terdiri dari berbagai jenis data seperti data terstruktur, semi-struktur, dan tidak terstruktur. Terdapat minimal 3V dalam big data yakni volume atau jumlah, variety atau variasi, dan velocity atau kecepatan. Sehingga mustahil bagi ahli atau profesional IT sekalipun untuk mampu mengolah big data tanpa menggunakan bantuan alat yang canggih dan modern. Dari sanalah, para ahli IT mencari tahu kira-kira software atau piranti IT apa saja yang bisa dipergunakan untuk mengolah big data. Hingga akhirnya diketahui bahwa belajar Python untuk analisis data dapat dilakukan. Proses mengolah data dengan Python dapat dilakukan dengan jauh lebih mudah, termasuk visualisasi data dengan Python.

Python adalah suatu bahasa pemrograman *scripting language* yang mempunyai orientasi atau fokus terhadap objek data. Jadi, selain bisa dipakai untuk mengembangkan dan memproduksi software, kita pun bisa mengolah dan visualisasi data dengan Python. Berkat fungsi ini pun, Python akhirnya terkenal sebagai bahasa pemrograman yang kerap dipakai dalam analisis big data dan metode di ilmu pengolahan data. Python untuk analisis data memang dimungkinkan sebab bahasa programing ini mempunyai aneka *library* dengan fungsi-fungsi untuk analisis data, data *pre-processing tools*, *machine learning* dan visualisasi data.

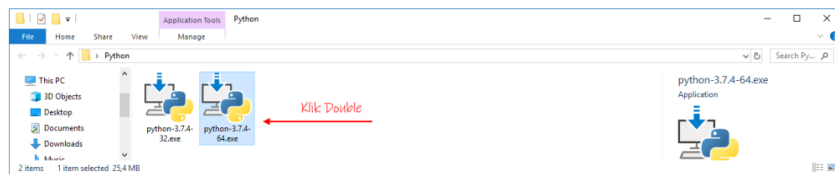
Python tentunya mempunyai ciri-ciri khusus yang membedakannya dengan bahasa programming lainnya. Adapun beberapa ciri dan karakteristik khas dari Python yaitu bersifat *open-source*, portable dan mudah dipahami. Python memiliki dan mendukung banyak *library*, selain itu juga mudah untuk melakukan pengecekan kode karena ada aturan layout *source code*, dan Python merupakan bahasa sudah diinterpretasikan (interpreted), sehingga memudahkan dalam proses *debugging* (penanganan bug).

Instalasi Python

Cara penginstalan Python cukup mudah, download aplikasi Python kemudian install seperti pada umumnya, klik next, next lagi hingga selesai. Berikut alamat link untuk mengdownload software Python [Python Releases for Windows | Python.org](https://www.python.org/downloads/windows/).

Adapun langkah – langkah menginstall Python sebagai berikut :

- Buka file Python yang telah selesai diunduh. Sekali lagi, pada tutorial ini menggunakan aplikasi Python versi 3.7.4-64bit. Klik double pada file Python tersebut (yang telah didownload) atau klik kanan kemudian open.



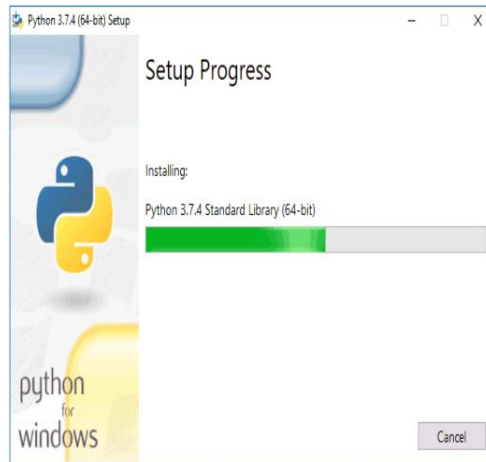
Gambar 1. 3 Install Phyton

- Setup, pada tahapan ini ada pengaturan khusus yang harus dipilih saat proses instalasi, agar perintah Python dapat dikenali di CMD. Pertama ceklist Add python 3.7 to PATH kemudian klik install now.



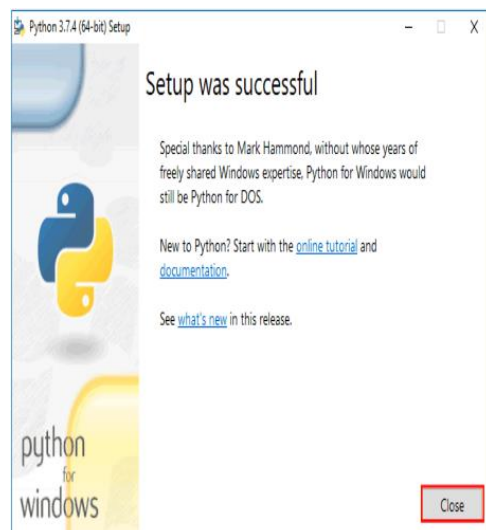
Gambar 1. 4 Setup Python

- Proses instalasi, pada Tahapan (Setup) kali ini, tunggu hingga proses instalasi selesai.



Gambar 1. 5 Progress Install

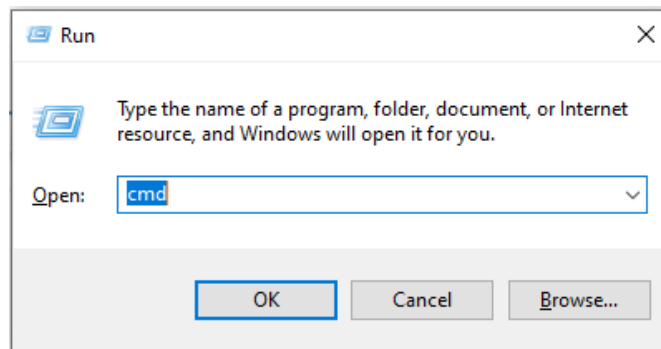
- *Setup was Successful*, instalasi Python berhasil, kemudian Klik close. Setelah ini Anda bisa menguji python apakah sudah bisa digunakan atau belum, bisa menggunakan CMD atau Python Shell.



Gambar 1. 6 Instalasi Sukses

Uji Coba Python dengan CMD

Setelah berhasil menginstall Python pada windows maupun OS lainnya. Selanjutnya, Karena tutorial ini menggunakan Windows 10, maka saya akan mengujinya menggunakan CMD. Buka CMD dengan cara klik windows+R kemudian ketik cmd lalu enter atau ok.



Gambar 1. 7 Membuka CMD

Setelah berhasil masuk ke CMD, ketikkan perintah "Python" pada CMD untuk masuk ke mode Python. perintah tersebut juga berguna untuk cek versi Python. Selanjutnya ketikkan perintah 'print("Halo Indonesia!")' kemudian enter untuk melakukan pengujian. Jika berhasil akan bisa dilihat pada gambar 1.8 berikut.

```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 10.0.17763.475]
(c) 2018 Microsoft Corporation. All rights reserved.

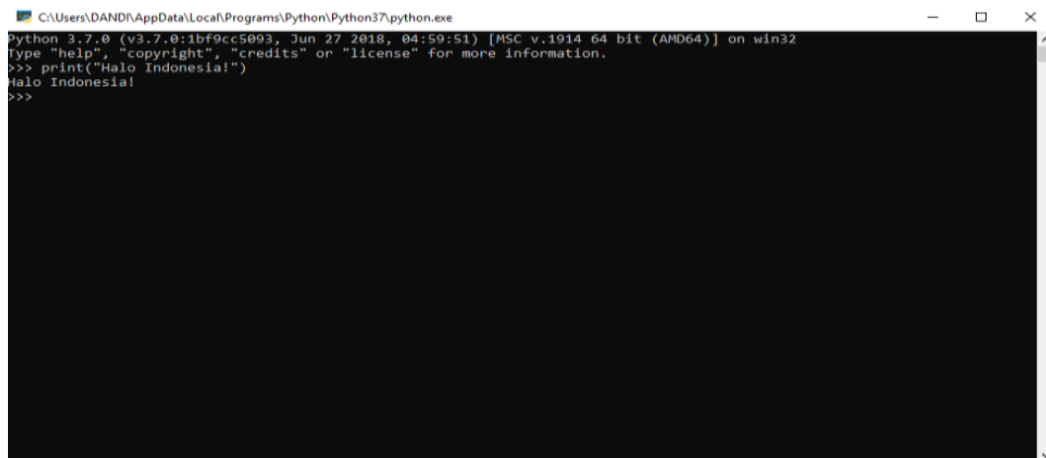
C:\Users\DANDI>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Halo Indonesia!")
Halo Indonesia!
>>> _
```

Gambar 1. 8 Cek Python di CMD

Uji Coba Python di Shell

Alternatif lainnya selain CMD yaitu, Anda bisa juga menggunakan Python Shell yang sudah otomatis terinstal saat selesai instalasi aplikasi Python. proses uji coba dengan python shell juga sama dengan CMD. Pertama cari dan buka Python shell, jika pada windows, ketik perintah 'python shell' pada fitur windows search. setelah ketemu, klik kanan pada python shell, lalu run as administrator. Jika command dari python shell sudah terbuka, Selanjutnya Anda bisa men-inputkan perintah python untuk cek versi python.

Lalu ketik perintah `print("Halo Indonesia!")` untuk uji coba. berikut contohnya bisa dilihat pada gambar 1.9.



```
C:\Users\DANDI\AppData\Local\Programs\Python\Python37\python.exe
Python 3.7.0 (tags/v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Halo Indonesia!")
Halo Indonesia!
>>>
```

Gambar 1. 9 Cek Python di Shell

1.2 Jupyter Notebook

Pengertian Jupyter Notebook

Jupyter Notebook merupakan tools yang sangat populer untuk mengolah data dibahasa pemograman Python. Jupyter Notebook adalah sebuah open-source, berbasis browser alat yang berfungsi sebagai notebook lab virtual untuk mendukung alur kerja, kode, data, dan visualisasi yang merinci penelitian proses.

Jupyter Notebook memungkinkan kita untuk dapat mengintegrasikan antara kode dengan output yang di dalam satu dokumen secara interaktif. Jupyter adalah organisasi non-profit untuk mengembangkan software yang interaktif dalam berbagai macam-macam bahasa pemrograman.

Sedangkan Notebook merupakan suatu software buatan Jupyter, ialah aplikasi web open-source yang memungkinkan kita membuat dan berbagi dokumen secara interaktif yang berisi kode live, persamaan, visualisasi, dan teks naratif yang kaya.

Komponen Jupyter Notebook

1.) Teks dan HTML

Teks biasa, atau teks yang dianotasikan ke dalam sintaks Markdown untuk menghasilkan suatu HTML, dan juga dapat dimasukkan ke dalam dokumen kapan saja. Kode program CSS juga dapat dimasukkan ke

inline atau ditambahkan ke tempat yang akan digunakan untuk menghasilkan notebook.

2.) Kode dan Hasil

Kode yang ada pada Jupyter Notebook biasanya merupakan kode Python, walaupun kita juga dapat menambahkan kode bahasa pemrograman lainnya seperti bahasa R atau Julia. Hasil output kode yang telah dieksekusi akan muncul segera setelah blok kode di running.

3.) Visualisasi

Sebuah grafik dan bagan dapat dihasilkan dari sebuah kode program, dengan menggunakan modul seperti library Matplotlib dan juga Plotly. Seperti halnya output sebuah kode, visualisasi ini akan muncul dibawah kode yang menghasilkannya. Namun, sebuah kode juga bisa di konfigurasi untuk menuliskannya ke dalam sebuah file eksternal jika nantinya diperlukan.

4.) Multimedia

Mengingat Jupyter Notebook dibangun dengan teknologi web, maka Jupyter Notebook dapat menampilkan semua jenis-jenis multimedia yang didukung di halaman web tersebut. Kita bisa memasukkannya ke dalam sebuah buku catatan sebagai elemen HTML, atau kita dapat juga membuatnya secara langsung atau terprogram melalui modul `IPython.display`.

5.) Data

Suatu data dapat kita berikan di dalam sebuah file yang terpisah di samping file `.ipynb` yang merupakan notebook di Jupyter Notebook, atau dapat kita import secara langsung atau terprogram di suatu kode, misalnya dengan memasukkan sebuah kode program dalam notebook 33 untuk mengunduh data dari repositori Internet publik atau juga dapat mengaksesnya melalui koneksi basis data.

Kelebihan Jupyter Notebook

Ada berbagai macam kegunaan dari Jupyter Notebook dalam semua jenis-jenis proyek, antara lain :

1.) Visualisasi data

Kebanyakan dari orang memilih eksposur pertama mereka menggunakan Jupyter Notebook dengan cara visualisasi data.. Pada Jupyter Notebook tidak hanya untuk memungkinkan kita membuat sebuah visualisasi data saja, tetapi juga akan membagikannya dan membuat perubahan interaktif pada kode program dan juga kumpulan data yang akan dibagikan.

2.) Berbagi kode

Pada layanan cloud seperti di GitHub dan Pastebin sudah menyediakan berbagai cara untuk kita dapat bisa berbagi suatu kode program, akan tetapi sebagian besar itu ialah non-interaktif. Dengan Jupyter Notebook ini, kita dapat melihat suatu kode, menjalankannya kode, dan menampilkan hasil outputnya secara langsung pada browser web kita.

3.) Interaksi langsung dengan kode

Suatu kode yang ada pada Jupyter Notebook sifatnya itu tidak statis, maksudnya kode tersebut dapat kita edit dan kita dijalankan kembali secara bertahap. Jupyter Notebook juga bisa melakukan kontrol pengguna yang dapat kita gunakan sebagai sumber input pada suatu kode program.

4.) Mendokumentasikan contoh kode

Apabila kita mempunyai suatu kode dan kita ingin menjelaskannya secara bertahap bagaimana cara kerjanya, Dengan Jupyter Notebook selain kita dapat membuat suatu kode program, kode tersebut akan tetap berfungsi penuh dan kita juga dapat menambahkan suatu interaktivitas bersama dengan penjelasan code tersebut, yang selanjutnya akan ditampilkan dan diceritakan pada saat yang sama.

Keterbatasan Jupyter Notebook

Selain itu Jupyter Notebook memiliki beberapa keterbatasan antara lain :

1.) Jupyter Notebook tidak mandiri

Satu-satunya kelemahan terbesar dari penggunaan Jupyter Notebook: Notebook membutuhkan suatu runtime Jupyter, bersamaan dengan library yang kita gunakan. Ada beberapa strategi untuk kita dapat membuat Jupyter Notebook itu bisa mandiri, akan tetapi tidak ada satupun dari strategi tersebut yang didukung secara resmi.

2.) Status sesi tidak dapat disimpan dengan mudah

Keadaan kode program apapun itu yang berjalan di dalam Jupyter Notebook tidak dapat dikembalikan dan dipertahankan dengan toolset default dari Jupyter Notebook itu sendiri. Pada saat kita ingin memuat lagi notebook tersebut , kita harus menjalankan kembali kode tersebut yang ada dalamnya agar dapat memulihkan kondisi urutan kode.

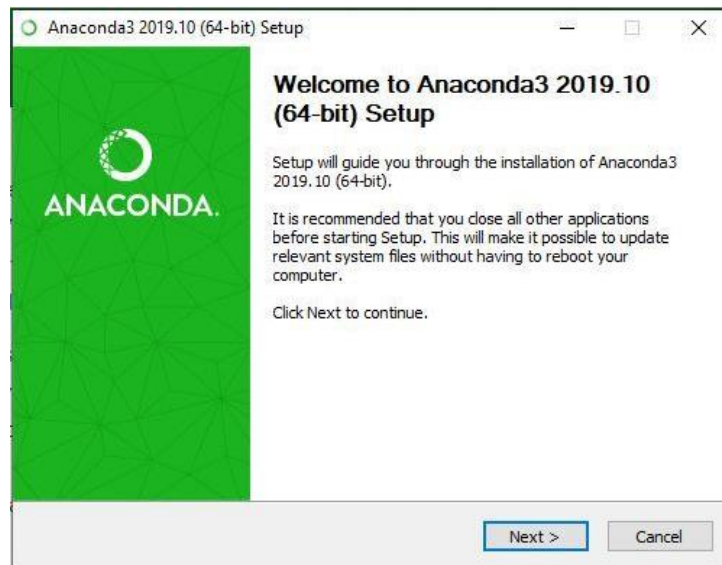
Instalasi Jupyter Notebook

- Pada kesempatan kali ini kita akan menginstall jupyter notebook di anaconda. Anaconda merupakan suatu perangkat lunak sumber terbuka (*Open Source*) yang berisi Jupyter Notebook, spyder, Orange, dll yang digunakan untuk pemrosesan big data, analisis data, komputasi ilmiah berat, dan sebagainya. Kembali ke instalasi, buka [Anaconda | The World's Most Popular Data Science Platform](#) install sesuai OS dan arsitektur kalian, di sini saya menggunakan windows 64-bit.



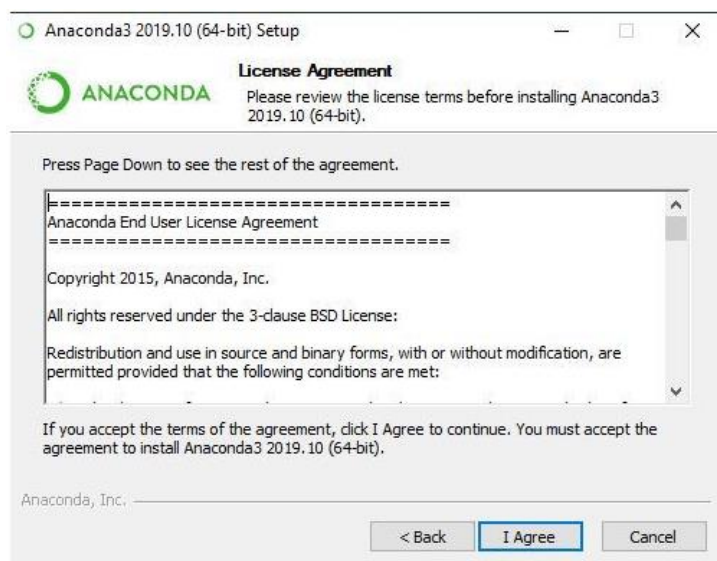
Gambar 1. 10 Pilihan Anaconda

- Temukan anaconda yang telah di download, kemudian double klik dan mulai instalasi



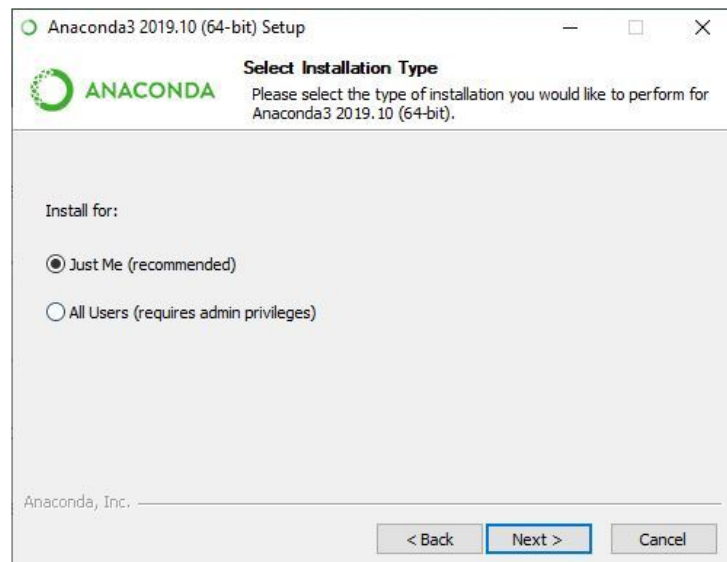
Gambar 1. 11 Welcome Anaconda

- Terdapat penjanjian penggunaan atau lisesnsi, pilih I Agree. Silahkan jika mau dibaca



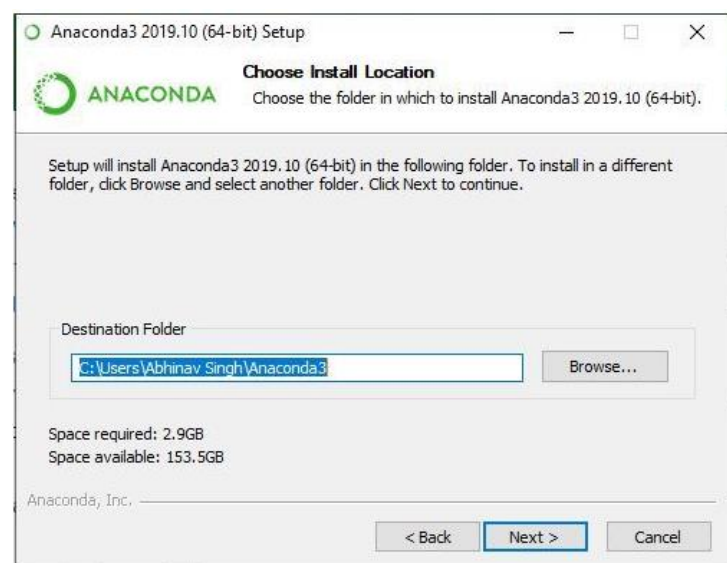
Gambar 1. 12 Persetujuan Anaconda

- Silahkan pilih jenis instalasi. Terdapat dua pilihan *Just me* atau *All User*. Pilih *Just me* jika perangkat lunak hanya digunakan oleh satu pengguna dan *All user* jika beberapa pengguna.



Gambar 1. 13 Pilih Instalasi Anaconda

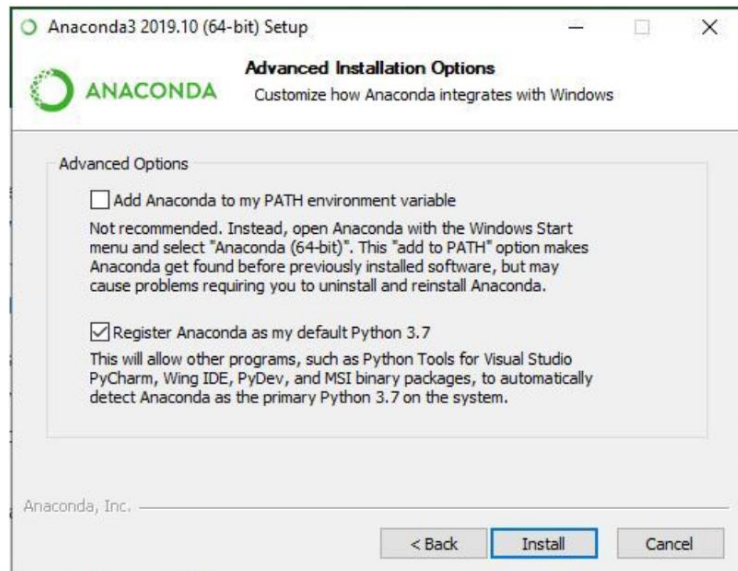
- Silahkan pilih lokasi instalasi, jika ingin dirubah silahkan pilih *Browse*. Jika sudah sesuai silahkan next



Gambar 1. 14 Lokasi Anaconda

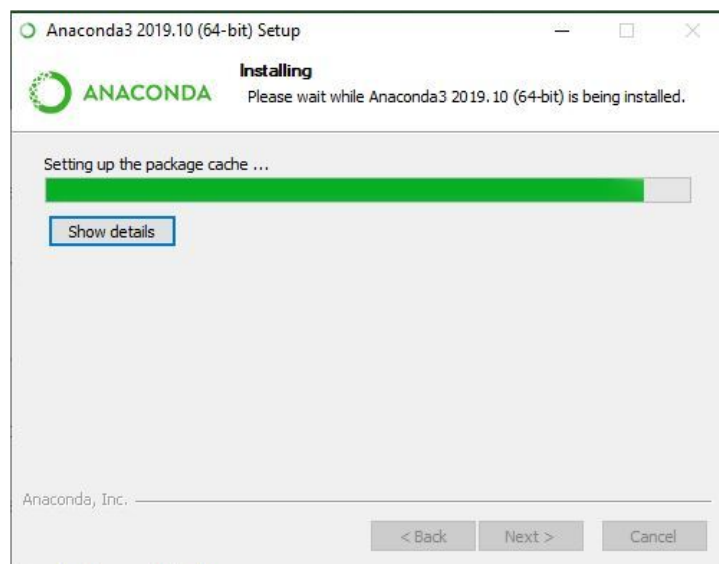
- Terdapat opsi instalasi lanjutan, berikut penjelasanya
Add anaconda to my PATH environment variable, opsi ini membuat anaconda ditemukan sebelum perangkat lunak yang diinstall sebelumnya, tetapi ini mungkin akan menyebabkan masalah yang mengharuskan Anda untuk menghapus dan menginstall perangkat lunak anaconda tersebut.

Register Anaconda as my default Python 3.7, opsi ini akan memungkinkan program lain, seperti Python tools untuk Visual Studio, PyCharm, Wing IDE, PyDev, dan paket biner MSI. Opsi ini akan otomatis mendeteksi Anaconda sebagai Python 3.7 utama pada system yang Anda gunakan.



Gambar 1. 15 Opsi Anaconda

- Silahkan menunggu proses instalasi



Gambar 1. 16 Proses Anaconda

- Terdapat rekomendasi untuk menginstal perangkat lunak PyCharm, jika berminat silahkan kunjungi link yang diberikan. Jika tidak ingin

menginstall atau sudah menginstall silahkan melanjutkan ke tahap berikutnya.



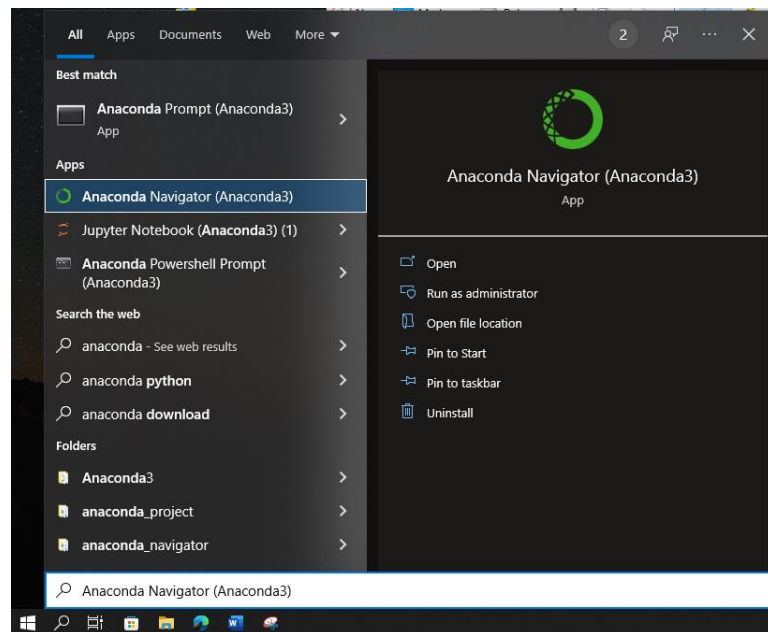
Gambar 1. 17 Rekomendasi Anaconda

- Anaconda telah berhasil terinstall,
ceklis *Learn more about Anaconda Cloud* jika Anda ingin mempelajari tentang anaconda cloud.
ceklis *Learn how to get started with Anaconda* jika Anda ingin belajar untuk memulai Anaconda pertama Anda.
Jika sudah klik finish.



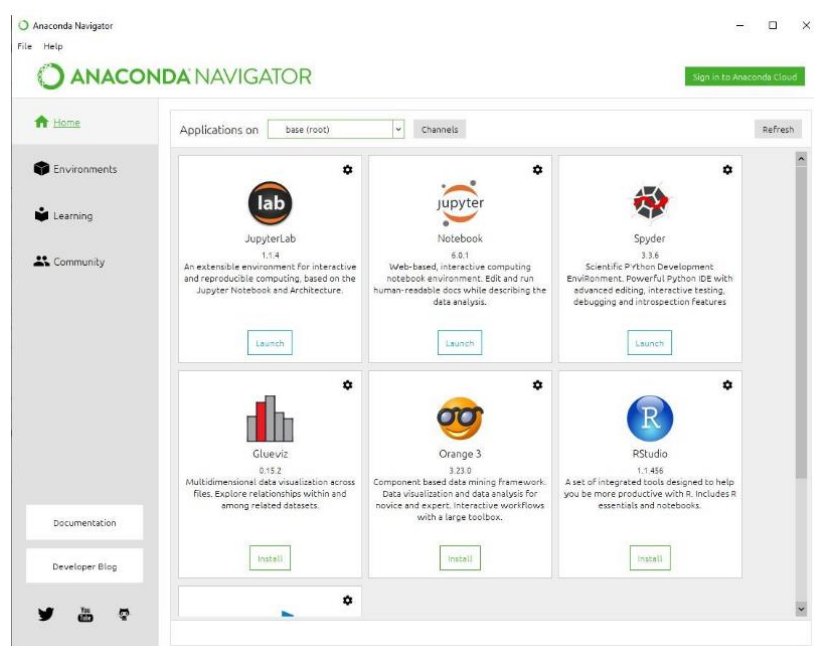
Gambar 1. 18 Selesai Anaconda

- Setelah proses instalasi Anaconda selesai, Anaconda dapat digunakan untuk melakukan beberapa operasi yang Anda inginkan. Untuk memulai menggunakan Anaconda, cari Anaconda Navigator dari Start Menu di Windows.



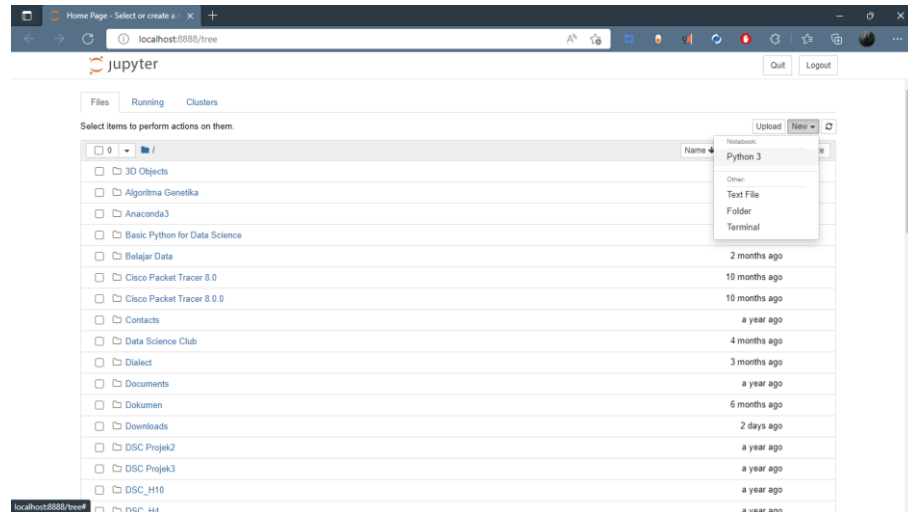
Gambar 1. 19 Anaconda Navigator

- Anda akan disuguhkan beberapa perangkat lunak yang disediakan Anaconda, karena kita akan menggunakan Jupyter Notebook silahkan jalankan Jupyter Notebook



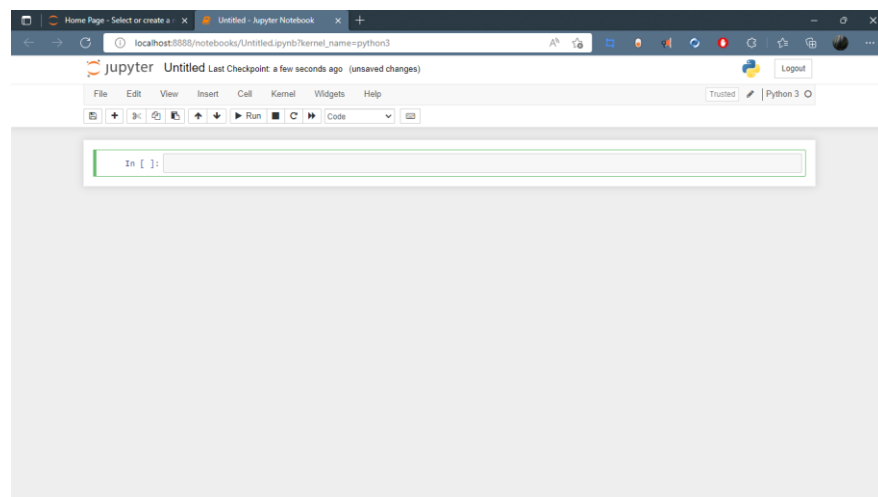
Gambar 1. 20 Lingkungan Anaconda

- Maka akan tampil seperti gambar dibawah. Silahkan menjelajahi fitur yang telah disediakan oleh Jupyter Notebook. Untuk membuat lingkungan untuk tutorial kali ini pilih Python 3, jika ingin merapikan silahkan buat folder baru terlebih dahulu kemudian pilih Python 3. Sebaiknya membuat folder baru karena data yang akan kita gunakan akan diletakkan dalam satu folder tersebut.



Gambar 1. 21 Tampilan JN

- Maka akan tampil seperti gambar dibawah dan siap digunakan



Gambar 1. 22 Lingkungan JN

1.3 Library Python

Python banyak digunakan dalam pengolahan data, apalagi data tersebut merupakan big data karena excel dan sejenisnya tidak dapat

mengolah big data. Oleh karena itu banyak yang menggunakan Python khususnya dalam proses pengolahan data. Python memiliki library yang cukup lengkap, library python adalah kumpulan modul terkait yang berisi kumpulan kode yang dapat digunakan berulang kali dalam pembuatan program. Adanya library tersebut membuat pengguna lebih efisien dan cepat dalam mengolah data, programmer juga merasa nyaman dan sederhana dalam penggunaannya.

Library Pandas

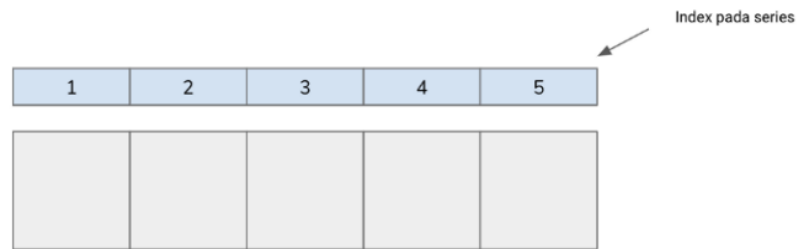
Library Python pertama yang akan kita bahas yakni pandas, pandas merupakan library Python yang digunakan untuk bekerja dengan kumpulan data. Pandas akan berfungsi untuk menganalisis, membersihkan, menjelajah, dan memanipulasi suatu data. Nama “Pandas” sendiri memiliki rujukan ke data panel dan analisis data python yang dibuat oleh Wes McKinney pada tahun 2008.

Pandas memungkinkan kita untuk dapat menganalisis big data dan membuat kesimpulan berdasarkan teori statistic. Pandas juga dapat membersihkan kumpulan data yang berantakan dan membuat data tersebut menjadi relevan dan mudah dibaca. Data yang relevan tersebut sangat penting dalam pengolahan data.

Dengan kata lain, library Pandas merupakan library untuk menganalisis data yang mempunyai struktur data yang diperlukan untuk membersihkan data yang mentah ke dalam sebuah bentuk yang cocok untuk di analisis yaitu tabel dari kumpulan data tersebut. Library Pandas melakukan tugas-tugas yang penting seperti menyelaraskan data untuk perbandingan dan penggabungan set data, serta juga untuk penanganan data yang hilang, dll. Library pandas ini telah menjadi sebuah library de facto untuk pemrosesan data tingkat tinggi dalam Python (yaitu statistik). Library Pandas pada mulanya didesain untuk menangani data finansial, dikarenakan alternatif umum adalah menggunakan spreadsheet (misalnya Microsoft Excel).

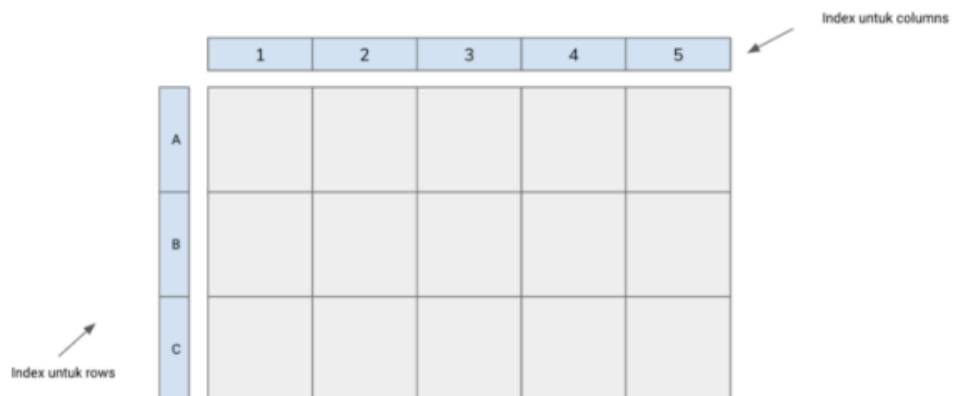
Dengan menggunakan Pandas, dapat memanfaatkan lima fitur utama dalam pemrosesan dan analisis data, yaitu load, prepare, manipulate, modelling, dan analysis data. Pandas menggunakan konsep array dari NumPy namun memberikan index kepada array tersebut, sehingga disebut series ataupun data frame. Sehingga bisa dikatakan Pandas menyimpan data dalam dictionary-based NumPy arrays. 1-Dimensi labelled array dinamakan sebagai

Series. Sedangkan 2-Dimensi dinamakan sebagai Data Frame. Bentuk dari series diilustrasikan sebagai berikut :



Gambar 1. 23 Series

Struktur data dasar pada library pandas dinamakan DataFrame, yaitu sebuah koleksi kolom berurutan dengan nama dan jenis, dengan demikian merupakan sebuah tabel yang tampak seperti database, dimana sebuah baris tunggal mewakili sebuah contoh tunggal dan kolom mewakili atribut tertentu. Harus dicatat di sini bahwa elemen dalam berbagai kolom mungkin berupa jenis yang berbeda. Bentuk dari data frame diilustrasikan sebagai berikut:



Gambar 1. 24 DataFrame

Dengan adanya fitur dataframe yang ada di library pandas memudahkan untuk membaca sebuah file dan menjadikannya sebuah table, kita juga dapat mengolah suatu data di table tersebut dengan menggunakan operasi seperti join, distinct, group by, agregasi, dan teknik lainnya yang terdapat pada SQL. Banyak format file yang bisa dibaca menggunakan library Pandas ini, seperti file .txt, .csv, .html, dan lainnya.

Berikut cara instalasi Pandas pada Jupyter Notebook. Hasil kemungkinan berbeda, disini saya sudah pernah menginstall pandas.

```
pip install pandas
Requirement already satisfied: pandas in c:\users\25and\anaconda3\lib\site-packages (1.1.3)Note: you may need to restart the ke
rnel to use updated packages.
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\25and\anaconda3\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: numpy>=1.15.4 in c:\users\25and\anaconda3\lib\site-packages (from pandas) (1.22.3)
Requirement already satisfied: pytz>=2017.2 in c:\users\25and\anaconda3\lib\site-packages (from pandas) (2020.1)
Requirement already satisfied: six>=1.5 in c:\users\25and\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas) (1.
15.0)
```

Gambar 1. 25 Install Pandas

```
pip install pandas
```

Library NumPy

Lirary Python selanjutnya yakni numpy, NumPy (Numerical Python) merupakan salah satu library python yang digunakan untuk bekerja dengan array. Library numpy memiliki fungsi untuk bekerja dalam domain aljabar linear, transformasi fourier, dan matriks. Library numpy dibuat pada tahun 2005 oleh Travis Oliphant. Numpy merupakan proyek sumber terbuka (open source) dan Anda dapat menggunakannya secara bebas.

Python memiliki daftar yang melayani tujuan dari array, tetapi mereka lambat untuk diproses. Oleh karena itu, numpy bertujuan untuk menyediakan objek array hingga 50x lebih cepat daripada list python sederhana. Pada array terdapat objek array yang disebut ndarray, ini menyediakan banyak fungsi pendukung yang membuat bekerja dengan ndarray sangat mudah. Array sangat sering digunakan dalam mengolah data, di mana kecepatan dan sumber daya sangat penting.

Numpy lebih cepat dari lists karena array numpy disimpan di satu tempat secara terus menerus dalam memori tidak seperti lists, sehingga proses manipulasi dan mengaksesnya sangat efisien. Perilaku seperti ini dapat disebut dengan lokalitas, ini merupakan alasan utama mengapa numpy lebih cepat daripada lists.

Numpy dapat digunakan untuk memproses operasi vektor, matriks, dan juga operasi matematika atau statistik. Beberapa tipe data pada Numpy yaitu boolean, integer, unsigned integer, dan float. Sintaks untuk menggunakan library Numpy sama dengan library lainnya yaitu `import numpy as np`. Penggunaan sebutan `np` umum digunakan ketika menggunakan Numpy. Kita juga bisa menggunakan Numpy untuk melakukan operasi sederhana dengan menggunakan simbol yaitu (+) untuk penjumlahan, (-) untuk pengurangan, (*)

untuk perkalian, dan (/) untuk pembagian. Operasi lain seperti pangkat bisa dituliskan dengan dua bintang (**). Numpy juga menyediakan fungsi universal function (ufunc) untuk menjalankan operasi seperti sin dan cos.

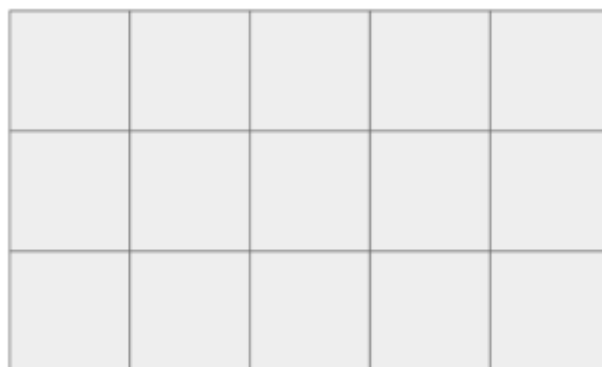
Sesuai namanya Numpy berfungsi sebagai library untuk melakukan proses komputasi numerik terutama dalam bentuk array multidimensional (1-Dimensi ataupun 2-Dimensi). Numpy array merupakan salah satu library yang disediakan oleh Python dalam memudahkan operasi komputasi tipe data numerik. Numpy bisa digunakan sebagai pengganti matlab ketika digunakan dengan Scipy dan matplotlib. Numpy menyimpan data dalam bentuk array yakni kumpulan variabel yang memiliki tipe data yang sama. Bentuk numpy array adalah multidimensional yang berarti bentuknya dapat berupa 1- dimensi, 2-dimensi, bahkan lebih. Di numpy sendiri berisi daftar panjang fungsi matematika yang berguna, termasuk beberapa fungsi untuk aljabar linier, transformasi fourier, dan rutinitas pembuatan bilangan acak.

Array merupakan kumpulan dari variabel yang memiliki tipe data yang sama. NumPy menyimpan data dalam bentuk arrays. Bentuk 1 Dimensi NumPy array dapat diilustrasikan sebagai berikut:



Gambar 1. 26 Array 1D

Sedangkan bentuk 2 Dimensi NumPy array dapat diilustrasikan sebagai berikut:



Gambar 1. 27 Array 2D

Numpy dapat digunakan sebagai pengganti matlab ketika digunakan dengan Scipy dan matplotlib. Biasanya library numpy dituliskan dengan cara singkatan np di code nya. Numpy dapat menyimpan data dalam bentuk array yaitu kumpulan variable-variable yang memiliki tipe data yang sama. Bentuk library numpy array yaitu multidimensional yang berarti bentuknya dapat berupa 1-dimensi, 2-dimensi, atau lebih.

Berikut cara instalasi Numpy pada Jupyter Notebook. Hasil kemungkinan berbeda, disini saya sudah pernah menginstall numpy.

```
pip install numpy
```

Requirement already satisfied: numpy in c:\users\25and\anaconda3\lib\site-packages (1.22.3)
Note: you may need to restart the kernel to use updated packages.

Gambar 1. 28 Install Numpy

```
pip install numpy
```

Library Matplotlib

Library Python selanjutnya yakni Matplotlib, Matplotlib adalah library plotting grafik tingkat rendah dengan python yang berfungsi sebagai utilitas visualisasi. Library matplotlib dibuat oleh D. Hunter. Library matplotlib merupakan sumber terbuka (open source) dan Anda dapat menggunakannya secara bebas. Library matplotlib Sebagian besar ditulis dalam python, beberapa segemen ditulis dalam C, Objective-C, dan JavaScript untuk kompatibilitas platform.

Plotting biasa dikenal dengan visualisasi pada data numerik, contohnya visualisasi fungsi garis linear dan non-linear. Visualisasi data tentu bermanfaat untuk memudahkan pemahaman posisi maupun nilai dari data dalam sebuah model suatu bidang, entah itu 2-dimensi (kartesius), 3-dimensi, ataupun yang bersifat kategori seperti bar chart, line chart, box plot chart, violin chart, errorbar chart, scatter chart dan pie chart. Library Matplotlib sangat berguna sekali dalam menciptakan suatu image yang akan digunakan pada dokumentasi apapun itu.

Library Matplotlib pertama kali diciptakan oleh John D. Hunter dan sekarang telah dikelola serta di kembangkan oleh tim developer yang besar. Awalnya matplotlib hanya dirancang untuk menghasilkan plot grafik yang

sesuai pada publikasi jurnal saja atau artikel ilmiah. Matplotlib dapat digunakan dalam skrip code Python, Python dan IPython shell, server aplikasi web, dan juga beberapa toolkit graphical user interface (GUI) lainnya.

Visualisasi data dari matplotlib merupakan sebuah gambar grafik yang memiliki satu sumbu atau lebih. Setiap sumbunya memiliki sumbu horizontal (x) dan sumbu vertikal (y), dan data yang direpresentasikan menjadi warna dan glyphs seperti marker (contohnya seperti berbentuk lingkaran) atau lines (garis) atau poligon.

Berikut cara instalasi Matplotlib pada Jupyter Notebook. Hasil kemungkinan berbeda, disini saya sudah pernah menginstall matplotlib.

```
pip install matplotlib

Requirement already satisfied: matplotlib in c:\users\25and\anaconda3\lib\site-packages (3.3.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\25and\anaconda3\lib\site-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\users\25and\anaconda3\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\25and\anaconda3\lib\site-packages (from matplotlib) (8.0.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\25and\anaconda3\lib\site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: numpy>=1.15 in c:\users\25and\anaconda3\lib\site-packages (from matplotlib) (1.22.3)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\25and\anaconda3\lib\site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: certifi>=2020.06.20 in c:\users\25and\anaconda3\lib\site-packages (from matplotlib) (2020.6.20)
Requirement already satisfied: six in c:\users\25and\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

Gambar 1. 29 Install Matplotlib

```
pip install matplotlib
```

Library Scikit-Learn

Scikit-learn merupakan library untuk pembelajaran mesin (*Machine Learning*) sumber terbuka yang mendukung pembelajaran terawasi (*Supervised Learning*) dan tidak terawasi (*Unsupervised Learning*). Library ini juga menyediakan berbagai alat untuk pemasangan model, pra-pemrosesan data, pemilihan model, evaluasi model, dan banyak utilitas lainnya..

Scikit-learn menyediakan lusinan algoritma dan model pembelajaran mesin bawaan yang disebut estimator, serta scikit-learn menyediakan data yang dapat digunakan secara langsung. Setiap estimator dapat dipasang ke beberapa data menggunakan metode fit atau pelatihannya.

Sejak dimulainya proyek pada tahun 2010, scikit-learn telah menjadi toolkit pembelajaran mesin utama bagi programmer Python. Hanya dalam tujuh tahun, scikit-learn telah memiliki lebih dari 1.500 kontributor dari seluruh dunia. Ini termasuk submodul untuk model diantaranya.

- Classification: SVM, nearest neighbors, random forest, logistic regression, dan sebagainya.
- Regression: Lasso, ridge regression, dan sebagainya.
- Clustering: k-means, spectral clustering, dan sebagainya.
- Dimensionality reduction: PCA, feature selection, matrix factorization, dan sebagainya.
- Model selection: Grid search, metrics, cross-validation.
- Preprocessing: notmalixation, feature extraction.

```

pip install scikit-learn

Requirement already satisfied: scikit-learn in c:\users\25and\anaconda3\lib\site-packages (0.23.2)Note: you may need to restart
the kernel to use updated packages.
Requirement already satisfied: numpy>=1.13.3 in c:\users\25and\anaconda3\lib\site-packages (from scikit-learn) (1.22.3)
Requirement already satisfied: scipy>=0.19.1 in c:\users\25and\anaconda3\lib\site-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: joblib>=0.11 in c:\users\25and\anaconda3\lib\site-packages (from scikit-learn) (0.17.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\25and\anaconda3\lib\site-packages (from scikit-learn) (2.1.0)

```

Gambar 1. 30 Install Scikit-Learn

```

pip install scikit-learn

```

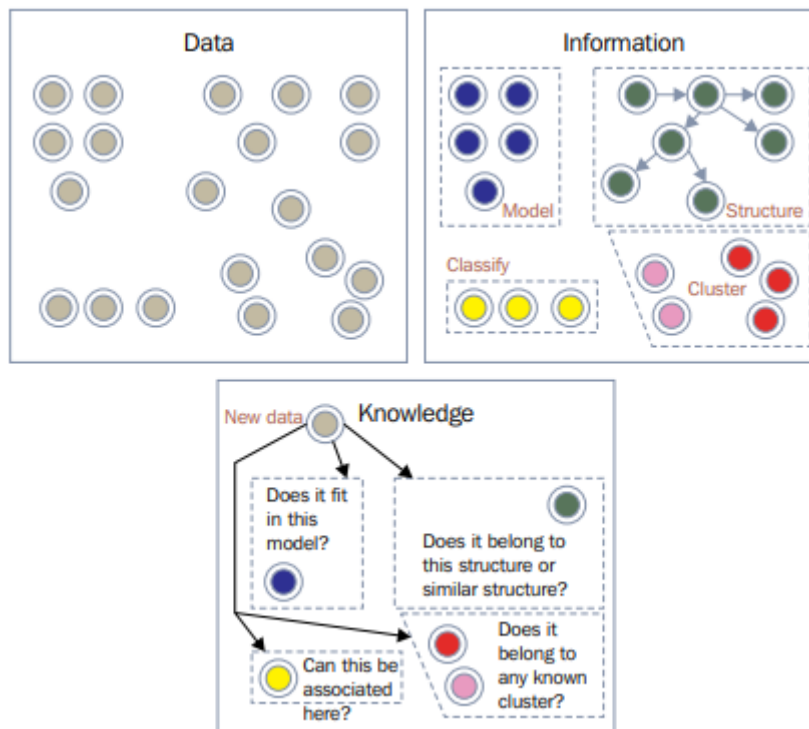
Analisis dan Visualisasi Data

2.1 Data, Analisis dan Visualisasi

Istilah data menyiratkan suatu premis dimana seseorang dapat menarik kesimpulan. Data sendiri ialah sekumpulan informasi atau fakta mentah yang belum diolah berupa angka, kata-kata, citra, dan lain sebagainya. Data sebenarnya mengacu pada fakta-fakta objektif yang diskrit dalam bentuk digital. Data ibarat blok bangunan dasar, Ketika data tersebut dapat diolah dan divisualisasikan maka akan memberi informasi yang berguna dalam menjawab beberapa pertanyaan tentang bisnis.

Data dapat menjadi sesuatu yang sangat sederhana, banyak, dan tidak terorganisir. Proses pengumpulan, penyimpanan, dan pendistribusian data sangat bervariasi menurut jenis data dan metode penyimpanannya. Data dapat berupa berbagai bentuk, seperti CSV, database, gambar, format dokumen, dan lain sebagainya. Informasi ialah data yang diproses untuk disajikan sebagai jawaban atas pertanyaan bisnis. Data akan menjadi informasi ketika kita menambahkan hubungan atau asosiasi. Asosiasi sendiri dilakukan dengan menyediakan konteks atau latar belakang data. Latar belakang data sangat membantu karena memungkinkan kita menjawab pertanyaan tentang data tersebut.

Pengetahuan akan muncul ketika kita menafsirkan dan mengatur informasi dalam menggunakannya untuk mendorong pengambilan keputusan. Pengetahuan ialah data, keterampilan, dan informasi yang diperoleh melalui pengalaman. Pengetahuan terdiri dari kemampuan untuk membuat suatu keputusan yang tepat serta keterampilan untuk melaksanakannya.



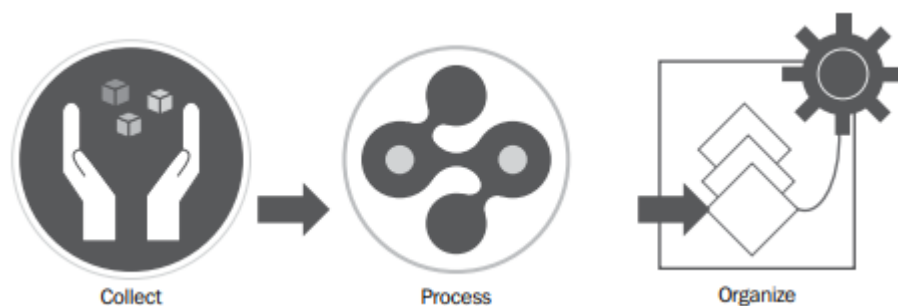
Gambar 2. 1 Data, Informasi, Pengetahuan

Analisis data merupakan proses pengolahan data yang bertujuan untuk menemukan suatu informasi yang berguna dan bisa menjadi dasar dalam pengambilan sebuah keputusan untuk solusi dari suatu permasalahan. Proses analisis ini merupakan kegiatan pengelompokan data berdasarkan karakteristiknya, yaitu proses pembersihan data, transformasi data, dan memodelkan data yang bertujuan untuk mendapatkan informasi penting dari data tersebut. Dan juga data yang telah di proses harus disajikan secara menarik dan mudah dipahami oleh orang lain, contohnya dalam bentuk plot maupun grafik. Saat ini segala aktivitas tidak jauh dengan teknologi. Termasuk dengan yang namanya data dimana akan terus bertambah setiap waktu. Apabila data terus menerus dibiarkan hingga menumpuk, maka data-data tersebut menjadi hal yang tidak berguna. Yang seharusnya sebuah data dapat diolah dan dimanfaatkan untuk mendapatkan informasi yang berguna. Sehingga, analisis data menjadi bagian penting dalam pengolahan data.

Tujuan dari analisis data adalah untuk mengekstrak informasi yang berguna dari data dan mengambil keputusan berdasarkan analisis data. Kapan pun kita mengambil keputusan dalam kehidupan kita sehari-hari adalah dengan memikirkan apa yang terjadi terakhir kali atau apa yang akan terjadi

dengan memilih keputusan itu. Ini tidak lain adalah menganalisis masa lalu atau masa depan kita dan membuat keputusan berdasarkan itu. Untuk itu, kita mengumpulkan kenangan masa lalu atau impian masa depan kita. Jadi itu tidak lain adalah analisis data. Sekarang hal yang sama dilakukan analisis untuk tujuan bisnis, disebut Analisis Data.

Sekarang kita sudah tau tentang apa itu data, sekarang apa tujuan dari pengumpulan data? Data berguna untuk menggambarkan fenomena fisik atau sosial dan untuk lebih menjawab pertanyaan tentang fenomena itu. Untuk alasan tersebut penting untuk memastikan bahwasanya data tidak salah, tidak akurat, ataupun tidak lengkap.



Gambar 2. 2 Tahapan Data

Data yang sudah terkumpul memerlukan beberapa pemrosesan dan pengorganisasian, dimana nantinya memiliki struktur, model, atau pola. Proses ini setidaknya memberi kita cara yang terorganisir untuk menemukan jawaban atas pertanyaan tentang data. Visualisasi data seperti seni, didorong oleh data dan belum dibuat oleh manusia dengan bantuan berbagai alat komunikasi. Misalnya seorang seniman melukis gambar menggunakan alat dan bahan seperti kuas dan warna. Demikian pula seniman lain mencoba membuat visualisasi data dengan bantuan alat komputasi.

2.2 Memilih Visual yang Efektif

Terdapat berbagai grafik yang berbeda dan jenis tampilan visual informasi, untuk memilih visual akan disesuaikan dengan kebutuhan.

91%

Simple text



Scatterplot

	A	B	C
Category 1	15%	22%	42%
Category 2	40%	36%	20%
Category 3	35%	17%	34%
Category 4	30%	29%	26%
Category 5	55%	30%	58%
Category 6	11%	25%	49%

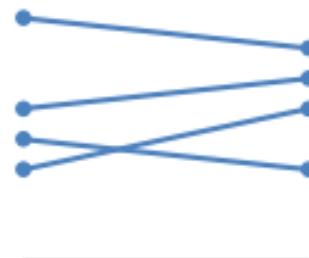
Table



Line

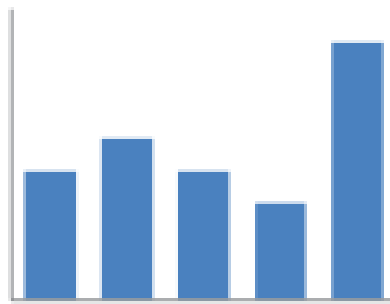
	A	B	C
Category 1	15%	22%	42%
Category 2	40%	36%	20%
Category 3	35%	17%	34%
Category 4	30%	29%	26%
Category 5	55%	30%	58%
Category 6	11%	25%	49%

Heatmap

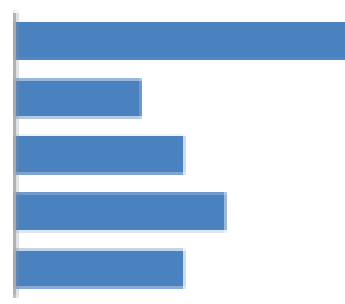


Slopegraph

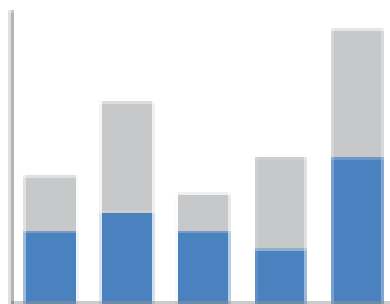
Gambar 2. 3 Grafik 1



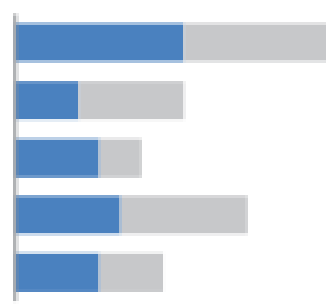
Vertical bar



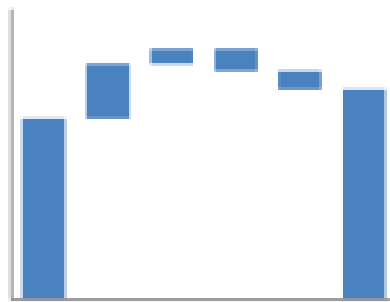
Horizontal bar



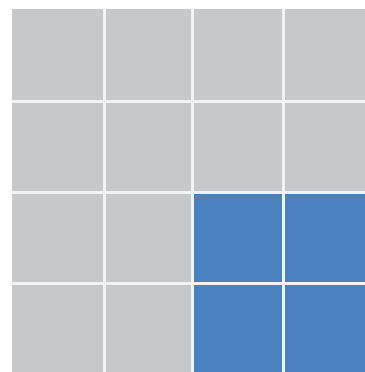
Stacked vertical bar



Stacked horizontal bar



Waterfall



Square area

Gambar 2. 4 Grafik 2

Simple text, ketika anda hanya memiliki satu atau dua nomor saja untuk diinformasikan, teks sederhana ini dapat menjadi cara yang bagus untuk berkomunikasi.

Tables, tampilan ini sangat bagus untuk berkomunikasi dengan audiens campuran dimana masing-masing anggotanya akan mencari baris minat khusus mereka. Jika Anda ingin mengomunikasikan beberapa unit ukuran yang berbeda, table juga biasanya lebih mudah daripada menggunakan grafik.

Heatmap, salah satu pendekatan untuk mencampur detail yang dapat disertakan dalam table serta memanfaatkan isyarat visualnya. Heatmap merupakan suatu cara untuk memvisualisasikan data dalam format table, dimana sebagai ganti angka atau tambahan, Anda juga dapat menyertakan sel berwarna yang menyampaikan besaran relative angka.

Scatterplot, digunakan untuk menunjukkan hubungan antara dua hal, karena kemungkinan Anda untuk mengkodekan data secara bersamaan pada sumbu x horizontal dan y vertical untuk melihat apakah ada hubungan. Scatterplot cenderung digunakan pada bidang ilmiah.

Line, sering digunakan untuk memplot data kontinu. Karena titik-titiknya terhubung secara fisik mealui garis, ini menyiratkan bahwasanya ada hubungan antara titik-titik yang mungkin tidak masuk akal untuk data yang diurutkan atau dibagi ke dalam kategori yang berbeda). Data kontinu seperti unit waktu: hari, bulan, kuartal, atau tahun.

Slopegraph, berguna ketika memiliki dua periode waktu atau titik perbandingan dan ingin dengan cepat menunjukkan kenaikan dan penurunan relative atau perbedaanya di berbagai kategori antara dua titik data.

Vertical bar, diagram ini dapat berupa rangkaian tunggal, dua rangkaian, atau beberapa rangkaian. Ketika Anda menambahkan lebih banyak rangkaian data, menjadi lebih sulit untuk fokus ke data yang disajikan dan mengambil informasi dari data, jadi gunakan beberapa diagram batang rangkaian dengan hati-hati. Pertimbangkan juga apa yang Anda ingin audiens Anda dapat bandingkan dan susun hierarki kategoris Anda untuk membuatnya mudah untuk dimengerti.

Stacked vertical bar, gunakan diagram ini lebih terbatas untuk memungkinkan Anda membandingkan total diseluruh kategori tertentu. Namun, ini juga dapat dengan cepat menjadi berlebihan secara visual, terutama mengingat skema warna default yang bervariasi di sebagian besar aplikasi grafik.

Waterfall, dapat digunakan untuk memisahkan potongan-potongan bagan batang bertumpuk untuk fokus pada satu per satu atau untuk menunjukkan titik awal kenaikan dan penurunan, serta titik akhir yang dihasilkan.

Horizontal bar, digunakan untuk data kategorikal karena lebih mudah dibaca. Bagan batang horizontal sangat berguna jika nama kategori memiliki kata yang cukup panjang, karena teks ditulis dari kiri ke kanan. Selain itu, informasinya diproses mulai dari atas dan membuat z dengan mata yang melintasi layer atau halaman, berarti saat kita mendapatkan data, kita sudah tahu apa yang diwakilinya.

Stacked horizontal bar, ini mirip dengan horizontal bar dimana dapat digunakan untuk menunjukkan total di berbagai kategori tetapi juga memberikan gambaran tentang bagian subkomponen. Bar ini dapat disusun untuk menunjukkan nilai absolut atau jumlah hingga 100%.

Square area, grafik ini sering dihindari karena mata manusia tidak melakukan pekerjaan yang baik untuk menghubungkan nilai kuantitatif ke ruang dua dimensi, yang dapat membuat grafik area lebih sulit dibaca daripada jenis tampilan visual lainnya. Grafik ini dapat digunakan ketika saya perlu memvisualisasikan sejumlah besaran yang sangat berbeda.

2.3 Langkah-Langkah Analisis Data

Pada umumnya terdapat lima langkah utama yang terlibat dalam proses analisis data, yang meliputi:

1. Pengumpulan Data

Langkah pertama dalam analisis data adalah mengumpulkan data yang akan dianalisis, data dapat berasal dari berbagai sumber. Data dapat berasal dari database yang berbeda, server web, file log, media sosial, file excel, CSV, dan sebagainya.

2. Persiapan Data

Langkah selanjutnya dalam proses ini adalah menyiapkan data. Ini melibatkan pembersihan data untuk menghapus nilai yang tidak diinginkan dan berlebihan, mengubahnya menjadi format yang tepat, dan membuatnya siap untuk dianalisis. Ini juga membutuhkan perselisihan data.

3. Eksplorasi Data

Setelah data siap, eksplorasi data dilakukan dengan menggunakan berbagai teknik visualisasi data untuk menemukan tren yang tidak terlihat dari data.

4. Pemodelan Data

Langkah selanjutnya adalah membangun model menggunakan algoritma pembelajaran mesin untuk membuat pemodelan pada data yang akan digunakan.

5. Interpretasi Hasil

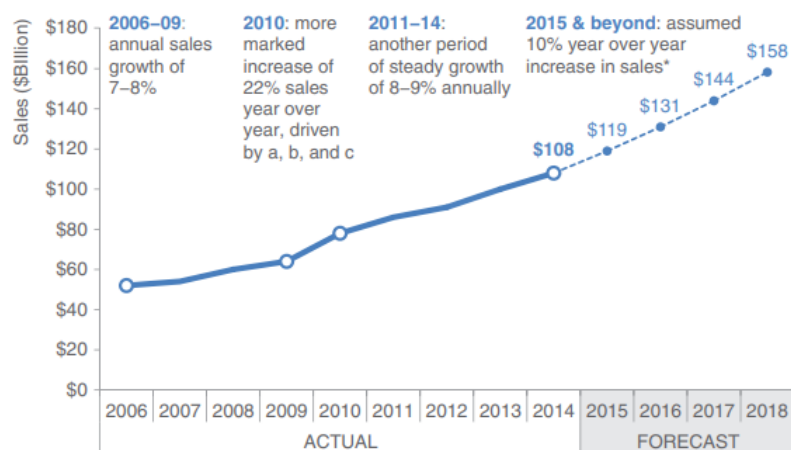
Langkah terakhir dalam proses analitik data apa pun adalah mendapatkan hasil yang berarti dan memeriksa apakah hasilnya sesuai dengan hasil yang Anda harapkan.

2.4 Membedah Model Visual

Sebelumnya kami telah membahas beberapa visual yang dapat Anda terapkan untuk meningkatkan kemampuan Anda dalam berkomunikasi dengan data. Sekarang setelah anda memahami dasar-dasar apa yang membuat visual dapat menjadi efektif, mari pertimbangkan beberapa contoh tambahan tentang apa yang membuat visualisasi data menjadi “baik”.

Setiap visual yang dibahas diciptakan untuk memenuhi kebutuhan pada situasi tertentu. Disini akan membahas scenario yang relevan secara singkat. Pikirkan kita meluangkan waktu untuk melihat dan memikirkan setiap visual model. Pertimbangkan juga tantangan visualisasi data apa yang dihadapi dengan pendekatan yang diberikan agar dapat bermanfaat.

- Line Graph dengan model peramalan

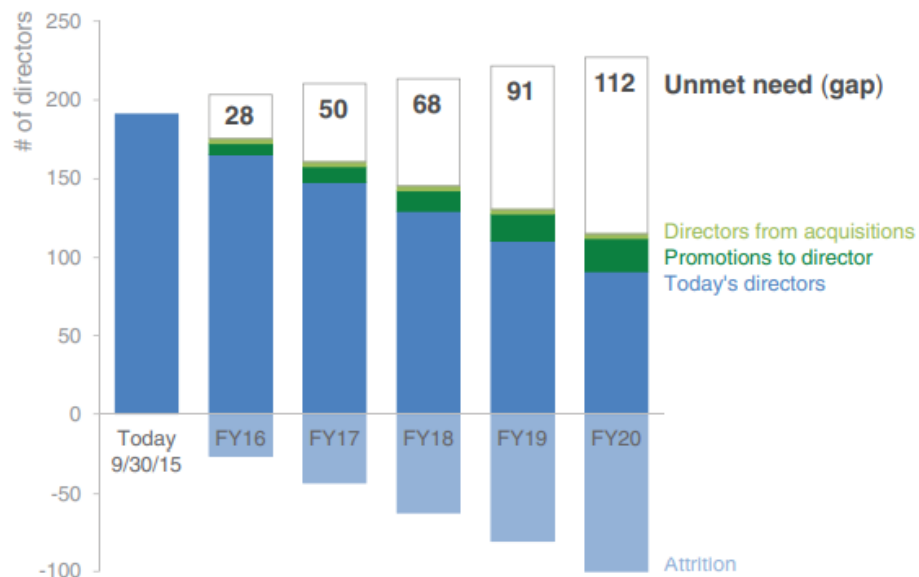


Gambar 2. 5 Peramalan Geafik Garis

Kita dapat memanfaatkan isyarat visual untuk membedakan antara data perkiraan dan aktual, mempermudah interpretasi informasi. Pada gambar diketahui bahwasanya garis padat mewakili data aktual dan garis putus-putus yang lebih tipis mewakili data perkiraan. Pelabelan yang jelas dari Aktual dan Prakiraan di bawah sumbu x membantu memperkuat ini, dengan bagian prakiraan dipisahkan secara visual sedikit melalui bayangan latar belakang yang terang.

Dalam visual ini, dikategorikan melalui penggunaan font dan elemen abu-abu kecuali judul grafik, tanggal di dalam kotak teks, data (baris), penanda data pilihan, dan label data numerik mulai tahun 2014 dan seterusnya. Ketika kita melihat visualnya, mungkin kebanyakan mata kita pertama-tama tertuju ke judul grafik di kiri atas (karena posisi dan teks abu-abu gelap yang lebih besar yang dibahas dalam contoh sebelumnya), kemudian ke tanggal biru di kotak teks, pada titik mana dapat berhenti sejenak dan membaca untuk sedikit konteks sebelum menggerakkan mata ke bawah untuk melihat titik atau tren yang sesuai dalam data.

- Stacked Bars positif dan negatif



A footnote explaining relevant forecast assumptions and methodology would go here.

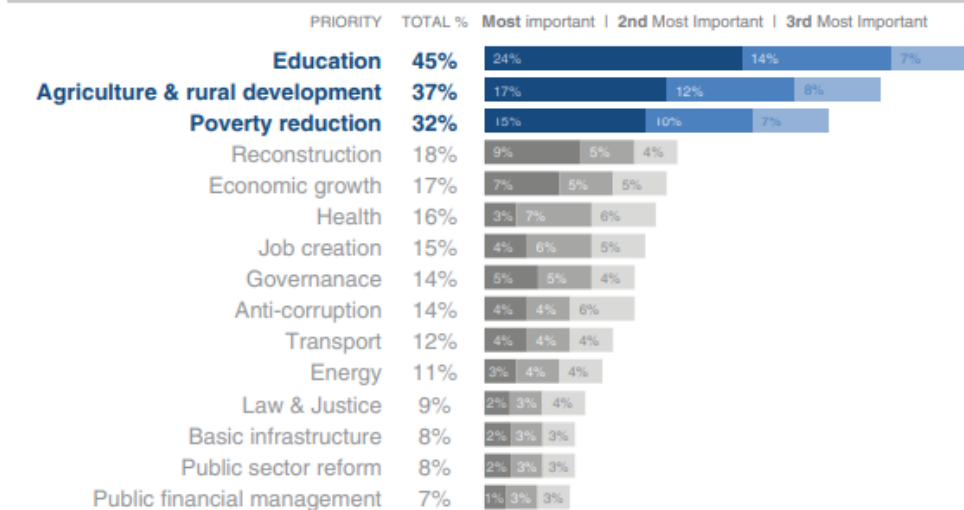
Gambar 2. 6 Bar Bertumpuk Vertikal

Gambar 2.6 menunjukkan contoh dari ruang analisis. Dapat diketahui bahwasanya akan ada peningkatan kebutuhan yang tidak terpenuhi dengan asumsi untuk penambahan yang diharapkan ke kumpulan direktur dari waktu ke waktu melalui akuisisi, promosi, dan penurunan ke kumpulan dari waktu ke waktu. Pilihan yang sengaja dibuat dalam hal penggunaan warna di seluruh visual ini yang mana ditampilkan dengan warna biru. Seiring waktu, akan melihat lebih sedikit warna biru yang jatuh di atas sumbu dan proporsi yang meningkat jatuh di bawah sumbu karena semakin banyak yang tertarik.

Semua teks dalam visual memiliki ukuran yang sama kecuali di mana keputusan dibuat untuk lebih menekankan ataupun mengurangi komponen. Judul dituliskan lebih besar untuk memudahkan pembacaan teks yang ditampilkan. Kita dapat tau bahwasanay catatan kaki ditulis dalam teks yang lebih kecil, sehingga sesuai kebutuhan tetapi tidak menarik perhatian.

- Horizontal Stacked Bars

Top 15 development priorities, according to survey



N = 4,392. Based on responses to item, When considering development priorities, which one development priority is the most important? Which one is the second most important priority? Which one is the third most important priority? Respondents chose from a list. Top 15 shown.

Gambar 2. 7 Bar Bertumpuk Horizontal

Dari grafik tersebut menunjukkan hasil pertanyaan terkait survey tentang prioritas yang relative di negara berkembang. Ini merupakan informasi yang baik, tetapi karena penekanan strategis dan pengurangan penekanan pada komponen, tidak menjadi berlebihan secara visual.

Dalam penggunaan batang tumpuk sudah sesuai mengingat sifat dari apa yang sedang digambarkan: prioritas utama, hingga bukan prioritas utama. Kategori diatur secara vertikal dalam urutan "% Total" yang akan memberikan audiens pemahaman yang jelas untuk digunakan saat mereka menafsirkan data tersebut.

2.5 Jenis Analisis Data

Saat melakukan penelitian, terdapat beberapa jenis analisis data yaitu analisis kualitatif dan analisis kuantitatif.

a.) Analisis Kuantitatif

Analisis kuantitatif adalah analisis yang menggunakan model matematika atau statistika dalam memproses datanya. Hasil analisis biasanya berupa angka-angka yang akan disajikan dan diuraikan oleh peneliti. Adapun teknik yang digunakan dalam analisis kuantitatif yaitu teknik analisis deskriptif dan teknik analisis inferensial yang memiliki fungsinya masing-masing.

1. Statistik Deskriptif

Analisis data deskriptif pada penelitian kuantitatif ialah analisis data dengan cara menggambarkan atau mendeskripsikan data-data yang ditemukan secara apa adanya. Deskripsi pada penelitian kuantitatif ialah menggambarkan data-data yang berupa angka-angka dengan deskripsi berdasarkan data tersebut secara jelas. Contoh penelitian mengenai analisis deskriptif kuantitatif ialah perhitungan data atau jumlah profesi, dan sebagainya.

2. Statistik Inferensial

Salah satu tugas statistik inferensial ialah menarik simpulan mengenai suatu variabel yang diteliti berdasarkan data yang diperoleh untuk digeneralisasikan pada populasi. Generalisasi pada penelitian kuantitatif ialah suatu cara pengambilan simpulan terhadap kelompok individu yang lebih luas 16 jumlahnya berdasarkan data yang diperoleh dari sekelompok individu yang sedikit jumlahnya.

Pada statistik inferensial, bertujuan untuk menentukan sejauh mana data data penelitian tersebut mewakili atau merepresentasikan populasi.

Statistik inferensial tidak dapat dilakukan dengan cara menggunakan metode dan teknik yang sama pada data yang berbeda. Data nominal menggunakan analisis kategori, data ordinal menggunakan non-parametrik, dan data interval & rasio, menggunakan parametrik.

b.) Analisis Kualitatif

Analisis kualitatif adalah analisis secara sistematis yang tidak menggunakan model matematika atau statistika. Dengan kata lain analisis ini dilakukan dengan membaca tabel, grafik, atau data lainnya yang sudah tersedia yang diperoleh dari berbagai sumber dengan teknik pengumpulan data tertentu. Tujuan analisis kualitatif adalah untuk menemukan makna dari data-data tersebut. Pada teknik analisis data kualitatif menganalisis atau membahas mengenai konsep-konsep suatu permasalahan dan tidak disertai data-data berupa angka-angka. Teknik analisis data pada penelitian kualitatif ada 3, yaitu analisis konten, analisis wacana, dan analisis naratif.

1. Analisis Konten

Analisis konten berasal dari komunikasi penelitian dan berpotensi menjadi salah satu yang paling penting menjadi teknik penelitian dalam ilmu sosial. Analisis konten berusaha untuk menganalisis data-data dalam konteks tertentu, berkaitan dengan individu-kelompok atau atribut-budaya mereka. Pada analisis konten, data biasanya dihasilkan atau didapatkan oleh pengamat yang merekam atau mentranskripsikan menjadi materi tekstual, bisa berupa gambar atau suara yang sesuai untuk analisis

2. Analisis Wacana

Teknik analisis wacana pada penelitian kualitatif bertujuan untuk menganalisis wacana-wacana atau komunikasi antarorang dalam suatu konteks 17 sosial tertentu. Bidang yang dikaji pada analisis wacana yaitu berupa pidato, tulisan, bahasa, percakapan (baik verbal dan nonverbal), dan sebagainya.

3. Analisis Narasi

Teknik analisis data naratif pada penelitian kualitatif bertujuan untuk menganalisis atau meneliti mengenai kumpulan deskripsi suatu peristiwa atau fenomena yang terjadi, kemudian menyajikannya dengan

bentuk narasi atau cerita. Contoh analisis naratif ini ialah mengenai kajian biografi.

2.6 Kebutuhan Eksplorasi Analisis Data

Eksplorasi Analisis Data adalah langkah penting sebelum beralih ke machine learning atau pemodelan data. Dengan melakukan ini, kita dapat mengetahui apakah fitur yang dipilih cukup baik untuk dimodelkan, apakah semua fitur yang diperlukan, apakah ada korelasi berdasarkan mana kita dapat kembali ke langkah pra-pemrosesan data atau beralih ke pemodelan.

Setelah analisis data eksplorasi selesai dan wawasan diperoleh, fiturnya dapat digunakan untuk pemodelan machine learning yang diawasi dan tidak. Dalam setiap alur kerja machine learning, langkah terakhir adalah melaporkan atau memberikan wawasan kepada Pemangku Kepentingan dan sebagai Ilmuwan Data, kita dapat menjelaskan setiap bagian kode tetapi kita harus mengingat audiens. Dengan menyelesaikan Analisis Data Eksplorasi, kita akan memiliki 105 banyak plot, peta panas, distribusi frekuensi, grafik, matriks korelasi bersama dengan hipotesis di mana setiap individu dapat memahami tentang data kita dan wawasan apa yang didapatkan dari menjelajahi kumpulan data.

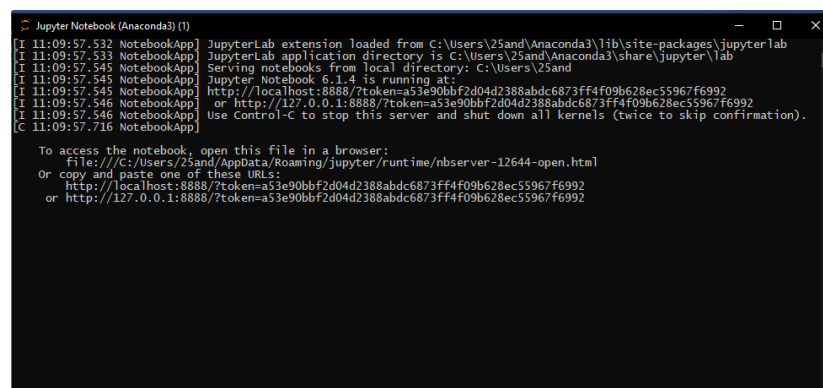
Ada pepatah yang mengatakan "Sebuah gambar bernilai seribu kata ". Saya ingin memodifikasinya untuk ilmuwan data sebagai " Plot bernilai seribu baris ". Dalam Contoh Perjalanan kami, kami melakukan semua penjelajahan tempat yang dipilih berdasarkan mana kami akan mendapatkan kepercayaan diri untuk merencanakan perjalanan dan bahkan berbagi dengan teman-teman kami wawasan yang kami dapatkan tentang tempat itu sehingga mereka juga dapat bergabung.

Konsep Dasar Python

3.1 Import Library

Sebelum kita mengimport library python, pertama-tama kita buka terlebih dahulu jupyter notebooknya. Contoh langkah-langkah nya sebagai berikut:

- Buka CMD lalu ketikan “jupyter notebook” setelah itu enter. Tunggu hingga proses membuka jupyter notebook selesai.



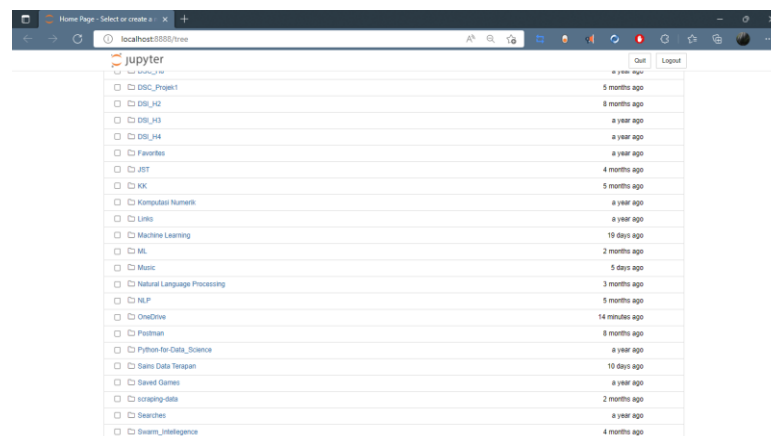
```

Jupyter Notebook (Anaconda3) (1)
[I 11:09:57.532 NotebookApp] JupyterLab extension loaded from C:\Users\25and\Anaconda3\lib\site-packages\jupyterlab
[I 11:09:57.533 NotebookApp] JupyterLab application directory is C:\Users\25and\Anaconda3\share\jupyter\lab
[I 11:09:57.545 NotebookApp] Serving notebooks from local directory: C:\Users\25and
[I 11:09:57.545 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 11:09:57.545 NotebookApp] http://localhost:8888/?token=a53e90bbf2d04d2388abdc6873ff4f09b628ec55967f6992
[I 11:09:57.546 NotebookApp] or http://127.0.0.1:8888/?token=a53e90bbf2d04d2388abdc6873ff4f09b628ec55967f6992
[I 11:09:57.546 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 11:09:57.716 NotebookApp]

To access the notebook, open this file in a browser:
file:///C:/Users/25and/AppData/Roaming/jupyter/runtime/nbserver-12644-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=a53e90bbf2d04d2388abdc6873ff4f09b628ec55967f6992
or http://127.0.0.1:8888/?token=a53e90bbf2d04d2388abdc6873ff4f09b628ec55967f6992
  
```

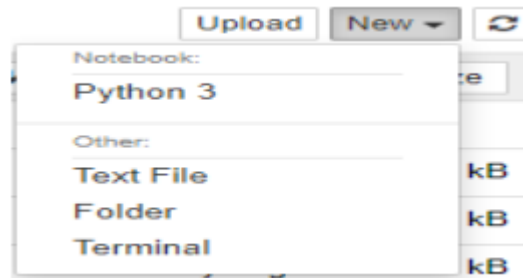
Gambar 3. 1 Proses Membuka Jupyter Notebook

- Jika jupyter notebook berhasil di buka maka tampilannya bisa dilihat pada gambar berikut :



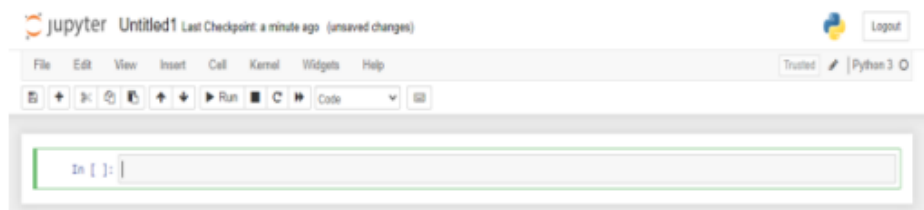
Gambar 3. 2 Tampilan Jupyter Notebook

- Setelah jupyter notebook terbuka, pilih “new” lalu pilih Python 3.



Gambar 3. 3 Pilih Python 3

- Setelah itu kita sudah bisa mulai mengimport library pandasnya di jupyter notebook dan python.



Gambar 3. 4 Berhasil Membuka Python 3 di Jupyter Notebook

Selanjutnya kita melakukan import library python, contohnya library pandas sebagai pd untuk mempermudah penulisan, jadi cukup dengan menuliskan pd.function sebagai alias. Pada baris selanjutnya kita menambahkan library NumPy dan seterusnya yang kita butuhkan. Contoh codenya sebagai berikut ini.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

3.2 Membaca Format File

Di Library python memungkinkan kita dapat membaca dan menganalisis file dari berbagai jenis format file data seperti CSV, JSON, XLSX, dan HTML.

1) CSV

Untuk membaca atau mengimport sebuah file CSV kita ke dalam Jupyter Notebook yaitu dengan cara menggunakan fungsi `read_csv()`. Pada code program di bawah ini saya menggunakan contoh nama file csv nya yaitu "data".

```
pd.read_csv("data.csv")
```

2) JSON

Untuk membaca atau mengimport sebuah file JSON kita ke dalam Jupyter notebook yaitu dengan cara menggunakan fungsi `read_json()`. Pada code program di bawah ini saya menggunakan contoh nama file JSON nya yaitu "data".

```
pd.read_json("data.json")
```

3) XLSX

Untuk membaca atau mengimport sebuah file Excel atau XLSX kita ke dalam Jupyter notebook yaitu dengan cara menggunakan fungsi `read_excel()`. Pada code program di bawah ini saya menggunakan contoh nama file Excel nya yaitu "data".

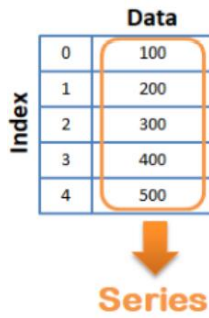
```
pd.read_excel("data.xlsx")
```

4) HTML

Untuk membaca atau mengimport sebuah file html kita ke dalam Jupyter notebook yaitu dengan cara menggunakan fungsi `read_html()`. Pada code program di bawah ini saya menggunakan contoh nama file html nya yaitu "data".

```
pd.read_html("data.html")
```

3.3 Series Objek Pandas



Gambar 3. 5 Series Pandas

Series merupakan struktur data dasar dalam Pandas, Series bisa juga diibaratkan sebagai array satu dimensi yang mampu menampung semua tipe data (bilangan bulat, string, bilangan float, Objek Python, dll.). Label sumbu secara kolektif disebut sebagai indeks.

Metode dasar untuk membuat Series adalah memanggil, contoh code programnya seperti berikut:

```
s = pd.Series(data, index=index)
```

Kita juga bisa membuat series dengan mendenifikasikan:

```
import pandas as pd
series = pd.Series()
print(series)
```

Pada code di atas, saya membuat Series kosong tanpa di isi data.

Series Pandas merupakan array satu dimensi dari data yang diindeks. Itu dapat dibuat dari list atau array seperti program berikut:

```
import pandas as pd
import numpy as np
data = pd.Series ([0.25, 0.5, 0.75, 1.0 ])
data
```

Hasil output program di atas bisa dilihat pada gambar berikut:

```
0    0.25
1    0.50
2    0.75
3    1.00
dtype: float64
```

Gambar 3. 6 Output code program array

Seperti yang kita lihat di output sebelumnya, series ini membungkus urutan nilai dan urutan indeks, yang dapat kita akses dengan nilai dan atribut indeks. Nilainya (value) hanyalah sebuah array NumPy:

```
data.values
```

Hasil output program di atas bisa dilihat pada gambar berikut:

```
array([0.25, 0.5 , 0.75, 1.  ])
```

Gambar 3. 7 Menampilkan value array

Index adalah objek mirip array dari tipe `pd.Index`, yang akan kita bahas secara lebih rinci saat ini:

```
data.index
```

Hasil output program di atas bisa dilihat pada gambar berikut:

```
RangeIndex(start=0, stop=4, step=1)
```

Gambar 3. 8 Menampilkan Index

Seperti halnya array NumPy, data dapat diakses oleh indeks terkait, melalui notasi square-bracket Python:

```
data[1]  
data[1:3]
```

Hasil output program di atas bisa dilihat pada gambar berikut:

```
data[1]  
0.5  
  
data[1:3]  
1    0.50  
2    0.75  
dtype: float64
```

Gambar 3. 9 Mengakses value dengan index

Seperti yang kita lihat, Pandas Series jauh lebih umum dan fleksibel daripada NumPy array satu dimensi yang ditiru.

Contoh lain penggunaan series:

```
import pandas as pd
import numpy as np
data = pd.Series(['a','b','c','d'])
np1 = np.array(['a','b','c','d'])
data_np = pd.Series(np1)
print(data)
print(data_np)
```

Hasil output program di atas bisa dilihat pada gambar berikut:

```
0    a
1    b
2    c
3    d
dtype: object
0    a
1    b
2    c
3    d
dtype: object
```

Gambar 3. 10 Hasil output code program penggunaan series

Pada code program serta gambar di atas kita tidak mendefinisikan indeksinya, secara default indeks data akan diberikan seperti halnya indeks array yang bermula dari 0-N dengan RangeIndex(start, stop, step).

```
import pandas as pd
import numpy as np
data = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])
data
```

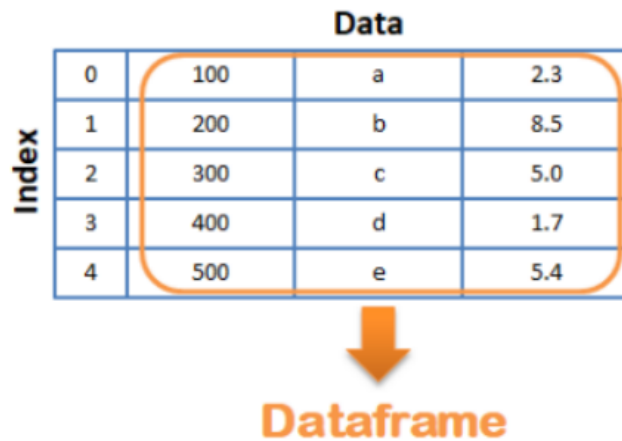
Hasil output program di atas bisa dilihat pada gambar berikut:

```
a    0.214167
b   -0.141125
c   -0.428143
d    0.911005
e    1.530669
dtype: float64
```

Gambar 3. 11 Menampilkan nilai secara random

Pada code program serta gambar di atas kita menggunakan series untuk mendapatkan nilai secara random dengan index yang sudah di tentukan.

3.4 DataFrame



Gambar 3. 12 Dataframe Pandas

Seperti objek Series yang dibahas di bagian sebelumnya, DataFrame dapat dianggap sebagai generalisasi dari array NumPy, atau sebagai spesialisasi dictionary Python. Baik sekarang lihatlah masing-masing dari perspektif ini.

DataFrame sebagai array NumPy

Jika Series adalah analog dari array satu dimensi dengan indeks fleksibel, DataFrame adalah analog dari array dua dimensi dengan indeks baris fleksibel dan nama kolom fleksibel. Sama seperti Anda mungkin menganggap array dua dimensi sebagai suatu urutan kolom satu dimensi selaras, Anda bisa menganggap DataFrame sebagai urutan objek Series selaras. Di sini, dengan “disejajarkan” kami maksudkan bahwa mereka berbagi indeks yang sama.

Untuk mendemonstrasikan ini, pertama mari kita membangun Series baru yang mencantumkan area masing-masing lima kota dan populasinya :

```
import pandas as pd
import numpy as np
area_dict = {'Surabaya': 350,
             'Malang': 145,
             'Pasuruan': 1474,
             'Gresik': 1191,
             'Bangkalan': 1001}
area = pd.Series(area_dict)
area
```

Hasil output program di atas bisa dilihat pada gambar berikut:

```

Surabaya    350
Malang       145
Pasuruan    1474
Gresik      1191
Bangkalan   1001
dtype: int64

```

Gambar 3. 13 Membuat series baru area

Kita sudah buat series baru area, dan selanjutnya membuat series baru populasi:

```

populasi_dict = {'Surabaya': 2874314,
                  'Malang': 875771,
                  'Pasuruan': 1876881,
                  'Gresik': 1326420,
                  'Bangkalan': 994212}

populasi = pd.Series(populasi_dict)
populasi

```

Hasil output program di atas bisa dilihat pada gambar berikut:

```

Surabaya    2874314
Malang       875771
Pasuruan    1876881
Gresik      1326420
Bangkalan   994212
dtype: int64

```

Gambar 3. 14 Membuat series baru populasi

Sekarang kita memiliki Series area bersama dengan Series populasi, kita dapat menggunakan DataFrame untuk membangun objek dua dimensi tunggal yang berisi informasi ini:

```

city = pd.DataFrame({'population': populasi, 'area': area})
city

```

Hasil output program di atas bisa dilihat pada gambar berikut:

	population	area
Surabaya	2874314	350
Malang	875771	145
Pasuruan	1876881	1474
Gresik	1326420	1191
Bangkalan	994212	1001

Gambar 3. 15 Membangun objek dua dimensi tunggal

Seperti objek Series, DataFrame memiliki atribut indeks yang memberikan akses ke label indeks:

```
city.index
```

Hasil output program di atas bisa dilihat pada gambar berikut:

```
Index(['Surabaya', 'Malang', 'Pasuruan', 'Gresik', 'Bangkalan'], dtype='object')
```

Gambar 3. 16 Menampilkan kota sesuai index

Selain itu, DataFrame memiliki atribut kolom, yang merupakan objek Indeks label kolom:

```
city.columns
```

Hasil output program di atas bisa dilihat pada gambar berikut:

```
Index(['population', 'area'], dtype='object')
```

Gambar 3. 17 Menampilkan kolom dataframe

Dengan demikian DataFrame dapat dianggap sebagai generalisasi dua dimensi Array NumPy, di mana baris dan kolom memiliki indeks umum untuk mengakses data.

Menambah Objek DataFrame

DataFrame dapat dibangun dengan berbagai macam cara. Disini kita beri beberap contoh:

- Dari objek Series tunggal

DataFrame adalah kumpulan dari objek Series, dan kolom tunggal DataFrame dapat dibangun dari Series tunggal:

```
pd.DataFrame (populasi ,columns =['population'])
```

population	
Surabaya	2874314
Malang	875771
Pasuruan	1876881
Gresik	1326420
Bangkalan	994212

Gambar 3. 18 Menambah objek Series tunggal

- List dictionary

List dictionary dapat dibuat menjadi DataFrame. Kita akan menggunakan pemahaman list sederhana untuk membuat beberapa data:

```
data = [{'a':i, 'b':2*i}for i in range (3)]  
pd.DataFrame(data)
```

	a	b
0	0	0
1	1	2
2	2	4

Gambar 3. 19 Menambah List dictionary menjadi DataFrame

Bahkan jika beberapa keys pada dictionary hilang, Pandas akan mengisinya dengan values NaN(yaitu “note a number”):

```
pd.DataFrame ([{'a':1, 'b':2}, {'b':3, 'c':4 }])
```

	a	b	c
0	1.0	2	NaN
1	NaN	3	4.0

Gambar 3. 20 Values NaN

- Dari dictionary objek Series

Seperti yang kita lihat sebelumnya, DataFrame dapat dibangun dari dictionary objek Series juga:

```
pd.DataFrame ({'population':populasi , 'area':area})
```

	population	area
Surabaya	2874314	350
Malang	875771	145
Pasuruan	1876881	1474
Gresik	1326420	1191
Bangkalan	994212	1001

Gambar 3. 21 Dictionary objek Series

- Dari array NumPy dua dimensi

Diberikan data array dua dimensi, kita bisa membuat DataFrame dengan kolom dan nama indeks yang ditentukan. Jika dihilangkan, integer akan digunakan untuk masing-masing:

```
pd.DataFrame(np.random.rand(3,2), columns=['foo', 'bar'], index=['a', 'b', 'c'])
```

	foo	bar
a	0.760405	0.988475
b	0.378031	0.819854
c	0.498128	0.490997

Gambar 3. 22 Menambah Dataframe

3.5 Tipe Data

Tipe data atau dtypes adalah hasil dari gabungan arsitektur pandas dengan numpy, dtype dari sebuah kolom tidak harus berkorelasi dengan tipe python dari objek yang terdapat dalam kolom tersebut.

Memeriksa jenis kolom

Jenis kolom dapat diperiksa dengan atribut .dtypes dari DataFrames.

```
import pandas as pd
df = pd.DataFrame({'A': [1, 2, 3], 'B': [1.0, 2.0, 3.0], 'C': [True, False, True]})
df
df.dtypes
```

	A	B	C
0	1	1.0	True
1	2	2.0	False
2	3	3.0	True

```
df.dtypes
A      int64
B     float64
C        bool
dtype: object
```

Gambar 3. 23 dtypes

Untuk satu seri, kita dapat menggunakan atribut “dtype”.

```
df['A'].dtype
```

Mengubah tipe data

Metode `astype()` mengubah tipe Seri dan mengembalikan sebuah Seri baru.

```
df = pd.DataFrame({'A': [1, 2, 3],
                    'B': [1.0, 2.0, 3.0],
                    'C': ['1.1.2010', '2.1.2011', '3.1.2011'],
                    'D': ['1 days', '2 days', '3 days'],
                    'E': ['1', '2', '3']
})
```

	A	B	C	D	E
0	1	1.0	1.1.2010	1 days	1
1	2	2.0	2.1.2011	2 days	2
2	3	3.0	3.1.2011	3 days	3

```
df.dtypes
A      int64
B    float64
C      object
D      object
E      object
dtype: object
```

Gambar 3. 24 Mengubah tipe seri

Ubah tipe kolom A menjadi float, dan tipe kolom B menjadi integer:

```
df['A'].astype('float')
df['B'].astype('int')
```

```
df['A'].astype('float')
0      1.0
1      2.0
2      3.0
Name: A, dtype: float64
```

```
df['B'].astype('int')
0      1
1      2
2      3
Name: B, dtype: int32
```

Gambar 3. 25 Ubah tipe data kolom

Mengubah tipe menjadi numerik

`pd.to_numeric` mengubah nilai menjadi tipe numerik.

```
pd.to_numeric(df['E'])
```

```
pd.to_numeric(df['E'])
0    1
1    2
2    3
Name: E, dtype: int64
```

Gambar 3. 26 Mengubah nilai menjadi tipe numerik

Secara default, `pd.to_numeric` menimbulkan error jika input tidak dapat diubah menjadi angka. Kita dapat mengubah masalah itu dengan menggunakan parameter `error`.

```
pd.to_numeric(pd.Series(['1', '2', 'a']), errors='ignore')
pd.to_numeric(pd.Series(['1', '2', 'a']), errors='coerce')
```

```
pd.to_numeric(pd.Series(['1', '2', 'a']), errors='ignore')
0    1
1    2
2    a
dtype: object

pd.to_numeric(pd.Series(['1', '2', 'a']), errors='coerce')
0    1.0
1    2.0
2    NaN
dtype: float64
```

Gambar 3. 27 Penggunaan parameter error

3.6 Pengoperasian Data

Salah satu bagian penting dari NumPy adalah kemampuan untuk melakukan operasi elemen cepat, baik dengan aritmatika dasar 83 (penambahan, pengurangan, perkalian, dll.) dan dengan operasi yang lebih canggih (fungsi trigonometrik, fungsi eksponensial dan logaritma, dll.).

Pandas mencakup beberapa siklus yang berguna, namun: untuk operasi unary seperti fungsi negasi dan trigonometrik, `ufunc` ini akan mempertahankan label indeks dan kolom pada output, dan untuk operasi biner seperti penambahan dan perkalian, Pandas akan secara otomatis menyelaraskan indeks ketika

meneruskan objek ke ufunc. Ini berarti bahwa menjaga konteks data dan menggabungkan data dari sumber yang berbeda - baik tugas yang berpotensi rawan kesalahan dengan array NumPy mentah - pada dasarnya menjadi sangat mudah dengan Pandas. Kami juga akan melihat bahwa ada operasi yang terdefinisi dengan baik antara struktur Series satu dimensi dan struktur DataFrame dua dimensi.

UFuncs: Pemeliharaan Indeks

Karena Pandas dirancang untuk bekerja dengan NumPy, setiap ufunc NumPy akan bekerja pada Pandas Series dan objek DataFrame. Mari kita mulai dengan mendefinisikan Series dan DataFrame sederhana untuk mendemonstrasikan ini:

```
import pandas as pd
import numpy as np
rng = np.random.RandomState(42)
ser = pd.Series(rng.randint(0, 10, 4))
ser

df = pd.DataFrame(rng.randint(0, 10, (3,4)),
                  columns=['A', 'B', 'C', 'D'])
df
```

```
import pandas as pd
import numpy as np

rng = np.random.RandomState(42)
ser = pd.Series(rng.randint(0, 10, 4))
ser
0    6
1    3
2    7
3    4
dtype: int32

df = pd.DataFrame(rng.randint(0, 10, (3,4)),
                  columns=['A', 'B', 'C', 'D'])
df
```

	A	B	C	D
0	6	9	2	6
1	7	4	3	7
2	7	2	5	4

Gambar 3. 28 Membuat Series dan DataFrame dengan nilai random

Jika kita menerapkan ufunc NumPy pada salah satu objek ini, hasilnya akan menjadi objek Pandas lainnya dengan indeks dipertahankan:

```
np.exp(ser)
```

```
np.exp(ser)
0      403.428793
1       20.085537
2     1096.633158
3       54.598150
dtype: float64
```

Gambar 3. 29 Menerapkan ufunc NumPy di Series

Atau, untuk perhitungan yang sedikit lebih rumit:

```
np.sin(df * np.pi / 4)
```

```
np.sin(df * np.pi / 4)
```

	A	B	C	D
0	-1.000000	7.071068e-01	1.000000	-1.000000e+00
1	-0.707107	1.224647e-16	0.707107	-7.071068e-01
2	-0.707107	1.000000e+00	-0.707107	1.224647e-16

Gambar 3. 30 Menerapkan ufunc NumPy di Dataframe

Ufuncs:Penjajaran Indeks

Untuk operasi biner pada dua objek Series atau DataFrame, Pandas akan menyelaraskan indeks dalam proses melakukan operasi. Ini sangat nyaman ketika Anda bekerja dengan data yang tidak lengkap, seperti yang akan kita lihat dalam beberapa contoh berikut.

Penyelarasan indeks dalam Series

Sebagai contoh, misalkan kita menggabungkan dua sumber data yang berbeda, dan hanya menemukan tiga kota bagian Jawa Timur teratas menurut wilayah dan tiga kota bagian Jawa Timur teratas berdasarkan populasi:

```
area = pd.Series({'Pasuruan': 1474, 'Gresik': 1191, 'Bangkalan': 1001},
                  name='area')
population = pd.Series({'Surabaya': 2874314, 'Pasuruan': 1876881,
```

```
'Gresik': 1326420}, name='population'))  
  
population / area
```

```
population / area  
Bangkalan      NaN  
Gresik         1113.702771  
Pasuruan       1273.324966  
Surabaya       NaN  
dtype: float64
```

Gambar 3. 31 Menghitung kepadatan populasi

Array yang dihasilkan berisi gabungan indeks dari dua array input, yang dapat kita tentukan menggunakan aritmatika himpunan Python standar pada indeks ini:

```
area.index | population.index
```

Setiap item yang satu atau lainnya tidak memiliki entri ditandai dengan NaN, atau “Bukan Angka,” yang merupakan cara Pandas menandai data yang hilang. Pencocokan indeks ini diimplementasikan dengan cara ini untuk setiap ekspresi aritmatika bawaan Python; nilai yang hilang diisi dengan NaN secara default:

```
A = pd.Series ([2, 4, 6], index = [0, 1, 2])  
B = pd.Series ([1, 3, 5], index = [1, 2, 3])  
A + B
```

```
0      NaN  
1      5.0  
2      9.0  
3      NaN  
dtype: float64
```

Gambar 3. 32 Penjumlahan dengan cara pencocokan index

Jika menggunakan nilai NaN bukan hasil nilai yang diinginkan, kita dapat memodifikasi nilai pengisian menggunakan metode objek yang sesuai di tempat operator. Misalnya, memanggil `A.add (B)` sama dengan memanggil `A + B`, tetapi memungkinkan spesifikasi eksplisit opsional dari nilai pengisian untuk elemen apa pun di `A` atau `B` yang mungkin tidak ada:


```
A.add(B, fill_value=0)
```

```
A.add(B, fill_value=0)
0    2.0
1    5.0
2    9.0
3    5.0
dtype: float64
```

Gambar 3. 33 Memodifikasi nilai pengisian dengan metode objek

3.7 Pivot Table

Tabel pivot atau pivot table merupakan salah satu fitur dari Excel yang paling canggih. Tabel pivot memungkinkan kita untuk menarik wawasan dari berbagai data. Pandas sudah menyediakan fungsi serupa yang disebut *pivot_table()*. Pandas *pivot_table()* ialah fungsi yang sangat sederhana tetapi dapat menghasilkan analisis data yang sangat kuat dan juga sangat cepat.

Sebelum kita menggunakan table pivot, terlebih dahulu kita buat dataframe baru, berikut code program untuk membuat dataframe barunya:

```
import pandas as pd
df = pd.DataFrame({
    "A": ["foo", "foo", "foo", "foo", "foo", "bar", "bar", "bar", "bar"],
    "B": ["one", "one", "one", "two", "two", "one", "one", "two", "two"],
    "C": ["small", "large", "large", "small", "small", "large", "small", "small", "large"],
    "D": [1, 2, 2, 3, 3, 4, 5, 5, 6], "E": [2, 4, 5, 5, 6, 6, 8, 9, 9]})
df
```

	A	B	C	D	E
0	foo	one	small	1	2
1	foo	one	large	2	4
2	foo	one	large	2	5
3	foo	two	small	3	5
4	foo	two	small	3	6
5	bar	one	large	4	6
6	bar	one	small	5	8
7	bar	two	small	6	9
8	bar	two	large	7	9

Gambar 3. 34 Membuat dataframe baru

Contohnya kita akan menggabungkan nilai sesuai jenisnya dan lalu kita akan menggunakan fungsi sum, selanjutnya kita tampilkan dalam bentuk pivot table.

```
table = pd.pivot_table(df, values='D', index=['A', 'B'], columns=['C'],
aggfunc=np.sum)
table
```

		C	
		large	small
A	B		
bar	one	4.0	5.0
	two	7.0	6.0
foo	one	4.0	1.0
	two	NaN	6.0

Gambar 3. 35 Menggabungkan & menjumlahkan di pivot table

3.8 Loc dan Iloc

Konvensi slicing dan indexing ini dapat menjadi sumber kebingungan. Misalnya, jika Series Anda memiliki indeks integer eksplisit, operasi indexing seperti data[1] akan menggunakan indeks eksplisit, sementara operasi slicing seperti data[1:3] akan menggunakan indeks gaya Python implisit.

```
data = pd.Series (['a', 'b', 'c'], index=[1,3,5])
data

# explicit index when indexing
data[1]

# implicit index when slicing
data[1:3]
```

```
data = pd.Series(['a', 'b', 'c'], index=[1,3,5])
data
1    a
3    b
5    c
dtype: object

# explicit index when indexing
data[1]
'a'

# implicit index when slicing
data[1:3]
3    b
5    c
dtype: object
```

Gambar 3. 36 Menggunakan Indeks eksplisit dan implicit

Karena potensi kebingungan ini dalam kasus indeks integer, Pandas menyediakan beberapa atribut pengindeks khusus yang secara eksplisit mengekspos skema pengindeksan tertentu. Ini bukan metode fungsional, tetapi atribut yang mengekspos antarmuka pengiris tertentu ke data dalam Series.

Pertama, atribut `loc` memungkinkan pengindeksan dan slicing yang selalu merujuk pada indeks eksplisit:

```
data.loc[1]
data.loc[1:3]
```

```
data.loc[1]
'a'

data.loc[1:3]
1    a
3    b
dtype: object
```

Gambar 3. 37 Penggunaan atribut `loc`

Atribut `iloc` memungkinkan pengindeksan dan slicing yang selalu merujuk pada indeks style python implisit:

```
data.iloc[1]
data.iloc[1:3]
```

```
data.iloc[1]
'b'

data.iloc[1:3]
3    b
5    c
dtype: object
```

Gambar 3. 38 Penggunaan atribut iloc

Salah satu prinsip panduan kode Python adalah bahwa "eksplisit lebih baik daripada implisit." Sifat eksplisit loc dan iloc membuatnya sangat berguna dalam menjaga kode yang bersih dan mudah dibaca; terutama dalam kasus indeks integer, saya sarankan menggunakan keduanya untuk membuat kode lebih mudah dibaca dan dipahami, dan untuk mencegah subtle bug karena konvensi indexing/slicing campuran.

Mengelola Data

4.1 Persiapan Data

Sumber Data

Kita dapat menemukan data dimanapun itu, kegiatan kita sehari-hari juga dapat disebut dengan data. Terdapat tiga jenis data diantaranya.

a.) Data tidak terstruktur

Pada data tidak terstruktur tidak memiliki bentuk atau struktur khusus. Contohnya gambar, video, dan pdf.

b.) Data terstruktur

Pada data terstruktur sudah siap untuk diproses, karena memiliki format yang tetap juga dapat langsung disimpan dalam bentuk digital. Contohnya excel.

c.) Data semi terstruktur

Pada data semi terstruktur mengandung kedua informasi diatas namun belum dimasukkan ke dalam database tetapi memiliki informasi penting didalamnya. Contohnya csv, xml, dan json.

Data dapat berasal dari eksternal ataupun internal dengan bantuan library python yakni sklearn.dataset.

a.) Internal

Library sklearn pada python menyediakan contoh dataset yang siap untuk digunakan. Data tersebut seperti data iris, boston, dan sebagainya. Pada bagian contoh kasus dalam buku ini menggunakan dataset boston yang disediakan oleh sklearn.dataset.

```
import pandas as pd
import numpy as np
from sklearn import datasets
```

```

omah = datasets.load_boston()
omah.data.shape
omah.feature_names
bstn = pd.DataFrame(omah.data)
bstn.columns = omah.feature_names
bstn.head()

```

```
omah.data.shape
```

```
(506, 13)
```

```
omah.feature_names
```

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',  
      'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

```
bstn = pd.DataFrame(omah.data)
bstn.columns = omah.feature_names
```

```
bstn.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

Gambar 4. 1 Data Internal

b.) Eksternal

Data eksternal dapat kita dapatkan dimanapun seperti data perusahaan, universitas, dan juga data open-source yang dibagikan melalui internet. Pada bagian contoh kasus dalam buku ini menggunakan dataset tips yang didapatkan melalui halaman website internet kaggle.com.

```

import pandas as pd
import numpy as np

dataTip = pd.read_csv('tips.csv')
dataTip.head()

```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Gambar 4. 2 Data Eksternal

Seleksi Fitur

Pada proses mengolah mungkin memiliki banyak data. Tetapi apakah semuanya berguna dan relevan untuk digunakan? Kolom dan fitur mana yang mungkin berkontribusi pada hasil? Seringkali, beberapa data tidak relevan dengan analisis yang telah dilakukan. Misalnya, apakah nama startup mempengaruhi keberhasilan pendanaannya oleh investor? Apakah ada hubungan antara warna kesukaan seseorang dengan kecerdasannya? Memilih fitur yang paling relevan juga merupakan tugas penting dalam memproses data. Mengapa membuang waktu dan sumber daya komputasi yang berharga untuk menyertakan fitur ataupun kolom yang tidak relevan dalam analisis? Lebih buruk lagi, apakah fitur yang tidak relevan akan mencondongkan analisis yang dilakukan? Jawabannya iya. Jika menyertakan fitur yang tidak relevan dalam analisis, mungkin juga mendapatkan hasil yang tidak akurat dan tidak relevan. Komputer dan algoritma yang digunakan akan "belajar dari contoh buruk" yang menghasilkan hasil yang salah. Untuk menghilangkan Sampah dan meningkatkan akurasi dan relevansi analisis kami, Seleksi Fitur sering dilakukan.

Sesuai dengan istilahnya, memilih "fitur" yang memiliki kontribusi terbesar dan relevansi langsung dengan output. Ini akan membuat model prediksi lebih sederhana dan lebih mudah dipahami. Misalnya, kami mungkin memiliki 20+ fitur yang menggambarkan pelanggan. Fitur-fitur ini termasuk usia, kisaran pendapatan, lokasi, jenis kelamin, apakah mereka memiliki anak atau tidak, tingkat pengeluaran, pembelian terakhir, pencapaian pendidikan tertinggi, apakah mereka memiliki rumah atau tidak, dan lebih dari selusin atribut lainnya yang terdapat pada data. Namun, tidak semua ini mungkin

memiliki relevansi dengan analisis atau model prediksi. Meskipun mungkin semua fitur ini memiliki beberapa efek, analisisnya mungkin terlalu rumit untuk menjadi berguna. Seleksi Fitur adalah cara menyederhanakan analisis dengan berfokus pada fitur yang relevan. Tapi bagaimana kita tahu jika fitur tertentu relevan? Di sinilah pengetahuan dan keahlian domain berperan. Misalnya, analis data atau dalam tim harus memiliki pengetahuan tentang ritel (dalam contoh kami di atas). Dengan cara ini, tim dapat dengan tepat memilih fitur yang paling berdampak pada model atau analisis prediktif. Bidang yang berbeda sering kali memiliki fitur relevan yang berbeda pula. Misalnya, menganalisis data ritel mungkin sama sekali berbeda dengan mempelajari data kualitas pada anggur. Di ritel kami fokus pada fitur yang memengaruhi pembelian orang (dan dalam jumlah berapa). Di sisi lain, menganalisis data kualitas anggur mungkin memerlukan dan mempelajari konstituen kimia anggur dan pengaruhnya terhadap preferensi orang. Selain itu, diperlukan beberapa pengetahuan domain untuk mengetahui fitur mana yang saling bergantung satu sama lain. Dalam contoh kami di atas tentang kualitas anggur, zat dalam anggur mungkin bereaksi satu sama lain dan karenanya mempengaruhi jumlah zat tersebut. Ketika Anda meningkatkan jumlah suatu zat, itu dapat menambah atau mengurangi jumlah yang lain.

Begitu juga dengan menganalisis data pada suatu bisnis. Lebih banyak pelanggan juga berarti lebih banyak penjualan. Orang-orang dari kelompok pendapatan yang lebih tinggi mungkin memiliki tingkat pengeluaran yang lebih tinggi. Fitur-fitur ini saling bergantung dan mengecualikan beberapa di antaranya sehingga dapat menyederhanakan analisis. Memilih fitur yang paling tepat mungkin juga membutuhkan waktu ekstra terutama ketika sedang berurusan dengan kumpulan data yang sangat besar (dengan ratusan atau bahkan ribuan kolom). Profesional sering mencoba kombinasi yang berbeda dan melihat mana yang memberikan hasil terbaik (atau mencari sesuatu yang paling masuk akal). Secara umum, keahlian domain bisa lebih penting daripada keterampilan analisis data itu sendiri. Bagaimanapun, kita harus mulai dengan mengajukan pertanyaan yang tepat daripada berfokus pada penerapan algoritma yang paling rumit pada data. Untuk mengetahui pertanyaan yang tepat dan yang paling penting, seseorang dari tim harus memiliki keahlian dalam subjek tersebut.

Pembersihan Data

Selama melakukan analisis dan pemodelan data, sebagian besar waktu dihabiskan untuk persiapan data: memuat, membersihkan, mengubah, dan menata ulang. Tugas-tugas seperti itu sering menghabiskan 80% atau lebih waktu seorang analis data. Terkadang cara data disimpan dalam file atau database tidak dalam format yang tepat untuk tugas tertentu yang dibutuhkan. Banyak peneliti memilih untuk melakukan pemrosesan data dari satu bentuk ke bentuk lainnya menggunakan bahasa pemrograman umum, seperti Python, Perl, R, atau Java, atau alat pemrosesan teks Unix seperti sed atau awk. Untungnya, pandas, bersama dengan fitur bahasa Python bawaan, memberi seperangkat alat tingkat tinggi, fleksibel, dan cepat untuk memungkinkan dalam memanipulasi data ke dalam bentuk yang tepat.

Menangani data yang hilang

Data yang hilang sering terjadi Ketika sedang menganalisis data. Salah satu tujuan pandas adalah membuat bekerja dengan data yang hilang semudah mungkin. Misalnya, semua statistik deskriptif pada objek pandas mengecualikan data yang hilang secara default. Cara data yang hilang direpresentasikan dalam objek pandas agak tidak sempurna, tetapi berfungsi untuk banyak pengguna. Untuk data numerik, pandas menggunakan nilai floating-point NaN (Not a Number) untuk mewakili data yang hilang. Biasanya disebut nilai sentinel yang dapat dengan mudah dideteksi.

```
import pandas as pd
import numpy as np

string_data = pd.Series(['aardvark', 'artichoke', np.nan, 'avocado'])
string_data
```

```
0    aardvark
1    artichoke
2         NaN
3     avocado
dtype: object
```

Gambar 4. 3 Data Hilang

```
string_data.isnull()
```

```
0    False
1    False
2     True
3    False
dtype: bool
```

Gambar 4. 4 Menangani Data Hilang

Pandas telah mengadopsi konvensi yang digunakan dalam bahasa pemrograman R dengan merujuk ke data yang hilang sebagai NA, yang berarti tidak tersedia. Dalam statistik, data NA dapat berupa data yang tidak ada atau yang ada tetapi tidak diamati. Saat membersihkan data untuk analisis, seringkali penting dalam melakukan analisis pada data yang hilang itu sendiri untuk mengidentifikasi masalah pengumpulan data atau potensi bias dalam data yang disebabkan oleh data yang hilang.

```
string_data[0] = None
string_data.isnull()
```

```
0     True
1    False
2     True
3    False
dtype: bool
```

Gambar 4. 5 Menambah Data Hilang

Ada yang sedang berlangsung di proyek pandas untuk meningkatkan detail internal tentang bagaimana data yang hilang ditangani, tetapi fungsi API pengguna, seperti `pandas.isnull`, mengabstraksikan banyak detail yang mengganggu. Berikut daftar beberapa fungsi yang terkait dengan penanganan data yang hilang.

- **Dropna** : Filter label sumbu berdasarkan apakah nilai untuk setiap label memiliki data yang hilang, dengan ambang batas yang bervariasi untuk berapa banyak data yang hilang untuk ditoleransi.
- **Fillna** : Isi data yang hilang dengan beberapa nilai atau gunakan metode interpolasi seperti 'fill' atau 'bfill'.
- **IsNull** : Kembalikan nilai boolean yang menunjukkan nilai mana yang hilang/NA.
- **Notnull** : Negasi dari isnull

Filtering Out Missing Data

Ada beberapa cara untuk menyaring data yang hilang. Meskipun selalu memiliki opsi untuk melakukannya dengan menggunakan `pandas.isnull` dan pengindeksan boolean, `dropna` dapat membantu. kali ini mengembalikan dengan hanya data non-null dan nilai indeks:

```
from numpy import nan as NA
data = pd.Series([1, NA, 3.5, NA, 7])
data[data.notnull()]
```

```
0    1.0
2    3.5
4    7.0
dtype: float64
```

Gambar 4. 6 Not Null

Dengan objek `DataFrame`, semuanya sedikit lebih kompleks. Mungkin ingin menghapus baris atau kolom yang semuanya NA atau hanya yang berisi NA apa pun. `dropna` secara default menghapus setiap baris yang berisi nilai yang hilang.

```
data = pd.DataFrame([[1., 6.5, 3.],
                     [1., NA, NA],
                     [NA, NA, NA],
                     [NA, 6.5, 3.]])
data
```

	0	1	2
0	1.0	6.5	3.0
1	1.0	NaN	NaN
2	NaN	NaN	NaN
3	NaN	6.5	3.0

Gambar 4. 7 Filter Missing Data

```
cleaned = data.dropna()
cleaned
```

	0	1	2
0	1.0	6.5	3.0

Gambar 4. 8 Dropna Data

Jika how='all' hanya akan menjatuhkan baris yang semuanya NA.

```
data.dropna(how='all')
```

	0	1	2
0	1.0	6.5	3.0
1	1.0	NaN	NaN
3	NaN	6.5	3.0

Gambar 4. 9 All Dropna Data

Mengisi data yang hilang

Daripada memfilter data yang hilang (dan berpotensi membuang data lain bersamanya), mungkin ingin mengisi "data kosong" dengan berbagai cara. Untuk sebagian besar tujuan, metode fillna adalah fungsi yang cocok untuk digunakan. Memanggil fillna dengan konstanta akan menggantikan nilai yang hilang.

```
df = pd.DataFrame(np.random.randn(7, 3))
df.iloc[:4, 1] = NA
df.iloc[:2, 2] = NA
df
```

	0	1	2
0	-0.204708	NaN	NaN
1	-0.555730	NaN	NaN
2	0.092908	NaN	0.769023
3	1.246435	NaN	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

Gambar 4. 10 Missing Random Data

```
df.fillna(0)
```

	0	1	2
0	-0.204708	0.000000	0.000000
1	-0.555730	0.000000	0.000000
2	0.092908	0.000000	0.769023
3	1.246435	0.000000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

Gambar 4. 11 Rubah Data Missing ke Nol

Memanggil fillna dengan dict, dapat menggunakan nilai isian yang berbeda untuk setiap kolom.

```
df.fillna({1: 0.5, 2: 0})
```

	0	1	2
0	-0.204708	0.500000	0.000000
1	-0.555730	0.500000	0.000000
2	0.092908	0.500000	0.769023
3	1.246435	0.500000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

Gambar 4. 12 Rubah Data Missing Rentang

fillna mengembalikan objek baru, tetapi Anda dapat memodifikasi objek yang ada di tempat. Jadi inplace=True akan merubah data dari variable yang dirubah

```
df.fillna(0, inplace=True)
df
```

	0	1	2
0	-0.204708	0.000000	0.000000
1	-0.555730	0.000000	0.000000
2	0.092908	0.000000	0.769023
3	1.246435	0.000000	-1.296221
4	0.274992	0.228913	1.352917
5	0.886429	-2.001637	-0.371843
6	1.669025	-0.438570	-0.539741

Gambar 4. 13 Tetapkan Perubahan Data

Terdapat fungsi pada fillna diantaranya.

- Value : Nilai skalar atau objek seperti dict yang digunakan untuk mengisi nilai yang hilang
- Method : Interpolasi; secara default 'mengisi' jika fungsi dipanggil tanpa argumen lain
- Axis : Sumbu untuk diisi; sumbu default = 0
- Inplace : Ubah objek panggilan tanpa membuat Salinan
- Limit : Untuk pengisian maju dan mundur, jumlah maksimum periode berurutan yang harus diisi

Menghapus duplikat data

Baris duplikat dapat ditemukan di DataFrame karena sejumlah alasan seperti kesalahan dalam memasukkan data dan sebagainya.

```
data = pd.DataFrame({'k1': ['one', 'two'] * 3 + ['two'],
                     'k2': [1, 1, 2, 3, 3, 4, 4]})
data
```

	k1	k2
0	one	1
1	two	1
2	one	2
3	two	3
4	one	3
5	two	4
6	two	4

Gambar 4. 14 Duplikasi Data

Metode DataFrame yang diduplikasi mengembalikan Seri boolean yang menunjukkan apakah setiap baris adalah duplikat atau tidak.

```
data.duplicated()
```

0	False
1	False
2	False
3	False
4	False
5	False
6	True

dtype: bool

Gambar 4. 15 Menampilkan Duplikasi Data

Terkait, drop_duplicates atau menghapus data duplikasi dapat mengembalikan DataFrame di mana array yang digandakan adalah False:

```
data.drop_duplicates()
```

	k1	k2
0	one	1
1	two	1
2	one	2
3	two	3
4	one	3
5	two	4

Gambar 4. 16 drop_duplicates

Kedua metode ini secara default mempertimbangkan semua kolom; sebagai alternatif, dapat menentukan subset apa pun untuk mendeteksi duplikat. Misalkan kita memiliki kolom nilai tambahan dan ingin memfilter duplikat hanya berdasarkan kolom 'k1':

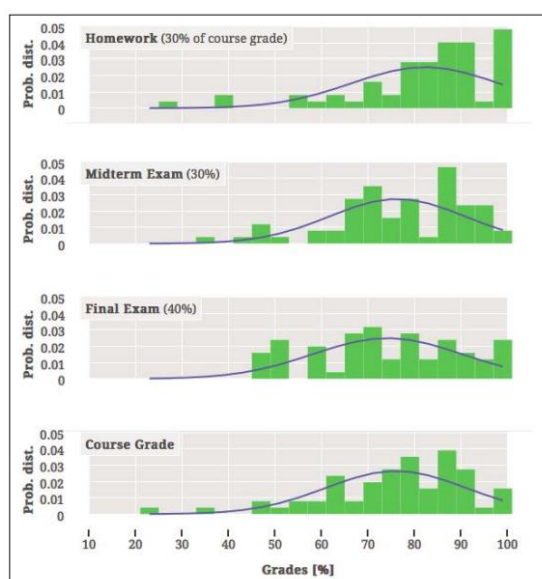
```
data['v1'] = range(7)
data.drop_duplicates(['k1'])
```

	k1	k2	v1
0	one	1	0
1	two	1	1

Gambar 4. 17 Filter Duplikasi Data

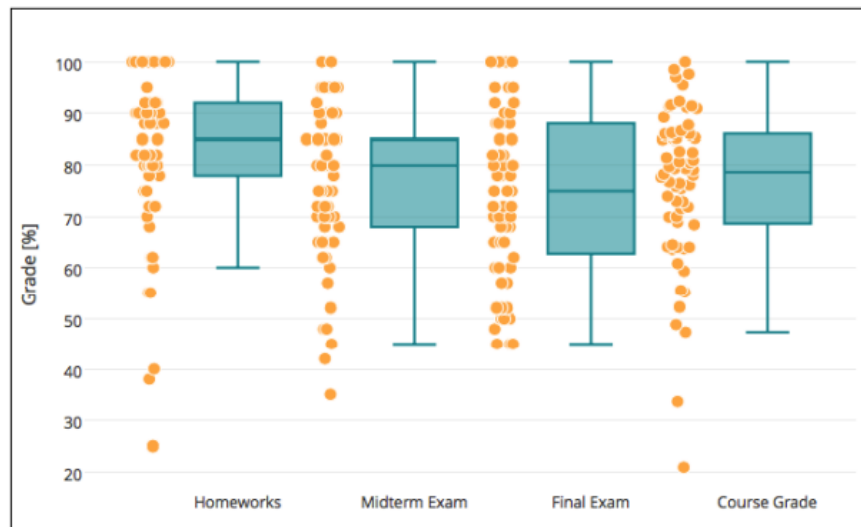
4.2 Distribusi Data

Analisis distribusi menunjukkan bagaimana nilai kuantitatif didistribusikan di seluruh rentangnya, oleh karena itu, sangat berguna dalam analisis data. Misalnya, bandingkan distribusi nilai pekerjaan rumah, ujian tengah semester, ujian akhir, dan total nilai mata pelajaran suatu kelas siswa. Dalam contoh ini, kita akan membahas dua jenis grafik yang paling umum digunakan untuk tujuan ini. Salah satunya adalah histogram (seperti yang ditunjukkan pada gambar berikut), dan yang lainnya adalah plot *box-and-whisker*.



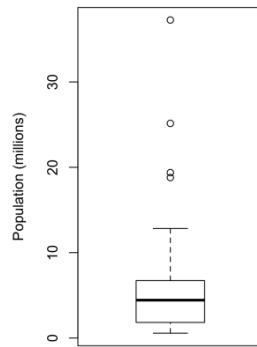
Gambar 4. 18 Distribusi Pada Data

Bentuk histogram sangat bergantung pada ukuran bin dan lokasi yang ditentukan. Plot *box-and-whisker* sangat baik untuk menampilkan banyak distribusi. Mereka mengemas semua titik data, dalam hal ini nilai per siswa ke dalam tampilan *box-and-whisker*. Sekarang Anda dapat dengan mudah mengidentifikasi nilai rendah, nilai persentil ke-25, median, persentil ke-75, dan nilai maksimum di semua kategori semuanya pada saat yang bersamaan.



Gambar 4. 19 box-and-whisker

Dari boxplot di bawah ini kita dapat langsung melihat bahwa median populasi negara bagian adalah sekitar 5 juta, setengah negara bagian berada di antara sekitar 2 juta dan sekitar 7 juta, dan ada beberapa outlier populasi yang tinggi. Bagian atas dan bawah kotak masing-masing adalah persentil ke-75 dan ke-25. Median ditunjukkan oleh garis horizontal di dalam kotak. Garis putus-putus, memanjang dari bagian atas dan bawah kotak untuk menunjukkan rentang sebagian besar data.



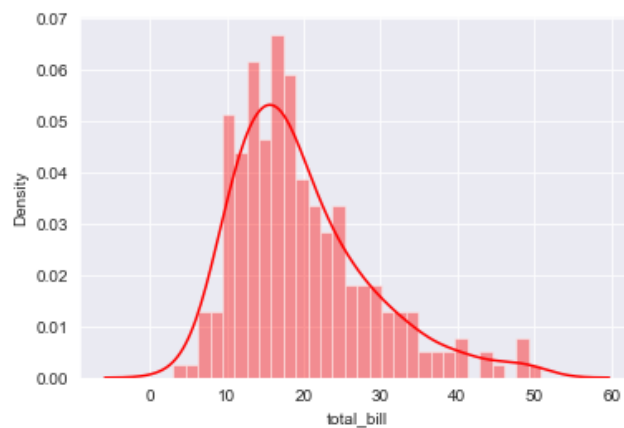
Gambar 4. 20 Boxplot

Salah satu dari banyak cara untuk dengan mudah memplot ini dengan Python adalah dengan menggunakan Plotly, yang merupakan alat analitik dan visualisasi online. Plotly menyediakan grafik online, salah satu alat dan analitik statistik serta perpustakaan plot ilmiah untuk Python.

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

dataTip = pd.read_csv('tips.csv')

sns.set_style('darkgrid')
sns.distplot(dataTip['total_bill'],
              kde = True,
              color = 'red',
              bins = 30)
```



Gambar 4. 21 Distribusi pada Data

4.3 Deskripsi Data

Deskripsi data statistic ialah cara untuk melihat, mengatur, dan mendeskripsikan kumpulan data menggunakan tabel bagan dan banyak parameter numerik lainnya. Terdapat istilah statistik yang termasuk dalam deskripsi data dan contoh penerapannya.

```
import pandas as pd
import numpy as np

dataTip = pd.read_csv('tips.csv')
dataTip.describe()
```

Out[18]:

	total_bill	tip	size	total_tip
count	243.000000	243.000000	243.000000	243.000000
mean	19.813868	3.002387	2.572016	0.677844
std	8.910071	1.385002	0.952356	0.638875
min	3.070000	1.000000	1.000000	0.030700
25%	13.380000	2.000000	2.000000	0.283375
50%	17.810000	2.920000	2.000000	0.492681
75%	24.175000	3.575000	3.000000	0.835275
max	50.810000	10.000000	6.000000	5.081000

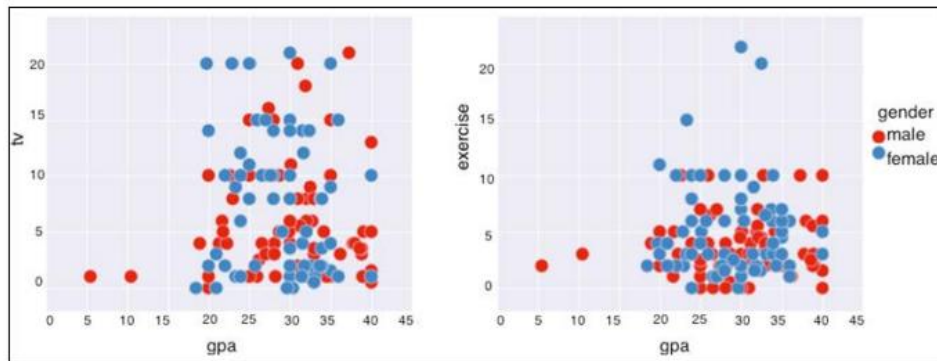
Gambar 4. 22 Deskripsi Data

- Count : Jumlah dari keseluruhan data.
- Mean : Nilai rata-rata dari data yang bertipe numerik/angka.
- Std : Rata-rata kuadrat dari selisih/jarak setiap observasi dengan nilai mean.
- Min : Nilai terkecil dari data
- 25% : Dikenal dengan kuartal 1
- 50% : Dikenal dengan kuartal 2 atau nilai tengah
- 75% : Dikenal dengan kuartal 3

- Max : Nilai terbesar dari data

4.4 Korelasi Data

Analisis korelasi adalah awal yang baik untuk mengidentifikasi hubungan antara ukuran atau data, meskipun korelasi tidak menjamin suatu hubungan. Untuk mengkonfirmasi bahwa hubungan itu benar-benar ada, metodologi statistik sering digunakan. Seperti mean dan standar deviasi, koefisien korelasi sensitif terhadap outlier atau data yang jauh dari kumpulan data yang lain dalam data.. Berikut ini adalah contoh untuk membangun scatter plot sederhana untuk mendeteksi korelasi antara dua faktor, katakanlah gpa atau lebih kita kenal dengan ipk dan tv atau gpa dan latihan di antara mahasiswa dari sebuah universitas:



Gambar 4. 23 Korelasi Scatter Plot

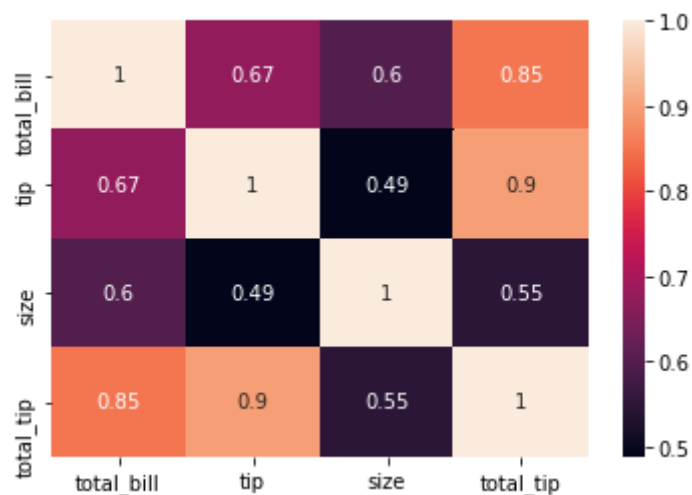
Analisis data eksplorasi dalam banyak proyek pemodelan (baik dalam ilmu data atau dalam penelitian) melibatkan pemeriksaan korelasi antar prediktor, dan antara prediktor dan variabel target. Variabel X dan Y (masing-masing dengan data terukur) dikatakan berkorelasi positif jika nilai X yang tinggi diikuti dengan nilai Y yang tinggi, dan nilai X yang rendah dengan nilai Y yang rendah. Jika nilai X yang tinggi diikuti dengan nilai Y yang rendah, nilai Y, dan sebaliknya, variabel berkorelasi negatif. Jika nilai mendekati positif 1 ataupun negatif 1 maka korelasi tinggi begitupula pula sebaliknya, jika mendekati 0 maka korelasi rendah.

Namun, kita juga dapat menggunakan cara lain untuk menampilkan matriks korelasi. Misalnya, seseorang dapat menggunakan plot pencar, peta panas, atau beberapa contoh spesifik untuk menunjukkan pengaruh terhadap

data. Untuk lebih menekankan, matriks korelasi melibatkan data dalam bentuk matriks. Data dikorelasikan dengan menggunakan peta warna berskala, seperti yang ditunjukkan pada contoh berikut.

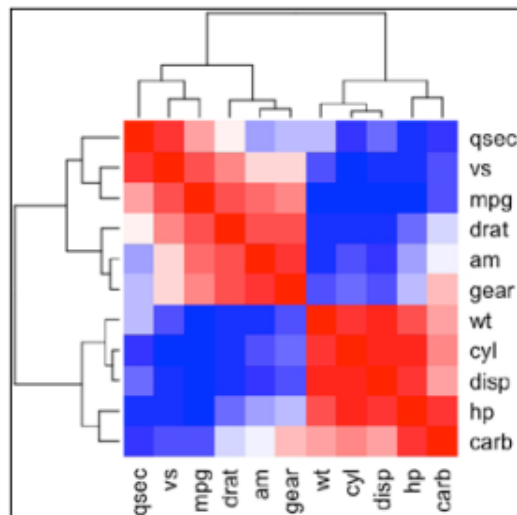
```
import pandas as pd
import numpy as np
import seaborn as sns

dataTip = pd.read_csv('tips.csv')
sns.heatmap(data=dataTip.corr(),annot=True)
```



Gambar 4. 24 Korelasi Peta Panas

Sebuah matriks korelasi digunakan untuk menyelidiki ketergantungan antara beberapa variabel pada waktu yang sama. Hasilnya adalah tabel yang berisi koefisien korelasi antara masing-masing variabel dengan variabel lainnya. Peta panas berasal dari tampilan 2D dari nilai-nilai dalam matriks data. Ada banyak skema warna berbeda yang dapat digunakan untuk mengilustrasikan peta panas, dengan kelebihan dan kekurangan persepsi masing-masing.



Gambar 4. 25 Korelasi Data

4.5 Visualisasi Data

Membuat visualisasi yang informatif (kadang-kadang disebut plot) adalah salah satu tugas terpenting dalam analisis data. Ini mungkin menjadi bagian dari proses eksplorasi misalnya, untuk membantu mengidentifikasi outlier atau transformasi data yang diperlukan, atau sebagai cara menghasilkan ide untuk model. Python memiliki banyak pustaka tambahan untuk membuat visualisasi statis atau dinamis

Mengeksplorasi serta mengkomunikasikan data adalah tujuan utama dari visualisasi data. Ketika data divisualisasikan (dalam diagram batang, histogram, ataupun bentuk lain), pola menjadi jelas. Maka akan diketahui dengan cepat apakah ada tren naik atau besaran relatif dari sesuatu sehubungan dengan faktor lain (misalnya menggunakan diagram lingkaran). Alih-alih memberi tahu orang-orang daftar angka yang panjang, mengapa tidak menunjukkannya saja kepada mereka untuk kejelasan yang lebih baik menggunakan visualisasi?

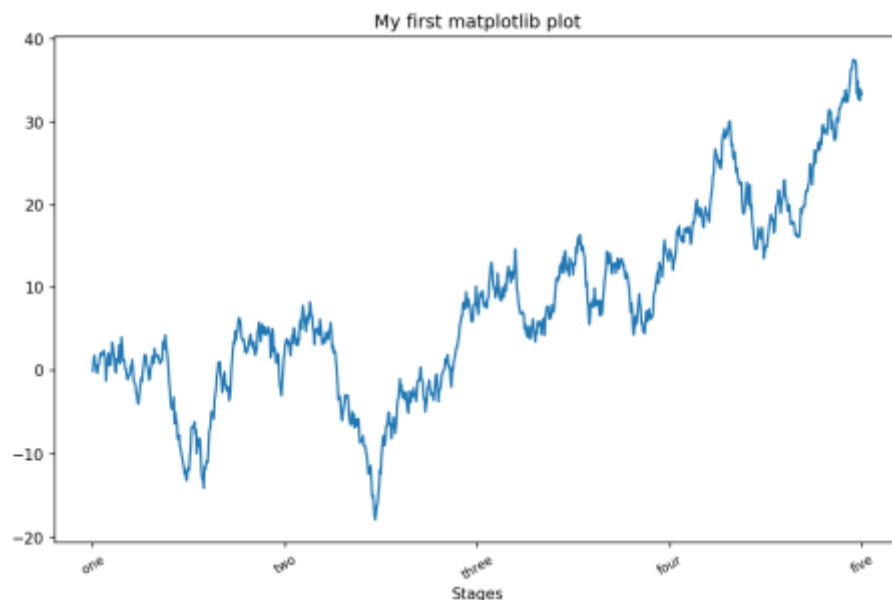
Ini juga penting ketika presentasi ke client atau publik. Orang lain mungkin lebih suka ikhtisar data singkat tanpa terlalu banyak membahas detailnya. Anda tidak ingin mengganggu mereka dengan teks dan angka yang membosankan. Yang membuat dampak lebih besar adalah bagaimana Anda menyajikan data sehingga orang akan segera mengetahui pentingnya data tersebut. Di sinilah visualisasi data dapat terjadi di mana Anda memungkinkan orang untuk dengan

cepat menjelajahi data dan secara efektif mengomunikasikan apa yang Anda coba katakan.

Ada beberapa cara untuk memvisualisasikan data seperti dapat langsung membuat plot dan grafik dengan Microsoft Excel. Anda juga dapat menggunakan D3, seaborn, Bokeh, dan matplotlib. Dalam bab ini dan berikutnya, kami akan fokus menggunakan library python khususnya seaborn dan matplotlib.

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

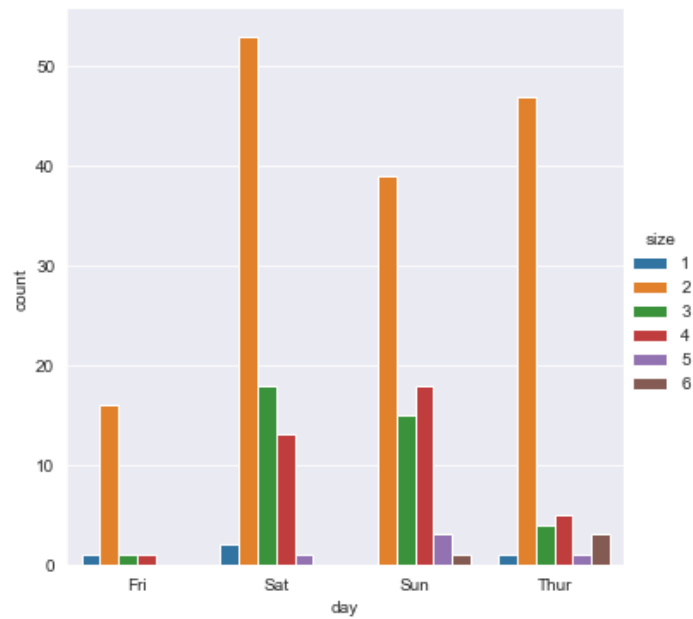
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(np.random.randn(1000).cumsum())
ticks = ax.set_xticks([0, 250, 500, 750, 1000])
labels = ax.set_xticklabels(['one', 'two', 'three', 'four', 'five']
                             rotation=30,
                             fontsize='small')
ax.set_title('My first matplotlib plot')
ax.set_xlabel('Stages')
```



Gambar 4. 26 Visualisasi Matplotlib

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

data = pd.read_csv('tips.csv')
sns.catplot(x="day", hue="size",
            kind="count", data=data)
```



Gambar 4. 27 Visualisasi Seaborn

Eksplorasi Data Tip

5.1 Pemahaman Bisnis

Pada contoh kasus kali ini, saya akan menggunakan dataset tip untuk mengeksplor data tersebut. Tip sendiri ialah uang dari konsumen yang diberikan kepada pemberi jasa sebagai tambahan dari harga yang diberikan, pada contoh ini adalah restoran. Tip makanan di restoran dapat dipengaruhi oleh banyak faktor diantaranya seperti sifat restoran, ukuran, dan meja restoran. Sebagai seorang manajer perlu mengetahui faktor penting saat menetapkan meja. Untuk mematuhi hukum setempat yakni di Amerika, restoran menawarkan tempat duduk bebas rokok kepada pelanggan yang meminta.

5.2 Pemahaman Data

Data diperoleh dari Kaggle dengan tautan [seaborn tips dataset | Kaggle](#). Terdapat tujuh fitur atau kolom pada data diantaranya total_bill, total tagihan dari biaya makanan yang dipesan dalam dolar AS dan termasuk pajak. Tip, uang yang diberikan konsumen atau gratifikasi. Sex, jenis kelamin konsumen. Smoker, keterangan konsumen apakah merokok di saat berada di restoran atau tidak. Day, hari kunjungan ke restoran. Time, waktu kunjungan ke restaurant. Size, kapasitas meja di restaurant.

5.3 Persiapan Data

Pertama import semua library yang dibutuhkan serta membaca data

```
import pandas as pd
```

```

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from plotnine import ggplot, aes, facet_grid, labs, geom_col
%matplotlib inline

dataTip = pd.read_csv('tips.csv')
dataTip.head()

```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Gambar 5. 1 Membaca Data Tip

Menambahkan kolom mungkin informasi ini dapat berguna, berapa persen tip yang diberikan dari total tagihan.

```

dataTip['total_tip'] = dataTip['total_bill'] * dataTip['tip'] /
100
dataTip.head()

```

	total_bill	tip	sex	smoker	day	time	size	total_tip
0	16.99	1.01	Female	No	Sun	Dinner	2	0.171599
1	10.34	1.66	Male	No	Sun	Dinner	3	0.171644
2	21.01	3.50	Male	No	Sun	Dinner	3	0.735350
3	23.68	3.31	Male	No	Sun	Dinner	2	0.783808
4	24.59	3.61	Female	No	Sun	Dinner	4	0.887699

Gambar 5. 2 Data Tip Baru

Merubah tipe data, pada kasus ini tipe data objek dirubah menjadi kategori

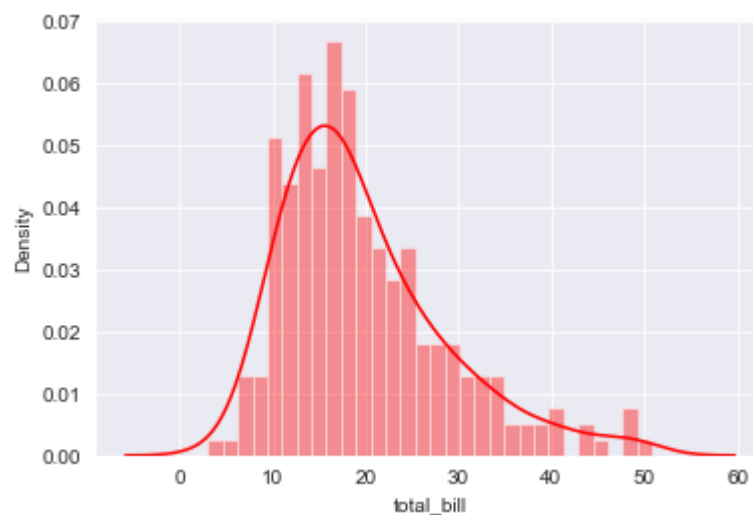
```
dataTip['sex'] = dataTip['sex'].astype('category')
dataTip['smoker'] = dataTip['smoker'].astype('category')
dataTip['day'] = dataTip['day'].astype('category')
dataTip['time'] = dataTip['time'].astype('category')
dataTip.dtypes
```

```
Out[10]: total_bill    float64
         tip          float64
         sex          category
         smoker        category
         day          category
         time          category
         size          int64
         total_tip     float64
         dtype: object
```

Gambar 5. 3 Tipe Data Tip

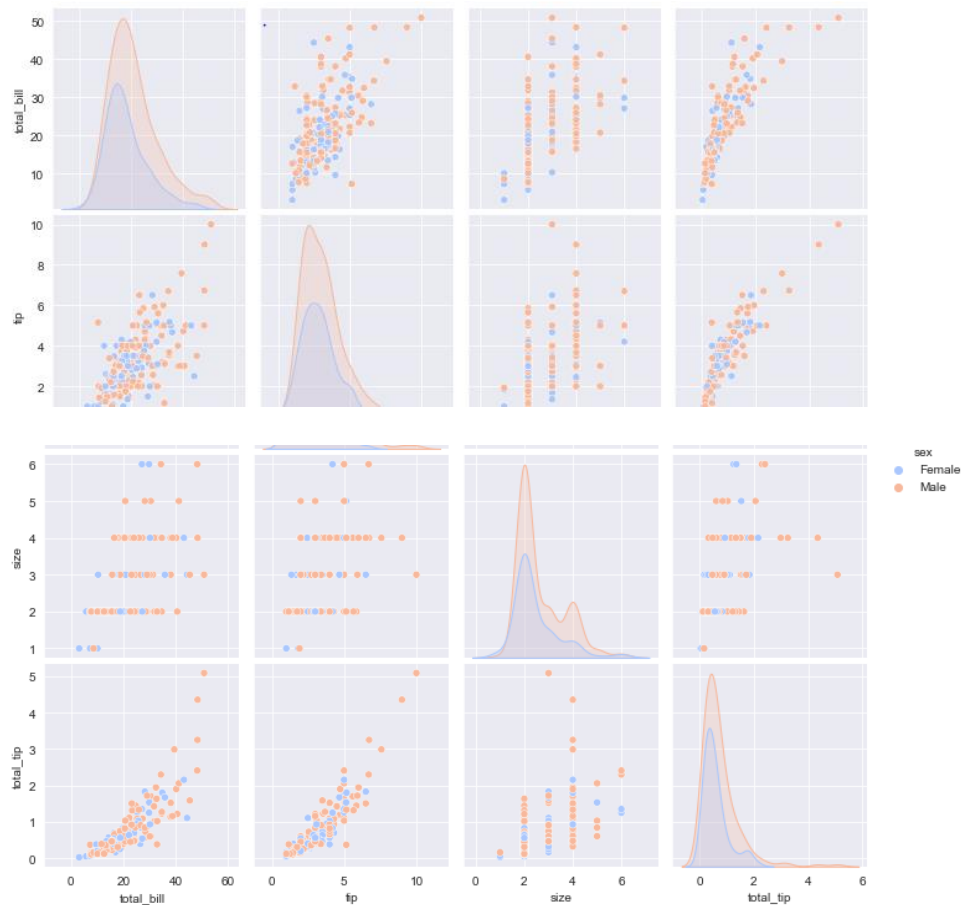
Menampilkan distribusi data

```
sns.set_style('darkgrid')
sns.distplot(dataTip['total_bill'], kde = True, color = 'red', bins
= 30)
```



Gambar 5. 4 Distribusi Data Tip

```
sns.pairplot(dataTip, hue = "sex", palette = 'coolwarm')
```



Gambar 5. 5 Grafik Distribusi Data Tip

Melihat dan memperbaiki jika terdapat outlier, noise, missing value, atau duplikasi data.

```
dataTip.isnull().sum()
```

```
Out[14]: total_bill    0
         tip          0
         sex          0
         smoker        0
         day           0
         time          0
         size          0
         total_tip     0
         dtype: int64
```

Gambar 5. 6 Data Tip Kosong

```
dataTip.duplicated().sum()
```

Out[15]: 1

Gambar 5. 7 Data Tip Duplikat

```
dataTip.drop_duplicates(inplace=True)  
dataTip.duplicated().sum()
```

Out[16]: 0

Gambar 5. 8 Data Tip Hapus Duplikat

Menampilkan deskripsi statistika sederhana dari data numerik

```
dataTip.describe()
```

Out[18]:

	total_bill	tip	size	total_tip
count	243.000000	243.000000	243.000000	243.000000
mean	19.813868	3.002387	2.572016	0.677844
std	8.910071	1.385002	0.952356	0.638875
min	3.070000	1.000000	1.000000	0.030700
25%	13.380000	2.000000	2.000000	0.283375
50%	17.810000	2.920000	2.000000	0.492681
75%	24.175000	3.575000	3.000000	0.835275
max	50.810000	10.000000	6.000000	5.081000

Gambar 5. 9 Deskripsi Data Tip

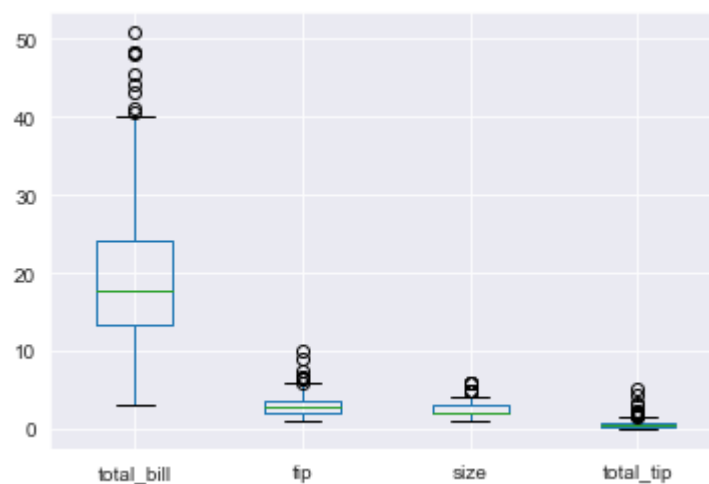
Menampilkan data unik dan memahami karakteristik dari distribusi data atau kesimetrisan sebaran data

```
dataTip['size'].unique()
```

```
Out[19]: array([2, 3, 4, 1, 6, 5], dtype=int64)
```

Gambar 5. 10 Data Tip Unik

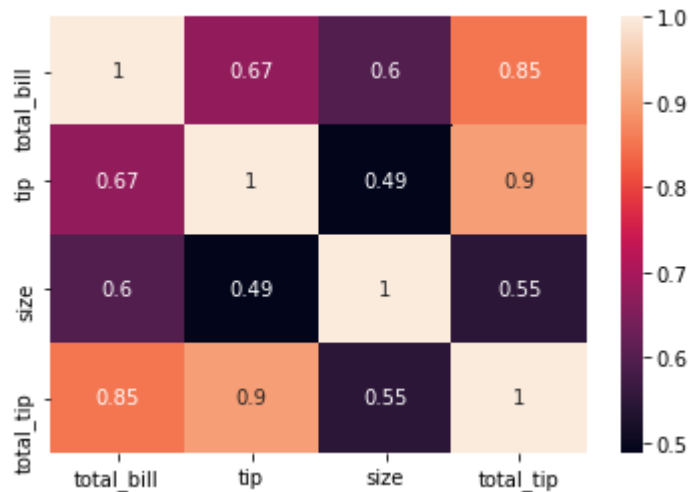
```
dataTip.boxplot()
```



Gambar 5. 11 Boxplot Data Tip

Dengan menggunakan heatmap dapat diketahui korelasi antar data

```
sns.heatmap(data=dataTip.corr(),annot=True)
```

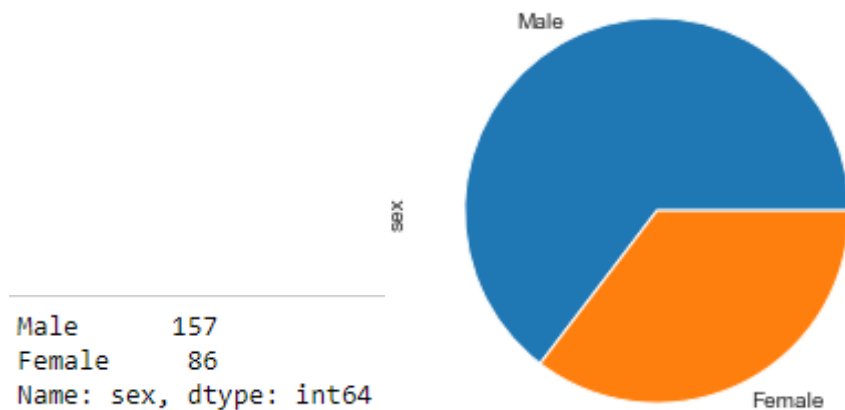


Gambar 5. 12 Korelasi Data Tip

5.4 Eksplorasi dan Penyajian Data

Apakah pelanggan pria dan wanita cenderung proporsional (balance)?

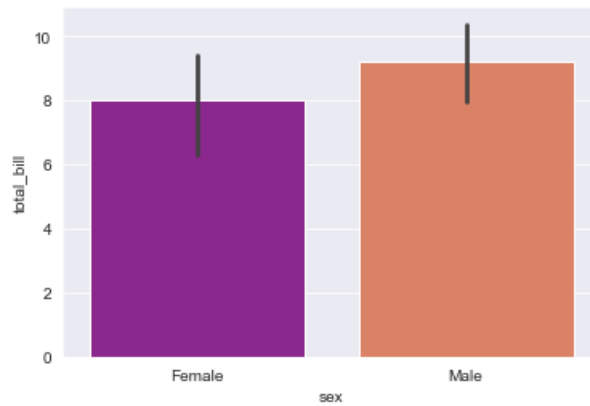
```
dataTip['sex'].value_counts()
dataTip['sex'].value_counts().plot(kind='pie')
```



Gambar 5. 13 Data Tip Jenkel

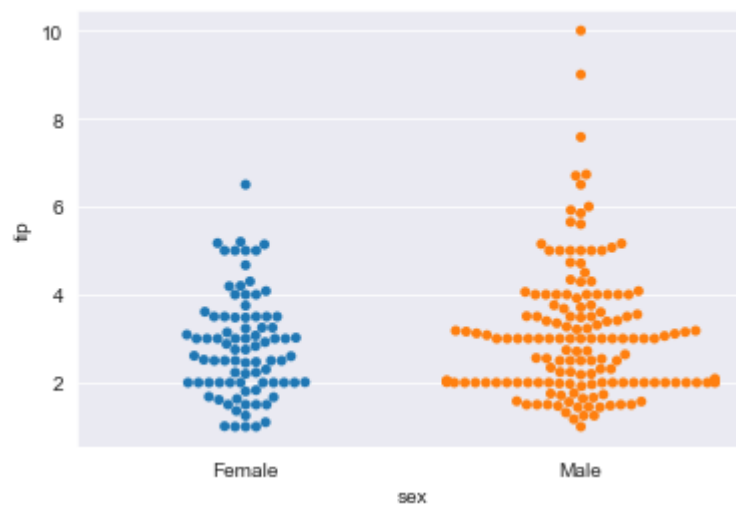
Dari data yang ada apakah Pria atau wanita ada kecenderungan memberi tips lebih besar?

```
sns.barplot(x='sex', y='tip', data=dataTip, palette='plasma')
```



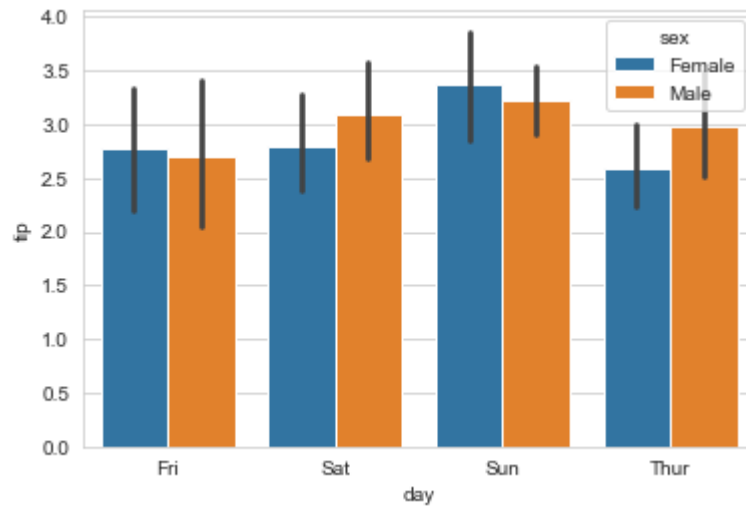
Gambar 5. 14 Data Tips

```
sns.swarmplot(x='sex', y='tip', data = dataTip)
```



Gambar 5. 15 Data Tips 1

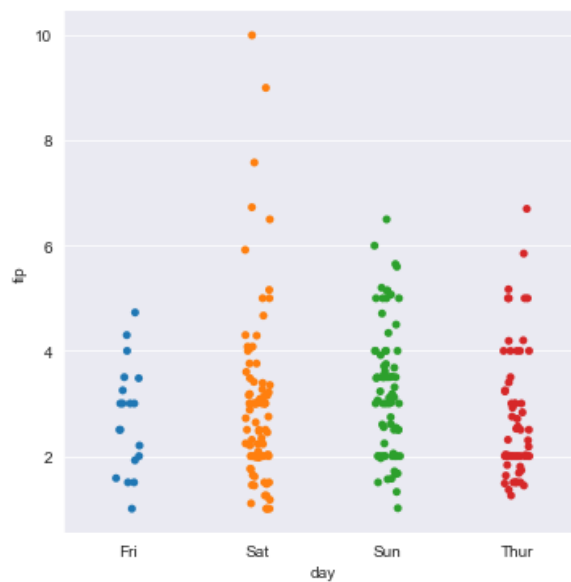
```
sns.barplot(x='day', y='tip', data=dataTip, hue='sex')
```

Gambar 5. 16 Data Tips 2

Dari data yang ada apakah ada kecenderungan tips lebih besar di hari-hari tertentu?

```
sns.catplot(x='day', y='tip', data=dataTip)
```



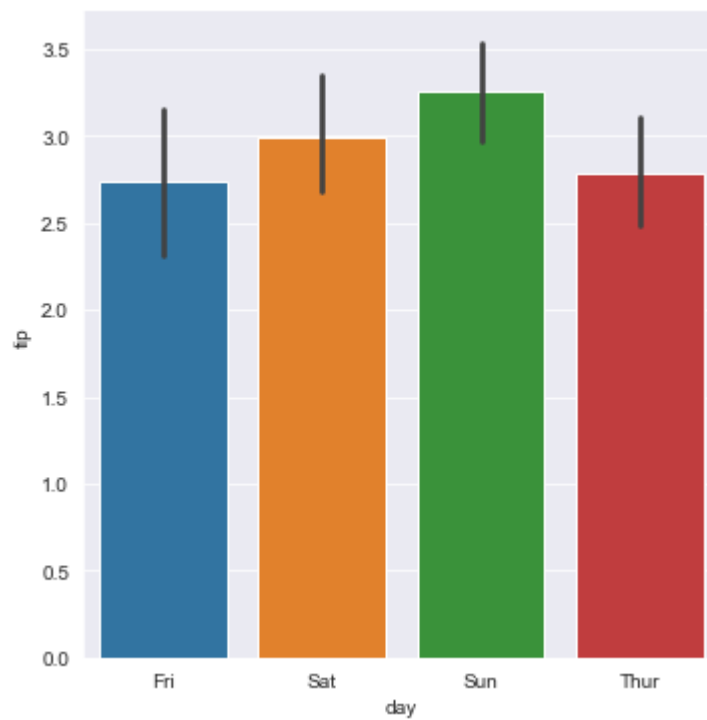
Gambar 5. 17 Hari Tip Besar

```
data.pivot_table(index='day',
                  values='tip',
                  aggfunc='mean').round(2)
```

tip	
day	
Fri	2.73
Sat	2.99
Sun	3.26
Thur	2.78

Gambar 5. 18 Pivot Data Tip

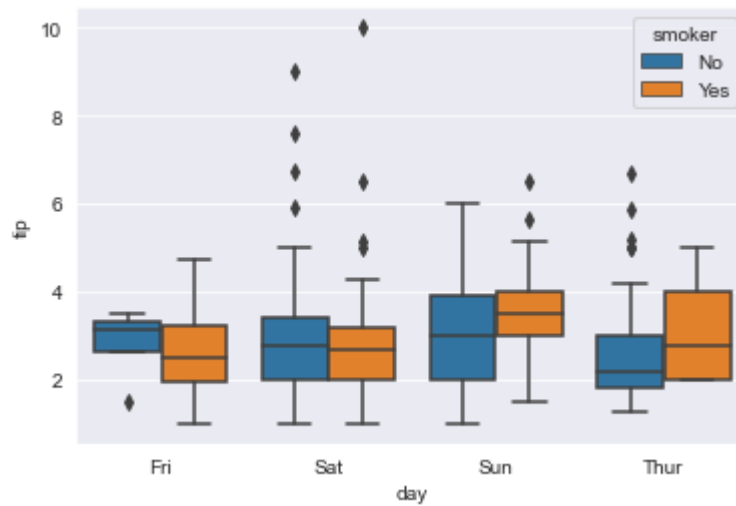
```
sns.factorplot(x='day', y='tip', data=data, kind='bar')
```



Gambar 5. 19 Hari Tip Besar 1

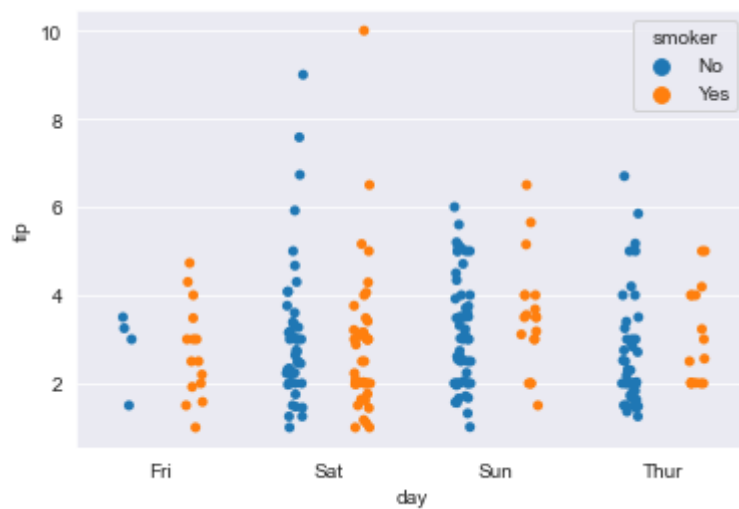
Dari data yang ada apakah customer perokok cenderung memberi tips lebih besar?

```
sns.boxplot(x='day', y='tip', data=data, hue='smoker')
```



Gambar 5. 20 Tip Perokok

```
sns.stripplot(x='day', y='tip', data=data,
              jitter=True, hue='smoker', dodge=True)
```



Gambar 5. 21 Hari Tip Perokok

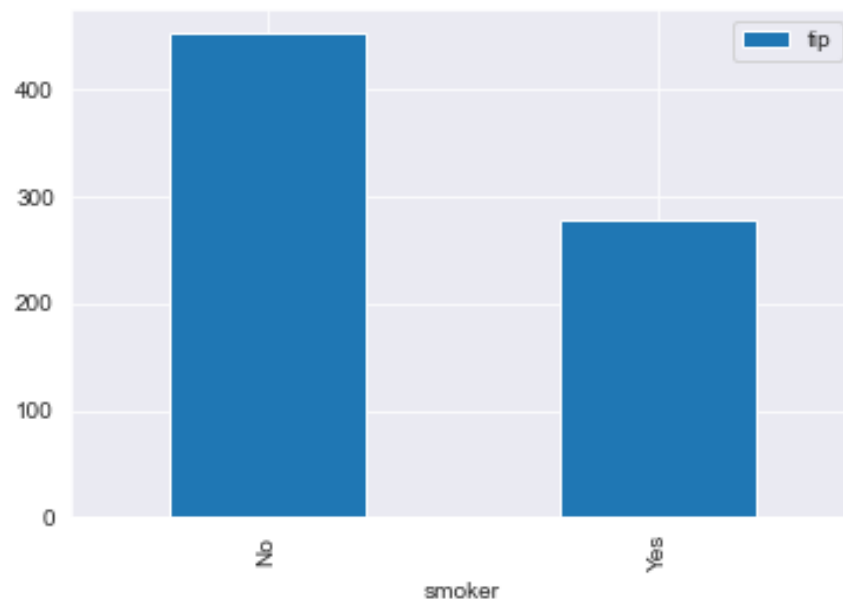
```
perokok = pd.pivot_table(data,
                          index=['smoker'],
                          values=['tip'],
                          aggfunc='sum').round(2)
```

```
perokok
```

tip	
smoker	
No	451.77
Yes	277.81

Gambar 5. 22 Pivot Data Tip 1

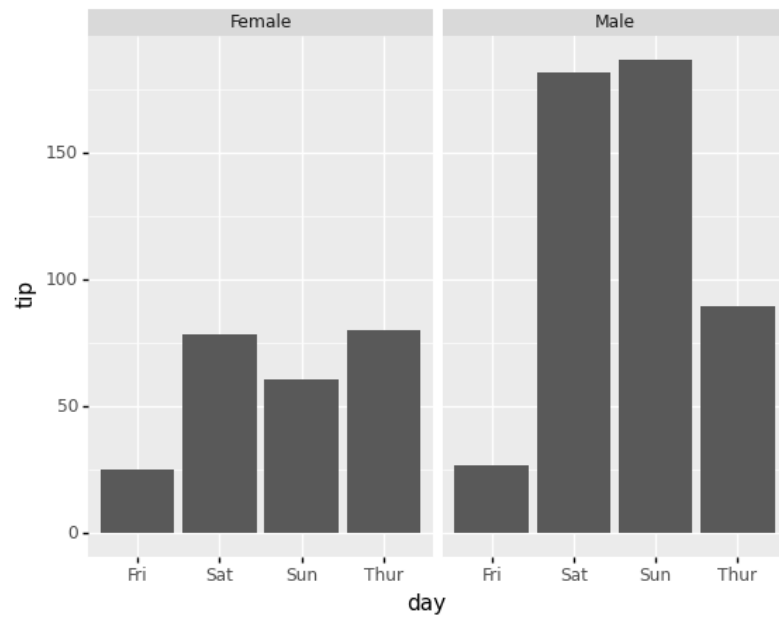
```
perokok.plot(kind='bar')
```



Gambar 5. 23 Tip Perokok 1

Apakah pola tip bagi pria dan wanita yang dipengaruhi hari?

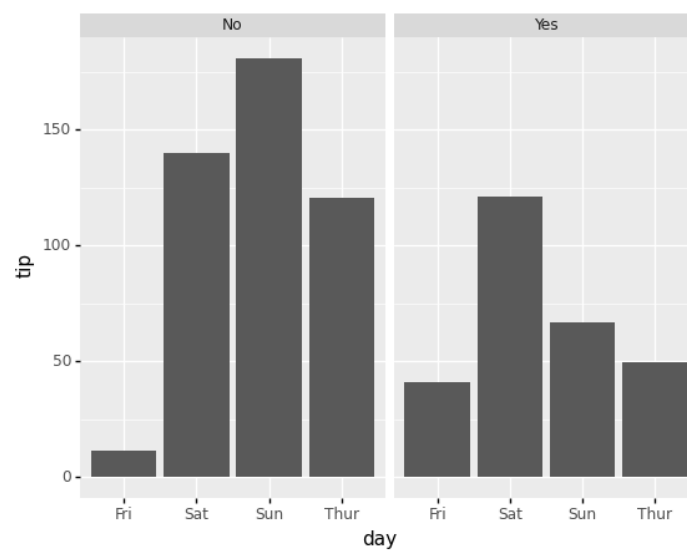
```
(  
  ggplot(data)  
  + facet_grid(facets="~sex")  
  + aes(x="day", y="tip")  
  + geom_col()  
)
```



Gambar 5. 24 Tip Jenkel

Apakah pola tip bagi perokok yang dipengaruhi hari

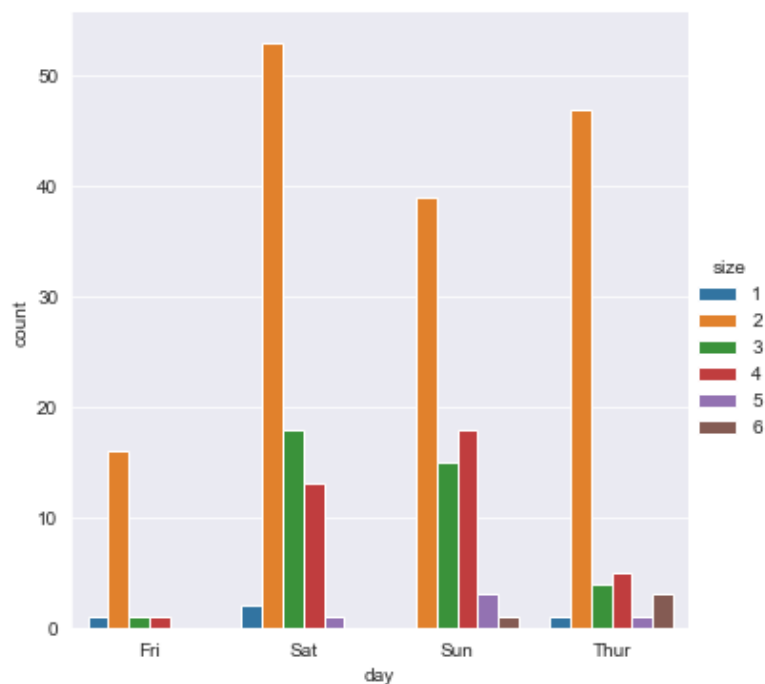
```
(
  ggplot(data)
  + facet_grid(facets="~smoker")
  + aes(x="day", y="tip")
  + geom_col()
)
```



Gambar 5. 25 Tip Perokok

bisakah anda menyarankan tata letak meja restaurant dari data ini?

```
sns.catplot(x="day", hue="size",  
            kind="count", data=data)
```



Gambar 5. 26 Data Tip Meja

5.5 Penarikan Kesimpulan dan Informasi

Setelah proses pengolahan data dari awal hingga akhir, kita mencoba menggali informasi dari data yang kita miliki. Dalam bisnis informasi ini berguna bagi pengembangan bisnis tersebut. Dari data yang masih sulit dipahami dimana masih berupa baris dan kolom, menjadi data yang menghasilkan informasi dan visual agar mudah dipahami. Jika data tidak diolah sehingga tidak menghasilkan informasi apapun akan menjadikan data yang tidak berguna.

Dari data restoran kita dapat informasi seperti menambah jumlah pegawai restoran di hari Sabtu dan Minggu karena pada hari tersebut pengunjung ramai otomatis juga pesanan bertambah banyak, jadi dapat mengatur jadwal serta jumlah pegawai di hari yang ramai dan kurang ramai.

Menambah meja dengan kapasitas 2 dan mengurangi meja dengan kapasitas 1, 5 dan 6 karena lebih banyak customer yang menggunakan meja berdua, dari sini kita dapat mengatur ulang berapa kapasitas meja yang lebih dibutuhkan, khususnya dengan ukuran meja 1, 5, dan 6 jarang digunakan sehingga kita dapat mengurangi jumlah meja dan menggantinya menjadi meja dengan kapasitas 2. Kita juga dapat membagi karyawan dengan perbandingan kerja malam yang lebih banyak, karena pelanggan lebih banyak datang berkunjung pada waktu makan malam. Kita juga dapat mengetahui bahwa apa yang sudah dilakukan sesuai dengan keinginan pelanggan yakni membuat ruangan khusus merokok agar pelanggan yang bukan perokok tidak terganggu dengan pelanggan perokok.

Prediksi Harga Rumah

6.1 Pemahaman Bisnis

Sebelumnya kita telah mencoba untuk menganalisa data tip dan menemukan informasi yang saya harap dapat bermanfaat. Selanjutnya kita akan mencoba untuk prediksi harga rumah. Tepatnya menyangkut harga perumahan di kota Boston, Amerika Serikat. Prediksi harga rumah sering digunakan sebagai contoh pembelajaran Model Regresi pada Pembelajaran Mesin (Machine Learning). Mungkin kita bertanya apa itu prediksi? Nah prediksi adalah proses dalam memperkirakan secara sistematis tentang sesuatu yang paling mungkin terjadi berdasarkan informasi yang telah diberikan.

6.2 Pemahaman Data

Data sebelumnya yang kita gunakan didapatkan dari Kaggle kali ini kita akan mencoba dengan data yang sudah disediakan library sklearn datasets yakni data boston. Data boston menyangkut harga rumah perumahan di kota Boston. Dataset yang disediakan ini memiliki 506 baris dan 13 kolom dan 1 target. Berikut deskripsi dari dataset

- CRIM : Tingkat kejahatan per capita di kota tersebut
- ZN : Proporsi tanah perumahan
- INDUS : Proporsi hektar bisnis non-ritel per kota
- CHAS : Variabel dummy sungai charles (jika saluran membatasi sungai = 1, sebaliknya = 0)
- NOX : Konsentrasi oksida nitrat (per 10 juta bagian)
- RM : Rata-rata jumlah kamar per hunian

- Age : Proporsi unit yang ditempati pemilik yang dibangun sebelum tahun 1940
- DIS : Jarak ke lima pusat kerja boston
- RAD : Indeks akses ke jalan raya radial
- TAX : Tarif dari pajak properti dengan nilai penuh \$10.000
- PTRATIO : Rasio murid-guru menurut kota
- B : $1000(B_k 0,36)^2$, dimana B_k ialah proporsi kulit hitam menurut kota
- LSTAT : Persentasi status pada populasi
- MEDV : Nilai tengah rumah yang ditempati oleh pemilik di \$1000

6.3 Persiapan Data

Pertama kita import library yang dibutuhkan serta membaca data dari library serta menjadikanya menjadi dataframe, feature names merupakan fitur dari data sedangkan targetnya yakni price.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import datasets
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

omah = datasets.load_boston()
omah.data.shape
omah.feature_names
bstn = pd.DataFrame(omah.data)
bstn.columns = omah.feature_names
bstn.head()
```

```

omah.data.shape

(506, 13)

omah.feature_names

array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
      'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')

bstn = pd.DataFrame(omah.data)
bstn.columns = omah.feature_names

bstn.head()

```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

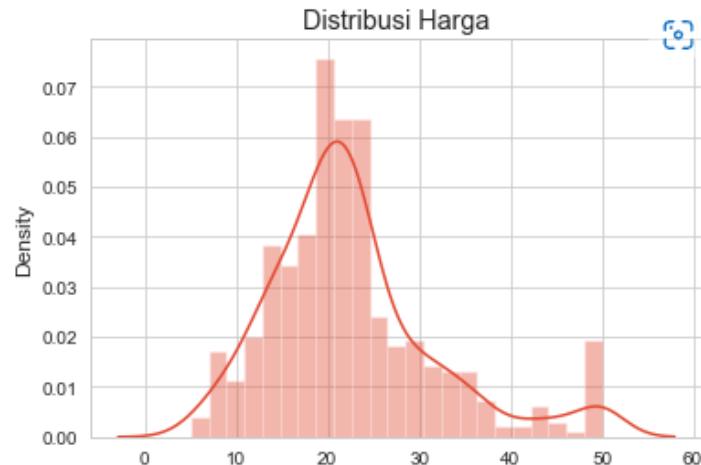
Gambar 6. 1 Data Fitur Boston

Kita coba menampilkan distribusi dari kolom harga yakni yang berperan sebagai target.

```

sns.set_style('whitegrid')
harga = sns.distplot(omah.target)
harga.set(title = "Distribusi Harga")

```



Gambar 6. 2 Distribusi Boston

Kita dapat melihat distribusi kolom target sedikit miring ke kanan dengan mean 22-23 dan standar deviasi 9-10. Tampaknya ada beberapa outlier di ujung yang lebih tinggi dari distribusi harga. Kita tambahkan kolom 'price' ke dalam data bstn.

```

omah.target.shape

bstn['Price'] = omah.target

```

```
bstn.head()
```

```
bstn.shape
```

```
omah.target.shape
```

```
(506,)
```

```
bstn['Price'] = omah.target  
bstn.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

```
bstn.shape
```

```
(506, 14)
```

Gambar 6. 3 Data Boston

Kita akan mencoba melihat apakah ada data yang kosong, jumlah dari data yang unik dari setiap kolom, serta informasi dari data.

```
bstn.isnull().sum()
```

```
bstn.nunique()
```

```
bstn.info()
```

```
bstn.isnull().sum()
```

CRIM	0
ZN	0
INDUS	0
CHAS	0
NOX	0
RM	0
AGE	0
DIS	0
RAD	0
TAX	0
PTRATIO	0
B	0
LSTAT	0
Price	0
dtype:	int64

```
bstn.nunique()
```

CRIM	504
ZN	26
INDUS	76
CHAS	2
NOX	81
RM	446
AGE	356
DIS	412
RAD	9
TAX	66
PTRATIO	46
B	357
LSTAT	455
Price	229
dtype:	int64

```
bstn.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 506 entries, 0 to 505  
Data columns (total 14 columns):  
#   Column      Non-Null Count  Dtype  
---  -  
0   CRIM        506 non-null    float64  
1   ZN          506 non-null    float64  
2   INDUS       506 non-null    float64  
3   CHAS        506 non-null    float64  
4   NOX         506 non-null    float64  
5   RM          506 non-null    float64  
6   AGE         506 non-null    float64  
7   DIS         506 non-null    float64  
8   RAD         506 non-null    float64  
9   TAX         506 non-null    float64  
10  PTRATIO     506 non-null    float64  
11  B           506 non-null    float64  
12  LSTAT       506 non-null    float64  
13  Price       506 non-null    float64  
dtypes: float64(14)  
memory usage: 55.5 KB
```

Gambar 6. 4 Persiapan Data Boston

Berikut deskripsi atau statistika dasar dari data

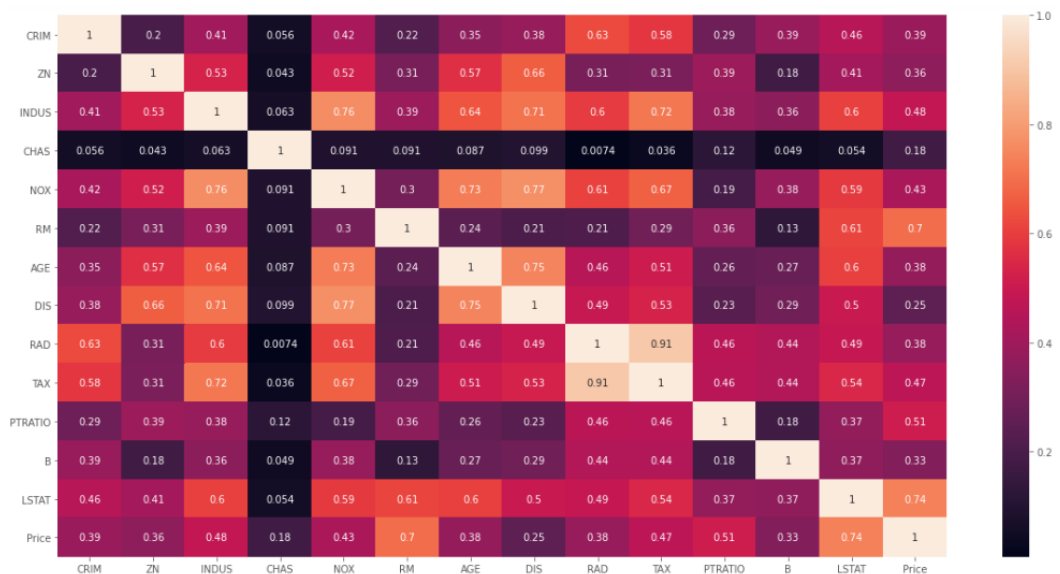
```
bstn.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000

Gambar 6. 5 Deskripsi Data Boston

Dapat disimpulkan dari deskripsi di atas bahwa kita memiliki 13 variabel independen dan satu variabel dependen (harga rumah). Sekarang kita perlu memeriksa korelasi antara variabel independen dan dependen. Kita bisa menggunakan scatterplot/corrplot untuk ini.

```
plt.figure(figsize=(20, 10))
sns.heatmap(bstn.corr().abs(), annot=True)
```

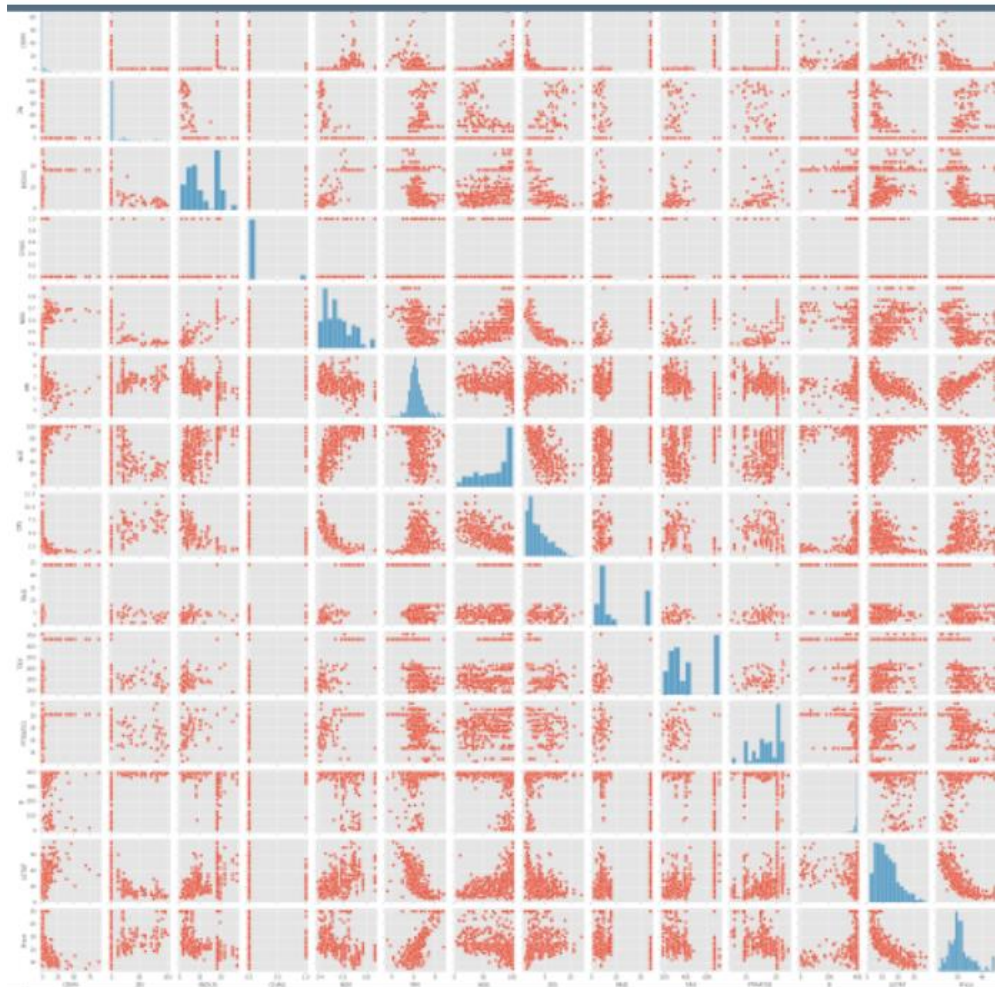


Gambar 6. 6 Korelasi Data Boston

6.4 Eksplorasi dan Penyajian Data

Kita akan menampilkan visualisasi pairplot keseluruhan data

```
sns.pairplot(bstn, size=2)
```



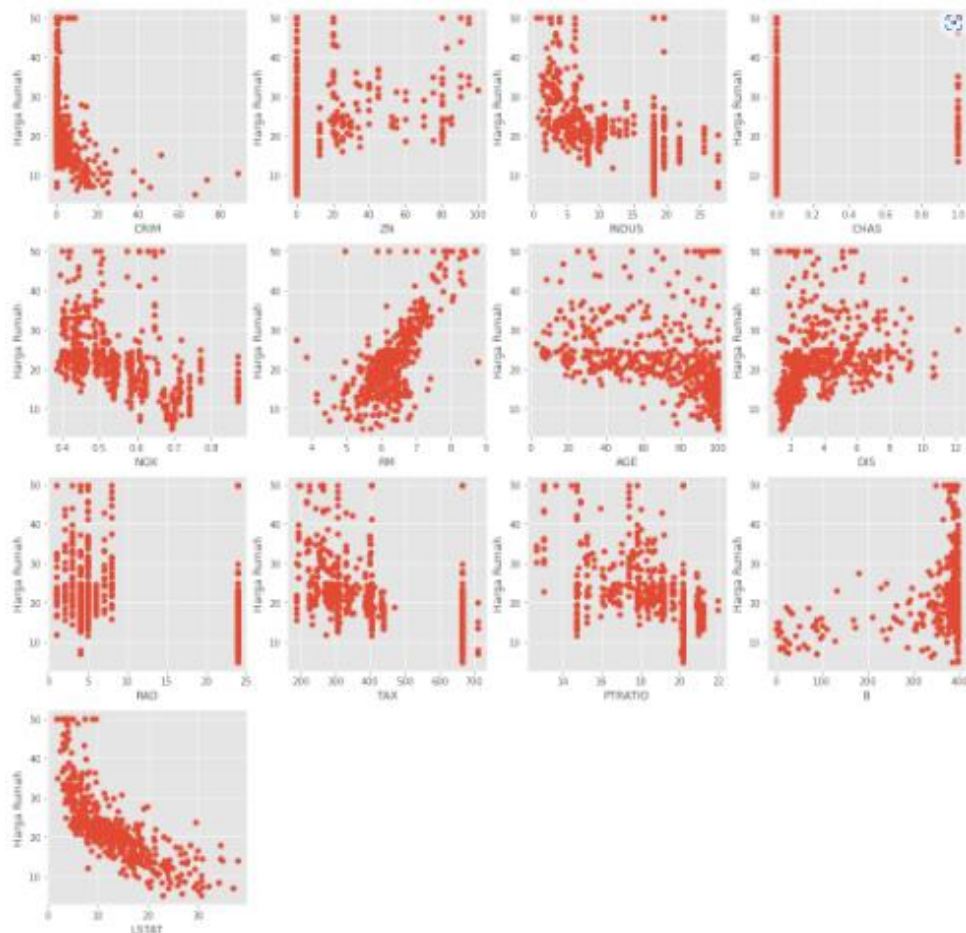
Gambar 6. 7 Pairplot Boston

Sekarang kita akan menampilkan plot sebar variable independent dengan dependen

```
plt.style.use('ggplot')
fig = plt.figure(figsize = (18, 18))

for index, feature_name in enumerate(omah.feature_names):
    ax = fig.add_subplot(4, 4, index + 1)
    ax.scatter(omah.data[:, index], omah.target)
    ax.set_ylabel('Harga Rumah', size = 12)
    ax.set_xlabel(feature_name, size = 12)

plt.show()
```



Gambar 6. 8 Fitur Price

6.5 Pemodelan Regresi

Pembelajaran Mesin (Machine Learning) merupakan ilmu yang mempelajari tentang algoritma dan pemrograman komputer agar dapat belajar dari data. Penggunaan Machine Learning seperti pengenalan pola ataupun identifikasi email spam atau tidak. Regresi linear termasuk model yang paling sederhana di dalam algoritma machine learning. Prediksi nilai real value berdasarkan fitur atau atribut dari data pelatihan. Model yang dibuat berbentuk linear. Dalam machine learning model regresi linear merupakan bagian dari supervised learning (terawasi) dimana memiliki target berupa real value. Contohnya seperti memprediksi harga rumah.

Regresi adalah suatu permasalahan yang outputnya berupa real value. Dimana training dan testing berupa fitur dan target berupa real value. Misal prediksi harga rumah fitur misal area, jumlah kamar, dan sebagainya sedangkan target berupa harga rumah tersebut. Model Regresi Linear akan

memetakan satu atribut (univariate) atau beberapa atribut (multivariate) menjadi satu nilai output (target) berupa nilai real.

Model

Univariate Linear Regression

$$f : \mathbb{R}^1 \rightarrow \mathbb{R} \quad f(x; w) = w_0 + w_1 x$$

$$w = [w_0 \quad w_1] \quad X = \begin{bmatrix} 1 \\ x \end{bmatrix}$$

Multivariate Linear Regression

$$f : \mathbb{R}^d \rightarrow \mathbb{R} \quad f(x; w) = w_0 + w_1 x_1 + \dots + w_d x_d$$

$$w = [w_0 \quad w_1 \dots w_d] \quad X = \begin{bmatrix} 1 \\ x_1 \\ \dots \\ x_d \end{bmatrix}$$

Training

$$w_j = w_j - \eta(f(x^j) - y^j)x_j^j$$

Gambar 6. 9 Regresi

Sekarang, kita menerapkan train-test split untuk membagi dataset menjadi dua bagian, satu untuk pelatihan dan satu lagi untuk pengujian. Kita akan menggunakan 30% data untuk pengujian.

```
x = omah.data
y = omah.target

x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.3, random_state=0)

print("x_train, x_test shape : ", x_train.shape, x_test.shape)
print("y_train, y_test shape : ", y_train.shape, y_test.shape)
```

```
x_train, x_test shape : (354, 13) (152, 13)
y_train, y_test shape : (354,) (152,)
```

Gambar 6. 10 Hasil Split Boston

Sekarang kita mulai menggunakan model regresi linear sekaligus melatih dan memprediksi data. Dari gambar 7.9 diketahui bahwasanya w_0 = intercept/bias, w_1 = koefisien

```

regressor = LinearRegression()
regressor.fit(x_train, y_train)

y_pred = regressor.predict(x_test)
regressor.score(x_train, y_train)
coef = regressor.coef_
coef
intercept = regressor.intercept_
intercept

```

```

regressor = LinearRegression()
regressor.fit(x_train, y_train)

y_pred = regressor.predict(x_test)

regressor.score(x_train, y_train)

0.7645451026942549

coef = regressor.coef_
coef
array([-1.21310401e-01,  4.44664254e-02,  1.13416945e-02,  2.51124642e+00,
        -1.62312529e+01,  3.85906801e+00, -9.98516565e-03, -1.50026956e+00,
         2.42143466e-01, -1.10716124e-02, -1.01775264e+00,  6.81446545e-03,
        -4.86738066e-01])

intercept = regressor.intercept_
intercept

37.937107741832804

```

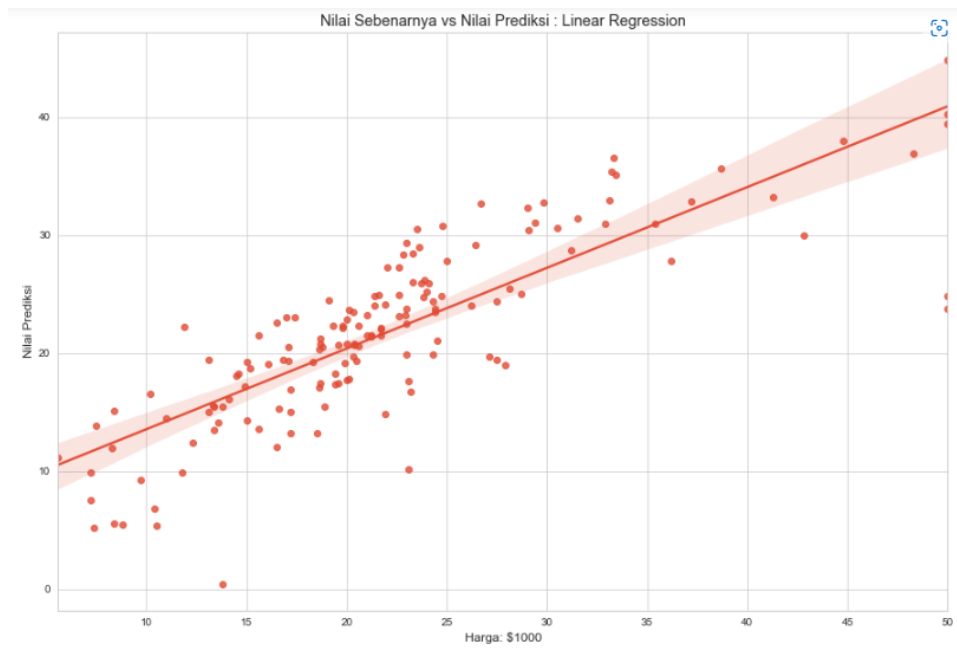
Gambar 6. 11 Regresi Linear Boston

Sekarang kita coba visualisasikan nilai sebenarnya dengan nilai prediksi

```

plt.figure(figsize=(15, 10))
sns.regplot(x=y_test, y=y_pred, fit_reg=True)
plt.xlabel("Harga: $1000")
plt.ylabel("Nilai Prediksi")
plt.title("Nilai Sebenarnya vs Nilai Prediksi : Linear
Regression")
plt.show()

```

Gambar 6. 12 Regresi Plot

Daftar Pustaka

- [1]. Morgan, Peters. 2016. Data Analysis from Scratch with Python. AI Sciences LLC.
- [2]. Raman, Kirthi. 2015. Mastering Python Data Visualization. Mumbai: Pact Publishing Ltd.
- [3]. Li, Rongpeng. 2020. Essential Statistic for Non-STEM Data Analyst. Mumbai: Pact Publishing Ltd.
- [4]. Bruce, Peter. Bruce, Andrew. Gedeck, Peter. 2020. Practical Statistics for Data Scientists. Sebastopol: O'Reilly Media, Inc.
- [5]. W3Schools. 1999-2022. Programming. Data Analytics. w3shools.com.
- [6]. Schmuller, Joseph, PhD. 2017. Statistical Anlysis with R. Hoboken: John Wiley & Sons, Inc.
- [7]. Bakti. 2019. Bahasa Pemrograman Python : Pengertian, Sejarah, Kelebihan dan Kekurangannya. baktikominfo.com.
- [8]. Wiley. 2015. Storytelling with Data. Hoboken: John Wiley & Sons, Inc.
- [9]. McKinney, Wes. 2018. Python for Data Analysis. Sebastopol: O'Reilly Media, Inc.