Startup Campus

Kampus Merdeka
INDONESIA JAYA

FOUNDATIONS FOR DATA SCIENCE

MODULE

DATA SCIENCE TRACK

Table of Contents

Tujuan dan Deskripsi Pembelajaran

Deskripsi Pembelajaran
Merupakan pembelajaran awal bagi para peserta untuk memperkenalkan gambaran umum tentang *data science* serta kaitannya dengan industri. Peserta akan diberikan bahan ajar berupa buku panduan, *live session* , menonton video, serta berbagai macam referensi lain nya. Selanjutnya peserta akan diberikan tugas yang nantinya akan dibahas secara sinkronus oleh mentor pada sesi mentoring. Peserta akan belajar mengenai cara menggunakan aplikasi pendukung/*tools* dalam *Data Science.* Pada materi ini peserta akan mendapatkan pembelajaran mandiri melalui beberapa media seperti YouTube maupun modul dan akan dibantu nantinya dengan pelatihan online bersama expert. Kemudian peserta akan belajar mengenai pengubahan dan penggabungan bingkai data baik melalui pembelajaran mandiri, video pembelajaran maupun *Live Session.*
Tujuan Pembelajaran
Adapun tujuan pembelajaran topik ini adalah sebagai berikut. • Peserta memahami informasi umum tentang ilmu data, beberapa teori dasar, dan pentingnya ilmu data di berbagai industri • Peserta memiliki kemampuan berpikir komputasional dan mampu menerapkannya untuk mendekati suatu masalah • Peserta memiliki pemahaman tentang Machine Learning Foundations yang dapat mereka gunakan dalam materi lebih lanjut mengenai Data Science • Peserta memiliki pemahaman tentang Python Foundations (Python MySQL and Python PostgreSQL) • Peserta memiliki kemampuan berpikir statistik dan mampu menerapkannya untuk mendekati suatu masalah • Peserta dapat memahami data dalam konteks bisnis • Peserta memiliki pemahaman mengenai Statistical Foundations yang dapat mereka gunakan dalam materi lebih lanjut mengenai Data Science • Peserta memiliki kemampuan design thinking dan mampu menerapkannya untuk mendekati suatu masalah • Peserta memiliki kemampuan kepemimpinan digital dan mampu menerapkannya dalam kehidupan nyata

Silabus

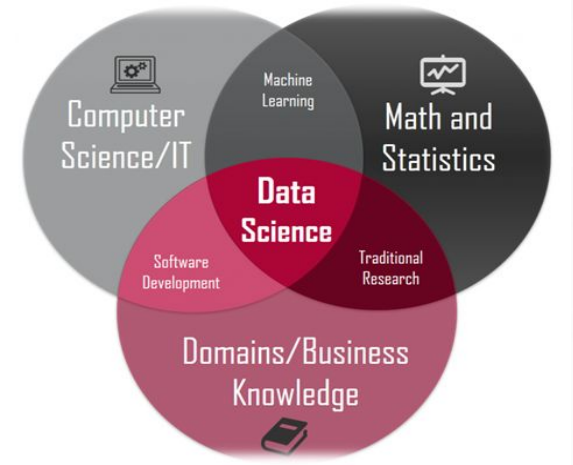
Silabus
Adapun materi dan sub materi yang akan dipelajari peserta ialah sebagai berikut: **1.1 Introduction to Data Science** Pengantar umum, teori dasar, pentingnya ilmu data dalam industri **1.2 Computational Thinking** Memberikan pendekatan pemecahan masalah yang terintegrasi melalui berbagai aktivitas komputasi 1.2.1 Python Foundations - Libraries: Pandas, NumPy, Arrays and Matrix handling, Data Discovery Pengenalan dasar Python, cara mulai menggunakan Python, berbagai library yang akan digunakan dalam program. Tidak terbatas pada pengantar Python dasar (tipe data, operasi, dll), penggabungan data dengan Python, transformasi data dengan Python) 1.2.2 Python MySQL and Python PostgreSQL Memberikan pengetahuan desain penelitian yang cermat untuk mengumpulkan data yang bermakna untuk menjawab pertanyaan penelitian yang terfokus, analisis rinci pola dalam data, dan menarik kesimpulan yang melampaui data yang diamati kepada peserta **1.3 Statistical Thinking** Memperkenalkan konsep pemikiran statistik untuk memecahkan masalah ilmu data 1.3.1 Understanding Data and Business Memberikan siswa konteks data dalam proses bisnis dan memahami peran data dalam proses bisnis 1.3.2Statistics Foundations: Basic/Descriptive Statistics, Distributions (Binomial, Poisson, etc.), Bayes, Inferential Statistics Memberikan mahasiswa dasar statistika yang akan digunakan dalam proses ilmu data **1.4 Design Thinking** Memperkenalkan konsep Design Thinking **1.5 Digital Leadership** Memperkenalkan konsep Digital Leadership
Durasi Pembelajaran
110 jam

Referensi

No	SubTopik	Referensi
1	Intro Data Science & Getting Data Ready - Intro Data Science [1]	Live Session: https://drive.google.com/file/d/1cXFZC_uFh_Wsb4XSWBoe3OShi7uZmTUm/view?usp=sharing Materi: https://drive.google.com/file/d/1tNlWgjarC8N2-3-GUY6VJOgYZhsXpNOe/view
2	Intro Data Science & Getting Data Ready - Intro Data Science [2]	Live Session: https://drive.google.com/file/d/1UtYn_FG8Gvfjb_tt2l5CzDnKmLntaqeW/view?usp=sharing Materi: https://drive.google.com/file/d/1pWA7PyY4pX0nAIIh8OkyErsKbfZkT9lc/view?usp=drive_web&authuser=0
3	What is Computational Thinking	https://digitalpromise.org/initiative/computational-thinking/computational-thinking-for-next-generation-science/what-is-computational-thinking/
4	Computational Thinking Defined	https://towardsdatascience.com/computational-thinking-defined-7806ffc70f5en
5	Developing Computational Thinking on AI and Big Data Era for Digital Society	https://www.apec.org/docs/default-source/Publications/2021/3/Developing-Computational-Thinking-on-AI-and-Big-Data-Era/221_HRD_Developing-Computational-Thinking-on-AI-and-Big-Data-Era.pdf
6	MySQL in Python Tutorial: Getting Started	https://www.datacamp.com/tutorial/mysql-python

Introduction to Data Science

Data Mining vs Data Science



Data Science adalah sebuah ilmu yang lahir dari gabungan antara ilmu statistik dan ilmu komputer. Dengan kata lain, ini adalah ilmu mengekstraksi pola yang berguna dari kumpulan data dengan menggunakan kekuatan komputer.

Ketika organisasi dan bisnis mulai menyadari bahwa ada nilai yang tersembunyi dalam sejumlah besar data yang mereka tangkap secara teratur, mereka mencoba menggunakan teknik yang berbeda untuk mewujudkan nilai tersebut. Sementara tujuan utamanya adalah untuk menghasilkan gagasan yang dapat ditindaklanjuti dari data tersebut, dunia teknologi dipenuhi dengan banyak istilah teknis. Dan di antara semua istilah ini, mungkin istilah yang paling banyak dibicarakan adalah *Data Science* dan *Data Mining*. Meskipun beberapa orang menggunakannya secara bergantian, mereka memiliki perbedaan yang signifikan. Inilah tujuh perbedaan paling menonjol antara *Data Science* dan *Data Mining*.

Introduction to Data Science

Data Mining vs Data Science

- Perbedaan terbesar antara *Data Science* dan *Data Mining* terletak pada istilahnya. *Data Science* adalah bidang yang luas yang mencakup proses menangkap data, menganalisis, dan memperoleh wawasan darinya. Di sisi lain, *Data Mining* mengenai bagaimana menemukan informasi yang berguna dalam kumpulan data dan memanfaatkan informasi itu untuk mengungkap pola tersembunyi.
- Perbedaan utama lainnya antara *Data Science* dan *Data Mining* adalah bahwa *Data Science* adalah bidang multidisiplin yang terdiri dari statistik, ilmu sosial, visualisasi data, pemrosesan bahasa alami, Data Mining, dll, sedangkan *Data Mining* adalah bagian dari *Data Science*.
- Peran seorang profesional *Data Science* dapat dianggap sebagai kombinasi dari peneliti AI, *deep learning engineer*, *machine learning engineer*, atau *data analyst*. Orang tersebut mungkin dapat melakukan peran sebagai *data engineer* juga. Sebaliknya, seorang profesional *data mining* tidak harus mampu melakukan semua peran ini.

Introduction to Data Science

Data Mining vs Data Science

- Perbedaan penting lainnya antara *Data Science* dan *Data Mining* terletak pada jenis data yang digunakan. Biasanya, *Data Science* berurusan dengan setiap jenis data baik terstruktur, semi-terstruktur, atau tidak terstruktur. Di sisi lain, *Data Mining* sebagian besar berkaitan dengan data terstruktur.
- Jika kamu mempertimbangkan sifat pekerjaan di lapangan, Anda akan dapat menemukan perbedaan lain. Dalam *Data Science*, kamu tidak hanya menemukan pola dan menganalisisnya yang merupakan komponen kunci dari *Data Mining*. Sebaliknya, dengan bantuan alat dan teknologi *Data Science*, kamu diharapkan dapat memperkirakan peristiwa masa depan dengan memanfaatkan data saat ini dan data historis.
- Istilah *Data Science* telah ada sejak lama (sejak tahun 1960-an), sedangkan *Data Mining* mulai dikenal pada tahun 1990-an, di antara komunitas basis data.
- Sementara bidang *Data Science* berkonsentrasi pada *Data Science*, bidang *data mining* terutama berkaitan dengan proses.

Introduction to Data Science

Data Mining vs Data Science

Dalam menangani jumlah data yang terus meningkat, baik *data science* maupun *data mining* memainkan peran penting dalam membantu bisnis dalam mengidentifikasi peluang dan membuat keputusan yang efektif. Sehingga, meskipun tujuan kedua bidang ini tetap sama (untuk memperoleh gagasan yang dapat membantu bisnis berkembang), perbedaan utama terletak pada alat dan teknologi yang digunakan, sifat pekerjaan, dan langkah-langkah untuk melakukan tanggung jawab masing-masing untuk mencapai tujuan tersebut.

Data Science merupakan ilmu yang penting karena perusahaan besar dan kecil sangat membutuhkan peran data. Jika Perusahaan tidak dapat mengolah data, Perusahaan tidak membantu dalam menjalankan bisnis yang efektif dan efisien untuk mendapatkan keuntungan.

Saat ini, organisasi di seluruh dunia semakin menyadari pentingnya *Data Science, Artificial Intelligence,* dan *Machine Learning*. Jika perusahaan ingin tetap kompetitif dan relevan, mereka harus dapat menerapkan *Data Science.*

Introduction to Data Science

Data Science dalam Industri

Banyak sekali kegunaan *Data Science* dalam berbagai macam posisi dalam industri. Beberapa diantaranya adalah:

- *Marketing and Sales*
  Penggunaan *Data Science* dalam pemasaran dan penjualan adalah rekomendasi produk. Saat kita memeriksa produk di *E-commerce* dan mereka memberitahu kita bahwa ada produk lain yang mungkin saja kita sukai, kerja algoritma yang memberikan rekomendasi produk tersebut menganggap kita menyukai produk tersebut berdasarkan apa yang sebenarnya dibeli oleh pelanggan lain yang juga melihat produk tersebut.
- *Finance*
  Pencegahan penipuan adalah salah satu tugas bank yang paling vital. Dengan menerapkan algoritma *Data Science* bank dapat mendeteksi penipuan dan menghindari kerugian secara *real time*. Karena sebagian besar aktivitas terkait transaksi perbankan dilakukan oleh nasabah melalui platform digital dan *online*, memastikan keamanan *real-time* dan akurasi transaksi menjadi bagian yang sangat penting.
- Kesehatan
  Salah satu contoh penerapan terbesar *data science* adalah pada sektor industri kesehatan. Bahkan, menurut laman Built In, *Data Science* pertama kali dikenalkan pada dunia lewat industri kesehatan pada tahun 2008. Pada tahun tersebut, Google menemukan bahwa mereka dapat memetakan wabah flu secara *real time* dengan melacak data lokasi pada pencarian terkait flu.

Computational Thinking

Python Foundations - Intro to Python

Python diciptakan pada awal 1990-an oleh Guido van Rossum di Stichting Mathematisch Centrum, Belanda sebagai penerus bahasa yang disebut ABC. Nama Python diambil dari sebuah acara televisi yang bernama Monty Python Flying Circus. Saat ini Python merupakan salah satu bahasa pemrograman yang paling dicari karena kemudahannya dan dapat digunakan untuk berbagai macam hal.

Adapun beberapa hal yang merupakan keunggulan dari Python adalah:

- Serbaguna.
- Memiliki syntax yang simpel mirip seperti bahasa pada umumnya.
- High End Programming (pemrograman tingkat tinggi).
- Open source.
- Bisa digunakan dengan berbagai cara seperti procedural way, object-oriented way, atau function way.

Computational Thinking

Environment untuk Python

Dalam proses pembelajaran di Startup Campus, penggunaan interface yang digunakan dibebaskan sesuai kenyamanan dan kemudahan. Namun ada dua environment yang kami sarankan untuk digunakan oleh peserta yakni a) Google Collaboratory dan b) Jupyter Notebook/Lab

Untuk cara download dan penggunaan dapat diakses disini

Computational Thinking

Dasar-dasar Python - Variable

Dalam Python, kita dapat membuat sebuah variabel dengan cara menugaskan nilai atau assigning value terhadap sebuah element. Setelah kita menentukan nilai dari suatu element maka kita dapat memanggil kembali variabel tersebut jika dibutuhkan. Variable dapat terus diubah dengan cara "meng-assign" nilai baru.

Beberapa aturan dalam pembuatan variable adalah:

1. Nama variabel tidak boleh diawali oleh angka.
2. Nama variabel hanya bisa terdiri dari karakter alphanumeric dan underscore (A-z, 0-9, dan _)
3. Nama variabel bersifat case sensitive. Artinya variabel nama berbeda dengan Nama atau naMA
4. Nama variabel tidak boleh reserved keywords https://realpython.com/lessons/reserved-keywords/

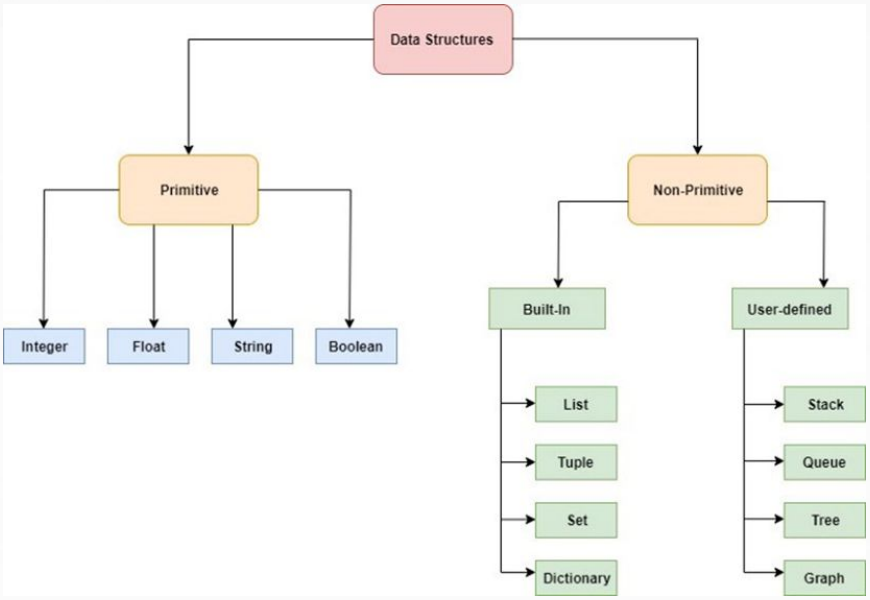
Contoh variable:

```
[ ] variabel_1 = 1

[ ] variabel_1

    1
```

```
[ ] # Menentukan nama variabel lalu mengisikan sebuah nilai
    variabel = "Hello World!"
    print(variabel)

    Hello World!

[ ] nama = "Lingga"

[ ] print("nama saya adalah " + nama)

    nama saya adalah Lingga

[ ] # TO DO: Buat variabel dan isikan nama kalian
    namaku = 'Lingga'
    print("perkenalkan saya: " + namaku)

    perkenalkan saya: Lingga
```

Computational Thinking

Dasar-dasar Python - Struktur Data



Struktur data memberi kita cara khusus dan khusus untuk menyimpan dan mengatur data sedemikian rupa sehingga dapat dengan mudah diakses dan dikerjakan secara efisien. Pada artikel ini, Anda akan belajar tentang berbagai struktur data Python dan bagaimana penerapannya.

Secara umum, struktur data dapat diklasifikasikan menjadi dua jenis - primitif dan non-primitif. Yang pertama adalah cara dasar untuk merepresentasikan data yang berisi nilai-nilai sederhana. Yang terakhir adalah cara yang lebih maju dan kompleks untuk merepresentasikan data yang berisi kumpulan nilai dalam berbagai format.

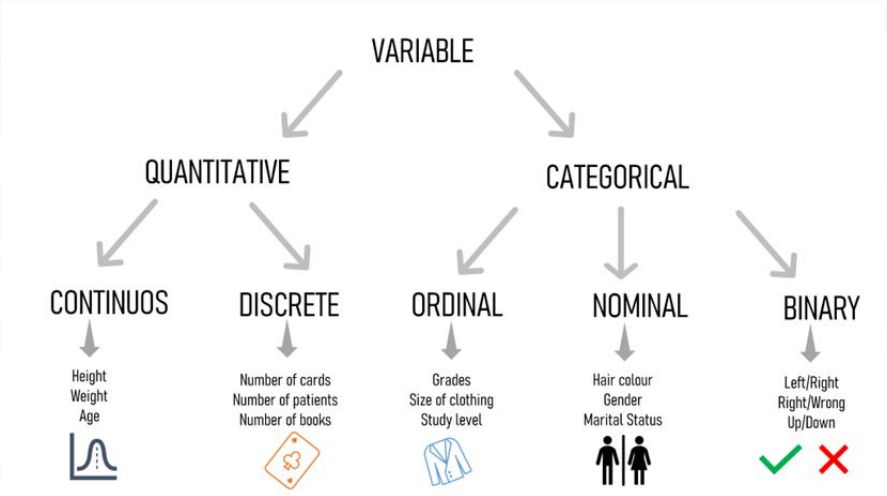
Computational Thinking

Dasar-dasar Python - Struktur Data

Struktur data non primitif selanjutnya dapat dikategorikan ke dalam struktur bawaan dan struktur yang ditentukan pengguna. Python menawarkan dukungan implisit untuk struktur bawaan yang mencakup Daftar, Tuple, Set, dan Kamus. Pengguna juga dapat membuat struktur data mereka sendiri (seperti Stack, Tree, Queue, dll.) yang memungkinkan mereka memiliki kontrol penuh atas fungsinya.

Computational Thinking

Dasar-dasar Python - Tipe Variabel



KUANTITATIF: menyatakan jumlah benda (misalnya jumlah rokok dalam satu bungkus). Dua jenis variabel kuantitatif yang berbeda adalah:

1. CONTINUOUS (a.k.a Ratio): digunakan untuk menggambarkan ukuran dan biasanya dapat dibagi menjadi unit yang lebih kecil dari satu (misalnya 1,50 kg).
2. DISCRETE (a.k.a Interval): digunakan untuk menggambarkan jumlah dan biasanya tidak dapat dibagi menjadi unit yang lebih kecil dari satu (mis. 1 batang rokok).

KATEGORIK: mengungkapkan pengelompokan benda-benda (mis. jenis buah yang berbeda). Tiga jenis variabel kategori adalah:

1. ORDINAL: mewakili data dengan urutan (misalnya peringkat).
2. NOMINAL: mewakili nama kelompok (misalnya merek atau nama spesies).
3. BINER: mewakili data dengan hasil ya/tidak atau 1/0 (mis. KIRI atau KANAN).

Computational Thinking

Dasar-dasar Python - Tipe Data

Tipe Data		Contoh	Keterangan
Text	str (string)	“hello world”	diapit oleh kutip (‘) / (“)
Numeric	int (integer)	1 atau 1680	bilangan bulat tanpa decimal
	float	1.6 atau 18.9	bilangan decimal
Boolean	bool	True atau False	menyatakan benar salah
Sequence	list []	[1,2,3,4,5] atau [‘hasan’,’bamba ng’]	Menggunakan kurung siku
	tuple ()	(1,2,3,4,5)	Menggunakan kurung
	range	range(5)	
Mapping	dict	{‘nama’:’daffa’} , {‘jumlah’:5}	menggunakan kurung kurawal
Set	set	{‘hello’, 3,4,6.6}	
	frozen set		

Computational Thinking

Dasar-dasar Python - Tipe Data

Python mengikuti built-in data types:

1. Text Type: str
2. Numeric Types: int, float, complex
3. Sequence Types: list, tuple, range
4. Mapping Type: dict
5. Set Types: set, frozenset
6. Boolean Type: bool
7. Binary Types: bytes, bytearray, memoryview
8. None Type: NoneType

Cara mencari data type:

```
[ ]  # Int
     x = 1
     print(type(1))
```

```
     <class 'int'>
```

```
[ ]  # float
     y = 1.5
     print(type(y))
```

```
     <class 'float'>
```

Computational Thinking

Dasar-dasar Python - Operasi Python

		Apabila x=4 dan y=2	Hasil
Operator Aritmatika	Penjumlahan	x+y	6
	Pengurangan	x-y	2
	Perkalian	x*y	8
	Pembagian	x/y	2
	Pangkat	x**y	16
	Modulus (sisa)	x//y	0
Operator Pembanding	Sama dengan	x==y	False
	Tidak sama dengan	x!=y	True
	Perbandingan (lebih besar/kecil/etc)	x>y x>= x<y x<=y	True True False False

Computational Thinking

Dasar-dasar Python - Operasi Python

Operator Penugasan		x=x+2	5
Logical operator	if	x = 5 y = 3 if x>y: print(‘benar’)	benar
	AND (kedua syarat harus dipenuhi, maka TRUE)	x = 5 y = 2 z = 1 x>y AND x<z	False
	OR (salah satu syarat dipenuhi maka TRUE)	OR = x < 4 or y < 3 # False, True print(OR)	True

Strings

- Operasi print(“string”); bisa dilakukan dengan single quote (‘) maupun double quote (“), dan bisa digabungkan apabila membutuhkan quote di dalam { print(“you’re awesome”) } dan bisa menggunakan \n untuk memisahkan string ke dalam dua baris

```
print('First line.\n Second line.') # \n
```

```
First line.
 Second line.
```

Computational Thinking

Dasar-dasar Python - String

- Operasi print("string"); bisa dilakukan dengan single quote (') maupun double quote ("), dan bisa digabungkan apabila membutuhkan quote di dalam { print("you're awesome") } dan bisa menggunakan \n untuk memisahkan string ke dalam dua baris

```
print('First line.\n Second line.') # \n
```

```
First line.
 Second line.
```

- menghitung jumlah karakter dalam string dapat menggunakan operasi len("string")

```
#string length

a = "ZulfikarLazuardiMaulana"
print(len(a))
```

```
23
```

Computational Thinking

Dasar-dasar Python - List

List [] adalah tipe data dalam python yang dapat digunakan untuk menyimpan berbagai jenis tipe objek. Menggunakan [] dan , untuk memisahkan objek dalam list. Kita dapat melakukan indexing dan slicing untuk tipe data list. List juga bersifat mutable yang artinya kita bisa menambahkan nilai baru didalam list tersebut dengan menggunakan berbagai operasi python.

List juga mendukung nesting, artinya kita bisa memiliki list di dalam list. contoh list:

```
type([1,2,3])
```

```
list
```

```
data = [1,2,3,4,5]
print(str(data))
```

```
[1, 2, 3, 4, 5]
```

- Dapat menggunakan operator len() untuk mengetahui jumlah objek di dalam list
- Untuk mengakses list dapat menggunakan metode indexing dimana objek pertama pada list memiliki nomor index 0

```
list=["ini", "contoh", "list"]
list[0]
```

```
'ini'
```

Computational Thinking

Dasar-dasar Python - List

- Menambahkan/Mengeluarkan object dalam list
  - .append(item) → menambahkan objek di akhir list
  - .insert(index,item) → menambahkan objek sesuai dengan index
  - .pop(index) mengeluarkan objek pada index tertentu, default -1
  - .remove(item) → mengeluarkan item spesifik pada list
    - Tuple
    - set
    - Dictionary
    - If Else
    - Looping
    - Function

Computational Thinking

Transforming Data: Python

Akses notebook disini

Apa itu Pandas?

Pandas adalah pustaka perangkat lunak yang ditulis untuk bahasa pemrograman Python untuk manipulasi dan analisis data. pandas menyediakan struktur data yang cepat, fleksibel, dan ekspresif yang dirancang untuk membuat pekerjaan dengan data "relasional" atau "berlabel" menjadi mudah dan intuitif.

Here are just a few of the things that pandas does well:

- Easy handling of missing data (represented as `NaN`, `NA`, or `NaT`) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let `Series`, `DataFrame`, etc. automatically align the data for you in computations
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets
- Intuitive merging and joining data sets
- Flexible reshaping and pivoting of data sets
- Hierarchical labeling of axes (possible to have multiple labels per tick)
- Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving/loading data from the ultrafast HDF5 format
- Time series-specific functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging

```
# panggil package
import pandas as pd
```

Computational Thinking

Transforming Data: Python

Apa itu dataframe?
Singkatnya, dataframe adalah sebuah tabel dua dimensional yang berisi data. Terdapat row dan column yang berisikan informasi. Contohnya adalah file excel/csv

Cara import data csv/excel ke environment python:

- Excel : pd.read_excel(file_name.xlxs')
- CSV : pd.read_csv('file_name.csv')

Apabila file yang dituju terdapat di file berbeda dengan file notebook maka harus menuliskan directory secara lengkap.
Warning: jangan salah tipe file ya!

Data kemudian dapat dipanggil kembali dengan cara menetapkan variabel.

```
[ ] df = pd.read_csv('/content/Sample.csv', sep =',')
```

```
[ ] df
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States
2	3	CA-2016-	6/12/2016	6/16/2016	Second	DV-13045	Darrin Van	Corporate	United

Computational Thinking

Transforming Data: Python

Operasi dasar dalam pandas/dataframe.

- **Info mengenai dataframe**
  - .shape dapat digunakan untuk mengetahui dimensi dari dataframe yang ada (jumlah row dan column).
  - .info: dapat digunakan untuk mengetahui informasi dasar tentang dataframe seperti jumlah kolom, nama kolom, tipe kolom, dan null value.
  - .describe: dapat digunakan untuk mengetahui beberapa statistik numerikal dasar mengenai dataframe.
  - .sort_values(): dapat digunakan untuk mengurutkan dataframe berdasarkan kolom dan nilai tertentu.
  - df['nama_kolom'].value_counts():dapat digunakan untuk menjabarkan jumlah dari suatu value pada kolom tertentu.

- **Sampling**
  - .head(x): mengambil x data teratas dari sebuah dataframe, dimana x merupakan jumlah baris yang ingin diambil. apabila x tidak dituliskan default nya 5
  - .tail(x): mengambil x data terbawah dari sebuah dataframe, dimana x merupakan jumlah baris yang ingin diambil. apabila x tidak dituliskan default nya 5
  - sample(x): mengambil x baris data secara acak, dimana x merupakan jumlah baris yang ingin diambil

Computational Thinking

Transforming Data: Python

- **Indexing :** : memanggil beberapa kolom / nilai dalam dataframe
  - df.nama_kolom_1→
  - df[['nama_kolom_1','nama_kolom_2','nama_kolom_3']]→ memanggil kolom yang ditulis saja pada dataframe (filter)
  - df['nama_kolom_1'][x]→ akses value pada kolom 1 di row x

- **iloc :** *index based location*
- **Selecting/Filtering :** df[df[‘nama_kolom_1’] <operator> value] dapat digunakan untuk menyeleksi data sehingga data yang ditampilkan memenuhi syarat tertentu (operator + value) seperti:
  - df['City'] == 'New York City' → akan memunculkan semua kolom di dataframe dengan city value “New York City”
  - df[['City','Category','Customer Name']][(df['City'] == 'New York City') & (df['Category'] == 'Technology')] → akan memunculkan kolom city, category, dan customer name di dataframe yang memiliki value city “new york city” dan category “technology”. *AND(&) disini bisa juga diganti OR(|) atau keduanya, dan akan mengikuti operasi logical.
  - df[df[‘nama_kolom_1’].isnull()] → akan menampilkan row yang memiliki nilai null di sebuah kolom
  - df[df[‘nama_kolom’].str.contains(pola)] → akan menampilkan value dengan pola tertentu. Dapat menambahkan ^ / $ untuk menentukan letak pola

Computational Thinking

Transforming Data: Python

- **Change data type:**
  - .astype(): merubah tipe data float/int/str
  - .str.lower() / .str.upper(): merubah menjadi huruf besar/kecil semua

- **Manipulasi kolom**
  - Menambah kolom baru: dapat dilakukan dengan mendefinisikan sebuah nama kolom baru dengan value dari kolom lainnya.
  - Menghapus kolom: df = df.drop(‘nama_kolom’, axis=1)
  - Menghapus row dengan value duplicate: df[df.duplicated(keep=False) == True]
  - Mengubah nama kolom di suatu dataframe: df = df.rename(columns = { ‘nama_lama’ : ‘nama_baru’ })
  - Mengubah beberapa nama kolom di suatu dataframe: df = df.rename(columns = { ‘nama_lama_1’ : ‘nama_baru_1’,‘nama_lama_2’ : ‘nama_baru_2’, ‘nama_lama_3’ : ‘nama_baru_3’,‘nama_lama_N’ : ‘nama_baru_N’ })
  - Mengubah semua nama kolom: df.columns = [‘name_1’, ‘name_2’, … ‘name_N’]

Computational Thinking

Transforming Data: Python

- **Grouping/aggregation**
  - Mengelompokan dataset berdasarkan satu kolom dan satu statistik: df.groupby(‘nama_kolom’).agg({‘kolom’ : ‘statistics’})
  - Mengelompokan dataset berdasarkan satu kolom dengan beberapa statistik: df.groupby(‘nama_kolom’).agg({‘kolom’ : [‘statistik_1’, ‘statistik_2’, ...] })
  - Mengelompokan dataset berdasarkan beberapa kolom dengan beberapa statistik: df.groupby(‘nama_kolom’).agg({‘nama_kolom_1’ : [‘statistik_1’, ‘statistik_2’, ...],‘nama_kolom_2’ : [‘statistik_1’, ‘statistik_2’, ...] . . . .‘nama_kolom_n’ : [‘statistik_1’, ‘statistik_2’, ...] })
  - Reset index data frame: df = df.reset_index()
  - Pivot Table: pd.pivot_table(df, index = [col_as_rows], columns = [col_as_columns], values =[col_as_values])

- **Menggabungkan dataset**
  - Append: df.append(df_2)
  - Merge: df_1.merge(df_2, left_on = kolom_1,right_on = kolom_2, how = ‘inner’) *inner bisa diganti dengan left/right

- **Exporting dataset to CSV/Excel**
  - df.to_csv(‘nama_file.csv’, index=False)
  - df.to_excel(‘nama_file.xlsx’, index = False)

Computational Thinking

Data Discovery - Definisi

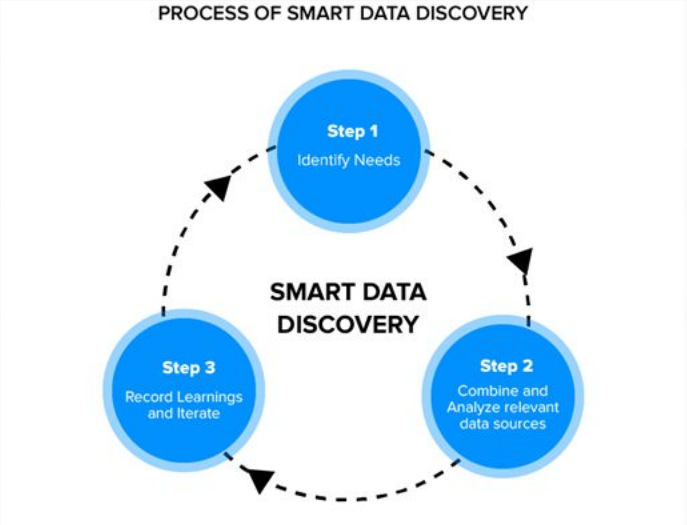
- **Grouping/aggregation**
  - Mengelompokan dataset berdasarkan satu kolom dan satu statistik: df.groupby(‘nama_kolom’).agg({‘kolom’ : ‘statistics’})
  - Mengelompokan dataset berdasarkan satu kolom dengan beberapa statistik: df.groupby(‘nama_kolom’).agg({‘kolom’ : [‘statistik_1’, ‘statistik_2’, ...] })
  - Mengelompokan dataset berdasarkan beberapa kolom dengan beberapa statistik: df.groupby(‘nama_kolom’).agg({‘nama_kolom_1’ : [‘statistik_1’, ‘statistik_2’, ...],‘nama_kolom_2’ : [‘statistik_1’, ‘statistik_2’, ...] . . . .‘nama_kolom_n’ : [‘statistik_1’, ‘statistik_2’, ...] })
  - Reset index data frame: df = df.reset_index()
  - Pivot Table: pd.pivot_table(df, index = [col_as_rows], columns = [col_as_columns], values =[col_as_values])

- **Menggabungkan dataset**
  - Append: df.append(df_2)
  - Merge: df_1.merge(df_2, left_on = kolom_1,right_on = kolom_2, how = ‘inner’) *inner bisa diganti dengan left/right

- **Exporting dataset to CSV/Excel**
  - df.to_csv(‘nama_file.csv’, index=False)
  - df.to_excel(‘nama_file.xlsx’, index = False)

Computational Thinking

Data Discovery - Pengertian



Apa itu Data Discovery?

Data Discovery adalah proses adalah sebuah istilah yang digunakan para analis untuk menggambarkan proses pengumpulan data dari berbagai sumber dengan mendeteksi berbagai pola yang ada. Yang terpenting, Data Discovery tidak memerlukan untuk membangun model yang rumit. Sebagian besar perusahaan menggunakan Data Discovery sebagai bagian dari perangkat lunak intelijen bisnis (BI), yang memberi mereka pandangan lengkap tentang organisasi mereka dalam dasbor sederhana atau format visual

Computational Thinking

Data Discovery - Pengertian

Pendekatan dalam Data Discovery

Dalam proses nya, data discovery dikategorikan menjadi dua jenis pendekatan. Masing-masing pendekatan menjelaskan proses yang berbeda untuk memahami kumpulan data untuk integrasi bisnis dan analisis data. Langkah-langkah melaksanakannya juga harus menggabungkan pencarian teknis dari tools terbaik dengan keahlian materi para pakar dan analis.

Ada dua jenis pendekatan utama dalam data discovery adalah seperti berikut ini.

1. Manual Discovery

   Manual data discovery adalah jenis pendekatan pertama dalam proses Data Discovery. Sederhananya, dalam pendekatan ini, para analis akan melihat jenis data apa yang ada, di mana data tersebut disimpan, dan apa yang perlu diberikan kepada perusahaan Perusahaan akan memantau metadata dan penemuan data lineage untuk mempelajari kategorisasi data.

   Data stewards, atau pengurus data, adalah mereka yang bertanggung jawab untuk menangani aturan serta standar dokumen aset data yang akan melakukan proses data discovery.

   Selanjutnya, dalam pendekatan ini, para Data scientist akan membuat konsep peta agar mereka dapat memahami semua data yang diperoleh suatu perusahaan.

Computational Thinking

Data Discovery - Pengertian

2. Smart Data Discovery

Seiring berkembangnya zaman, data discovery semakin memudahkan para data scientist dalam melakukan penemuan data. Pada akhirnya, proses dalam perusahaan semakin efektif dan efisien. Smart data discovery merupakan sebuah pendekatan baru, di mana proses data scientist akan dibantu oleh proses machine learning dan teknologi lainnya. Analis akan memanfaatkan augmented intelligence untuk mempersiapkan, membuat konsep, mengintegrasikan, dan menyajikan data melalui visual, pola dan wawasan yang tersembunyi.

Pendekatan ini juga mendorong perusahaan untuk mempertimbangkan bahwa seluruh analisis dari kumpulan data. Untuk itu, perusahaan dan analisis akan memanfaatkan mesin untuk menerima pertanyaan, melakukan pemrosesan dalam black box, dan menghasilkan jawaban yang masuk akal.

Computational Thinking

Data Discovery - Mengapa?

Mengapa Data Discovery Penting?

Suatu bisnis yang sukses salah satunya dinilai dari ketangkasan nya. Data Discovery adalah bagian dari dasar ketangkasan bisnis. Data Discovery memberikan perusahaan pandangan menyeluruh tentang operasional bisnis perusahaan sehingga mereka dapat lebih memahami dan menangani masalah apa pun.

Data Discovery terus tumbuh dalam popularitas karena lebih banyak perusahaan memperlakukan data mereka seperti aset. Informasi yang dikumpulkan perusahaan tentang customer dan operasional mereka yang digunakan untuk membedakan model bisnis perusahaan mereka dari pesaing. Data Discovery memungkinkan mereka mengubah bisnis mereka menjadi keunggulan kompetitif, baik dalam bentuk inovasi produk, User experience yang lebih baik, atau peningkatan efisiensi.

Kategori Data Discovery

Data Discovery datang dalam berbagai bentuk, menggabungkan analisis, pemodelan dan keluaran visual. Untuk mendapatkan nilai maksimal dari proses tersebut, bisnis perlu memahami interaksi di antara berbagai aliran data mereka. Dengan bantuan alat penemuan visual dan perangkat lunak intelijen bisnis (BI), tiga kategori Data Discovery berikut dapat membantu perusahaan mendapatkan gambaran besar tentang datanya dalam satu format yang mudah dicerna.

Computational Thinking

Data Discovery - Kategori

1. **Persiapan Data**

   Data Discovery datang dalam berbagai bentuk, menggabungkan analisis, pemodelan dan keluaran visual. Untuk mendapatkan nilai maksimal dari proses tersebut, bisnis perlu memahami interaksi di antara berbagai aliran data mereka. Dengan bantuan alat penemuan visual dan perangkat lunak intelijen bisnis (BI), tiga kategori Data Discovery berikut dapat membantu perusahaan mendapatkan gambaran besar tentang datanya dalam satu format yang mudah dicerna.

2. **Analisis Visual**

   Memvisualisasikan data adalah salah satu cara paling efektif untuk memahami isi yang ada di dalam nya. Baik dalam bentuk bagan, diagram aliran data, atau dashboard, visualisasi data membantu mereka yang tidak terlatih dalam Data Science untuk memahami hubungan di antara perspektif data mereka dengan cara yang terasa efektif. Misalnya, tim User Interface dapat dengan mudah mempelajari bagaimana pelanggan menggunakan produk mereka dan menyesuaikan pekerjaan mereka. Dan tim keuangan dapat memperoleh gambaran tentang biaya pendapatan untuk setiap departemen dalam bisnis dan menunjukkan area yang perlu ditingkatkan.

Computational Thinking

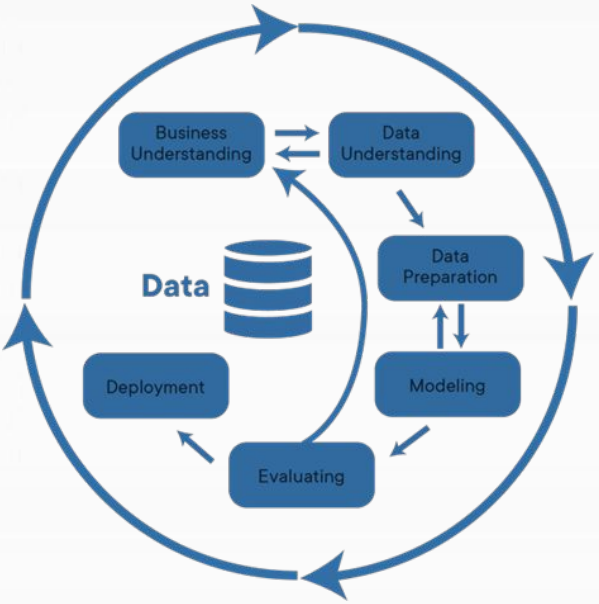
Data Discovery - Kategori

3. Analisis Visual Lanjutan

Analisis lanjutan yang dipandu menggabungkan deskripsi dan visual untuk memberikan gambaran lengkap tentang data perusahaan. Jika keluaran analitik berfokus pada deskripsi kecil dari data itu sendiri, analitik terpandu memungkinkan bisnis untuk melihat implikasi yang lebih besar dari upaya Data Discovery mereka, termasuk hubungan antara perspektif data science yang berbeda. Analisis lanjutan yang dipandu sangat berharga untuk bisnis yang menavigasi peralihan ke digital area, di mana integrasi data web dengan aliran data yang ada sangat penting untuk pengambilan keputusan strategis.

Computational Thinking

Data Discovery - Proses Data Discovery



Ada beberapa langkah dalam Data Discovery Ini juga merupakan proses berulang, yang berarti perusahaan dapat terus mengumpulkan, menganalisis, dan menyempurnakan pendekatan Data Discovery mereka dari waktu ke waktu dengan memanfaatkan hasil dan saran dari pemangku kepentingan bisnis.

Computational Thinking

Data Discovery - Proses Data Discovery

- Langkah 1: Identifikasi kebutuhan. Data Discovery yang efektif dimulai dengan tujuan yang jelas, Ini berarti mempertimbangkan jenis data apa yang akan berguna untuk diketahui, sambil tetap terbuka terhadap wawasan tak terduga Misalnya, distributor barang konsumen yang pada industri FMCG mungkin memutuskan untuk memeriksa kembali data logistiknya dalam upaya mengurangi limbah makanan selama pengiriman. Atau industri perbankan mungkin menganalisis data webnya dengan tujuan untuk menganalisis customer mereka.
- Langkah 2: Gabungkan data dari sumber yang relevan. Agar Data Discovery menjadi efektif, penting untuk menggabungkan dan mengintegrasikan data dari berbagai sumber karena tidak ada aliran data tunggal yang mencakup semua nya. Proses ini kadang-kadang disebut sebagai data crunching.
- Langkah 3: Bersihkan dan siapkan data. Ini adalah bagian penting dari Data Discovery. Membersihkan data dan menyiapkannya untuk analisis membantu perusahaan mengurangi "gangguan" dalam data mereka dan mendapatkan hasil yang lebih jelas dari analisis data mereka.
- Langkah 4: Analisis data. Dengan informasi yang digabungkan dari beberapa departemen, terintegrasi dengan data eksternal dan dibersihkan untuk analisis, perusahaan dapat memperoleh gambaran lengkap tentang operasi mereka dan memecahkan permasalahan operasional yang menghalangi efisiensi.
- Langkah 5: Rekam pembelajaran dan ulangi. Data Discovery bukanlah proses satu kali; itu adalah komitmen untuk perbaikan terus-menerus.

Computational Thinking

Python MySQL - Installing MySQL

Prosedur yang harus diikuti dengan Python untuk bekerja dengan MySQL

1. Hubungkan ke database.
2. Buat objek untuk database Anda.
3. Jalankan query SQL.
4. Ambil catatan dari hasilnya.
5. Menginformasikan Database jika membuat perubahan pada tabel.

MySQL adalah salah satu sistem manajemen basis data (DBMS) yang paling populer di pasaran saat ini. Ini peringkat kedua setelah Oracle DBMS di Peringkat DB-Engine tahun ini. Karena sebagian besar aplikasi perangkat lunak perlu berinteraksi dengan data dalam beberapa bentuk, bahasa pemrograman seperti Python menyediakan alat untuk menyimpan dan mengakses sumber data ini.

1. Installing MySQL

MySQL adalah salah satu database yang paling populer. Unduh dan instal MySQL dari situs web resmi MySQL. Kira perlu menginstal server MySQL untuk mengikuti tutorial ini. Selanjutnya, harus menginstal mysql.connector untuk Python. Kita membutuhkan mysql.connector untuk menghubungkan Python Script ke database MySQL. Unduh mysql.connector dari sini dan instal di komputer. Sekarang, periksa apakah telah menginstal mysql.connector dengan benar atau tidak menggunakan kode berikut.

```
import mysql.connector
```

Computational Thinking

Python MySQL - Connecting and Creating

Sekarang, kita akan terhubung ke database menggunakan username dan password MySQL. Jika tidak ingat nama pengguna atau kata sandi, buat pengguna baru dengan kata sandi. Untuk membuat pengguna baru, lihat dokumentasi resmi MySQL. Sekarang, sambungkan ke database menggunakan nama pengguna dan kata sandi.

```
## Connecting to the database
## importing 'mysql.connector' as mysql for convenient
import mysql.connector as mysql

## connecting to the database using 'connect()' method
## it takes 3 required parameters 'host', 'user', 'passwd'
db = mysql.connect(
        host = "localhost",
        user = "root",
        passwd = "dbms"
)
print(db) # it will print a connection object if everything is fine
```

```
<mysql.connector.connection_cext.CMySQLConnection object at
0x0000020C26A84C50>
```

Computational Thinking

Python MySQL - Create Database

Sekarang, kita akan membuat database dengan nama datacamp. Untuk membuat database di MySQL, kita menggunakan pernyataan CREATE DATABASE database_name.

```
import mysql.connector as mysql
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms"
)

## creating an instance of 'cursor' class which is used to execute the 'SQL'
statements in 'Python'
cursor = db.cursor()

## creating a databse called 'datacamp'
## 'execute()' method is used to compile a 'SQL' statement
## below statement is used to create tha 'datacamp' database
cursor.execute("CREATE DATABASE datacamp")
```

Jika database sudah ada Anda akan mendapatkan kesalahan. Pastikan bahwa database tidak ada. Lihat semua database yang ada di MySQL menggunakan kode berikut. Untuk melihat semua database kami menggunakan pernyataan SHOW DATABASES.

Computational Thinking

Python MySQL - Create Database

```
import mysql.connector as mysql
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms"
)
cursor = db.cursor()

## executing the statement using 'execute()' method
cursor.execute("SHOW DATABASES")

## 'fetchall()' method fetches all the rows from the last executed statement
databases = cursor.fetchall() ## it returns a list of all databases present

## printing the list of databases
print(databases)

## showing one by one database
for database in databases:
        print(database)
```

```
[('datacamp',), ('information_schema',), ('mysql',), ('performance_schema',),
('sakila',), ('sys',), ('world',)]
('datacamp',)
('information_schema',)
('mysql',)
('performance_schema',)
('sakila',)
('sys',)
('world',)
```

Computational Thinking

Python MySQL - Create Table

Membuat tabel dalam database untuk menyimpan informasi. Sebelum membuat tabel, kita harus memilih database terlebih dahulu. Jalankan kode berikut, untuk memilih database datacamp yang telah kita buat satu menit sebelumnya.

```
import mysql.connector as mysql
db = mysql.connect(
   host = "localhost",
   user = "root",
   passwd = "dbms",
   database = "datacamp"
)
```

Kode di atas akan dijalankan tanpa kesalahan jika database ada. Sekarang, Anda telah terhubung ke database yang disebut datacamp. Gunakan nama_tabel CREATE TABLE untuk membuat tabel di database yang dipilih.

```
import mysql.connector as mysql
db = mysql.connect(
   host = "localhost",
   user = "root",
   passwd = "dbms",
   database = "datacamp" )

cursor = db.cursor()

## creating a table called 'users' in the 'datacamp' database
cursor.execute("CREATE TABLE users (name VARCHAR(255), user_name
VARCHAR(255))")
```

Computational Thinking

Python MySQL - Create Table

Anda telah berhasil membuat tabel user di database datacamp. Lihat semua tabel yang ada dalam database menggunakan pernyataan SHOW TABLES.

```
import mysql.connector as mysql
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms",
    database = "datacamp"
)

cursor = db.cursor()
## getting all the tables which are present in 'datacamp' database

cursor.execute("SHOW TABLES")
tables = cursor.fetchall() ## it returns list of tables present in the database

## showing all the tables one by one
for table in tables:
        print(table)
```

Computational Thinking

Python MySQL - Create Table

Primary Key adalah nilai unik dalam tabel. Ini membantu untuk menemukan setiap baris secara unik dalam tabel. Untuk membuat Primary Key, kami menggunakan pernyataan PRIMARY KEY saat membuat tabel.

```
import mysql.connector as mysql
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms",
    database = "datacamp"
)
cursor = db.cursor()

## first we have to 'drop' the table which has already created to create it
again with the 'PRIMARY KEY'
## 'DROP TABLE table_name' statement will drop the table from a
database
cursor.execute("DROP TABLE users")

 ## creating the 'users' table again with the 'PRIMARY KEY'
cursor.execute("CREATE TABLE users (id INT(11) NOT NULL
AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), user_name
VARCHAR(255))")
```

Computational Thinking

Python MySQL - Create Table

Untuk melihat tabel jalankan kode berikut.

```
import mysql.connector as mysql
 db = mysql.connect(
   host = "localhost",
   user = "root",
   passwd = "dbms",
   database = "datacamp"
)
cursor = db.cursor()

## 'DESC table_name' is used to get all columns information
cursor.execute("DESC users")

## it will print all the columns as 'tuples' in a list
print(cursor.fetchall())
```

```
[('id', 'int(11)', 'NO', 'PRI', None, 'auto_increment'), ('name', 'varchar(255)',
'YES', '', None, ''), ('user_name', 'varchar(255)', 'YES', '', None, '')]
```

Computational Thinking

Python MySQL - Inserting Data

Mari kita lihat bagaimana menyisipkan satu baris ke dalam tabel. Memasukkan data ke dalam tabel untuk menyimpannya. Gunakan pernyataan INSERT INTO table_name (column_names) VALUES (data) untuk dimasukkan ke dalam tabel.

```
import mysql.connector as mysql

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms",
    database = "datacamp"
)

cursor = db.cursor()

## defining the Query
query = "INSERT INTO users (name, user_name) VALUES (%s, %s)"

## storing values in a variable
values = ("Hafeez", "hafeez")

## executing the query with values
cursor.execute(query, values)

## to make final output we have to run the 'commit()' method of the
database object
db.commit()

print(cursor.rowcount, "record inserted")
```

Computational Thinking

Python MySQL - Inserting Data

Mari kita lihat cara menyisipkan beberapa baris ke dalam tabel.

```
import mysql.connector as mysql

 db = mysql.connect(
   host = "localhost",
   user = "root",
   passwd = "dbms",
   database = "datacamp"
)

cursor = db.cursor()

## defining the Query
query = "INSERT INTO users (name, user_name) VALUES (%s, %s)"

## storing values in a variable
values = [
          ("Peter", "peter"),
          ("Amy", "amy"),
          ("Michael", "michael"),
          ("Hennah", "hennah")
]

## executing the query with values
cursor.executemany(query, values)

## to make final output we have to run the 'commit()' method of the
database object

db.commit()

print(cursor.rowcount, "records inserted")
```

Computational Thinking

Python MySQL - Select Data

To retrieve the data from a table we use, SELECT column_names FROM table_name statement. Untuk mendapatkan semua record dari sebuah tabel, kita menggunakan * sebagai pengganti nama kolom. Mari kita ambil semua data dari tabel users yang telah kita masukkan sebelumnya.

```
import mysql.connector as mysql

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms",
    database = "datacamp"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM users"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the data
for record in records:
        print(record)
```

```
(1, 'Hafeez', 'hafeez')
(2, 'Peter', 'peter')
(3, 'Amy', 'amy')
(4, 'Michael', 'michael')
(5, 'Hennah', 'hennah')
```

Computational Thinking

Python MySQL - Where

WHERE digunakan untuk memilih data pada suatu kondisi. Sekarang, kita akan memilih record dengan id 5.

```
import mysql.connector as mysql

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms",
    database = "datacamp"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM users WHERE id = 5"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the data
for record in records:
        print(record)
```

```
(5, 'Hennah', 'hennah')
```

Computational Thinking

Python MySQL - Order by

Gunakan ORDER BY untuk mengurutkan hasil dalam urutan menaik atau menurun. Ini mengurutkan hasil dalam urutan menaik secara default, untuk mengurutkan hasil dalam urutan menurun gunakan kata kunci DESC.

```
import mysql.connector as mysql

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms",
    database = "datacamp"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM users ORDER BY name"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the data
for record in records:
        print(record)
```

```
(3, 'Amy', 'amy')
(1, 'Hafeez', 'hafeez')
(5, 'Hennah', 'hennah')
(4, 'Michael', 'michael')
(2, 'Peter', 'peter')
```

Computational Thinking

Python MySQL - Delete

Kata kunci DELETE digunakan untuk menghapus record dari tabel. DELETE FROM table_name WHERE pernyataan kondisi digunakan untuk menghapus record.

Jika Anda tidak menentukan kondisinya, maka semua catatan akan dihapus. Mari kita hapus catatan dari tabel pengguna dengan id 5

```
import mysql.connector as mysql

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms",
    database = "datacamp"
)

cursor = db.cursor()

## defining the Query
query = "DELETE FROM users WHERE id = 5"

## executing the query
cursor.execute(query)

## final step to tell the database that we have changed the table data
db.commit()
```

Computational Thinking

Python MySQL - Update

Kata kunci UPDATE digunakan untuk memperbarui data suatu record atau record. UPDATE table_name SET column_name = new_value WHERE pernyataan kondisi digunakan untuk memperbarui nilai baris tertentu. Update nama record 1 dari Hafeez ke Kareem.

```
import mysql.connector as mysql

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms",
    database = "datacamp"
)

cursor = db.cursor()

## defining the Query
query = "UPDATE users SET name = 'Kareem' WHERE id = 1"

## executing the query
cursor.execute(query)

## final step to tell the database that we have changed the table data
db.commit()
```

Computational Thinking

Python MySQL - Update

Memeriksa data apakah diperbarui atau tidak dengan mengambil semua catatan dari data.

```
import mysql.connector as mysql

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "dbms",
    database = "datacamp"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM users"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the data
for record in records:
        print(record)
```

```
(1, 'Kareem', 'hafeez')
(2, 'Peter', 'peter')
(3, 'Amy', 'amy')
(4, 'Michael', 'michael')
```

Lihat, nama catatan pertama diubah.

Computational Thinking

Python PostgreSQL- Connecting

PostgreSQL adalah salah satu database relasional open-source yang paling populer. Perusahaan dari semua ukuran dan pengembang di seluruh dunia menggunakannya. Menurut DB-Engines, PostgreSQL menempati urutan keempat di antara database paling populer di dunia, dan memiliki tren yang meningkat. Ini tidak mengejutkan, karena kita dapat menemukan database PostgreSQL di balik banyak aplikasi web dan seluler — dan bahkan perangkat lunak analitik.

PostgreSQL juga memiliki ekosistem yang kaya dengan sejumlah besar alat dan ekstensi yang terintegrasi dengan baik dengan database inti. Untuk alasan ini, PostgreSQL membuat pilihan yang tepat apakah kita memerlukan database transaksional atau analitik, atau ingin membangun solusi database kustom kita sendiri.

Connect To PostgreSQL Database Server

Pertama, kunjungi psycopg2 package di Psycopg2. Kedua, gunakan baris perintah berikut dari terminal:

```
pip install psycopg2
```

Jika telah mengunduh package ke komputer, kita dapat menggunakan setup.py sebagai berikut:

```
python setup.py build
sudo python setup.py install
```

Computational Thinking

Python PostgreSQL- Create Database

Create a new database

Pertama, masuk ke server database PostgreSQL menggunakan alat klien apa pun seperti pgAdmin atau psql. Kedua, gunakan pernyataan berikut untuk membuat database baru bernama pemasok di server database PostgreSQL.

```
CREATE DATABASE suppliers;
```

Connect to PostgreSQL database menggunakan psycopg2

Untuk terhubung ke database pemasok, kita menggunakan fungsi connect() dari modul psycopg2. Fungsi connect() membuat sesi database baru dan mengembalikan instance baru dari kelas koneksi. Dengan menggunakan objek koneksi, kita dapat membuat kursor baru untuk mengeksekusi pernyataan SQL apa pun.

Untuk memanggil fungsi connect(), kita menentukan parameter database PostgreSQL sebagai string koneksi dan meneruskannya ke fungsi seperti ini:

```
conn = psycopg2.connect("dbname=suppliers user=postgres
password=postgres")
```

Atau dapat menggunakan keyword arguments:

```
conn = psycopg2.connect(
   host="localhost",
   database="suppliers",
   user="postgres",
   password="Abcd1234")
```

Computational Thinking

Python PostgreSQL- Create Table

Untuk membuat tabel baru di database PostgreSQL, Anda menggunakan langkah-langkah berikut:

1. Pertama, buat pernyataan CREATE TABLE.
2. Selanjutnya, sambungkan ke database PostgreSQL dengan memanggil fungsi connect(). Fungsi connect() mengembalikan objek koneksi.
3. Kemudian, buat objek kursor dengan memanggil metode kursor() dari objek koneksi.
4. Setelah itu, jalankan CREATE TABLE dengan memanggil metode execute() dari objek kursor.
5. Terakhir, tutup komunikasi dengan server database PostgreSQL dengan memanggil metode close() dari kursor dan objek koneksi.

```
#!/usr/bin/python

import psycopg2
from config import config


def create_tables():
    """ create tables in the PostgreSQL database"""
    commands = (
        """
        CREATE TABLE vendors (
            vendor_id SERIAL PRIMARY KEY,
            vendor_name VARCHAR(255) NOT NULL
        )
        """,
        """ CREATE TABLE parts (
                part_id SERIAL PRIMARY KEY,
                part_name VARCHAR(255) NOT NULL
                )
        """,
```

Computational Thinking

Python PostgreSQL- Create Table

```
    """
    CREATE TABLE part_drawings (
            part_id INTEGER PRIMARY KEY,
            file_extension VARCHAR(5) NOT NULL,
            drawing_data BYTEA NOT NULL,
            FOREIGN KEY (part_id)
            REFERENCES parts (part_id)
            ON UPDATE CASCADE ON DELETE CASCADE
    )
    """,
    """
    CREATE TABLE vendor_parts (
            vendor_id INTEGER NOT NULL,
            part_id INTEGER NOT NULL,
            PRIMARY KEY (vendor_id , part_id),
            FOREIGN KEY (vendor_id)
               REFERENCES vendors (vendor_id)
               ON UPDATE CASCADE ON DELETE CASCADE,
            FOREIGN KEY (part_id)
               REFERENCES parts (part_id)
               ON UPDATE CASCADE ON DELETE CASCADE
    )
    """)
conn = None
```

Computational Thinking

Python PostgreSQL- Create Table

```
try:
        # read the connection parameters
        params = config()
        # connect to the PostgreSQL server
        conn = psycopg2.connect(**params)
        cur = conn.cursor()
        # create table one by one
        for command in commands:
            cur.execute(command)
        # close communication with the PostgreSQL database server
        cur.close()
        # commit the changes
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()


if __name__ == '__main__':
    create_tables()
```

Computational Thinking

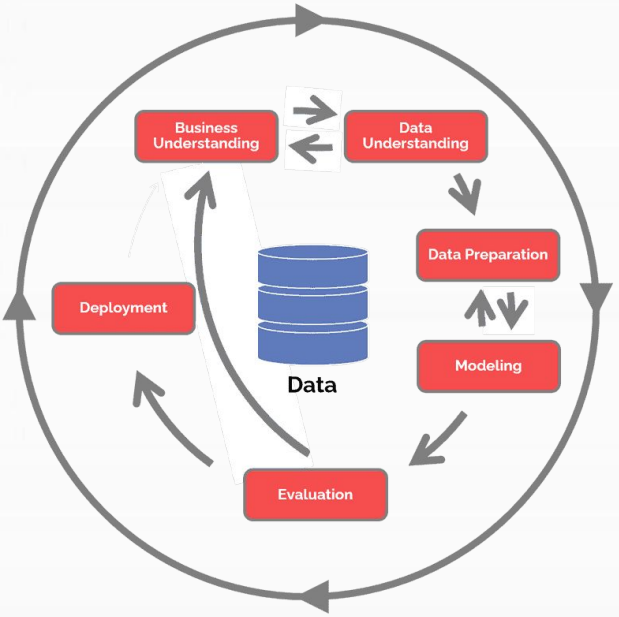
Python PostgreSQL- Insert data to table

Untuk menyisipkan baris ke dalam tabel PostgreSQL dengan Python, kita menggunakan langkah-langkah berikut:

```
#!/usr/bin/python

import psycopg2
from config import config


def insert_vendor(vendor_name):
    """ insert a new vendor into the vendors table """
    sql = """INSERT INTO vendors(vendor_name)
             VALUES(%s) RETURNING vendor_id;"""
    conn = None
    vendor_id = None
    try:
        # read database configuration
        params = config()
        # connect to the PostgreSQL database
        conn = psycopg2.connect(**params)
        # create a new cursor
        cur = conn.cursor()
        # execute the INSERT statement
        cur.execute(sql, (vendor_name,))
        # get the generated id back
        vendor_id = cur.fetchone()[0]
        # commit the changes to the database
        conn.commit()
        # close communication with the database
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()

    return vendor_id
```

Statistical Thinking

Understanding Data and Business



Seringkali, ketika berbicara tentang *Data Science*, Sepertinya tidak ada yang bisa memberikan penjelasan yang tepat tentang bagaimana keseluruhan proses berjalan. Mulai dari pengumpulan data, analisis hingga penyajian hasilnya.

Beberapa proses yang terdapat dalam siklus tahapan *data science* antara lain terdiri dari *business understanding, data understanding, data preparation, modeling, evaluation,* dan *deployment.* Berikut adalah penjelasan mengenai proses - proses apa saja yang ada pada tiap tahapan siklus *data science*.

Statistical Thinking

Understanding Data and Business

- *Business Understanding*
  - ➢ Fase *business understanding* berfokus pada pemahaman tujuan dan persyaratan proyek. Selain itu tugas lain dalam fase ini adalah aktivitas manajemen proyek dasar yang bersifat universal untuk sebagian besar proyek
  - ➢ Tentukan tujuan bisnis: Pertama-tama kamu harus memahami secara menyeluruh, dari perspektif bisnis, apa yang benar-benar ingin dicapai pelanggan (CRISP-DM Guide) dan kemudian tentukan kriteria keberhasilan bisnis.
  - ➢ Menilai situasi: Menentukan ketersediaan sumber daya, persyaratan proyek, menilai risiko dan kontingensi, dan melakukan analisis biaya-manfaat.
  - ➢ Tentukan tujuan *Data Science*: Selain menentukan tujuan bisnis, kamu juga harus menentukan seperti apa kesuksesan dari perspektif *Data Science*.
  - ➢ Menghasilkan rencana proyek: Pilih teknologi dan alat dan tentukan rencana terperinci untuk setiap fase proyek. Banyak orang yang terburu - buru dalam melakukan proses *business understanding*. Padahal, tahapan ini adalah pondasi dari tahapan - tahapan *data science* itu sendiri

Statistical Thinking

Understanding Data and Business

- *Data Understanding*
  - ➢ Pengumpulan data: Dapatkan data yang diperlukan dan (jika perlu) masukkan ke dalam alat analisis kamu.
  - ➢ Penjelasan data: Periksa data dan dokumentasikan properti permukaannya seperti format data, jumlah catatan, atau identitas.
  - ➢ *Explore* data: Gali data lebih dalam. *Query*, visualisasikan, dan identifikasi hubungan antar data.
  - ➢ Verifikasi kualitas data: Seberapa bersih/kotor datanya? Dokumentasikan masalah kualitas apa pun.
- *Data Preparation*
  - ➢ *Select data*: Tentukan kumpulan data mana yang akan digunakan dan dokumentasikan alasan penyertaan/pengecualian.
  - ➢ *Clean data*: Seringkali ini adalah tugas terlama. Hal yang umum dilakukan adalah mengoreksi, mengaitkan, atau menghapus nilai yang salah.
  - ➢ *Construct data*: Dapatkan atribut baru yang akan membantu. Misalnya, dapatkan indeks massa tubuh seseorang dari bidang tinggi dan berat badan.
  - ➢ *Integrate data*: Buat kumpulan data baru dengan menggabungkan data dari berbagai sumber.
  - ➢ *Format data*: Format ulang data seperlunya. Misalnya, dapat mengkonversi nilai *string* yang menyimpan angka menjadi nilai numerik sehingga Anda dapat melakukan operasi matematika.

Statistical Thinking

Understanding Data and Business

- *Modeling*
  Langkah pertama yang perlu lakukan dalam memodelkan data adalah mengurangi dimensi kumpulan data. Tidak semua fitur atau nilai penting untuk memprediksi model. Ada beberapa hal yang dapat kita lakukan dalam pemodelan. Salah satu contoh dari model adalah model untuk melakukan klasifikasi untuk membedakan email yang terima sebagai "Inbox" dan "Spam" menggunakan regresi logistik. Kita juga dapat memperkirakan nilai menggunakan regresi linier. Selain itu, kita juga dapat menggunakan pemodelan untuk mengelompokkan data untuk memahami logika di balik suatu *cluster*.
- *Evaluation*
  - ➢ Evaluasi hasil: Apakah model memenuhi kriteria keberhasilan bisnis? Yang mana yang harus kita setujui untuk bisnis ini?
  - ➢ Proses peninjauan: Tinjau pekerjaan yang diselesaikan. Apakah ada yang terlewatkan? Apakah semua langkah dijalankan dengan benar? Ringkas temuan dan perbaiki apapun jika diperlukan.
  - ➢ Tentukan langkah selanjutnya: Berdasarkan hal yang telah dilakukan, tentukan apakah akan melanjutkan penerapan, mengulangi lebih lanjut, atau memulai proyek baru.

Statistical Thinking

Understanding Data and Business

- *Modeling*
  Sebuah model berguna apabila para pengguna dapat mengaksesnya. Kompleksitas fase ini sangat bervariasi. Fase terakhir ini memiliki empat tugas:
  - ➢ *Plan Development*: Kembangkan dan dokumentasikan rencana untuk menerapkan model.
  - ➢ *Plan monitoring and maintenance*: Kembangkan rencana pemantauan dan pemeliharaan yang menyeluruh untuk menghindari masalah selama fase operasional (atau fase pasca proyek) suatu model.
  - ➢ *Produce final report*: Tim proyek mendokumentasikan ringkasan proyek yang mungkin mencakup presentasi akhir hasil *Data Science*.
  - ➢ *Review project*: Lakukan retrospeksi proyek tentang apa yang berjalan dengan baik, apa yang bisa lebih baik, dan bagaimana meningkatkannya di masa depan.

Pekerjaan organisasi kamu mungkin tidak berakhir di situ. Sebagai kerangka kerja proyek, CRISP-DM tidak menguraikan apa yang harus dilakukan setelah proyek. Tetapi jika model akan diproduksi, pemantauan secara rutin dan penyetelan model sesekali sering diperlukan.

Statistical Thinking

Statistics Foundation

Binomial

BI dalam kata BINOMIAL berarti dua. Hal ini merujuk kepada setiap kali percobaan atau kesempatan, hasil yang mungkin muncul hanya ada dua. Pertanyaan yang biasanya muncul adalah, kapan kita menggunakan menghitung peluang dengan menggunakan persamaan distribusi peluang Binomial? Jika kejadian tersebut memenuhi sifat-sifat di bawah ini maka ketika menghitung peluang kejadian tersebut terjadi maka persamaan yang digunakan adalah persamaan peluang dari distribusi binomial. Syarat kejadian binomial, yaitu:

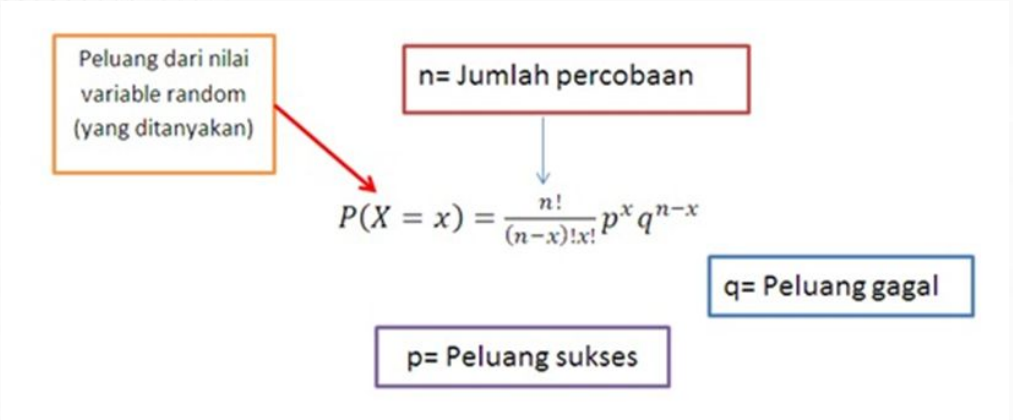
- Percobaan dilakukan sebanyak n kali.
- Setiap kali percobaan mempunyai dua kemungkinan hasil.
- Kemungkinan hasil dari masing-masing percobaan sama.
- Hasil yang diperoleh pada percobaan pertama tidak akan mempengaruhi hasil yang diperoleh pada percobaan-percobaan yang lain (saling independen)

Untuk persamaan hitung peluang dapat dilihat sebagai berikut. Misalkan X adalah variabel random diskret. Maka peluang dari X adalah:

$$P(X) = \frac{n!}{(n-x)!\,x!} p^x q^{n-x}$$

Statistical Thinking

Statistics Foundation



Misalkan kita mempunyai satu buah koin yang terdiri atas dua sisi, depan dan belakang. Misalkan kita mengundi sebanyak 10 kali. Pada undian pertama, kemungkinan hasilnya hanya sisi depan atau sisi belakang. Pada undian kedua, kemungkinan hasilnya hanya sisi depan atau sisi belakang. Demikian seterusnya. Setiap kali mengundi, kemungkinan hasilnya sama, hanya dua yaitu sisi depan atau sisi belakang. Dari sepuluh kali percobaan, berapa peluang sisi depan muncul sebanyak dua kali?

Diketahui:

- Jumlah percobaan = n = 10.
- Peluang sukses = peluang munculnya sisi depan dalam setiap percobaan = p = 0.5.
- Peluang gagal = peluang tidak munculnya sisi depan dalam setiap percobaan = q = 1-p = 0.5.

Statistical Thinking

Statistics Foundation

Ditanyakan:
Dari sepuluh kali percobaan, berapa peluang sisi depan muncul sebanyak dua kali? Atau P (X = 2) -> Yang ditanyakan adalah peluang munculnya sisi depan maka kejadian yang dianggap sukses adalah jika sisi depan muncul ketika diundi.

Jawaban:

$$P(X = 2) = \frac{10!}{(10-2)!\,2!}(0.5)^2(1-0.5)^{10-2} = 0.0439$$

Referensi: DISTRIBUSI PELUANG BINOMIAL

Statistical Thinking

Statistics Foundation

Poisson

Distribusi *poisson* merupakan suatu distribusi untuk peristiwa yang probabilitas kejadiannya kecil, di mana kejadian tergantung pada selang waktu tertentu atau di suatu daerah tertentu dengan hasil pengamatan berupa variabel diskret dan antar variabel prediktor saling independen. Selang waktu tersebut dapat berupa beberapa saja panjangnya, misalnya semenit, sehari, seminggu, sebulan bahkan setahun. Daerah tertentu yang dimaksudkan dapat berupa suatu garis, suatu luasan, suatu volume, atau mungkin sepotong bahan. Distribusi *Poisson* memiliki ciri – ciri sebagai berikut:

- Banyaknya percobaan yang terjadi dalam suatu selang waktu atau suatu daerah tertentu, tidak tergantung pada banyaknya hasil percobaan yang terjadi pada selang waktu atau daerah lain yang terpisah.
- Peluang terjadinya satu hasil percobaan selama suatu selang waktu yang singkat sekali atau dalam suatu selang yang kecil. Sebanding dengan panjang selang waktu tersebut atau besarnya daerah tersebut dan tidak tergantung pada banyak hasil percobaan yang terjadi diluar selang waktu dan daerah tertentu.
- Peluang bahwa lebih dari satu hasil percobaan akan terjadi dalam selang waktu yang singkat tersebut atau dalam daerah yang kecil tersebut diabaikan.

Referensi: Distribusi Poisson dan Hipergeometrik

Statistical Thinking

Statistics Foundation

Variabel Random Poisson

Percobaan yg menghasilkan variabel random X yg menyatakan banyaknya outcome selama interval waktu tertentu atau dalam "area" atau "luas" tertentu dinamakan percobaan Poisson.

Contoh:

X : banyak panggilan telepon per jam

X : banyak hari-hari sekolah tutup karena bencana alam dalam setahun

X : banyaknya penundaan pertandingan bola karena hujan dalam semusim pertandingan

X : banyak tikus per hektar

X : banyaknya kesalahan ketik per halaman

Sifat Proses *Poisson*:

1. Tidak punya memori atau ingatan, yaitu banyaknya outcome dalam satu interval waktu (atau daerah) tidak bergantung pada banyaknya outcome pada waktu atau daerah yg lain.
2. Probabilitas terjadinya 1 *outcome* dalam interval waktu (atau daerah) yg sangat pendek (kecil) sebanding dengan lama waktu interval waktu tsb (atau luas daerahnya). Dan tidak bergantung pada kejadian atau outcome di luar interval ini.
3. Probabilitas terjadinya lebih dari 1 *outcome* dalam interval waktu yg sangat pendek di (2) tsb sangat kecil atau bisa diabaikan.

Statistical Thinking

Statistics Foundation

X : variabel *random Poisson* yg menyatakan banyaknya *outcome* selama percobaan.
μ : rata-rata banyak *outcome* = λt dimana t adalah lama intervalnya dan λ adalah laju terjadinya *outcome*.

Distribusi probabilitas dari variabel *random Poisson* X yg menyatakan banyaknya *outcome* dalam interval waktu tertentu t (atau daerah tertentu) dengan λ menyatakan laju terjadinya *outcome* persatuan waktu atau per satuan daerah diberikan oleh (tidak diturunkan!) :

$$p(x;\lambda t) = \frac{e^{-\lambda t}(\lambda t)^x}{x!} \qquad \Rightarrow p(x;\mu) = \frac{e^{-\mu}(\mu)^x}{x!}$$

Statistical Thinking

Statistics Foundation

Hipergeometrik

Distribusi hipergeometrik digunakan ketika pengambilan sampel dilakukan tanpa pengembalian, informasi tentang susunan populasi harus diketahui untuk menentukan kembali probabilitas keberhasilan dalam setiap percobaan berturut-turut karena probabilitas berubah. Dengan demikian, distribusi hipergeometrik memiliki karakteristik sebagai berikut:

- Distribusi hipergeometrik merupakan distribusi diskrit
- Setiap hasil (*outcome*) terdiri dari keberhasilan atau kegagalan.
- Pengambilan sampel (*sampling*) dilakukan tanpa pengembalian.
- Populasi (N) adalah terbatas dan diketahui.
- Jumlah keberhasilan dalam populasi, k, diketahui.

Distribusi Hipergeometrik sangat serupa dengan distribusi binomial, terdapat persamaan, yakni keduanya menyatakan probabilitas sejumlah tertentu percobaan masuk dalam kategori tertentu. Sementara itu, terdapat beberapa perbedaan, diantaranya:

1. Binomial mengharuskan ketidakbergantungan dari satu percobaan (*trial*) ke percobaan berikutnya.
2. Sampling harus dilakukan dengan dikembalikan (*replaced*)
3. Hipergeometrik tidak mengharuskan ketidakbergantungan, jadi *sampling* dilakukan tanpa mengembalikan *outcome* yg sudah keluar.

Statistical Thinking

Statistics Foundation

Distribusi Hipergeometrik dari variabel random X yang menyatakan banyaknya outcome yang "sukses" dari sampel random sebanyak n yg diambil dari populasi sebanyak N, dimana dari N tsb sebanyak k buah adalah "sukses" dan sisanya "N-k" adalah "gagal":

$$h(x;N,n,k)=\frac{\binom{k}{x}\binom{N-k}{n-x}}{\binom{N}{n}}$$

- Suku pembagi (denominator) menyatakan banyak kombinasi yg terjadi jika dari N obyek diambil n tiap kali.
- Faktor pertama suku terbagi (numerator) menyatakan banyaknya kombinasi dari obyek berjenis "sukses" yg berjumlah k jika tiap kali diambil sebanyak x buah.

Statistical Thinking

Statistics Foundation

Contoh :
Suatu jenis suku cadang mobil dijual dalam bentuk paket yg isinya 10 buah. Produsen merasa bahwa bahwa paket tersebut dinyatakan "dapat diterima" jikalau tak lebih dari 1 buah suku cadang/paket yg cacat. Untuk memeriksa kualitasnya dilakukan sampling secara random diambil beberapa paket, dan di tiap paket dilakukan pemeriksaan terhadap 3 buah suku cadang dari paket yg di sampel. Kemudian paket dinyatakan baik jika dari 3 yg diperiksa tersebut tidak satupun yg cacat. Berapakah probabilitasnya seandainya sampel yg diambil sebenarnya mengandung 2 buah suku cadang cacat (jadi unacceptable), tapi ketika diambil sampel 3 ternyata tak satupun juga cacat, sehingga salah mengambil kesimpulan!

Jawab:
Misalkan bahwa ada lot yg benar-benar tak bisa diterima, karena 2 dari 10 isinya cacat. Kita hitung berapa probabilitasnya bahwa teknik sampling yg kita lakukan dapat menemukan hal ini.
Misal X adalah banyak suku cadang yg cacat, maka probabilitas bahwa dari 3 suku cadang yg diambil tak satupun cacat adalah sbb: Jumlah yg cacat di paket k=2, yg terambil tidak ada, X=0. Isi satu paket N=10, jadi yg baik N-k=10-2=8. Dari paket diambil n=3 sampel.

Statistical Thinking

Statistics Foundation

Banyaknya kombinasi bahwa dari k=2 cacat di paket tidak terambil sama sekali (x=0) adalah C2 0 = 2!/(0!2!)=1. Dan kombinasi dari 8 yg cacat diambil 3 buah ada sebanyak C8 3 = 8!/(3!5!) = 8x7x6/6=56. Sedangkan kalau dari 10 diambil 3 buah item, banyak kombinasi item yg mungkin adalah C10 3 = 10!/7!3! = 10x9x8/6 = 120. Jadi probabilitas bahwa yg terambil mengandung 3 buah item dan tak satupun cacat adalah:

$$h(x=0;N=10,n=3,k=2)=\frac{\binom{2}{0}\binom{8}{3}}{\binom{10}{3}}=\frac{1*56}{120}=0.467=47\%$$

Referensi: Distribusi Probabilitas Diskrit

Statistical Thinking

Statistics Foundation

Bayes

Statistika Bayes adalah sebuah teori di bidang statistika yang didasarkan pada interpretasi Bayes tentang probabilitas dimana probabilitas mengekspresikan tingkat kepercayaan pada suatu peristiwa. Tingkat kepercayaan dapat didasarkan pada pengetahuan sebelumnya tentang peristiwa tersebut seperti hasil percobaan sebelumnya, atau didasarkan pada keyakinan pribadi tentang peristiwa tersebut. Hal ini berbeda dari sejumlah interpretasi probabilitas lainnya, seperti interpretasi frekuensi yang memandang probabilitas sebagai batas frekuensi relatif dari suatu peristiwa setelah melakukan percobaan dalam jumlah yang besar.

Metode statistika Bayes menggunakan teorema Bayes untuk menghitung dan memperbarui probabilitas setelah mendapatkan data baru. Teorema Bayes menggambarkan probabilitas bersyarat pada suatu peristiwa berdasarkan data serta informasi atau keyakinan sebelumnya tentang peristiwa, atau kondisi yang terkait dengan peristiwa tersebut. Misalnya saja dalam inferensi Bayes, teorema Bayes dapat digunakan untuk memperkirakan parameter distribusi probabilitas atau model statistikal. Karena statistika Bayes memperlakukan probabilitas sebagai tingkat kepercayaan, teorema Bayes dapat secara langsung menetapkan distribusi probabilitas yang mengkuantifikasi keyakinan pada suatu parameter atau serangkaian parameter.

Referensi: Bayes' Theorem: What It Is, the Formula, and Examples

Statistical Thinking

Statistics Foundation

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Dengan menggunakan teorema Bayes, kita dapat mencari peluang terjadinya A, dengan syarat B telah terjadi. Di sini, B adalah bukti dan A adalah hipotesis. Asumsi yang dibuat di sini adalah bahwa prediktor/fiturnya independen. Artinya kehadiran satu fitur tertentu tidak mempengaruhi yang lain.

Contoh Numerik Teorema Bayes :
Sebagai contoh numerik, bayangkan ada tes narkoba yang akurat 98%, artinya 98% dari waktu, itu menunjukkan hasil positif yang benar untuk seseorang yang menggunakan narkoba, dan 98% dari waktu, itu menunjukkan hasil yang benar-benar negatif. untuk bukan pengguna obat.

Selanjutnya, asumsikan 0,5% orang menggunakan obat tersebut. Jika seseorang yang dipilih secara acak dites positif menggunakan narkoba, perhitungan berikut dapat dilakukan untuk menentukan probabilitas orang tersebut benar-benar pengguna narkoba.

(0,98 x 0,005) / [(0,98 x 0,005) + ((1 - 0,98) x (1 - 0,005))] = 0,0049 / (0,0049 + 0,0199) = 19,76%. Teorema Bayes menunjukkan bahwa bahkan jika seseorang dites positif dalam skenario ini, ada kemungkinan sekitar 80% orang tersebut tidak menggunakan obat tersebut.

Statistical Thinking

Statistics Foundation

Inferential Statistics

Statistik inferensial sering digunakan untuk membandingkan perbedaan antara kelompok perlakuan. Statistik inferensial menggunakan pengukuran dari sampel subjek dalam eksperimen untuk membandingkan kelompok perlakuan dan membuat generalisasi tentang populasi subjek yang lebih besar. Ada banyak jenis statistik inferensial dan masing-masing sesuai untuk desain penelitian dan karakteristik sampel tertentu. Peneliti harus berkonsultasi dengan banyak teks tentang desain eksperimental dan statistik untuk menemukan uji statistik yang tepat untuk eksperimen mereka.

Namun, sebagian besar statistik inferensial didasarkan pada prinsip bahwa nilai statistik uji dihitung berdasarkan rumus tertentu. Nilai itu bersama dengan derajat kebebasan, ukuran yang terkait dengan ukuran sampel, dan kriteria penolakan digunakan untuk menentukan apakah ada perbedaan antara kelompok perlakuan. Semakin besar ukuran sampel, semakin besar kemungkinan statistik menunjukkan bahwa ada perbedaan antara kelompok perlakuan. Jadi, semakin besar sampel subjek, semakin kuat statistiknya.

Hampir semua statistik inferensial memiliki asumsi dasar yang penting. Setiap replikasi dalam suatu kondisi diasumsikan independen. Artinya setiap nilai dalam suatu kondisi dianggap tidak berhubungan dengan nilai lain dalam sampel.

Statistical Thinking

Statistics Foundation

Statistik inferensial digunakan terutama untuk memperoleh perkiraan tentang kelompok besar (atau populasi) dan menarik kesimpulan pada data berdasarkan metode pengujian hipotesis.

Statistik inferensial menggunakan data sampel karena lebih hemat biaya dan tidak membosankan daripada mengumpulkan data dari seluruh populasi. Ini memungkinkan seseorang untuk sampai pada asumsi yang masuk akal tentang populasi yang lebih besar berdasarkan karakteristik sampel. Metode pengambilan sampel harus tidak bias dan acak agar kesimpulan dan inferensi statistik dapat divalidasi.

Referensi:
Inferential Statistics

Digital Leadership

Digital leadership atau yang dikenal juga dengan sebutan *e-Leadership* merupakan kepemimpinan digital yang timbul akibat dari berkembangnya lingkungan berbasis elektronik atau *e-Environment*. *Digital leadership* diperlukan dalam proses transformasi digital yang tengah berjalan saat ini untuk mengawal perubahan dan pemanfaatan teknologi dengan cepat di berbagai sektor. Hadirnya pemimpin digital dapat mendorong percepatan transformasi di dalam organisasi. Dengan hadirnya kepemimpinan digital, maka pemimpin mampu untuk mendayagunakan aset digital yang dimiliki oleh pegawainya untuk dapat mencapai tujuan organisasi. Pemimpin digital juga dapat memanfaatkan teknologi digital yang dihubungkan dengan proses bisnis masing-masing instansi pemerintah dalam melakukan transformasi layanan.

Terdapat empat karakteristik yang membedakan kepemimpinan biasa dengan *e-Leadership*, antara lain:

- *Digital leader* harus mampu berkomunikasi secara efektif menggunakan perangkat media sosial untuk terus terkoneksi dengan anggota di dalam maupun luar organisasi.
- *digital leader* harus memiliki kemampuan berpikir dan bekerja sama tanpa adanya batasan waktu, ruang, dan rintangan budaya dimana pengawasan dan interaksi tatap muka tidak lagi diperlukan.
- *Digital leader* harus memiliki kemampuan dalam memantau dan mengelola pekerjaan dengan efektif secara virtual.
- *Digital leader* harus memiliki kemampuan beradaptasi dengan perubahan lingkungan teknologi.

Referensi: Digital Leadership

EXERCISE

Berikut adalah *exercise* yang dapat digunakan untuk melatih pemahaman kamu:

1. Jelaskan tentang *skill - skill* yang diperlukan oleh *data scientist* untuk menyelesaikan pekerjaannya!
2. Berikan contoh-contoh penerapan *data science* pada bidang:
   - *Finance*
   - *Marketing*
   - *Operation*
   - *Sales*
3. Terdapat Data Harga Rumah di Boston, tugas kamu:
   - Berikan analisis CRISP-DM sederhana yang bisa kamu dapatkan dari data tersebut!
   - Jelaskan *feature - feature* data yang ada, dan berikan hipotesis *feature* apa yang berpengaruh terhadap harga rumah!
   - Apa *outcome* yang dapat diharapkan dari data tersebut?

Startup Campus

Kampus Merdeka
INDONESIA JAYA

MODULE

DATA SCIENCE TRACK