algoritma
learn data science by building

PT BANK RAKYAT INDONESIA (PERSERO) Tbk
DATA SCIENCE TRAINING
COURSEBOOK

JANUARY 3rd - JANUARY 11th, 2024

Copyright 2024, Algoritma

BRI Audit Analytics - Data Science Course in Python

Team Algoritma

December 28, 2023

Contents

1 Background **6**

2 Training Objectives **6**

3 Python for Data Analysts **6**
3.1 Variables and Keywords . . . 6

4 Introduction to Pandas Library **7**
4.1 Working with DataFrame . . . 7
4.2 Data Types . . . 8
4.2.1 Categorical and Numerical Variable . . . 10
4.3 Exploratory Data Analysis Tools . . . 11
4.4 Indexing & Subsetting with Pandas . . . 16
4.4.1 Conditional Subsetting . . . 20
4.4.2 Referencing and Copying . . . 21

5 Background **24**

6 Training Objectives **24**

7 Practical Statistics **24**

8 Study Case: Credit Card Balance Analysis **25**

9 Descriptive Statistics **27**
9.1 Measure of Central Tendency . . . 27
9.1.1 Mean . . . 27
9.1.2 Median . . . 28
9.1.3 Modus (Mode) . . . 28
9.1.4 Knowledge Check . . . 29
9.2 Measure of Spread . . . 29
9.2.1 Variance . . . 29
9.2.2 Standard Deviation . . . 30
9.2.3 Range using `boxplot()` . . . 31
9.3 Variable Relationship . . . 33
9.3.1 Covariance . . . 33
9.3.2 Correlation . . . 34

9.3.3 Knowledge Check . . . 36

10 Inferential Statistics **36**
10.1 Normal Distribution . . . 37
10.2 Probability Mass Function . . . 38
10.3 Probability Density Function . . . 38
10.4 Confidence Interval . . . 39
10.5 Hypothesis Testing . . . 41
10.5.1 Z-Test . . . 42
10.5.2 T-test . . . 44

11 Further Readings **46**

12 Background **47**

13 Training Objectives **47**

14 Reproducible Environment **47**
14.1 Importing Requirements . . . 48
14.2 Exporting Requirements . . . 48

15 Data Wrangling and Reshaping **49**
15.1 Slicing Multi-Index DataFrame . . . 51
15.1.1 Cross Section . . . 51
15.2 `stack()` and `unstack()` . . . 52
15.3 Melt . . . 54

16 Pandas dan Matplotlib **56**
16.1 Other Visualization . . . 60

17 Group By **61**
17.1 Visualisasi Barchart untuk Perbandingan . . . 64
17.2 Menggunakan Barchart Group . . . 69
17.3 Menggabungkan `agg` dan `groupby` . . . 72

18 Background **76**
18.1 Training Objectives . . . 76

19 Data Preparation **77**

20 Introduction to Machine Learning **80**

21 Regression Model **80**
21.1 Least Squares Regression . . . 81
21.2 Simple Linear Regression . . . 82
21.3 R-Squared . . . 87
21.4 [Additional] Leverage vs. Influence . . . 88

22 Multiple Linear Regression **90**
22.1 Adjusted R-Squared . . . 93

23 Encoding **95**

24 Regression Model Evaluation **98**

25 Limitations of Regression Models **99**
25.1 Linearity . . . 99
25.2 Normality of Residuals . . . 99
25.2.1 [Optional] Interpret the Residual Normality Indicators from `sm` results . . . 100
25.3 No-Heteroscedasticity . . . 101
25.4 No Multicolinearity . . . 103

26 Background **104**

27 Classification in Machine Learning **104**
27.1 Training Objectives . . . 104

28 Data Preparation and Exploratory Data **106**
28.1 Read The Dataset . . . 106
28.2 Descriptive Statistics . . . 109
28.3 Data Types Inspection and Preparation . . . 111
28.4 Check Missing Values . . . 113
28.5 Feature Selection . . . 115

29 Data Preprocessing **116**
29.1 Dummy Variables for Categorical Predictor . . . 117
29.2 Target & Predictors Splitting . . . 122
29.3 Cross Validation . . . 125
29.3.1 Train-Test Split . . . 125
29.3.2 Class Balance & Imbalance . . . 126

30 Model Classification 1 - Logistic Regression **127**
30.1 Probability . . . 127
30.2 Relating Probabilities to Odds . . . 128
30.2.1 Log of Odds . . . 129
30.3 Logistic Regression from First Principles . . . 132
30.4 Logistic Regression in Action . . . 133
30.4.1 Logistic Regression with One Categorical (Discrete) Predictor . . . 135
30.4.2 Logistic Regression with One Numerical (Continuous) Predictor . . . 138
30.4.3 Logistic Regression with Multiple Predictor . . . 140
30.5 Prediksi . . . 143
30.6 Evaluasi Model . . . 143
30.6.1 Confusion Matrix . . . 143
30.6.2 Accuracy . . . 144
30.6.3 Precision . . . 145
30.6.4 Recall/Sensitivity . . . 145
30.6.5 Specificity/True Negative Rate (TNR) . . . 145
30.7 Logistic Regression Key Assumptions . . . 146

31 Model Classification 2 - K-Nearest Neighbour **146**

31.1 Euclidean Distance . . . 147
31.2 Data Preparation . . . 148
31.2.1 Features rescaling . . . 149
31.3 Modeling . . . 149
31.3.1 Choosing an appropriate k . . . 149
31.3.2 Characteristics of k-NN . . . 150
31.4 Prediction . . . 151
31.5 Evaluation . . . 151

32 Modeling Summary **151**
32.1 [Additional] Classification Model Selection: Overfit, Underfit, Just Right . . . 152
32.1.1 Overfitting (High Variance) . . . 152
32.1.2 Underfitting/High Bias . . . 153
32.1.3 Trade-Off Underfitting and Overfitting (Bias and Variance Trade-Off) . . . 154

33 Wrap Up **154**

34 Background **156**
34.1 Training Objectives . . . 156

35 Unsupervised Learning **157**

36 Pengurangan Dimensi **157**
36.1 Prinsip dan Motivasi . . . 157

37 Import Library **159**

38 Langkah demi langkah: Analisis PCA **159**
38.1 Exploratory Data Analysis . . . 162
38.1.1 Cek Jumlah Transaksi . . . 162
38.1.2 Screening dataset secara cepat . . . 164
38.1.3 Melihat statistik dataset . . . 165
38.2 Data Perparation . . . 168
38.2.1 [Opsional] Nilai Eigen dan Vektor Eigen . . . 179
38.2.2 Intuisi dan Praktik: PCA . . . 181
38.2.3 Aplikasi PCA: Reduksi Dimensi pada Data credit . . . 181
38.2.4 Principal Component Analysis menggunakan library sklearn . . . 194
38.3 Visualisasi PCA . . . 199
38.3.1 Visualisasi Biplot . . . 199
38.3.2 [Optional] Visualisasi explained variance menggunakan library . . . 202
38.3.3 [Optional] Simple biplot : helper.py . . . 203
38.3.4 [Optional] Scree Plot Proporsi Eigenvalue . . . 203

39 Studi Kasus Local Outlier Factor (LOF): Deteksi Anomali dengan Menggunakan Library PyOD **206**
39.1 Overview . . . 206

40 Local Outlier Factor **207**
40.1 Membuat Model LOF . . . 208

40.2 Cek Index Anomali . . . 210
40.3 Visualisasi Plot LOF . . . 211
40.4 LOF Menggunakan Treshold . . . 212
40.4.1 Buat model LOF menggunakan treshold . . . 212
40.4.2 Visualisasi Plot LOF (treshold) . . . 212
40.4.3 Cek Index Anomali . . . 213
40.4.4 [Optional] Membandingkan kolom label dengan hasil lof . . . 215
40.5 Catatan Penting tentang Local Outlier Factor . . . 219

41 Kesimpulan **220**

42 Glossary **221**

43 Referensi **221**

Coursebook: Python for Data Analytics

- Bagian 1 Audit Analytics untuk Bank Rakyat Indonesia
- Durasi: 7 Jam
- *Last Updated*: December 2023

- *Coursebook* ini disusun dan dikurasi oleh tim produk dan instruktur dari Algoritma Data Science School

1 Background

Coursebook ini merupakan bagian dari **BRI Audit Analytics** yang disiapkan oleh Algoritma. *Coursebook* ini ditujukan hanya untuk khalayak terbatas, yaitu individu dan organisasi yang menerima *coursebook* ini langsung dari organisasi pelatihan. Tidak boleh direproduksi, didistribusikan, diterjemahkan, atau diadaptasi dalam bentuk apapun di luar individu dan organisasi ini tanpa izin.

Algoritma adalah pusat pendidikan *data science* yang berbasis di Jakarta. Kami menyelenggarakan *workshop* dan program pelatihan untuk membantu para profesional dan mahasiswa dalam menguasai berbagai sub-bidang *data science* yaitu: *data visualization, machine learning,* statistik, dan lain sebagainya.

2 Training Objectives

- Introduction to the `pandas` library.
- Introduction to `DataFrame`

- Data Types
- Exploratory Data Analysis I
- Indexing and Subsetting

3 Python for Data Analysts

Python adalah *high-level programming language*, yang memudahkan manusia untuk memprogram sesuatu dengan menggunakan bahasa yang mudah dipahami. Selain itu, Python juga menyediakan file notebook yang bisa digunakan untuk melakukan analisis . Python memiliki notebook dokumen, yang memudahkan pengguna dalam melakukan analisis.

> Notebook dalam Python adalah file yang mengandung **baris kode** (Python) dan **elemen text** (paragraf, heading, gambar). File notebook biasa digunakan dalam proses analisis karena selain bisa digunakan untuk menulis kode juga bisa digunakan untuk menulis analisa deskriptif. Ekstensi file notebook dalam Python adalah .ipynb.

Selanjutnya kita akan mengenal tentang dasar - dasar dalam pemrograman bahasa Python.

3.1 Variables and Keywords

Bahasa Python merupakan bahasa pemrograman yang sangat mudah dipahami. *Syntax* yang digunakan merupakan syntax umum yang sering kali ditemui di dunia nyata. Seperti *assignment*

(=).

Kita akan sering menggunakan operator *assignment* untuk membuat sebuah variabel baru. Variabel adalah nama yang merujuk kepada suatu nilai.

[1]:
```
perusahaan = 'Bank Rakyat Indonesia'

print(perusahaan)
```

```
Bank Rakyat Indonesia
```

Hal yang perlu diingat adalah Python merupakan bahasa pemrograman **case-sensitive**, sehingga perbedaan karakter akan merujuk pada hal yang berbeda. Seperti `perusahaan` dan `Perusahaan` adalah dua hal yang berbeda.

[2]:
```
'perusahaan' == 'Perusahaan'
```

[2]:
```
False
```

Berdasarkan sifat Python tersebut, penamaan variabel merupakan hal yang penting untuk diperhatikan. Sebagaimana Python yang bersifat *human-readable*. Terdapat beberapa kata yang tidak dapat digunakan sebagai nama variabel. Kata - kata tersebut adalah kata yang termasuk kedalam **Python Keywords**.

Saat ini terdapat 33 *Keywords*: `True`, `False`, `None`, `and`, `as`, `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else except`, `finally`, `for`, `from`, `global`, `if`, `import`, `in`, `is`, `lambda`, `nonlocal`, `not`, `or`, `pass raise`, `return`, `try`, `while`, `with`, `yield`.

Selain penggunaan *keyword*. terdapat beberapa aturan penamaan variable, diantaranya:

1. Tidak boleh diawali dengan angka.
2. Tidak menggunakan special karakter kecuali underscore (_).

4 Introduction to Pandas Library

`pandas` adalah *library* yang digunakan sebagai *tools* analisis data dan struktur data pada Python. Pandas memiliki kemampuan pengolahan data seperti agregasi, manipulasi data, join dan lain sebagainya. Pandas mengolah data dalam bentuk objek `Dataframe`, sehingga memungkinkan pandas mengolah data dari berbagai format file.

pandas official documentation

4.1 Working with DataFrame

Untuk menggunakan `pandas`, kita akan menggunakan fungsi `import`. Ketika kita sudah melakukan *import* library `pandas`, maka kita bisa menggunakan seluruh fungsi dari pandas.

syntax: *pandas.function_name*

[2]:
```
import pandas as pd
pd.__version__
```

[2]: '2.0.0'

Kita akan melakukan pembacaan data. Data yang digunakan adalah data `turnover.csv`

[3]:
```
rice = pd.read_csv("data_input/rice.csv", index_col=0)
rice.head()
```

[3]:
```
   receipt_id  receipts_item_id    purchase_time category sub_category
1     9622257          32369294  7/22/2018 21:19     Rice         Rice  \
2     9446359          31885876  7/15/2018 16:17     Rice         Rice
3     9470290          31930241  7/15/2018 12:12     Rice         Rice
4     9643416          32418582   7/24/2018 8:27     Rice         Rice
5     9692093          32561236  7/26/2018 11:28     Rice         Rice

        format  unit_price  discount  quantity yearmonth
1  supermarket    128000.0         0         1   2018-07
2   minimarket    102750.0         0         1   2018-07
3  supermarket     64000.0         0         3   2018-07
4   minimarket     65000.0         0         1   2018-07
5  supermarket    124500.0         0         1   2018-07
```

Pada code diatas, kita memanfaatkan fungsi `.read_csv()` dari *library* `pandas` yang digunakan untuk membuka file csv dari path yang diberikan dan menyimpannya dalam sebuah objek dataframe bernama `rice`.

Selanjutnya mari kita amati dataframe kita. Secara *default*, pandas akan mengenali baris pertama dari data kita sebagai *header* dari tabel. Tetapi apabila kita tidak mengingikan, kita bisa menambahkan parameter `header = None` pada fungsi awal.

Selain itu coba amati bahwa kita menggunakan `0` untuk menunjukkan baris pertama dari data kita. Hal ini dikarenakan Python menggunakan `0-based indexing`, sehingga index pertama akan dimulai dari angka 0

4.2 Data Types

Dataframe dalam pandas akan memiliki beberapa *series*. **Series** dalam dataframe mengacu kepada kolom. Secara *default*, `pandas` akan secara otomatis menentukan tipe data dari masing - masing kolom. Tetapi, itu tidak selalu benar. Sehingga kita perlu melakukan pengecekan maupun perubahan sehingga tipe data masing - masing kolom sesuai.

Pada bagian ini kita akan mempelajari bagaimana mengecek tipe data dari sebuah dataframe, Jenis tipe data, dan mengubah kolom menjadi tipe data yang sesuai

[4]:
```
print(rice.dtypes)
```

```
receipt_id            int64
receipts_item_id      int64
purchase_time        object
category             object
sub_category         object
format               object
```

```
unit_price          float64
discount              int64
quantity              int64
yearmonth            object
dtype: object
```

`dtypes` merupakan **atribut** dari objek dataframe yang menunjukkan jenis tipe data. Karena `rice` adalah objek pandas (dataframe), `dtypes` akan menunjukkan jenis tipe data untuk setiap kolomnya.

Knowledge check: `.dtypes` and pandas attributes Lihat kode di bawah ini - output apa yang akan muncul dari kode tersebut? Kenapa?

```
x = [2019, 4, 'data science']
x.dtypes
```

Hint: Coba ketik `type(x)` dan cek tipe data `x`.

[ ]:
```
## Your code below



## -- Solution code
```

Coba lihat beberapa contoh `DataFrame.dtypes`:

[5]:
```
employees = pd.DataFrame({
    'name': ['Anita', 'Brian'],
    'age': [34, 29],
    'joined': [pd.Timestamp('20190410'), pd.Timestamp('20171128')],
    'degree': [True, False],
    'hourlyrate': [35.5, 29],
    'division': ['HR', 'Product']
})
employees.dtypes
```

[5]:
```
name                  object
age                    int64
joined        datetime64[ns]
degree                   bool
hourlyrate           float64
division              object
dtype: object
```

[6]:
```
employees
```

[6]:

	name	age	joined	degree	hourlyrate	division
0	Anita	34	2019-04-10	True	35.5	HR
1	Brian	29	2017-11-28	False	29.0	Product

Selanjutnya kita akan melihat tipe data dan isi dari seluruh kolom pada dataframe di atas:

- `name` [`object`]: value berupa text

- `age` [`int`]: value berupa integer
- `joined` [`datetime`]: value berupa tanggal dan waktu
- `degree` [`bool`]: value berupa True/False
- `hourlyrate` [`float`]: value berupa angka floating point
- `division` [`object`]: value berupa text

Di antara kolom-kolom ini, hanya `age` dan `hourlyrate` yang merupakan kolom dengan angka numerik. Salah satu tipe data yang menarik untuk dibahas selanjutnya adalah **categorical values**.

4.2.1 Categorical and Numerical Variable

Tipe data **numeric** adalah tipe data yang berupa angka, tipe data ini meliputi tipe data **Integer** dan **Float**. Integer menyimpan nilai berupa angka bulat tanpa koma, sedangkan Float menyimpan nilai angka desimal.

Tipe data **categoric** adalah tipe data yang menyimpan nilai kategorik. Nilai kategorik merupakan nilai yang dapat dikelompokkan menjadi beberapa kategori dan biasanya berulang di data kita. Contoh nilai kategorik adalah jenis kelamin, golongan darah, rating.

Alasan mengapa kita perlu menggunakan tipe data categorical:

1. ***Dari sisi "Business Perspective"***, hal ini dapat menginformasikan dan memandu seorang *Analyst* pada pertanyaan seperti metode statistik atau tipe plot mana yang digunakan untuk mengolah data.
2. ***Dari sisi teknikal***, ketika kita bekerja dengan tipe data categorical pada pandas, hal ini akan jauh menghemat memori dan menambah kecepatan komputasional.

Dari tipe data kolom pada dataframe `employees` kita akan menganalisis apa saja kolom yang seharusnya memiliki tipe data kategorik. Selanjutnya kita akan mengubah tipe data tersebut dengan menggunakan fungsi `astype()`.

Hal yang harus diperhatikan ketika menggunakan fungsi `astype()` adalah kita harus melakukan assignment ulang pada kolom yang dilakukan perubahan.

Contoh:

```
# convert marital_status to category
employees['marital_status'] = employees['marital_status'].astype('category')

# convert experience to integer
employees['experience'] = employees['experience'].astype('int')
```

Kita akan melakukan perubahan terhadap kolom channel, packet_name, provinsi dan region.

[8]:
```
## Your code below
```

Selanjutnya kita akan cek kembali apakah tipe data kolom tersebut sudah berubah menggunakan `dtypes`

[9]:
```
## Your code below
```

4.3 Exploratory Data Analysis Tools

Exploratory Data Analysis (EDA) adalah proses kita melakukan eksplorasi awal terhadap data, untuk mengetahui lebih lanjut kondisi dan karakteristik data.

`pandas` memiliki built-in function untuk melakukan *Exploratory Data Analysis*. Berikut adalah fungsi yang umum digunakan untuk proses EDA.

- `.head()` dan `.tail()`
- `.shape` dan `.size`
- `.axes`
- `.dtypes`
- `.info()`
- `.describe()`

[8]:
```
rice.head()
```

[8]:
```
   receipt_id  receipts_item_id    purchase_time category sub_category
1     9622257          32369294  7/22/2018 21:19     Rice         Rice  \
2     9446359          31885876  7/15/2018 16:17     Rice         Rice
3     9470290          31930241  7/15/2018 12:12     Rice         Rice
4     9643416          32418582   7/24/2018 8:27     Rice         Rice
5     9692093          32561236  7/26/2018 11:28     Rice         Rice

        format  unit_price  discount  quantity yearmonth
1  supermarket    128000.0         0         1   2018-07
2   minimarket    102750.0         0         1   2018-07
3  supermarket     64000.0         0         3   2018-07
4   minimarket     65000.0         0         1   2018-07
5  supermarket    124500.0         0         1   2018-07
```

Fungsi `.head()` akan menampilkan 5 data pertama dari dataframe. Secara *default* fungsi ini akan menampilkan 5 data pertama. Tetapi apabila ingin mengubah banyak data yang ingin ditampilkan bisa ditambahkan sebuah nilai seperti `.head(3)`. Ini akan menampilkan 3 data pertama dalam daraframe `rice`.

Fungsi `.tail()` merupakan kebalikan dari `.head()`. Fungsi ini akan menampilkan 5 data terakhir.

Setelah mengetahui beberapa baris awal dari data kita. Seringkali muncul pertanyaan, berapa ukuran data kita. Untuk mengetahui hal tersebut kita akan melihat dari atribut `.shape` dan `.size` dari pandas dataframe.

Tidak seperti `.head()` dan `.tail()`, `.shape` dan `.size` merupakan atribut dari pandas dataframe. Perbedaan mendasar dari fungsi dan atribut adalah apabila fungsi selalu diakhiri dengan tanda kurung. Sehingga Anda harus lebih berhati - hati dalam mengenali fungsi dan atribut.

[9]:
```
rice.shape
```

[9]:
```
(12000, 10)
```

Atribut `.shape` menunjukkan berapa banyak baris dan kolom yang ada pada dataframe kita. *Output* dari atribut ini adalah `(baris, kolom)`. Sehingga dari dataframe `rice` kita memiliki 12000

baris data dengan 10 kolom.

[10]:
```
rice.size
```

[10]:
```
120000
```

Berbeda dengan `.shape`, atribut `.size` menunjukkan banyak elemen yang ada dalam dataframe. Karena pada dataframe `rice` memiliki 12000 baris dengan 10 kolom, maka banyak elemen dari dataframe `rice` adalah `baris x kolom` yaitu 120000.

Kedua atribut tersebut hanya akan menampilkan informasi tentang banyak baris, kolom dan elemen. Tetapi tidak dapat menunjukkan informasi nama baris atau kolom. Untuk itu, kita bisa menggunakan atribut `.axes`. Atribut `.axes` akan mengembalikan nilai sebuah list dari dua elemen yaitu sumbu baris dan sumbu kolom.

Secara default, elemen **pertama** dari fungsi `.axes` menunjukkan sumbu baris (nama/index) dan elemen **kedua** menunjukkan sumbu kolom. Untuk mengambil informasi tersebut secara terpisah, Anda bisa menggunakan operator `[]` dan memasukkan urutan dari list. Ingat bahwa python menggunakan **0-based indexing**.

[11]:
```
rice.axes
```

[11]:
```
[Index([    1,     2,     3,     4,     5,     6,     7,     8,     9,    10,
        …
        11991, 11992, 11993, 11994, 11995, 11996, 11997, 11998, 11999, 12000],
       dtype='int64', length=12000),
 Index(['receipt_id', 'receipts_item_id', 'purchase_time', 'category',
        'sub_category', 'format', 'unit_price', 'discount', 'quantity',
        'yearmonth'],
       dtype='object')]
```

Bagaimana jika ingin mengambil elemen dari sumbu baris baris atau kolom saja ?

[15]:
```
# Your code
```

Penggunaan atribut `.axes` untuk mengambil elemen dari sumbu kolom akan memberikan hasil berupa nama - nama kolom. Hal ini sama dengan penggunaan atribut `.columns`.

Apabila Anda ingin mendapatkan informasi lengkap terkait nama kolom, banyak baris serta tipe data untuk tiap baris pandas menyediakan fungsi `.info()` yang dapat digunakan. Jika Anda melakukan proses pengubahan tipe data dengan baik pada bagian sebelumnya, maka output yang Anda dapatkan akan seperti ini.

[12]:
```
rice.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 12000 entries, 1 to 12000
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   receipt_id        12000 non-null  int64
```

```
 1   receipts_item_id  12000 non-null  int64
 2   purchase_time     12000 non-null  object
 3   category          12000 non-null  object
 4   sub_category      12000 non-null  object
 5   format            12000 non-null  object
 6   unit_price        12000 non-null  float64
 7   discount          12000 non-null  int64
 8   quantity          12000 non-null  int64
 9   yearmonth         12000 non-null  object
dtypes: float64(1), int64(4), object(5)
memory usage: 1.0+ MB
```

Fungsi `.info()` akan memberikan informasi terkait kondisi data kita. Informasi yang diberikan meliputi banyak baris dalam data, tipe data setiap kolom dan nilai kosong / *missing value* setiap kolom.

Fungsi `.info()` hanya menampilkan informasi tentang dataframe saja. Untuk melakukan *Exploratory Data Analysis* sering kali membutuhkan perhitungan statistika deskriptif untuk menggambarkan kondisi dari data kita. Fungsi yang digunakan adalah `.describe()`

[13]:
```
rice.describe()
```

[13]:
```
         receipt_id  receipts_item_id     unit_price       discount  \
count  1.200000e+04      1.200000e+04   12000.000000   12000.000000
mean   7.650135e+06      2.457950e+07   70013.146313     835.305750
std    1.873838e+06      7.171105e+06   29905.391437    6207.475704
min    3.173994e+06      9.282023e+06    9395.000000   -4100.000000
25%    5.983209e+06      1.820694e+07   61900.000000       0.000000
50%    7.443618e+06      2.234337e+07   63500.000000       0.000000
75%    9.149786e+06      3.108379e+07   66000.000000       0.000000
max    1.146321e+07      3.842939e+07  219400.000000  320000.000000

           quantity
count  12000.000000
mean       1.332917
std        0.980304
min        1.000000
25%        1.000000
50%        1.000000
75%        1.000000
max       19.000000
```

Fungsi `.describe()` akan menampilkan informasi statistika deskriptif pada data kita. Secara default, fungsi `.describe()` hanya menampilkan statistika deskriptif untuk tipe data **numerik**. Informasi yang ditampilkan oleh fungsi `.describe()` meliputi:

- Count
- Mean
- Standard Deviation
- Minimum Value

- 25th Percentile (Q1)
- 50th Percentile (Q2/Median)
- 75th Percentile (Q3)
- Maximum Value

Untuk menampilkan statistika deskriptif pada kolom dengan tipe data categorik bisa menambahkan parameter `include` kedalam fungsi `.describe()`. Parameter ini bisa diisi **tipe data** yang ingin di tampilkan atau `all` jika untuk seluruh kolom

[14]:
```
rice.describe(include='all')
```

[14]:
```
          receipt_id  receipts_item_id   purchase_time category sub_category
count   1.200000e+04      1.200000e+04           12000    12000        12000  \
unique           NaN               NaN           11609        1            1
top              NaN               NaN  6/14/2018 0:00     Rice         Rice
freq             NaN               NaN               4    12000        12000
mean    7.650135e+06      2.457950e+07             NaN      NaN          NaN
std     1.873838e+06      7.171105e+06             NaN      NaN          NaN
min     3.173994e+06      9.282023e+06             NaN      NaN          NaN
25%     5.983209e+06      1.820694e+07             NaN      NaN          NaN
50%     7.443618e+06      2.234337e+07             NaN      NaN          NaN
75%     9.149786e+06      3.108379e+07             NaN      NaN          NaN
max     1.146321e+07      3.842939e+07             NaN      NaN          NaN

            format     unit_price       discount      quantity yearmonth
count        12000   12000.000000   12000.000000  12000.000000     12000
unique           3            NaN            NaN           NaN        12
top     minimarket            NaN            NaN           NaN   2018-07
freq          7088            NaN            NaN           NaN      1000
mean           NaN   70013.146313     835.305750      1.332917       NaN
std            NaN   29905.391437    6207.475704      0.980304       NaN
min            NaN    9395.000000   -4100.000000      1.000000       NaN
25%            NaN   61900.000000       0.000000      1.000000       NaN
50%            NaN   63500.000000       0.000000      1.000000       NaN
75%            NaN   66000.000000       0.000000      1.000000       NaN
max            NaN  219400.000000  320000.000000     19.000000       NaN
```

Coba gunakan fungsi `.describe()` untuk menampilkan statistika deskriptif untuk kolom - kolom yang bertipe `category`

[19]:
```
# Your code below
```

Kita sudah mempelajari `.dtypes` di subbab sebelumnya. Mari kita latihan dengan menggunakan data `rice`. Apakah kolom-kolom yang ada di Dataframe tersebut sudah benar? Jika belum, lis kolom apa saja yang harus diubah dan perubahan tipe datanya!

[15]:
```
rice.dtypes
```

[15]:
```
receipt_id            int64
receipts_item_id      int64
purchase_time        object
category             object
sub_category         object
format               object
unit_price          float64
discount              int64
quantity              int64
yearmonth            object
dtype: object
```

Bandingkan lis yang sudah Anda buat dengan kode di bawah ini. Kita ubah tipe data kolom `purchase_time` menjadi `datetime` dan mengubah tipe data kategorikal juga:

[16]:
```
rice['purchase_time'] = rice['purchase_time'].astype('datetime64[ns]')
rice[['category', 'sub_category', 'format']] = rice[['category',␣
↪'sub_category', 'format']].astype('category')
rice.dtypes
```

[16]:
```
receipt_id                   int64
receipts_item_id             int64
purchase_time       datetime64[ns]
category                  category
sub_category              category
format                    category
unit_price                 float64
discount                     int64
quantity                     int64
yearmonth                   object
dtype: object
```

Knowledge Check: Exploratory Data Analysis Misalkan kita memiliki dataframe pandas bernama `inventory`

1. Code `inventory.dtypes` dan mendapatkan keluaran berikut. Kolom manakah yang mungkin memerlukan konversi tipe karena sepertinya tipe datanya salah? Pilih semua yang sesuai.

   ☐ `unit_instock`: int64
   ☐ `harga_diskon`: float64
   ☐ `nama_item`: objek
   ☐ `unit_terjual`: objek

2. Apabila ingin mengetahui jumlah kolom di `inventory`. Manakah dari kode berikut yang akan mencetak jumlah kolom saja di `inventory`?

   ☐ `inventory.shape`
   ☐ `inventory.axes[0]`
   ☐ `inventory.axes[1]`

4.4 Indexing & Subsetting with Pandas

Indexing digunakan untuk memilih dan mengambil sebagian data yang diperlukan dalam proses analisa data yang sedang dikerjakan.

Contoh:

- Membandingkan sales pada tahun 2018 vs 2019
- Identifikasi peluang penjualan pada segmen pasar (ex: Wholesale vs Retail)
- Melihat quarter terbaik untuk setiap tahun yang dapat digunakan untuk tujuan promosi
- dan sebagainya

Terdapat beberapa cara untuk melakukan indexing dan subsetting menggunakan `pandas`:

- `.head()`
- `.tail()`
- `.select_dtypes()`
- `.drop()`
- `[]` operator
- `.loc`
- `.iloc`
- Conditional subsetting

Misalkan kita akan melakukan analisis terhadap data integer saja. Maka kita bisa menggunakan fungsi `.select_dtypes()` untuk memilih kolom yang sesuai dengan tipe data yang diinginkan. Parameter yang digunakan adalah `include` dan `exclude`.

Contoh:

- `include = 'category'` artinya kita memilih semua kolom yang bertipe kategorik saja
- `include = ['int', 'float']` artinya kita memilih semua kolom yang bertipe integer dan float
- `exclude = 'category'` artinya kita memilih semua kolom yang **bukan** bertipe kategorik

[17]:
```
rice.select_dtypes(include='int')
```

[17]:

	receipt_id	receipts_item_id	discount	quantity
1	9622257	32369294	0	1
2	9446359	31885876	0	1
3	9470290	31930241	0	3
4	9643416	32418582	0	1
5	9692093	32561236	0	1
…	…	…	…	…
11996	5760491	17555486	0	1
11997	5598782	16999147	0	1
11998	5735850	17434503	3000	1
11999	5678748	17317935	3000	1
12000	5702411	17341559	3000	1

[12000 rows x 4 columns]

Selain menggunakan `.select_dtypes()`, kita bisa menggunakan fungsi `.drop()` untuk membuang

baris atau kolom yang tidak ingin digunakan berdasarkan `axis`. Nilai `axis = 0` menunjukkan baris sedangkan `axis = 1` menunjukkan kolom. Secara *default*, parameter axis bernilai 0.

[18]:
```
rice.drop(2).head()
```

[18]:
```
   receipt_id  receipts_item_id       purchase_time category sub_category
1     9622257          32369294 2018-07-22 21:19:00     Rice         Rice  \
3     9470290          31930241 2018-07-15 12:12:00     Rice         Rice
4     9643416          32418582 2018-07-24 08:27:00     Rice         Rice
5     9692093          32561236 2018-07-26 11:28:00     Rice         Rice
6     9504155          32030785 2018-07-17 18:05:00     Rice         Rice

        format  unit_price  discount  quantity yearmonth
1  supermarket    128000.0         0         1   2018-07
3  supermarket     64000.0         0         3   2018-07
4   minimarket     65000.0         0         1   2018-07
5  supermarket    124500.0         0         1   2018-07
6   minimarket     63500.0         0         1   2018-07
```

Pada code diatas, kita menghilangkan baris index ke-2 dan menggabungkan dengan fungsi `.head()` untuk melihat 5 data pertama. Apabila kita ingin menghapus kolom, kita bisa mengubah nilai `axis` menjadi `1` dan menspesifikkan kolom mana yang ingin dihilangkan.

Apabila terdapat beberapa kolom / baris yang ingin dihilangkan, kita bisa memasukkannya kedalam sebuah list.

Contoh:

```
rice.drop(['purchase_time', 'category'], axis = 1)
```

[23]:
```
# Your code below
```

Perhatikan bahwa fungsi `.drop()` tidak akan mengubah dataframe awal `rice`. Fungsi tersebut hanya menghilangkan kolom/baris dari dataframe. Apabila kita ingin mengubah dataframe maka terdapat dua cara

1. Assignment ulang dataframe yang sudah di drop `py` `rice = rice.drop('category', axis = 1)`
2. Menambahkan parameter `inplace = True` pada fungsi `.drop()` `py` `rice.drop('category', axis = 1, inplace = True)`

Pada umumnya, seringkali Anda ingin melakukan *subsetting* pada sebagian baris tertentu saja. Proses *subsetting* tersebut dapat dilakukan dengan menggunakan operasi `[]`.

Operasi `[]` akan melakukan *subsetting* dengan cara mengiris (*slicing*) index pada dataframe. Formula penulisan operasi tersebut adalah `[start:end]` dengan tetap mengikuti aturan *indexing* dari Python (dimulai dari 0) dimana `start` *inclusive* dan `end` *exclusive*.

Code dibawah ini akan melakukan *slicing* untuk 4 data pertama yaitu index ke 0 ,1, 2, dan 3.

[19]:
```
rice[0:4]
```

[19]:
```
   receipt_id  receipts_item_id       purchase_time category sub_category
1     9622257          32369294 2018-07-22 21:19:00     Rice         Rice  \
2     9446359          31885876 2018-07-15 16:17:00     Rice         Rice
3     9470290          31930241 2018-07-15 12:12:00     Rice         Rice
4     9643416          32418582 2018-07-24 08:27:00     Rice         Rice

        format  unit_price  discount  quantity yearmonth
1  supermarket    128000.0         0         1   2018-07
2   minimarket    102750.0         0         1   2018-07
3  supermarket     64000.0         0         3   2018-07
4   minimarket     65000.0         0         1   2018-07
```

Berbeda dengan drop yang harus menentukan nilai axis, operator `[]` dapat melakukan pengambilan untuk nilai kolom dengan langsung memasukkan nama kolomnya. Seperti biasa, apabila ingin mengambil beberapa kolom harus dimasukkan ke dalam sebuah list.

contoh:

```
# Mengambil kolom category saja
rice['category']

# Mengambil kolom purchase_time dan category
rice[['purchase_time', 'category']]
```

Knowledge Check : Slicing

1. Proses slicing pada beberapa baris dalam dataframe menggunakan operator `[]` dengan memasukkan nilai start dan end. Bagaimana cara kita melakukan slicing pada dataframe apabila ingin mengambil baris ke 8 sampai 12 ?

Hint : Ingat bahwa Python menggunakan 0-based indexing

- ☐ `rice[7:12]`
- ☐ `rice[8:12]`
- ☐ `rice[7:13]`
- ☐ `rice[8:13]`

Selain menggunakan `drop()` dan operator `[]`. Kita bisa juga menggunakan `.loc` dan `.iloc`.

Perbedaan dari `.loc` dan `.iloc` adalah:

- `.loc` merujuk pada nama baris atau kolomnya
- `.iloc` merujuk pada index baris atau kolomnya (harus integer)

Kita akan berfokus pada penggunaan `.iloc` terlebih dahulu. *Syntax* `.iloc` > dataframe.iloc[baris, kolom]

[20]:
```
rice.iloc[3:5 , 1:3]
```

[20]:
```
   receipts_item_id       purchase_time
4          32418582 2018-07-24 08:27:00
5          32561236 2018-07-26 11:28:00
```

Pada *cell* diatas, kita akan melakukan subset pada data `rice` untuk index baris ke 3 dan 4, serta kolom index ke 1 dan 2. Output dari `.iloc` adalah `pandas` dataframe.

Berikut adalah contoh latihan penggunaan `.iloc` yang bisa Anda kerjakan:

1. Tampilkan baris dengan index 1 dan kolom dengan index 2, artinya adalah data pada baris kedua dan kolom `packet_name`.
2. Mengambil baris ke 2 sampai 5 dan kolom dengan `packet_name` sampai `revenue`.
3. Menampilkan semua data pada kolom `packet_name` dan `revenue` saja.

Gabungkan apa yang telah Anda pelajari, coba gunakan `.iloc` untuk subset **2 baris terakhir** dan **4 kolom pertama** dari dataframe di atas! Jika Anda melakukannya dengan benar, output `x` akan berbentuk `pandas` dataframe(`type(x)` adalah objek `DataFrame`).

Bonus: jika Anda ingin tantangan ekstra, coba lakukan operasi ini tiga kali dengan cara;

1. Gunakan hanya `.iloc[ , ]`
2. Gunakan `tail(2)` untuk mendapatkan 2 baris dan gabungkan dengan `.iloc`

3. Gunakan `rice.shape[0]-2:` untuk mendapatkan 2 baris terakhir di operasi `.iloc`

[26]:
```
## Your code below
```

Berbeda dengan `.iloc`, `.loc` melakukan *slicing* dengan menggunakan `label` atau nama baris dan kolom. Kita tetap bisa menggunakan `integer` dalam `.loc` tetapi itu akan diinterpretasikan sebagai label.

Syntax `.loc`: > dataframe.loc[baris , kolom]

Mari kita baca lagi `csv` yang sama, tetapi kali ini kita atur `receipt_id` sebagai label baris:

[21]:
```
rice = pd.read_csv("data_input/rice.csv", index_col=1)
rice = rice.drop('Unnamed: 0', axis=1)
rice.head()
```

[21]:
```
            receipts_item_id    purchase_time category sub_category
receipt_id
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice

                 format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0         0         1   2018-07
9446359      minimarket    102750.0         0         1   2018-07
9470290     supermarket     64000.0         0         3   2018-07
9643416      minimarket     65000.0         0         1   2018-07
9692093     supermarket    124500.0         0         1   2018-07
```

Subsetting baris transaksi yang mempunyai id receipt 9643416 and 5735850 dapat dilakukan dengan `.loc` sebagai berikut:

[22]:
```
rice.loc[[9643416, 5735850], :]
```

[22]:
```
            receipts_item_id     purchase_time category sub_category
receipt_id
9643416             32418582    7/24/2018 8:27     Rice         Rice  \
5735850             17434503  12/13/2017 19:17     Rice         Rice

                 format  unit_price  discount  quantity yearmonth
receipt_id
9643416      minimarket     65000.0         0         1   2018-07
5735850     supermarket     59990.0      3000         1   2017-12
```

4.4.1 Conditional Subsetting

Selain menggunakan `.loc` dan `.iloc`, kita dapat melakukan *subsetting* berdasarkan kondisi tertentu. Hal ini dinamakan *Conditional Subsetting*.

Dengan menggunakan *Conditional Subsetting*, Kita dapat memilih data menggunakan kriteria tertentu. Berikut adalah contoh *conditional subsetting* yang bisa dilakukan pada dataframe `rice`

- `.format == 'supermarket'` untuk memilih semua transaksi yang formatnya adalah supermarket
- `.unit_price >= 400000` untuk memilih semua transaksi dengan harga satuan sama dengan atau lebih besar dari 400000.
- `.quantity != 0` untuk memilih semua transaksi yang kuantitas pembeliannya **bukan** 0

Syntax penulisan untuk *conditional subsetting* adalah:

df[df['column_name']]

Contoh *comparison* operator:

- `==`: sama dengan
- `!=`: tidak sama dengan / bukan
- `>` : Lebih dari
- `<` : Kurang dari
- `>=`: Lebih dari sama dengan
- `<=`: Kurang dari sama dengan

[23]:
```
rice = pd.read_csv("data_input/rice.csv", index_col=1)
rice = rice.drop('Unnamed: 0', axis=1)
rice[rice.discount != 0].head()
```

[23]:
```
            receipts_item_id    purchase_time category sub_category
receipt_id
9767244             32763935   7/29/2018 8:25     Rice         Rice  \
9483559             31978405  7/16/2018 16:24     Rice         Rice
9841041             33010608   7/29/2018 8:21     Rice         Rice
```

```
9284943             31450853   7/6/2018 12:05     Rice         Rice
9853312             33070898  7/30/2018 13:54     Rice         Rice

                 format  unit_price  discount  quantity yearmonth
receipt_id
9767244     supermarket     59900.0      4100         1   2018-07
9483559     supermarket     46900.0      4300         1   2018-07
9841041     supermarket    117900.0     10100         1   2018-07
9284943     supermarket     59900.0      4100         1   2018-07
9853312     supermarket    119890.0      8110         1   2018-07
```

4.4.2 Referencing and Copying

Pada Python, tanda sama dengan (=) bisa digunakan untuk merujuk ke sebuah Dataframe. Sekarang mari kita coba untuk dengan menggunakan dataframe `rice`

[24]:
```
rice = pd.read_csv("data_input/rice.csv", index_col=1)
rice = rice.drop('Unnamed: 0', axis=1)

rice.head()
```

[24]:
```
            receipts_item_id    purchase_time category sub_category
receipt_id
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice

                 format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0         0         1   2018-07
9446359      minimarket    102750.0         0         1   2018-07
9470290     supermarket     64000.0         0         3   2018-07
9643416      minimarket     65000.0         0         1   2018-07
9692093     supermarket    124500.0         0         1   2018-07
```

Lalu, mari kita buat dataframe baru, `rice_july`, yang menyimpan data penjualan beras khususnya pada bulan Juli 2018. Misalkan pada periode tersebut, kita memberikan diskon 15 persen untuk semua produk kita, maka kita juga mengganti semua nilai dalam `rice_july.discount` menjadi 15.

[25]:
```
rice_july = rice
rice_july['discount'] = 15
rice_july.head()
```

[25]:
```
            receipts_item_id    purchase_time category sub_category
receipt_id
```

```
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice

                 format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0        15         1   2018-07
9446359      minimarket    102750.0        15         1   2018-07
9470290     supermarket     64000.0        15         3   2018-07
9643416      minimarket     65000.0        15         1   2018-07
9692093     supermarket    124500.0        15         1   2018-07
```

Data pada `rice_july` sudah berubah pada kolom `discount`. Tetapi apabila kita lihat pada data `rice`, nilai pada kolom `discount` juga berubah

[26]:
```
rice.head()
```

[26]:
```
            receipts_item_id    purchase_time category sub_category
receipt_id
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice

                 format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0        15         1   2018-07
9446359      minimarket    102750.0        15         1   2018-07
9470290     supermarket     64000.0        15         3   2018-07
9643416      minimarket     65000.0        15         1   2018-07
9692093     supermarket    124500.0        15         1   2018-07
```

Penjelasan dari apa yang terjadi pada dua kode sebelumnya adalah ketika kita mengeksekusi kode `rice_july = rice` kita **TIDAK** membuat sebuah objek baru. Kita hanya membuat sebuah variabel bernama `rice_july` yang me*refer* satu objek yang sama.

Untuk mengatasi permasalahan tersebut, kita bisa menggunakan fungsi `.copy()`. Fungsi `copy()` akan **menduplikasi objek** dan **membuat sebuah objek baru**, sehingga terdapat dua objek yang berbeda dan perubahan pada salah satu objek **tidak** akan berpengaruh pada objek yang lain.

[27]:
```
rice = pd.read_csv("data_input/rice.csv", index_col=1)
rice = rice.drop('Unnamed: 0', axis=1)

rice_july = rice.copy()
rice_july['discount'] = 15
rice_july.head()
```

[27]:

```
            receipts_item_id    purchase_time category sub_category
receipt_id
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice

                 format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0        15         1   2018-07
9446359      minimarket    102750.0        15         1   2018-07
9470290     supermarket     64000.0        15         3   2018-07
9643416      minimarket     65000.0        15         1   2018-07
9692093     supermarket    124500.0        15         1   2018-07
```

[28]:

```
rice.head()
```

[28]:

```
            receipts_item_id    purchase_time category sub_category
receipt_id
9622257             32369294  7/22/2018 21:19     Rice         Rice  \
9446359             31885876  7/15/2018 16:17     Rice         Rice
9470290             31930241  7/15/2018 12:12     Rice         Rice
9643416             32418582   7/24/2018 8:27     Rice         Rice
9692093             32561236  7/26/2018 11:28     Rice         Rice

                 format  unit_price  discount  quantity yearmonth
receipt_id
9622257     supermarket    128000.0         0         1   2018-07
9446359      minimarket    102750.0         0         1   2018-07
9470290     supermarket     64000.0         0         3   2018-07
9643416      minimarket     65000.0         0         1   2018-07
9692093     supermarket    124500.0         0         1   2018-07
```

Coursebook: Practical Statistics

- Bagian 2 Audit Analytics untuk Bank Rakyat Indonesia
- Durasi: 7 Jam
- *Last Updated*: December 2023

- *Coursebook* ini disusun dan dikurasi oleh tim produk dan instruktur dari Algoritma Data Science School

5 Background

Coursebook ini merupakan bagian dari **BRI Audit Analytics** yang disiapkan oleh Algoritma. *Coursebook* ini ditujukan hanya untuk khalayak terbatas, yaitu individu dan organisasi yang menerima *coursebook* ini langsung dari organisasi pelatihan. Tidak boleh direproduksi, didistribusikan, diterjemahkan, atau diadaptasi dalam bentuk apapun di luar individu dan organisasi ini tanpa izin.

Algoritma adalah pusat pendidikan *data science* yang berbasis di Jakarta. Kami menyelenggarakan *workshop* dan program pelatihan untuk membantu para profesional dan mahasiswa dalam menguasai berbagai sub-bidang *data science* yaitu: *data visualization, machine learning,* statistik, dan lain sebagainya.

6 Training Objectives

Descriptive Statistics

- Understanding 5 number summary
- Central tendency measure
- Measure of spread
- Variable relationship
- Z Score and Central Limit Theorem

Inferential Statistics

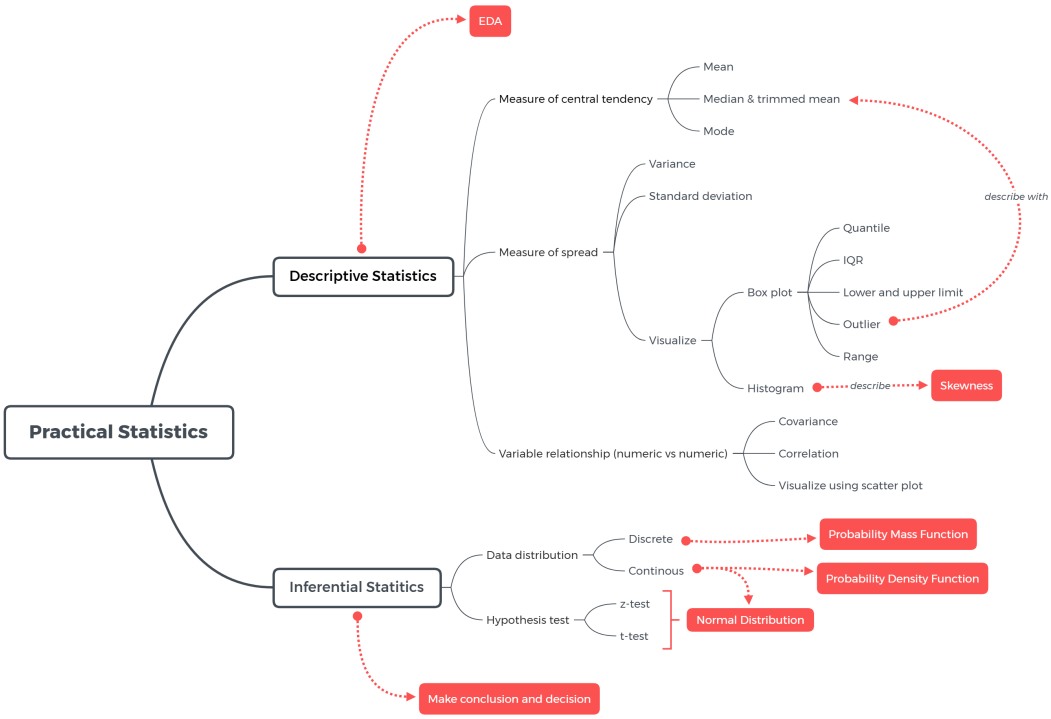
- Probability Density Function
- Data distributions
- Hypothesis test
- Error and confidence intervals

7 Practical Statistics

Practical Statistics merupakan salah satu bagian penting dalam pengolahan data sehingga mendukung *busines case* yang ingin diangkat. Practical Statistics berisi kaidah statistika yang banyak diterapkan dalam praktik data science agar dapat **memahami dan mengolah data dengan tepat**.

Secara umum, Practical Statistics terbagi 2, masing-masing membantu kita dalam hal tertentu:

- **Descriptive Statistics**: meringkas informasi dalam data agar terambil insight secara cepat. Nilai-nilai yang didapatkan merupakan rangkuman dari data, tujuannya untuk menggambarkan keadaan data secara umum.
- **Inferential Statistics**: menyimpulkan sesuatu tentang kondisi di lapangan, berdasarkan data yang kita punya (sample -> population).



Untuk lebih memahami *practical statistics*, mari kita melakukan analisis menggunakan data asli.

[1]:
```
import pandas as pd
import numpy as np #perhitungan statistik
import matplotlib.pyplot as plt # untuk visualisasi
import math # perhitungan statistik
from scipy import stats #untuk perhitungan statistik
import seaborn as sns # untuk visualisasi
```

8 Study Case: Credit Card Balance Analysis

1. Business Question

Credit Card Balance Analysis, atau Analisis Saldo Kartu Kredit, dilakukan sebagai bagian dari analisis debitur dalam sebuah perusahaan kartu kredit. Hasil analisis dapat menentukan debitur mana yang memiliki risiko pembayaran kredit yang tinggi, atau bagaimana behavior debitur. Selain itu, menggabungkan data saldo kredit dengan informasi seperti limit kredit dapat membantu menghitung pemanfaatan kredit kartu, informasi yang berpengaruh pada Rating kredit seorang pemegang kartu.

Asumsi data:

- Balance dihitung sebagai jumlah semua transaksi selama periode penagihan/billing cycle. Sebagai contoh, jika seorang pemegang kartu mengeluarkan `$400`, `$500`, dan `$600` dalam 3 bulan, maka saldo rata-rata akan dicatat sebagai `$500`.

Kita sebagai tim data diminta untuk **menganalisa performa Credit Card Balance** nasabah. Data tersimpan dalam folder data_input dengan nama file `credit_card.csv`, gunakan `stringAsFactors = T` supaya kolom bernilai string berubah langsung menjadi tipe factor.

2. Read Data

[2]:
```
cc = pd.read_csv('data_input/CC.csv')
cc.head()
```

[2]:
```
    Income  Limit  Rating  Cards  Age  Education  Gender Student Married
0   14.891   3606     283      2   34         11    Male      No     Yes  \
1  106.025   6645     483      3   82         15  Female     Yes     Yes
2  104.593   7075     514      4   71         11    Male      No      No
3  148.924   9504     681      3   36         11  Female      No      No
4   55.882   4897     357      2   68         16    Male      No     Yes

   Ethnicity  Balance
0  Caucasian      333
1      Asian      903
2      Asian      580
3      Asian      964
4  Caucasian      331
```

Deskripsi Kolom

- `Income`: Besaran gaji nasabah per tahun (dalam $10000)
- `Limit` : Besaran kredit limit
- `Rating` : Skor yang diberikan kepada individu berdasarkan kelayakan kreditnya. Semakin besar maka semakin baik
- `Cards` : Jumlah banyaknya kartu kredit yang dimiliki oleh nasabah
- `Age` : Usia nasabah
- `Education` : Level/lamanya pendidikan yang ditempuh oleh nasabah
- `Gender`: Jenis kelamin nasabah
  - Male
  - Female
- `Student` : Apakah nasabah seorang pelajar atau bukan
  - Yes → Pelajar
  - No → Bukan pelajar
- `Married`: Status pernikahan
  - Yes → Sudah menikah
  - No → Belum menikah
- `Ethnicity`: Etnis nasabah
  - African American
  - Asian

  - Caucasian
- `Balance`: Rata-rata jumlah saldo kartu kredit

9 Descriptive Statistics

Descriptive Statistics membantu kita **menggambarkan karakteristik** dari data, sehingga berguna dalam proses **Exploratory Data Analysis (EDA)**. Terdapat 3 hal pada descriptive statistics:

- Ukuran pemusatan data (Measure of Central Tendency)
- Ukuran penyebaran data (Measure of Spread)
- Hubungan antar data (Variable Relationship)

9.1 Measure of Central Tendency

Ukuran pemusatan data adalah **suatu nilai yang cukup untuk mewakili seluruh nilai pada data**.

9.1.1 Mean

Cara paling umum untuk membuat perkiraan nilai tunggal dari data yang banyak adalah dengan merata-ratakannya.

- Formula:

$$\frac{\sum x_i}{n}$$

- Fungsi pada Python: `mean()`

Contoh:

Berapa rata-rata `Rating` atau skor yang diberikan kepada nasabah berdasarkan kelayakan kreditnya?

[14]:
```
# code here



# end of the code
```

- Sifat nilai mean: **sensitif terhadap outlier**

  Outlier adalah nilai ekstrim yang jauh dari observasi lainnya. Kurang tepat apabila menggunakan nilai mean yang diketahui ada data outliernya.

Contoh lain:

Ada sebuah Kantor Cabang BRI di daerah Bekasi yang merekap jumlah pengunjung per bulan.

Dengan nilai mean:

[7]:
```
# data pengunjung
pengunjung = pd.Series([55, 50, 40, 70, 60, 45, 35, 35, 60, 1000, 250, 70])
```

[13]:
```
# rata-rata pengunjung


# end of the code
```

Apakah nilai mean di atas dapat diandalkan? _____

Nilai mean tidak dapat diandalkan karena terdapat outlier

Masalah ini dapat diatasi oleh nilai **median**.

9.1.2 Median

Median atau nilai tengah diperoleh dengan mengurutkan data terlebih dahulu kemudian mencari nilai tengah dari data.

- Baik untuk data yang memiliki **outlier** atau berdistribusi **skewed** (condong kiri/kanan)
- Fungsi pada Python: `median()`

Mari hitung ulang nilai pusat dari `pengunjung` menggunakan median:

[9]:
```
# median


# end of the code
```

[9]: 147.5

[10]:
```
# bandingkan dengan mean


# end of the code
```

[10]: 147.5

Untuk nilai desimal, apabila tidak sesuai dengan konteks bisnis (Pengunjung), chaining hasil dengan fungsi rounding `round()`

[11]:
```
pengunjung.mean().round()
```

[11]: 148.0

9.1.3 Modus (Mode)

Modus berguna untuk mencari nilai yang paling sering muncul (frekuensi tertinggi).

- Modus digunakan untuk data kategorik
- Fungsi pada Python: `mode()`

Contoh:

Berasal dari `Ethnicity` mana nasabah di Bank tersebut paling banyak berasal?

[12]:
```
cc['Ethnicity'].mode()
```

[12]:
```
0    Caucasian
Name: Ethnicity, dtype: object
```

Modus untuk `Ethnicity` adalah _____

9.1.4 Knowledge Check

Dari pernyataan berikut, jawablah benar atau salah. Apabila salah tuliskan pernyataan yang benar.

1. Median adalah pusat data yang hanya melibatkan sebagian data dalam perhitungannya.

☐ Benar
☐ Salah

2. Mean adalah pusat data yang sensitif terhadap outlier.

☐ Benar
☐ Salah

3. Nilai pusat data yang cocok untuk tipe data kategorik adalah modus.

☐ Benar
☐ Salah

9.2 Measure of Spread

Ukuran penyebaran data mewakili seberapa menyebar atau beragam data kita.

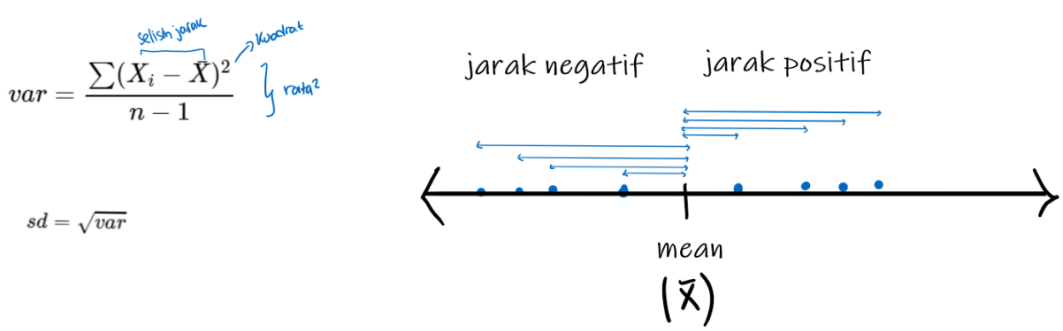
9.2.1 Variance

Variance menggambarkan seberapa beragam suatu data numerik tunggal menyebar dari pusat datanya

- Formula variance:

$$var = \frac{\sum(X_i - \bar{X})^2}{n-1}$$

- Variance tidak dapat diinterpretasikan karena satuannya dalam kuadrat.
- Fungsi di Python: `var()`



Contoh:

Bank BRI sedang dalam rencana membuka kantor cabang baru. Bank BRI menyeleksi daerah mana yang cocok untuk cabang baru mereka. Mereka mengumpulkan informasi harga sewa bangunan di daerah A dan B sebagai berikut:

[15]:
```
# harga dalam satuan juta
harga_A = pd.Series([400,410,420,400,410,420,400,410,420,400,410,420,400])
harga_B = pd.Series([130,430,650,540,460,320,380,550,650,470,330,140,270])
```

Bandingkan rata-rata harga bangunan kedua daerah:

[19]:
```
# code here



# end of the code
```

Mari bandingkan dari sisi lain, yaitu tingkat keberagaman data (variance). Daerah mana yang harganya lebih bervariasi?

[18]:
```
# code here



# end of the code
```

Daerah manakah yang lebih baik untuk dijadikan area perkantoran?

...

Karakteristik Variance

- Skala variance dari 0 sampai tak hingga. Semakin besar nilainya maka artinya semakin menyebar dari pusat datanya (mean).
- Variance memiliki satuan kuadrat, sehingga tidak dapat langsung diinterpretasikan. Biasanya digunakan untuk membandingkan dengan nilai var lain dengan satuan yang sama.
- **Nilai variansi sangat bergantung dengan skala data**. Hati-hati apabila membandingkan antar variabel yang berbeda skala.

9.2.2 Standard Deviation

Standard deviation menggambarkan **seberapa jauh simpangan nilai yang dianggap umum, dihitung dari titik pusat (mean) nya.** Kita dapat menentukan apakah suatu nilai dikatakan menyimpang dari rata-rata namun masih dikatakan umum, atau sudah tidak umum.

Karena dihitung dengan **mengakarkan variance**, satuannya sudah sesuai dengan data asli dan bisa diinterpretasikan.

- Formula standar deviasi:
$$sd = \sqrt{var}$$
- Fungsi di Python: `std()`

[20]:
```
# standar deviasi harga_A & harga_B



# end of code
```

[21]:
```
# tinjau nilai mean harga_A & harga_B



# end of the code
```

Interpretasi nilai normal/wajar : mean +- sd (karena satuan mean dan sd sama, yaitu jutaan rupiah)

- Harga sewa pada daerah A umumnya jatuh pada interval _____
- Harga sewa pada daerah B umumnya jatuh pada interval _____

Business question

Apabila kita ditawarkan suatu bangunan di daerah B dengan harga 800, apakah harga tersebut masih wajar? Apakah sebaiknya kita membeli bangunan tersebut? Hubungkan dengan nilai mean dan standar deviasi yang diperoleh.

Hitung range "harga normal" daerah B:

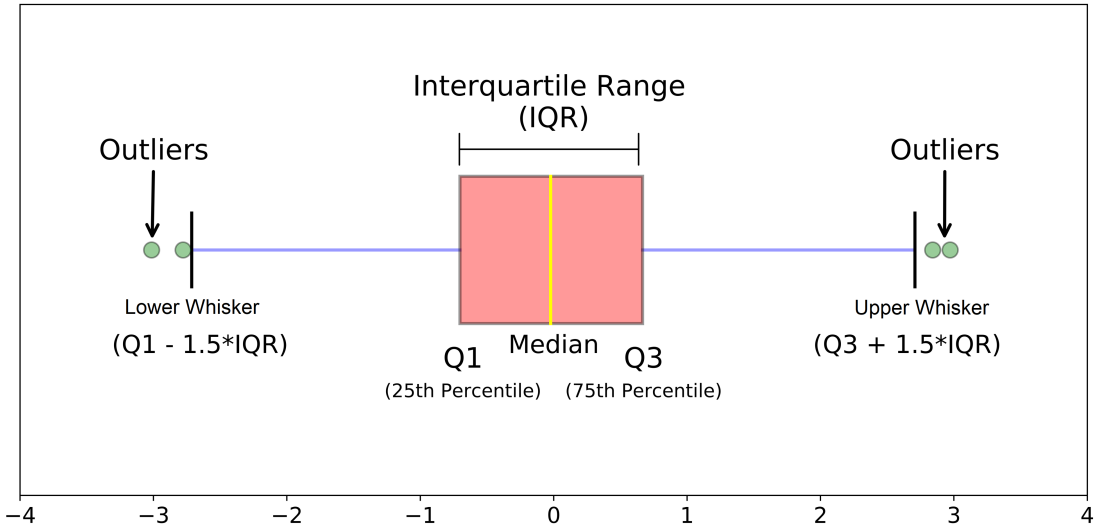
[ ]:
```
# code here



# end of the code
```

…

9.2.3 Range using `boxplot()`

Distribusi data numerik pada umumnya divisualisasikan dengan `boxplot()`, yang meliputi komponen:

- Box: menggambarkan Q1, Q2 (median), dan Q3
  - Kuartil 1 (Q1): nilai ke 25%
  - Kuartil 2 (Q2 atau median): nilai ke 50% (nilai tengah)
  - Kuartil 3 (Q3): nilai ke 75%
  - Interquartile Range (IQR): selisih antara Q3 dan Q1
- Whisker: pagar bawah dan atas (PENTING: hati-hati, nilai ini bukan nilai minimum dan maksimum data)
- Data outliers: nilai ekstrim data yang berada di luar pagar bawah dan atas



Beberapa hal yang harus diperhatikan dalam boxplot:

- Banyaknya data dari Q1 ke nilai minimum (bukan pagar bawah) adalah 25%
- Banyaknya data dari Q1 ke Q2 adalah 25%
- Banyaknya data dari Q2 ke Q3 adalah 25%
- Banyaknya data dari Q3 ke nilai maksimum (bukan pagar atas) adalah 25%

Insight yang dapat diperoleh dari boxplot:

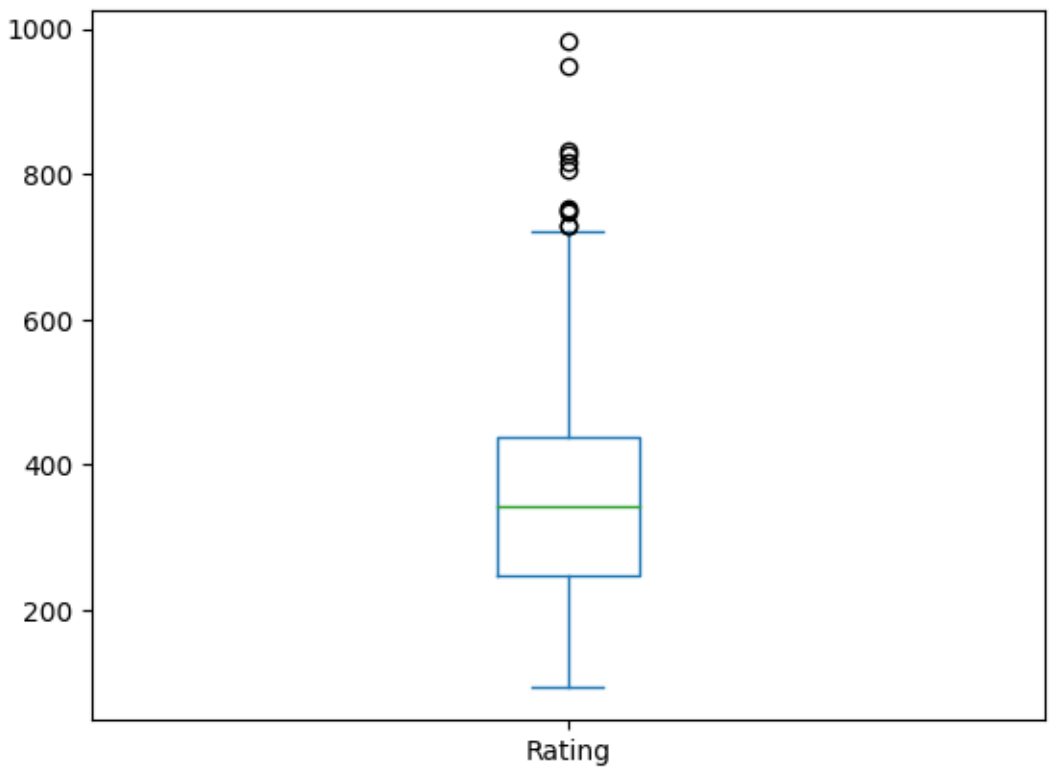
1. Pusat data dengan median (Q2)
2. Sebaran data dengan IQR (lebar kotak)
3. Outlier, nilai ekstrim pada data
4. Bentuk distribusi data:

- box yang berada ditengah = **distribusi normal**
- box yang mendekati batas bawah = **distribusi skewed kanan**
- box yang mendekati batas atas = **distribusi skewed kiri**

Contoh:

Visualisasikan sebaran data `Rating` dari data `cc`! Analisis informasi yang didapatkan.

[23]:
```
cc['Rating'].plot.box();
```



- Apakah data memiliki outlier?

  ...

- Central tendency (mean, median, modus) mana yang cocok dipakai untuk data ini?

  ...

- Bagaimana bentuk distribusi data?

  ...

9.3 Variable Relationship

Karena pada data kita punya banyak kolom atau variabel, kita juga ingin tahu hubungan antar variabel dalam data kita.

Ukuran yang digunakan untuk melihat **hubungan linear** antara dua variabel numerik.

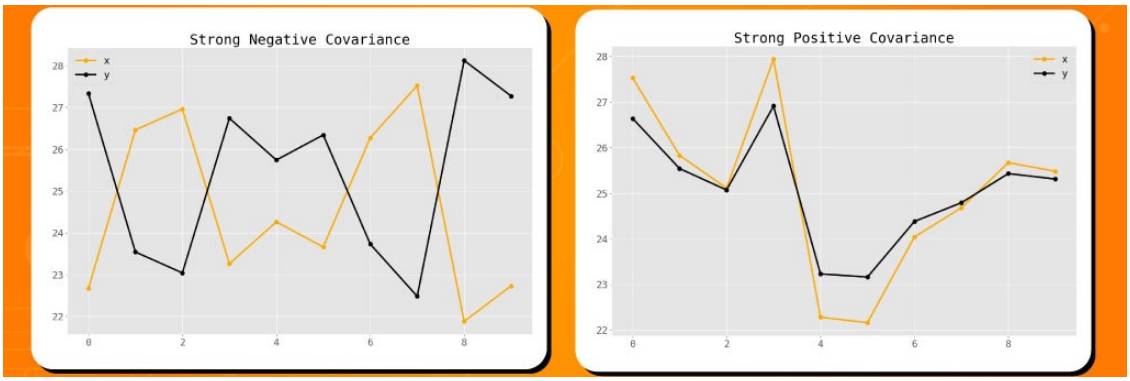
9.3.1 Covariance

Covariance menunjukkan bagaimana variansi 2 data (variable yang berbeda) bergerak bersamaan

- Formula Covariance:

$$Cov(X, Y) = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \mu_X)(Y_i - \mu_Y)$$

- Fungsi di Python: `cov()`



- Nilai covariance positif mengindikasikan pergerakan nilai yang searah / berbanding lurus.
- Nilai covariance negatif mengindikasikan pergerakan nilai yang berbalik arah.

Contoh:

Hitunglah covariance antara `Income` dengan `Rating` pada data `cc` . Bagaimana hubungannya?

[ ]:
```
# code here


# end of code
```

Interpretasi nilai: _____

> Kelemahan: Seperti variance, covariance tidak memiliki batasan nilai untuk mengukur kekuatan hubungan antar dua variabel (-inf s.d inf), sehingga kita hanya bisa mengetahui apakah hubungannya positif atau negatif. Oleh karena itu, hadir **correlation**.

9.3.2 Correlation

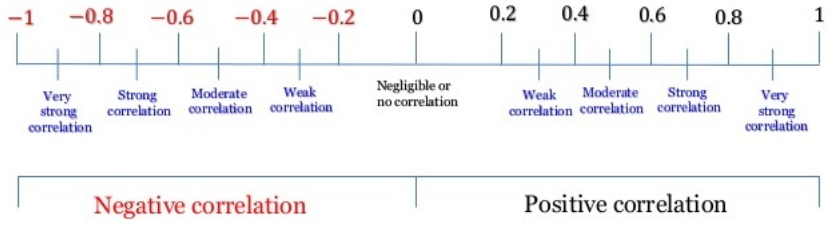
Correlation memampatkan nilai covariance dari -inf s.d inf menjadi **-1 s.d 1** sehingga bisa diukur kekuatan hubungan antar data (variable).

- Formula Correlation:

$$Cor(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}}$$

* Fungsi di Python: `corr()`

- Nilai korelasi mengindikasikan kekuatan hubungan antara dua variable numerik sebagai berikut:



Bila korelasi dua variable numerik mendekati: - -1 artinya korelasi negatif kuat - 0 artinya tidak berkorelasi - 1 artinya korelasi positif kuat

Contoh:

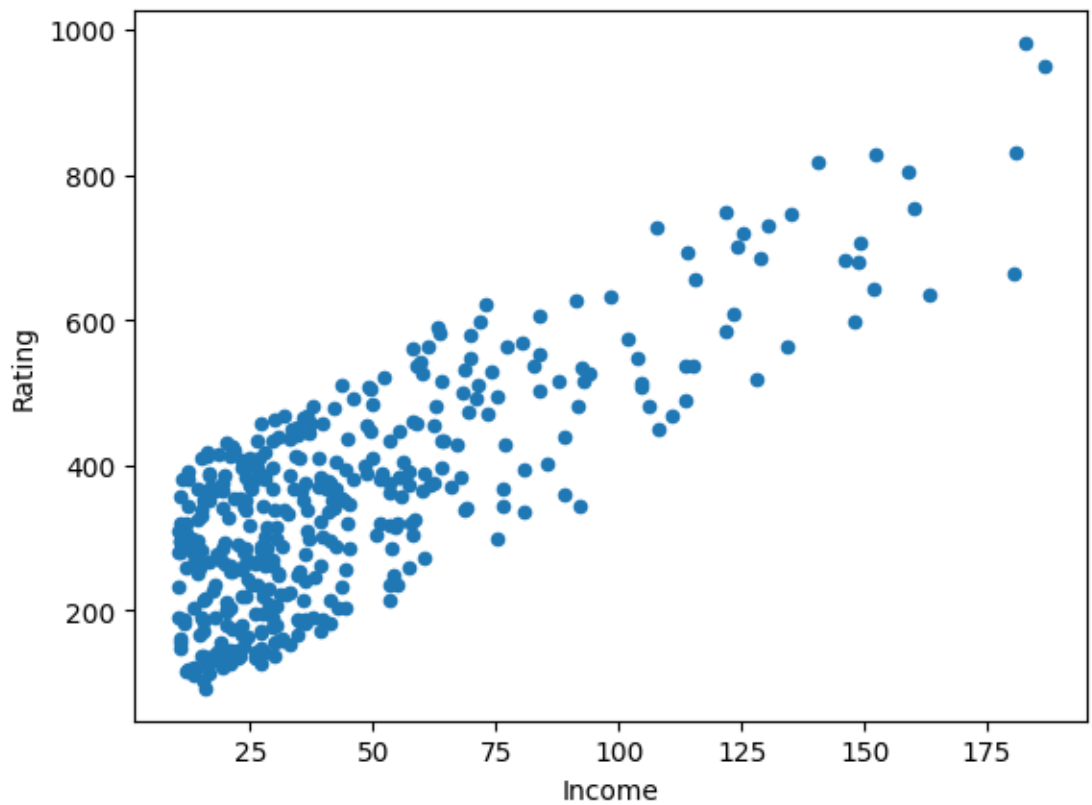
Adakah korelasi antara `Income` dengan `Rating` pada data `cc` . Bagaimana hubungan dan kekuatannya?

[ ]:
```
# code here


# end of code
```

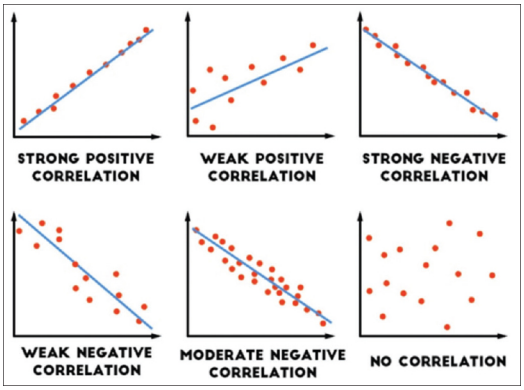
Jawaban: _____

Visualisasi korelasi dengan scatter plot:

[25]:
```
cc.plot.scatter(x='Income',
                y='Rating');
```



Ilustrasi correlation:



9.3.3 Knowledge Check

Dari pernyataan berikut, jawablah benar atau salah. Apabila salah, tuliskan pernyataan yang benar.

1. Ketika korelasi variabel A dan B bernilai -1 artinya tidak ada korelasi antara nilai A dan B.

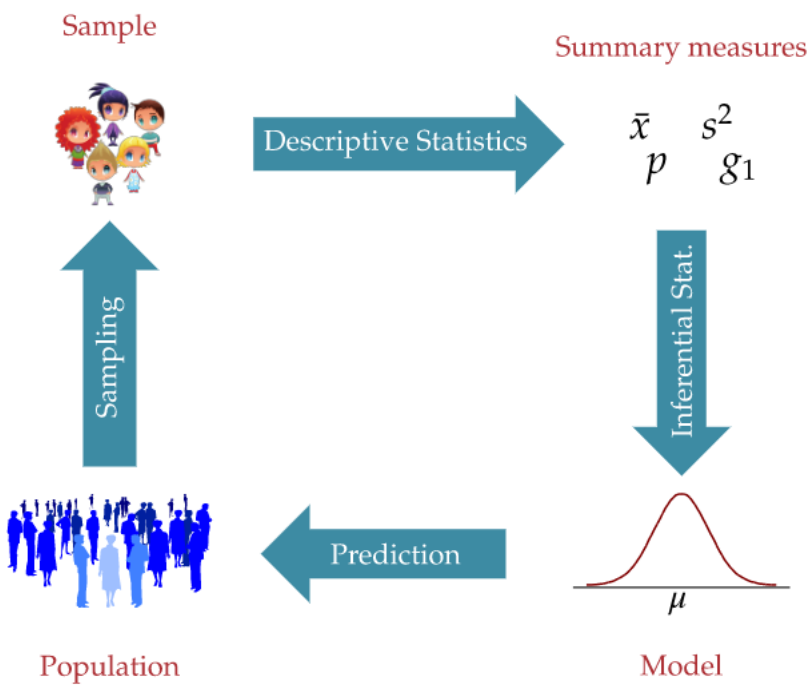
☐ Benar
☐ Salah

2. Scatter plot dapat digunakan untuk menggambarkan hubungan antara dua variabel numerik.

☐ Benar
☐ Salah

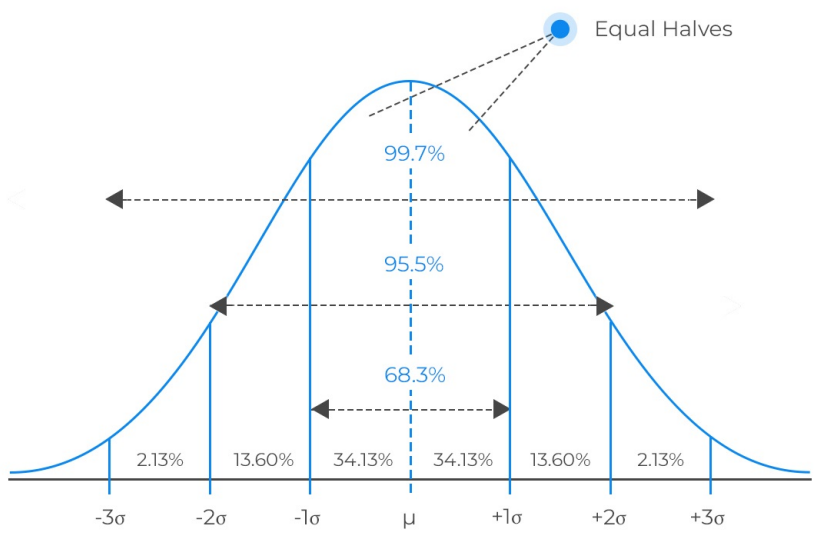
10 Inferential Statistics

Inferential Statistics membantu kita **menarik kesimpulan tentang keseluruhan data (populasi) dengan menggunakan sebagian informasinya saja (sampel)**



Setiap data memiliki distribusi. Distribusi data yang spresial dan berperan dalam inferential statistics adalah **distribusi normal**

10.1 Normal Distribution



Source: Quantitative Methods – Learning Sessions (Analyst Prep, 2019)

Karakteristik:

- Kurva membentuk lonceng simetris, artinya puncaknya adalah titik pusat (mean = median)
- Luas area dibawah kurva = 1 (menyatakan probabilitas)

- Persebaran data:
  - 68% data berada di rentang +- 1 standar deviasi dari mean
  - 95% data berada di rentang +- 2 standar deviasi dari mean
  - 99.7% data berada di rentang +- 3 standar deviasi dari mean
- **Standar normal baku** adalah distribusi normal dimana mean = 0 dan standar deviasi = 1.

Distribusi normal banyak digunakan pada inferensial statistik karena dicetuskannya **Central Limit Theorem**.

> Semakin bertambahnya jumlah sampel yang diambil secara acak, maka **distribusi rata-rata sampel** akan mengikuti distribusi normal

Karakteristik distribusi normal inilah yang dimanfaatkan untuk penghitungan inferensial statistik:

- **Menghitung Probabilitas:**
  - Probability Mass Function -> diskrit/kategorik
  - Probability Density Function -> kontinu/numerik
- **Membuat Confidence Interval**
- **Uji Hipotesis**

10.2 Probability Mass Function

- Menghitung peluang untuk data diskrit, contoh:
  - peluang hujan/tidak hujan
  - peluang produk yang terjual
  - peluang nasabah good credit/bad credit
- Formula: jumlah kejadian terjadi dibagi dengan jumlah kejadian total

Contoh:

Terdapat 100 nasabah dari sebuah Bank, 90 diantaranya merupakan nasabah dengan status good (good credit), sedangkan sisanya sebanyak 90 adalah status bad (bad credit). Berapakah peluang nasabah bad credit?

[ ]:
```
# code here



# end of code
```

10.3 Probability Density Function

- Menghitung probability data **kontinu**. Data kontinu merupakan data yang memiliki nilai dalam rentang tertentu, dan bisa memiliki angka desimal atau pecahan, contohnya:
  - tinggi badan
  - rating nasabah
  - profit/revenue
- Tahapan:
  1. Hitung Z-score (ubah nilai data asli ke standar normal baku = Z-score standardization)
  2. hitung peluang berdasarkan Z-score dengan menggunakan fungsi `pnorm()`
- Formula Z-score:

$$Z = \frac{x - \mu}{\sigma}$$

Keterangan:

- Z = Z-score
- x = titik data
- μ = mean
- σ = standar deviasi

  Z-score merupakan sebuah nilai yang merepresentasikan **berapa standard deviasi data tersebut menyimpang dari rata-ratanya**

Contoh

Tinggi badan pria dewasa di Indonesia berdistribusi normal dengan rata-rata 165 cm dan standar deviasi 10 cm. Berapa peluang pria dewasa di Indonesia memiliki tinggi badan > 180 cm?

Diketahui:

- mean = 165
- stdev = 10
- titik data = 180cm

[28]:
```
mean = 165
std = 10
titik_data = 180
```

[ ]:
```
# code here
from scipy.stats import norm

# hitung Z-score lalu ubah jadi peluang
Z_score = ....
```

[ ]:
```
# menghitung peluang
peluang = ....
```

Insight: Peluang pria dewasa di Indonesia memiliki tinggi badan > 180 cm _____

10.4 Confidence Interval

Confidence interval (selang kepercayaan) berguna untuk menduga nilai mean populasi dengan sebuah interval. Menebak dengan sebuah interval akan meminimalisir error dibandingkan hanya dengan menebak satu nilai.

- Formula:

$$CI = \bar{x} \pm Z_{\frac{\alpha}{2}} * SE$$

- Keterangan:
  - $\bar{x}$ = rata-rata sampel
  - $Z_{\frac{\alpha}{2}}$ = Z-score ketika alpha/2

  - α = tingkat error yang ditolerasi
  - tingkat kepercayaan = 1-α
  - SE = standard error

SE mengukur kebaikan sampel dalam mewakilkan populasi. Semakin kecil, maka sampel semakin representatif (baik).

$$SE = \frac{\sigma}{\sqrt{n}}$$

- Ket:
  - σ = standar deviasi populasi
  - n = jumlah sampel
- Tahapan:
  - hitung mean sampel
  - hitung standar deviasi sampel & SE
  - tentukan tingkat kepercayaan & α
  - tentukan Z alpha/2
  - hitung confidence interval

Contoh

Dari data `cc` yang berisikan sampel **400 nasabah** suatu Bank diketahui memiliki rata-rata `Balance` kredit sebesar **520**. Semisal diketahui Bank tersebut memiliki standard deviasi populasi untuk Balance sebesar **465**.

Berapakah confidence interval untuk rata-rata Balance seluruh nasabah? Gunakan tingkat kepercayaan 95%!

1. Diketahui:

- mean sampel = _____

- stdev populasi = _____
- jumlah sampel (n) = _____

2. Hitung nilai SE

[ ]:
```
# code here
import math
from scipy import stats

# std populasi dibagi akar n
SE = ...
```

2. Tentukan tingkat kepercayaan dan alpha

- Tingkat kepercayaan: 95%
- alpha (tingkat error): 100% - 95% = 5%, artinya kita mentoleransi error sebesar 5%, bahwa mungkin saja rata-rata Balance nasabah aslinya terletak di luar Confidence Interval

[ ]:
```
alpha =

print('alpha: ', alpha)
```

3. Hitung Z alpha/2

alpha dibagi 2 karena ingin membuat batas bawah dan batas atas (dalam dunia statistika dikenal sebagai two-tailed)

[ ]:
```
# code here
from scipy.stats import norm

# luas di bawah kurva - kedua bagian hijau
Z = ....
```

4. Hitung confidence interval

`CI = mean -+ (Z * SE)`

[ ]:
```
lower = mean - (Z*SE)
upper = mean + (Z*SE)
```

Kesimpulan: _____

10.5 Hypothesis Testing

Uji hipotesis bertujuan untuk menguji **dugaan**. Uji hipotesis sering disebut juga sebagai **uji signifikansi** yang digunakan untuk menguji apakah suatu treatment memberikan perubahan/pengaruh signifikan terhadap suatu kondisi.

Istilah-istilah:

- Hipotesis: dugaan sementara yang harus diuji
  - H_0 / null hypothesis:
    - kondisi awal
    - memiliki unsur kesamaan (=, >=, <=)
  - H_1 / alternative hypothesis:
    - kontradiktif dengan H_0
- α:
  - tingkat signifikansi yaitu tingkat error yang masih bisa ditoleransi
  - umumnya 0.05
- $1 - \alpha$: tingkat kepercayaan
- $p - value$:
  - hasil perhitungan statistik yang menunjukkan peluang data sampel terjadi dengan kondisi H0.

Pengambilan kesimpulan:

- Jika $p - value < \alpha$, maka tolak H_0 -> terima h1

- Jika $p - value > \alpha$, maka gagal tolak H_0 -> terima h0

Contoh Hipotesis

1. Hipotesis dua arah (!=)

- H_0 : Rata-rata saldo rekening tidak berbeda secara signifikan antara nasabah yang menggunakan layanan internet banking dan yang tidak menggunakan layanan tersebut. (=)
- H_1 : Rata-rata saldo rekening **berbeda secara signifikan** antara nasabah yang menggunakan layanan internet banking dan yang tidak menggunakan layanan tersebut. (!=)

2. Hipotesis satu arah (<)

- H_0 : Penambahan teller tidak memberikan perbedaan durasi pembayaran (>=)
- H_1 : Penambahan teller **menurunkan** durasi pembayaran (<)

3. Hipotesis satu arah (>)

- H_0: Penerapan diskon tidak memberikan perbedaan jumlah pembelian produk (<=)
- H_1: Penerapan diskon **meningkatkan** jumlah pembelian produk (>)

10.5.1 Z-Test

Uji hipotesis yang menggunakan Z-test bila:

- standar deviasi populasi diketahui
- jumlah sampel banyak (n > 30)

Contoh

BRI merupakan salah satu Bank terbaik di Indonesia. Bila diketahui rata-rata likes dari suatu post di platform mereka sebesar **14000** likes dengan standar deviasi **5000** likes.

Demi meningkatkan likes dari tiap post, BRI memutuskan untuk menggunakan influencer sebagai brand ambassador pemasaran produk. Setelah menggunakan influencer, diambil **50** postingan acak yang ternyata memiliki rata-rata likes **17500**.

Sebagai tim marketing, lakukan analisis apakah menggunakan jasa influencer secara signifikan meningkatkan customer engagement (dari sisi rata-rata jumlah likes) atau tidak? Gunakan tingkat kepercayaan **95%**.

Jawaban:

I. Tentukan hipotesis

- H_0:
- H_1:

II. Hitung nilai statistik

Diketahui deskriptif statistiknya:

- mean populasi =
- stdev populasi =
- n =
- mean sampel =

Ditentukan oleh user:

- tingkat kepercayaan =
- alpha =

[ ]:
```
# nilai statistic descriptive
mean_populasi =
std_populasi =
n =
mean_sample =

print('mean_populasi: ', mean_populasi)
print('std_populasi: ', std_populasi)
print('n: ', n)
print('mean_sample: ', mean_sample)
```

$$Z = \frac{\bar{X} - \mu}{SE}$$

Z = (rata2 sampel - rata2 populasi) / standar error

$$SE = \frac{\sigma}{\sqrt{n}}$$

Standar error = standar deviasi populasi / akar dari banyak sampel

[ ]:
```
# menghitung nilai SE
SE =

# menghitung nilai z
Z =
```

[ ]:
```
p_value =
```

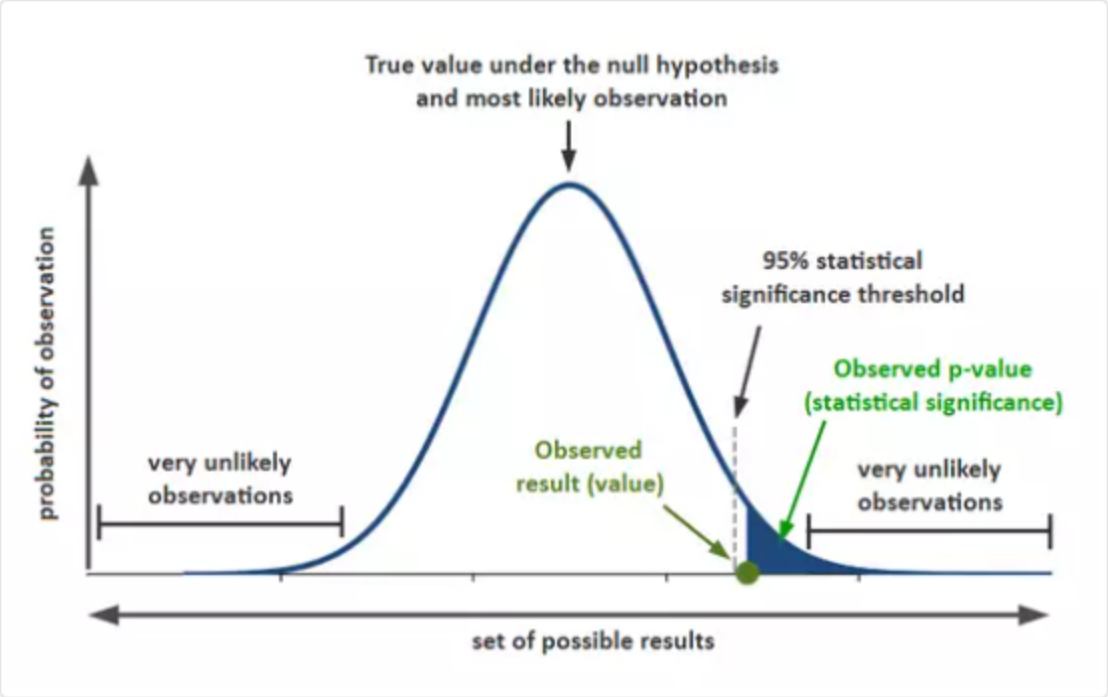
Gunakan fungsi `ztest()` untuk menghitung z-statistics dan p-value.

```
zstats, pval = ztest(x1=...,
                     value = ...,
                     alternative = ...)
```

parameter:

- `x1` : number of observations
- `value` : rata-rata dari x1 di H_0
- `alternative` :
  - jika H_1 tidak sama (!=) dengan nilai tertentu, isi dengan `two-sided`
  - jika H_1 lebih besar (>) dari suatu nilai, gunakan `larger`
  - jika H_1 lebih kecil (<) dari suatu nilai, gunakan `smaller`

```
[ ]: from statsmodels.stats.weightstats import ztest

     z_test =
```



c. Bandingkan P-value dengan alpha

Pengambilan kesimpulan:

- Jika $p - value < \alpha$, maka tolak H_0
- Jika $p - value > \alpha$, maka gagal tolak H_0

p-value = ____ (>/<) alpha = 0.05

IV. Kesimpulan

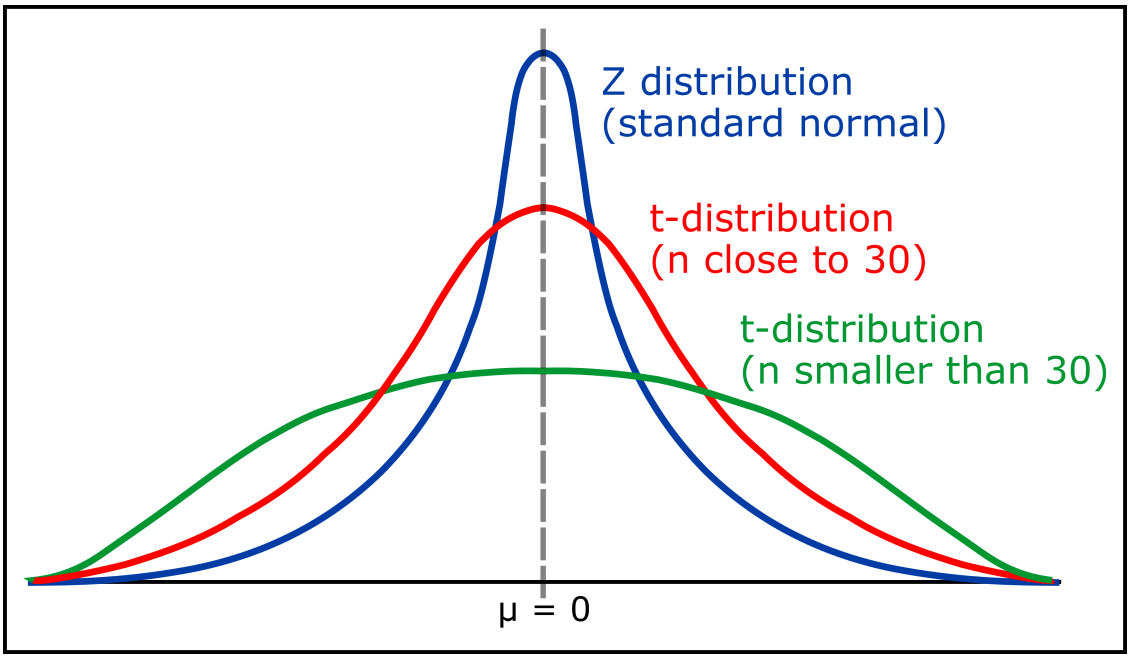
...

10.5.2 T-test

Uji hipotesis menggunakan T-test jika:

- standar deviasi populasi tidak diketahui atau
- jumlah sampel sedikit (n <= 30)

Bentuk t-distribution mirip dengan normal distribution, hanya saja lebih landai ketika jumlah sampel sedikit:



Contoh Kasus

Mari kita asumsikan Bank BRI memiliki dua kelompok nasabah bank, yaitu kelompok yang memiliki behavior scoring tinggi dan kelompok yang memiliki behavior scoring rendah.

Diketahui data saldo rekening antara kedua kelompok sebagai berikut:

[30]:
```
behavior_score_high = pd.Series([30.4, 52.7, 70.6, 55.7, 56.3, 34.2, 59.6, 42.
↪3, 21.1, 50.5, 12.2, 58.6, 12.0, 56.1, 49.4, 60.9, 60.0, 35.3, 15.0, 50.3])

behavior_score_low = pd.Series([6.5, 13.3, 6.8, 9.2, 10.0, 1.5, 21.7, 16.2, 5.
↪9, 25.0, 18.4, 12.6, 22.2, 22.0, 21.6, 20.5, 19.4, 14.5, 12.6, 12.0])
```

Tujuan kita adalah menguji **apakah terdapat perbedaan signifikan dalam rata-rata saldo rekening antara kedua kelompok tersebut?**

Jawab:

I. Tentukan hipotesis

- H_0: Rata-rata saldo rekening antara kelompok dengan behavior scoring tinggi dan kelompok dengan behavior scoring rendah tidak berbeda secara signifikan.
- H_1: Terdapat perbedaan signifikan dalam rata-rata saldo rekening antara kedua kelompok.

II. Hitung P-value dengan `ttest_ind()`

Gunakan fungsi `ttest_ind()` untuk menghitung t-statistics dan p-value dua independent sample

```
t_stats, pval = ttest_ind(a= ...,
                          b= ...,
                          alternative = ...)
```

parameter:

- `a` : data atau observasi sampel berbentuk Series atau array
- `b` : data atau observasi sampel berbentuk Series atau array
- `alternative` : tergantung hypothesis alternative (H_1)
  - jika H_1 tidak sama (!=) dengan nilai tertentu, isi dengan `two-sided`
  - jika H_1 lebih besar (>) dengan nilai tertentu, isi dengan `less`
  - Jika H_1 lebih besar (<) dengan nilai tertentu, isi dengan `greater`

[ ]:
```
# code here
from scipy import stats

t_test =
```

III. Bandingkan P-value dengan alpha

Dalam membuat keputusan uji statistik, kita dapat membandingan p-value dengan alpha:

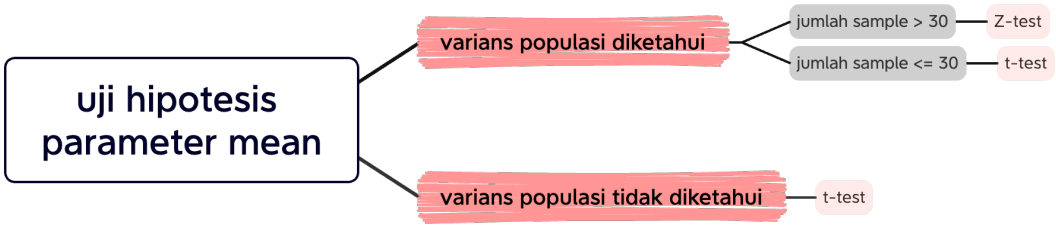
- selang kepercayaan = 95%
- alpha = 5% -> 0.05

p-value = ____ (>/<) alpha = 0.05, maka ____

IV. Kesimpulan

Dengan menggunakan tingkat kepercayaan 95% dapat disimpulkan ____

Summary penggunaan hipotesis testing:



11 Further Readings

- Descriptive Statistics: https://courses.lumenlearning.com/suny-natural-resources-biometrics/chapter/chapter-1-descriptive-statistics-and-the-normal-distribution/
- Dealing with small data set: https://measuringu.com/small-n/
- t-Distribution and some case examples: https://stattrek.com/probability-distributions/t-distribution.aspx

Coursebook: Data Wrangling and Visualization

- Bagian 3 Audit Analytics untuk Bank Rakyat Indonesia
- Durasi: 7 Jam
- *Last Updated*: December 2023

- *Coursebook* ini disusun dan dikurasi oleh tim produk dan instruktur dari Algoritma Data Science School

12 Background

Coursebook ini merupakan bagian dari **BRI Audit Analytics** yang disiapkan oleh Algoritma. *Coursebook* ini ditujukan hanya untuk khalayak terbatas, yaitu individu dan organisasi yang menerima *coursebook* ini langsung dari organisasi pelatihan. Tidak boleh direproduksi, didistribusikan, diterjemahkan, atau diadaptasi dalam bentuk apapun di luar individu dan organisasi ini tanpa izin.

Algoritma adalah pusat pendidikan *data science* yang berbasis di Jakarta. Kami menyelenggarakan *workshop* dan program pelatihan untuk membantu para profesional dan mahasiswa dalam menguasai berbagai sub-bidang *data science* yaitu: *data visualization, machine learning,* statistik, dan lain sebagainya.

13 Training Objectives

Coursebook ini berfokus pada:

- Stacking and Unstacking
- Reproducible Environment
- Working with MultiIndex DataFrames
- Reshaping your DataFrame with Melt
- Using Group By Effectively
- Visual Data Exploratory

14 Reproducible Environment

Ada beberapa paket baru yang akan digunakan dalam materi ini. Biasanya, kita dapat menggunakan `pip install/conda install` untuk menginstal library baru ke environment kita. Namun untuk saat ini, mari kita coba pendekatan lain dalam mempersiapkan library yang diperlukan untuk proyek tertentu.

Bayangkan Anda sedang bekerja dengan tim Anda dalam sebuah proyek kolaboratif. Anda menginisialisasi proyek dengan dependensi dan versi tertentu di komputer Anda dan semuanya berjalan dengan baik. Nantinya, Anda perlu 'mengirimkan' proyek itu ke tim Anda yang mengharuskan mereka menyiapkan environment yang sama dengan Anda. Lalu apa yang akan Anda lakukan untuk memastikan program itu juga berjalan lancar di mesin mereka?

Di sinilah Anda perlu membuat environment Anda dapat direproduksi dengan membuat file `requirements.txt`.

Lihat pada folder material utama, Anda akan menemukan file `requirements.txt` yang isinya seperti ini:

```
matplotlib==3.8.1
numpy==1.26.1
pandas==2.0.0
yfinance==0.2.31
```

Perhatikan kita memiliki baris untuk setiap library, lalu nomor versi. Hal ini penting karena saat Anda mulai mengembangkan aplikasi python, Anda akan mengembangkan aplikasi dengan mempertimbangkan versi library tertentu. Sederhananya, `requirements.txt` membantu melacak versi setiap library yang Anda gunakan untuk mencegah perubahan yang tidak terduga.

14.1 Importing Requirements

Kita sudah membahas untuk apa file persyaratan itu, tetapi bagaimana cara menggunakannya? Karena kita tidak ingin menginstal dan melacak secara manual setiap library yang diperlukan untuk proyek tertentu, mari kita coba mengimpor persyaratan dengan langkah-langkah berikut:

1. Aktifkan environment yang ingin digunakan:

   ```
   conda activate <ENV_NAME>
   ```

   Apabila belum ada, maka perlu membuat environment baru:

   conda create -n [ENV_NAME] python=[PYTHON_VERSION]

   Jangan lupa instalasi kernel di dalam environment tersebut apabila ingin dapat diakses menggunakan jupyter notebook:

   > pip install ipykernel

   > python -m ipykernel install –user –name=[ENV_NAME]

2. Navigasikan path ke folder di mana file `requirements.txt` berada

   ```
   cd <PATH_TO_REQUIREMENTS>
   ```

3. Instalasi packages dari file tersebut

   “‘ pip install -r requirements.txt

14.2 Exporting Requirements

Perintah `pip install` selalu menginstal versi terbaru dari sebuah library, namun terkadang, Anda mungkin ingin menginstal versi tertentu yang Anda tahu berfungsi pada proyek Anda.

File persyaratan memungkinkan Anda menentukan dengan tepat library dan versi mana yang harus diinstal. Anda dapat mengikuti langkah-langkah berikut untuk membuat file kebutuhan Anda:

1. Aktifkan environment

```
conda activate <ENV_NAME>
```

2. Navigasikan path ke folder tempat di mana file `requirements.txt` ingin disimpan

```
cd <PATH_TO_REQUIREMENTS_FOLDER>
```

3. Export environment: membuat daftar packages beserta versinya.

```
pip list --format=freeze > requirements.txt
```

Notes!

Anda dapat menyimpan file dengan nama lain, namun sebagai konvensi biasa digunakan penamaan requirements.txt

---

15 Data Wrangling and Reshaping

Dalam dua course sebelumnya, kita telah mempelajari beberapa teknik umum dan mempelajari cara mengeksplorasi data menggunakan metode bawaan `pandas`. Secara khusus, di bagian pertama dan kedua dari seri ini kita telah membahas cara menggunakan tools inspeksi, diagnostik, dan eksplorasi berikut:

Data Inspection

- `.head()` and `.tail()`
- `.describe()`
- `.shape` and `.size`
- `.axes`
- `.dtypes`
- Subsetting using `.loc`, `.iloc` and conditionals

Diagnostic and Exploratory

- Tables
- Cross-Tables and Aggregates
- Using `aggfunc` for aggregate functions
- Pivot Tables
- Working with DateTime
- Working with Categorical Data
- Duplicates and Missing Value Treatment

Paruh pertama course ini berfungsi sebagai perpanjangan dari materi terakhir. Kami akan mengenalkan beberapa teknik baru untuk melengkapi tools EDA. Mari kita mulai dengan teknik `reshaping` kembali.

[1]:
```
import pandas as pd
import yfinance as data
pd.set_option('display.float_format', '{:,.3f}'.format) #display setting␣
↪purpose only
```

[2]:
```
symbol = ['BRIS.JK' ,'BBRI.JK', 'BMRI.JK']
start_date = '2020-01-01' # 1 januari 2020
end_date = '2023-12-31' # 31 desember 2023
stock = data.download(tickers = symbol, start = start_date, end = end_date)
stock.columns.names = ['Attributes', 'Symbols']
stock.head()
```

```
[*********************100%%**********************]  3 of 3 completed
```

[2]:
```
Attributes Adj Close                        Close                        High
Symbols      BBRI.JK   BMRI.JK BRIS.JK   BBRI.JK   BMRI.JK BRIS.JK   BBRI.JK
Date
2020-01-02 3,772.726 3,239.328 326.177 4,410.000 3,875.000 332.000 4,410.000  \
2020-01-03 3,781.281 3,228.878 322.247 4,420.000 3,862.500 328.000 4,440.000
2020-01-06 3,738.507 3,176.631 318.318 4,370.000 3,800.000 324.000 4,390.000
2020-01-07 3,764.171 3,176.631 312.423 4,400.000 3,800.000 318.000 4,410.000
2020-01-08 3,747.061 3,134.833 306.528 4,380.000 3,750.000 312.000 4,400.000

Attributes                          Low                          Open
Symbols      BMRI.JK BRIS.JK   BBRI.JK   BMRI.JK BRIS.JK   BBRI.JK   BMRI.JK
Date
2020-01-02 3,887.500 336.000 4,360.000 3,825.000 330.000 4,400.000 3,837.500  \
2020-01-03 3,912.500 336.000 4,390.000 3,812.500 326.000 4,420.000 3,875.000
2020-01-06 3,837.500 334.000 4,320.000 3,762.500 320.000 4,360.000 3,825.000
2020-01-07 3,862.500 324.000 4,380.000 3,787.500 316.000 4,410.000 3,862.500
2020-01-08 3,775.000 318.000 4,340.000 3,687.500 312.000 4,380.000 3,775.000

Attributes              Volume
Symbols    BRIS.JK    BBRI.JK    BMRI.JK  BRIS.JK
Date
2020-01-02 330.000   41714100   37379800  1456400
2020-01-03 334.000   82898300   70294600  4989600
2020-01-06 328.000   44225100   61892000  6937900
2020-01-07 324.000  103948100   70895600  6319400
2020-01-08 318.000  171751200  105080600  4058800
```

Jika Anda belum menginstal modul `pandas_datareader`, atau jika Anda mengikuti buku pelajaran ini tanpa koneksi internet aktif, Anda dapat memuatnya dari objek serial yang kami simpan di folder `data_cache` Anda.

Membuat objek DataFrame dengan membaca dari `pickle`:

- `stock = pd.read_pickle('data_cache/stock')`

Membuat serial objek DataFrame menjadi aliran byte menggunakan `pickle`:

- `stock.to_pickle('data_cache/stock')`

[3]:
```
# buat dataframe kedalam pickle
# stock.to_pickle('data_cache/stock')
```

[4]:
```
# stock = pd.read_pickle('data_cache/stock')
# stock.head()
```

15.1 Slicing Multi-Index DataFrame

Multi-Index Dataframe adalah bentuk dataframe yang memiliki level indexing lebih dari 1 baik pada baris, kolom, ataupun keduanya. Hal yang perlu diperhatikan dalam MultiIndex Dataframe adalah bentuk dataframe ini terkadang tidak bisa langsung kita gunakan untuk menganalisis data, sehingga akan ada beberapa perlakuan untuk kita mengiris atau mengubah bentuknya ke dataframe yang lebih sederhana.

Perhatikan bahwa data `stock` adalah Multi-Index DataFrame, di mana level dari columnnya terdiri dari:

- `Attributes` = level 0
- `Symbols` = level 1

Perhatikan bagaimana Dataframe kita merupakan data multi-indeks. Jika Anda memperhatikan dengan seksama, Anda dapat melihat 2 tingkat sumbu kolom: `Attributes` dan `Symbols`. Jika Anda mengelompokkan data menggunakan tanda kurung siku [ ] , Anda akan mengakses indeks tingkat tertinggi:

[5]:
```
# Ambil kolom High dengan semua attributnya
stock['High']

# Jika tidak, kode ini akan menimbulkan kesalahan
# stock['AAPL']
```

[5]:
```
Symbols      BBRI.JK   BMRI.JK   BRIS.JK
Date
2020-01-02 4,410.000 3,887.500   336.000
2020-01-03 4,440.000 3,912.500   336.000
2020-01-06 4,390.000 3,837.500   334.000
2020-01-07 4,410.000 3,862.500   324.000
2020-01-08 4,400.000 3,775.000   318.000
…               …        …        …
2023-12-15 5,600.000 6,000.000 1,715.000
2023-12-18 5,575.000 5,950.000 1,705.000
2023-12-19 5,550.000 5,975.000 1,710.000
2023-12-20 5,700.000 5,975.000 1,765.000
2023-12-21 5,600.000 5,975.000 1,765.000

[970 rows x 3 columns]
```

15.1.1 Cross Section

Menggunakan method `.xs()` (cross section) untuk mengambil kolom (`axis = 1`) pada level dalam. Parameter:

- `key` : nama kolom/baris yang kita ingin ambil

- `level` : kolom/baris tersebut ada di level apa?
- `axis` : levelnya terdapat pada index kolom/baris
  - `0` untuk baris
  - `1` untuk kolom

[6]:
```
# mengambil seluruh nilai BBRI
stock.xs(key='BBRI.JK',
         level= 1,
         axis = 1)
```

[6]:
```
Attributes  Adj Close     Close      High       Low      Open     Volume
Date
2020-01-02  3,772.726 4,410.000 4,410.000 4,360.000 4,400.000   41714100
2020-01-03  3,781.281 4,420.000 4,440.000 4,390.000 4,420.000   82898300
2020-01-06  3,738.507 4,370.000 4,390.000 4,320.000 4,360.000   44225100
2020-01-07  3,764.171 4,400.000 4,410.000 4,380.000 4,410.000  103948100
2020-01-08  3,747.061 4,380.000 4,400.000 4,340.000 4,380.000  171751200
…               …       …       …       …       …        …
2023-12-15  5,550.000 5,550.000 5,600.000 5,550.000 5,575.000  252448800
2023-12-18  5,500.000 5,500.000 5,575.000 5,500.000 5,575.000  102780900
2023-12-19  5,550.000 5,550.000 5,550.000 5,450.000 5,450.000  135207300
2023-12-20  5,550.000 5,550.000 5,700.000 5,550.000 5,700.000  138470900
2023-12-21  5,575.000 5,575.000 5,600.000 5,525.000 5,550.000   99049600

[970 rows x 6 columns]
```

15.2 `stack()` and `unstack()`

`stack()` menumpuk level yang ditentukan dari kolom ke indeks dan sangat berguna pada DataFrames yang memiliki kolom multi-level. Ia melakukannya dengan "menggeser" kolom untuk membuat level baru pada indeksnya.

Hal ini lebih mudah dipahami bila kita hanya melihat contohnya. Perhatikan bahwa `stock` memiliki kolom 2 tingkat (Atribut dan Simbol) dan indeks 1 tingkat (Tanggal):

[7]:
```
stock.head(10)
```

[7]:
```
Attributes Adj Close                        Close                        High
Symbols      BBRI.JK   BMRI.JK BRIS.JK   BBRI.JK   BMRI.JK BRIS.JK   BBRI.JK
Date
2020-01-02 3,772.726 3,239.328 326.177 4,410.000 3,875.000 332.000 4,410.000  \
2020-01-03 3,781.281 3,228.878 322.247 4,420.000 3,862.500 328.000 4,440.000
2020-01-06 3,738.507 3,176.631 318.318 4,370.000 3,800.000 324.000 4,390.000
2020-01-07 3,764.171 3,176.631 312.423 4,400.000 3,800.000 318.000 4,410.000
2020-01-08 3,747.061 3,134.833 306.528 4,380.000 3,750.000 312.000 4,400.000
2020-01-09 3,764.171 3,218.429 312.423 4,400.000 3,850.000 318.000 4,420.000
2020-01-10 3,772.726 3,228.878 308.493 4,410.000 3,862.500 314.000 4,430.000
2020-01-13 3,858.275 3,228.878 314.388 4,510.000 3,862.500 320.000 4,510.000
2020-01-14 3,909.605 3,239.328 314.388 4,570.000 3,875.000 320.000 4,600.000
```

```
2020-01-15 3,918.160 3,197.530 314.388 4,580.000 3,825.000 320.000 4,580.000

Attributes                          Low                          Open
Symbols      BMRI.JK BRIS.JK   BBRI.JK   BMRI.JK BRIS.JK   BBRI.JK   BMRI.JK
Date
2020-01-02 3,887.500 336.000 4,360.000 3,825.000 330.000 4,400.000 3,837.500  \
2020-01-03 3,912.500 336.000 4,390.000 3,812.500 326.000 4,420.000 3,875.000
2020-01-06 3,837.500 334.000 4,320.000 3,762.500 320.000 4,360.000 3,825.000
2020-01-07 3,862.500 324.000 4,380.000 3,787.500 316.000 4,410.000 3,862.500
2020-01-08 3,775.000 318.000 4,340.000 3,687.500 312.000 4,380.000 3,775.000
2020-01-09 3,862.500 322.000 4,370.000 3,762.500 310.000 4,400.000 3,775.000
2020-01-10 3,862.500 324.000 4,390.000 3,825.000 314.000 4,430.000 3,850.000
2020-01-13 3,875.000 324.000 4,420.000 3,825.000 314.000 4,430.000 3,862.500
2020-01-14 3,900.000 326.000 4,520.000 3,850.000 318.000 4,540.000 3,900.000
2020-01-15 3,900.000 322.000 4,530.000 3,800.000 316.000 4,570.000 3,875.000

Attributes             Volume
Symbols    BRIS.JK    BBRI.JK    BMRI.JK   BRIS.JK
Date
2020-01-02 330.000   41714100   37379800   1456400
2020-01-03 334.000   82898300   70294600   4989600
2020-01-06 328.000   44225100   61892000   6937900
2020-01-07 324.000  103948100   70895600   6319400
2020-01-08 318.000  171751200  105080600   4058800
2020-01-09 312.000   72072000   75587000   4928400
2020-01-10 318.000  124809200   64231800   6175000
2020-01-13 316.000  111351400   75399600  10833200
2020-01-14 322.000  158414200  104134400   7388400
2020-01-15 320.000  102566700   89795200  18046800
```

Saat kita menumpuk `stock` DataFrame, kita mengecilkan jumlah level pada kolomnya sebanyak satu: `stock` sekarang memiliki 1 kolom level bernama `Attributes`:

[8]:
```
stock.stack()
```

[8]:
```
Attributes          Adj Close     Close      High       Low      Open
Date       Symbols
2020-01-02 BBRI.JK  3,772.726 4,410.000 4,410.000 4,360.000 4,400.000  \
           BMRI.JK  3,239.328 3,875.000 3,887.500 3,825.000 3,837.500
           BRIS.JK    326.177   332.000   336.000   330.000   330.000
2020-01-03 BBRI.JK  3,781.281 4,420.000 4,440.000 4,390.000 4,420.000
           BMRI.JK  3,228.878 3,862.500 3,912.500 3,812.500 3,875.000
…                       …       …       …       …       …
2023-12-20 BMRI.JK  5,925.000 5,925.000 5,975.000 5,900.000 5,900.000
           BRIS.JK  1,755.000 1,755.000 1,765.000 1,710.000 1,710.000
2023-12-21 BBRI.JK  5,575.000 5,575.000 5,600.000 5,525.000 5,550.000
           BMRI.JK  5,975.000 5,975.000 5,975.000 5,925.000 5,950.000
           BRIS.JK  1,690.000 1,690.000 1,765.000 1,680.000 1,765.000
```

```
Attributes            Volume
Date       Symbols
2020-01-02 BBRI.JK  41714100
           BMRI.JK  37379800
           BRIS.JK   1456400
2020-01-03 BBRI.JK  82898300
           BMRI.JK  70294600
…                       …
2023-12-20 BMRI.JK  65361000
           BRIS.JK  39163500
2023-12-21 BBRI.JK  99049600
           BMRI.JK  50363900
           BRIS.JK  25310000

[2910 rows x 6 columns]
```

`unstack()` melakukan yang sebaliknya: ia "menggeser" level dari sumbu indeks ke sumbu kolom. **Coba dan buat tumpukan DataFrame, lalu terapkan `unstack` pada DataFrame baru untuk melihatnya kembali ke bentuk aslinya:**

[9]:
```
## Write your code to try out .unstack() method here
```

Dive Deeper

Jawablah pertanyaan-pertanyaan berikut ini untuk memastikan Anda dapat melanjutkan sesi berikutnya:

1. Bagaimana cara menukar posisi (level) Symbols dan Attributes?
2. Berdasarkan pengetahuan Anda, (simbol) perusahaan apa yang layak untuk diinvestasikan? (Anda dapat melihat fluktuasinya, artinya, dll)

[10]:
```
# Write your solution code here
```

Knowledge Check: Stack and Unstack

Which of the following statement is correct? Manakah dibawah ini yang merupakan pernyataan yang benar?

- ☐ `stack()` mengubah DataFrame dari lebar ke panjang
- ☐ `unstack()` mengubah DataFrame dari panjang ke lebar
- ☐ `unstack()` mengubah DataFrame dari lebar ke panjang

15.3 Melt

Berbicara tentang membentuk kembali DataFrame dari format lebar ke format panjang, metode lain yang harus ada di perangkat Anda adalah `melt()`. Pertimbangkan DataFrame berikut, yang dibuat dari metode Pemotong MultiIndex `pandas`, `.xs()` (Singkatan dari 'Cross Section'):

[11]:
```
bbri = stock.xs(key = 'BBRI.JK', level='Symbols', axis=1)
bbri.head()
```

[11]:
```
Attributes  Adj Close     Close      High       Low      Open     Volume
Date
2020-01-02  3,772.726 4,410.000 4,410.000 4,360.000 4,400.000   41714100
2020-01-03  3,781.281 4,420.000 4,440.000 4,390.000 4,420.000   82898300
2020-01-06  3,738.507 4,370.000 4,390.000 4,320.000 4,360.000   44225100
2020-01-07  3,764.171 4,400.000 4,410.000 4,380.000 4,410.000  103948100
2020-01-08  3,747.061 4,380.000 4,400.000 4,340.000 4,380.000  171751200
```

[12]:
```
bbri.shape
```

[12]:
```
(970, 6)
```

DataFrame di atas lebar: memiliki 968 baris dan 6 kolom. Fungsi `melt()` mengumpulkan semua kolom menjadi satu dan menyimpan nilai yang sesuai dengan setiap kolom sehingga DataFrame yang dihasilkan memiliki 968 * 6 = 5.808 baris, bersama dengan kolom pengidentifikasi dan nilai:

[13]:
```
bbri_melted = bbri.melt()
bbri_melted
```

[13]:
```
     Attributes           value
0     Adj Close       3,772.726
1     Adj Close       3,781.281
2     Adj Close       3,738.507
3     Adj Close       3,764.171
4     Adj Close       3,747.061
…         …               …
5815     Volume 252,448,800.000
5816     Volume 102,780,900.000
5817     Volume 135,207,300.000
5818     Volume 138,470,900.000
5819     Volume  99,049,600.000

[5820 rows x 2 columns]
```

[14]:
```
bbri_melted.shape
```

[14]:
```
(5820, 2)
```

Knowledge Check : Apa yang membedakan antara melt dengan stack?

Secara opsional, kita dapat menentukan satu atau lebih kolom untuk menjadi variabel pengenal (`id_vars`), yang memperlakukan semua kolom lainnya sebagai variabel nilai (`value_vars`):

[15]:
```
bbri.reset_index().melt(id_vars=['Date'])
```

[15]:
```
           Date Attributes           value
0    2020-01-02  Adj Close       3,772.726
1    2020-01-03  Adj Close       3,781.281
2    2020-01-06  Adj Close       3,738.507
3    2020-01-07  Adj Close       3,764.171
4    2020-01-08  Adj Close       3,747.061
…           …          …               …
5815 2023-12-15     Volume 252,448,800.000
5816 2023-12-18     Volume 102,780,900.000
5817 2023-12-19     Volume 135,207,300.000
5818 2023-12-20     Volume 138,470,900.000
5819 2023-12-21     Volume  99,049,600.000

[5820 rows x 3 columns]
```

[16]:
```
bbri.reset_index().melt(value_vars=['High', 'Low'])
```

[16]:
```
     Attributes     value
0          High 4,410.000
1          High 4,440.000
2          High 4,390.000
3          High 4,410.000
4          High 4,400.000
…           …         …
1935        Low 5,550.000
1936        Low 5,500.000
1937        Low 5,450.000
1938        Low 5,550.000
1939        Low 5,525.000

[1940 rows x 2 columns]
```

16 Pandas dan Matplotlib

Tentunya ini adalah titik di mana seorang analis data menyiapkan beberapa grafik yang mencolok menggunakan perpustakaan `matplotlib` yang populer?

Baiklah. Bahkan lebih baik lagi, kita akan menggunakan metode `DataFrame.plot()`, yang ada di dalam `pandas` yang kemudian memanggil fungsi plotting `matplotlib` di baliknya. Perhatikan bahwa kita menambahkan `matplotlib.pyplot` sebagai impor, meskipun kode kita tidak akan memanggil `matplotlib` secara eksplisit tetapi bergantung pada implementasi `pandas`.

Sekarang mari kita lihat kerangka data saham BRI:

[17]:
```
bbri.head()
```

[17]:
```
Attributes  Adj Close     Close      High       Low      Open     Volume
Date
```

```
2020-01-02  3,772.726 4,410.000 4,410.000 4,360.000 4,400.000   41714100
2020-01-03  3,781.281 4,420.000 4,440.000 4,390.000 4,420.000   82898300
2020-01-06  3,738.507 4,370.000 4,390.000 4,320.000 4,360.000   44225100
2020-01-07  3,764.171 4,400.000 4,410.000 4,380.000 4,410.000  103948100
2020-01-08  3,747.061 4,380.000 4,400.000 4,340.000 4,380.000  171751200
```

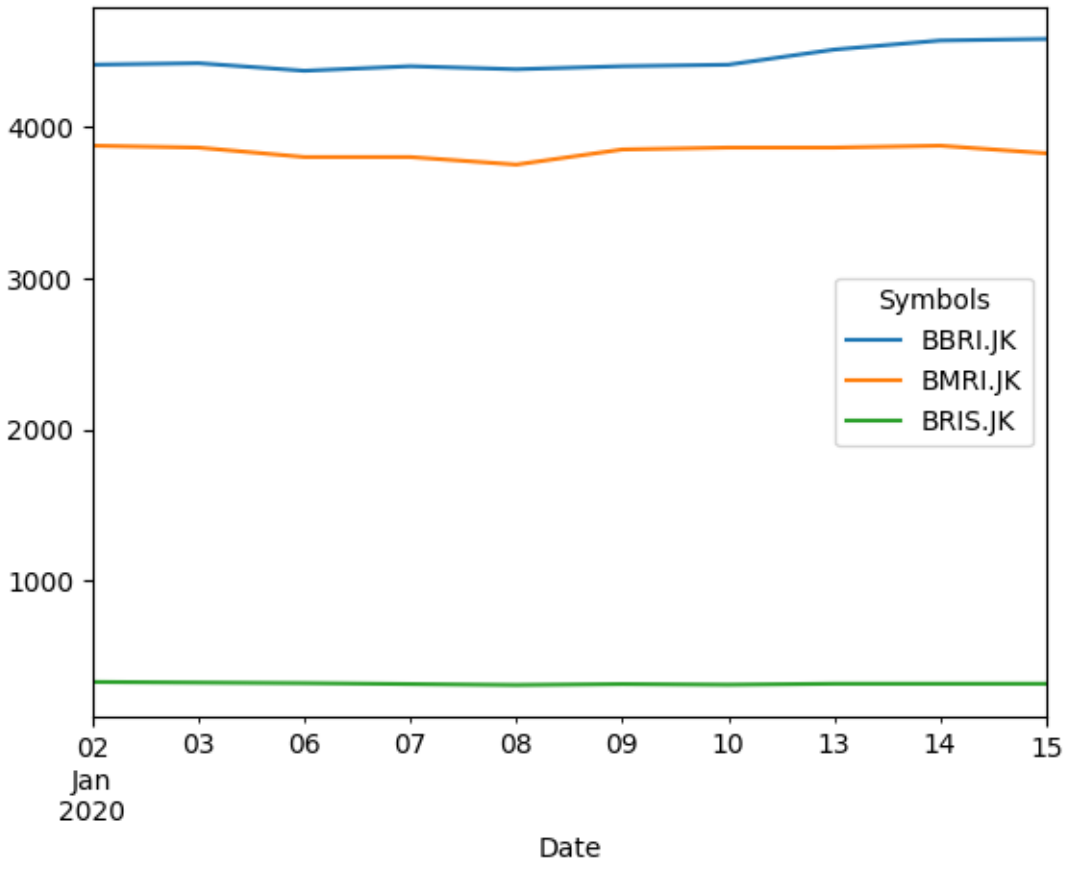
Cara terbaik untuk mendemonstrasikan peningkatan efisiensi `DataFrame.plot()` adalah dengan melihatnya bekerja. Kami akan memanggil `.plot()` langsung di `DataFrame` - `pandas` kami akan menangani kode matplotlib yang, menurut pengakuan matplotlib sendiri, *dapat menjadi hal yang menakutkan bagi banyak pengguna baru.*

[18]:
```
stock
```

[18]:
```
Attributes Adj Close                          Close
Symbols      BBRI.JK   BMRI.JK   BRIS.JK   BBRI.JK   BMRI.JK   BRIS.JK
Date
2020-01-02 3,772.726 3,239.328   326.177 4,410.000 3,875.000   332.000  \
2020-01-03 3,781.281 3,228.878   322.247 4,420.000 3,862.500   328.000
2020-01-06 3,738.507 3,176.631   318.318 4,370.000 3,800.000   324.000
2020-01-07 3,764.171 3,176.631   312.423 4,400.000 3,800.000   318.000
2020-01-08 3,747.061 3,134.833   306.528 4,380.000 3,750.000   312.000
…              …       …       …       …       …       …
2023-12-15 5,550.000 5,900.000 1,700.000 5,550.000 5,900.000 1,700.000
2023-12-18 5,500.000 5,925.000 1,695.000 5,500.000 5,925.000 1,695.000
2023-12-19 5,550.000 5,975.000 1,705.000 5,550.000 5,975.000 1,705.000
2023-12-20 5,550.000 5,925.000 1,755.000 5,550.000 5,925.000 1,755.000
2023-12-21 5,575.000 5,975.000 1,690.000 5,575.000 5,975.000 1,690.000

Attributes      High                           Low
Symbols      BBRI.JK   BMRI.JK   BRIS.JK   BBRI.JK   BMRI.JK   BRIS.JK
Date
2020-01-02 4,410.000 3,887.500   336.000 4,360.000 3,825.000   330.000  \
2020-01-03 4,440.000 3,912.500   336.000 4,390.000 3,812.500   326.000
2020-01-06 4,390.000 3,837.500   334.000 4,320.000 3,762.500   320.000
2020-01-07 4,410.000 3,862.500   324.000 4,380.000 3,787.500   316.000
2020-01-08 4,400.000 3,775.000   318.000 4,340.000 3,687.500   312.000
…              …       …       …       …       …       …
2023-12-15 5,600.000 6,000.000 1,715.000 5,550.000 5,900.000 1,700.000
2023-12-18 5,575.000 5,950.000 1,705.000 5,500.000 5,850.000 1,670.000
2023-12-19 5,550.000 5,975.000 1,710.000 5,450.000 5,925.000 1,690.000
2023-12-20 5,700.000 5,975.000 1,765.000 5,550.000 5,900.000 1,710.000
2023-12-21 5,600.000 5,975.000 1,765.000 5,525.000 5,925.000 1,680.000

Attributes      Open                           Volume
Symbols      BBRI.JK   BMRI.JK   BRIS.JK    BBRI.JK    BMRI.JK   BRIS.JK
Date
2020-01-02 4,400.000 3,837.500   330.000   41714100   37379800   1456400
2020-01-03 4,420.000 3,875.000   334.000   82898300   70294600   4989600
```

```
2020-01-06 4,360.000 3,825.000   328.000   44225100   61892000   6937900
2020-01-07 4,410.000 3,862.500   324.000  103948100   70895600   6319400
2020-01-08 4,380.000 3,775.000   318.000  171751200  105080600   4058800
…                …        …         …          …          …         …
2023-12-15 5,575.000 5,950.000 1,710.000  252448800  142382200  12317700
2023-12-18 5,575.000 5,850.000 1,690.000  102780900  101458600  12521500
2023-12-19 5,450.000 5,950.000 1,700.000  135207300   40763200   5861900
2023-12-20 5,700.000 5,900.000 1,710.000  138470900   65361000  39163500
2023-12-21 5,550.000 5,950.000 1,765.000   99049600   50363900  25310000

[970 rows x 18 columns]
```

[19]:
```
stock['Close'].head(10).plot();
```



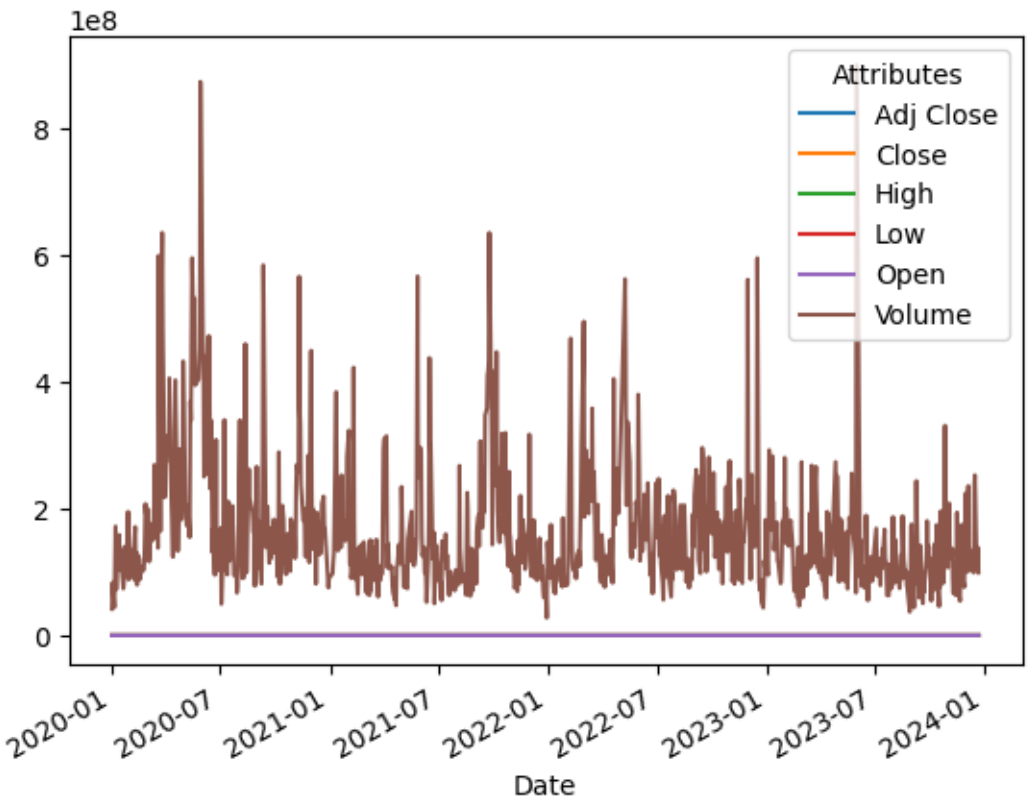
Kita dapat menyesuaikan plot kami dengan style sheet tetapi referensi praktis dapat dijangkau. Anda dapat mengganti 'default' dengan salah satu gaya yang tersedia dan menjalankan kembali kode plot untuk melihat gaya yang diterapkan.

[20]:
```
import matplotlib.pyplot as plt
print(plt.style.available)
plt.style.use('default')
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-
nogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight',
'ggplot', 'grayscale', 'seaborn-v0_8', 'seaborn-v0_8-bright',
'seaborn-v0_8-colorblind', 'seaborn-v0_8-dark', 'seaborn-v0_8-dark-palette',
'seaborn-v0_8-darkgrid', 'seaborn-v0_8-deep', 'seaborn-v0_8-muted',
'seaborn-v0_8-notebook', 'seaborn-v0_8-paper', 'seaborn-v0_8-pastel',
'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8-ticks',
'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

Karena metode `.plot()` dipanggil pada objek DataFrame, kita dapat memiliki DataFrame yang diindeks dengan beberapa kolom dan `plot` akan menanganinya menggunakan opsi defaultnya:

[21]:
```
bbri.plot();
```



[22]:
```
bbri.loc[:, ['High', 'Low', 'Adj Close']].plot();
```

16.1 Other Visualization

Visualisasi berikut hanya perlu menggunakan **satu** kolom:

- Data kategorik:
  - **`.plot(kind='bar')` untuk barplot (diagram batang)**
  - **`.plot(kind='barh')` untuk horizontal barplot**
  - **`.plot(kind='box')` untuk boxplot (five number summary)**
  - `.plot(kind='pie')` untuk pie chart
- Data numerik:
  - **`.plot(kind='hist')` untuk histogram**
  - `.plot(kind='density')` untuk density plot
  - `.plot(kind='area')` untuk area plot

Visualisasi berikut perlu menggunakan **dua** kolom:

- `.plot(kind='scatter')` untuk scatter plot
- `.plot(kind='hexbin')` untuk hexagonal bin plot

Panduan untuk menentukan tipe visualisasi yang tepat: https://www.data-to-viz.com/

Silahkan mengacu referensi lengkapnya di official documentation untuk method `plot` apabila ingin eksplor visualisasi yang ada di luar lingkup course ini

17 Group By

Membentuk ulang data adalah komponen penting dari setiap tahapan di data wrangling karena memungkinkan analis untuk "mengirimkan pesan" data ke dalam bentuk yang diinginkan untuk diproses lebih lanjut.

Teknik lain yang sama pentingnya adalah pengelompokan berdasarkan operasi. Analis yang memiliki pengalaman dengan SQL atau perangkat analisis data lainnya (misalnya `tidyverse` R) akan menganggap operasi grup sebagai strategi yang familier dalam banyak alur kerja yang banyak menganalisis.

Pertimbangkan DataFrame berikut:

[23]:
```
stock_adj = stock.stack()
stock_adj['Volume USD'] = stock_adj['Volume'] * stock_adj['Adj Close']
stock_adj = stock_adj.unstack()
stock_adj
```

[23]:
```
Attributes Adj Close                          Close
Symbols      BBRI.JK   BMRI.JK   BRIS.JK   BBRI.JK   BMRI.JK   BRIS.JK
Date
2020-01-02 3,772.726 3,239.328   326.177 4,410.000 3,875.000   332.000  \
2020-01-03 3,781.281 3,228.878   322.247 4,420.000 3,862.500   328.000
2020-01-06 3,738.507 3,176.631   318.318 4,370.000 3,800.000   324.000
2020-01-07 3,764.171 3,176.631   312.423 4,400.000 3,800.000   318.000
2020-01-08 3,747.061 3,134.833   306.528 4,380.000 3,750.000   312.000
…               …       …       …       …       …       …
2023-12-15 5,550.000 5,900.000 1,700.000 5,550.000 5,900.000 1,700.000
2023-12-18 5,500.000 5,925.000 1,695.000 5,500.000 5,925.000 1,695.000
2023-12-19 5,550.000 5,975.000 1,705.000 5,550.000 5,975.000 1,705.000
2023-12-20 5,550.000 5,925.000 1,755.000 5,550.000 5,925.000 1,755.000
2023-12-21 5,575.000 5,975.000 1,690.000 5,575.000 5,975.000 1,690.000

Attributes      High                          Low  …                Open
Symbols      BBRI.JK   BMRI.JK   BRIS.JK   BBRI.JK  …   BRIS.JK   BBRI.JK
Date                                               …
2020-01-02 4,410.000 3,887.500   336.000 4,360.000  …   330.000 4,400.000  \
2020-01-03 4,440.000 3,912.500   336.000 4,390.000  …   326.000 4,420.000
2020-01-06 4,390.000 3,837.500   334.000 4,320.000  …   320.000 4,360.000
2020-01-07 4,410.000 3,862.500   324.000 4,380.000  …   316.000 4,410.000
2020-01-08 4,400.000 3,775.000   318.000 4,340.000  …   312.000 4,380.000
…               …       …       …       …  …       …       …
2023-12-15 5,600.000 6,000.000 1,715.000 5,550.000  … 1,700.000 5,575.000
2023-12-18 5,575.000 5,950.000 1,705.000 5,500.000  … 1,670.000 5,575.000
2023-12-19 5,550.000 5,975.000 1,710.000 5,450.000  … 1,690.000 5,450.000
2023-12-20 5,700.000 5,975.000 1,765.000 5,550.000  … 1,710.000 5,700.000
2023-12-21 5,600.000 5,975.000 1,765.000 5,525.000  … 1,680.000 5,550.000

Attributes                          Volume
```

```
Symbols      BMRI.JK   BRIS.JK    BBRI.JK    BMRI.JK   BRIS.JK
Date
2020-01-02 3,837.500   330.000   41714100   37379800   1456400  \
2020-01-03 3,875.000   334.000   82898300   70294600   4989600
2020-01-06 3,825.000   328.000   44225100   61892000   6937900
2020-01-07 3,862.500   324.000  103948100   70895600   6319400
2020-01-08 3,775.000   318.000  171751200  105080600   4058800
…               …       …         …         …        …
2023-12-15 5,950.000 1,710.000  252448800  142382200  12317700
2023-12-18 5,850.000 1,690.000  102780900  101458600  12521500
2023-12-19 5,950.000 1,700.000  135207300   40763200   5861900
2023-12-20 5,900.000 1,710.000  138470900   65361000  39163500
2023-12-21 5,950.000 1,765.000   99049600   50363900  25310000

Attributes            Volume USD
Symbols                  BBRI.JK             BMRI.JK            BRIS.JK
Date
2020-01-02   157,375,882,916.675 121,085,428,320.947    475,044,585.657
2020-01-03   313,461,787,446.875 226,972,716,839.746  1,607,885,657.007
2020-01-06   165,335,827,872.876 196,608,067,369.141  2,208,455,440.582
2020-01-07   391,278,438,345.825 225,209,185,370.898  1,974,324,582.458
2020-01-08   643,562,271,192.773 329,410,184,669.629  1,244,136,031.702
…                            …                 …                 …
2023-12-15 1,401,090,840,000.000 840,054,980,000.000 20,940,090,000.000
2023-12-18   565,294,950,000.000 601,142,205,000.000 21,223,942,500.000
2023-12-19   750,400,515,000.000 243,560,120,000.000  9,994,539,500.000
2023-12-20   768,513,495,000.000 387,263,925,000.000 68,731,942,500.000
2023-12-21   552,201,520,000.000 300,924,302,500.000 42,773,900,000.000

[970 rows x 21 columns]
```

[24]:
```
volumeusd = stock_adj.xs(key='Volume USD',
                         level='Attributes',
                         axis=1)
volumeusd = volumeusd.round(2)
volumeusd
```

[24]:
```
Symbols                  BBRI.JK             BMRI.JK            BRIS.JK
Date
2020-01-02   157,375,882,916.670 121,085,428,320.950    475,044,585.660
2020-01-03   313,461,787,446.880 226,972,716,839.750  1,607,885,657.010
2020-01-06   165,335,827,872.880 196,608,067,369.140  2,208,455,440.580
2020-01-07   391,278,438,345.830 225,209,185,370.900  1,974,324,582.460
2020-01-08   643,562,271,192.770 329,410,184,669.630  1,244,136,031.700
…                            …                 …                 …
2023-12-15 1,401,090,840,000.000 840,054,980,000.000 20,940,090,000.000
2023-12-18   565,294,950,000.000 601,142,205,000.000 21,223,942,500.000
```

```
2023-12-19   750,400,515,000.000 243,560,120,000.000  9,994,539,500.000
2023-12-20   768,513,495,000.000 387,263,925,000.000 68,731,942,500.000
2023-12-21   552,201,520,000.000 300,924,302,500.000 42,773,900,000.000

[970 rows x 3 columns]
```

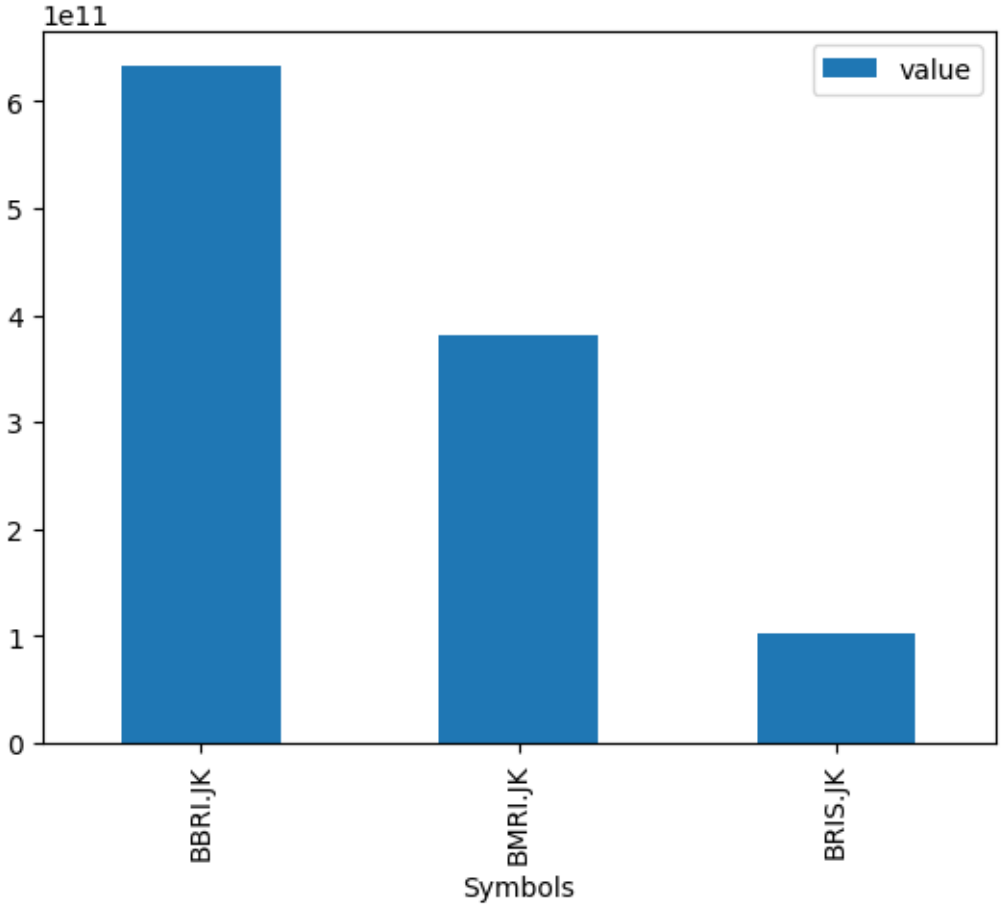
Perhatikan bagaimana bingkai data menunjukkan jumlah volume transaksi harian, misalnya kita ingin membandingkan rata-rata transaksi harian untuk BBRI.JK, BMRI.JK, dan BRIS.JK. Mari kita lakukan fungsi peleburan:

[25]:
```
volume_melted = volumeusd.melt()
volume_melted
```

[25]:
```
      Symbols               value
0     BBRI.JK 157,375,882,916.670
1     BBRI.JK 313,461,787,446.880
2     BBRI.JK 165,335,827,872.880
3     BBRI.JK 391,278,438,345.830
4     BBRI.JK 643,562,271,192.770
…       …                 …
2905  BRIS.JK  20,940,090,000.000
2906  BRIS.JK  21,223,942,500.000
2907  BRIS.JK   9,994,539,500.000
2908  BRIS.JK  68,731,942,500.000
2909  BRIS.JK  42,773,900,000.000

[2910 rows x 2 columns]
```

Misalkan kita ingin membandingkan rata-rata volume transaksi antara masing-masing harga saham. Rata-rata, manakah dari 3 saham tersebut yang memiliki rata-rata volume transaksi harian tertinggi?

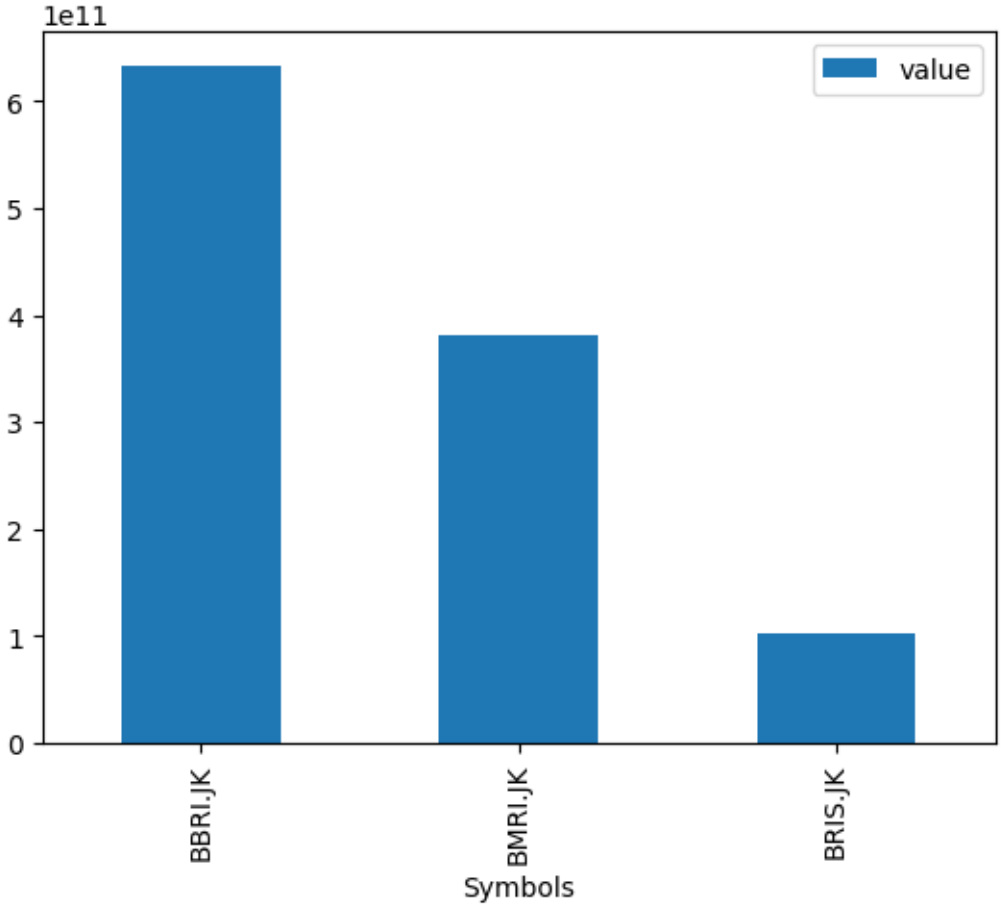
[26]:
```
volume_melted.groupby(['Symbols']).mean().plot.bar();
```



17.1 Visualisasi Barchart untuk Perbandingan

Katakanlah kita ingin membandingkan rata-rata volume penjualan harian dari perusahaan-perusahaan tersebut. Untuk melakukan itu, kita perlu mengekstrak atribut volume dari kerangka data kita, dan menjalankan fungsi leleh:

[27]:
```
volume_melted.groupby('Symbols').mean().plot.bar();
```



Jika kita membandingkan visualisasi dengan angka numerik, akan jauh lebih mudah untuk membandingkan volume rata-rata setiap saham. Sekarang mari kita pertimbangkan kerangka data berikut ini:

[28]:
```
bbri = stock.xs('BBRI.JK', level='Symbols', axis=1)
bbri = bbri.round(2)
bbri['Close_Diff'] = bbri['Close'].diff()
bbri['Weekday'] = bbri.index.day_name()
bbri
```

[28]:
```
Attributes  Adj Close     Close      High       Low      Open     Volume
Date
2020-01-02  3,772.730 4,410.000 4,410.000 4,360.000 4,400.000   41714100  \
2020-01-03  3,781.280 4,420.000 4,440.000 4,390.000 4,420.000   82898300
2020-01-06  3,738.510 4,370.000 4,390.000 4,320.000 4,360.000   44225100
2020-01-07  3,764.170 4,400.000 4,410.000 4,380.000 4,410.000  103948100
2020-01-08  3,747.060 4,380.000 4,400.000 4,340.000 4,380.000  171751200
…                 …         …         …         …         …          …
```

```
2023-12-15  5,550.000 5,550.000 5,600.000 5,550.000 5,575.000  252448800
2023-12-18  5,500.000 5,500.000 5,575.000 5,500.000 5,575.000  102780900
2023-12-19  5,550.000 5,550.000 5,550.000 5,450.000 5,450.000  135207300
2023-12-20  5,550.000 5,550.000 5,700.000 5,550.000 5,700.000  138470900
2023-12-21  5,575.000 5,575.000 5,600.000 5,525.000 5,550.000   99049600

Attributes  Close_Diff    Weekday
Date
2020-01-02         NaN   Thursday
2020-01-03      10.000     Friday
2020-01-06     -50.000     Monday
2020-01-07      30.000    Tuesday
2020-01-08     -20.000  Wednesday
…                 …         …
2023-12-15       0.000     Friday
2023-12-18     -50.000     Monday
2023-12-19      50.000    Tuesday
2023-12-20       0.000  Wednesday
2023-12-21      25.000   Thursday

[970 rows x 8 columns]
```

Berikan perhatian khusus pada bagaimana kolom Close_Diff dibuat. Ini adalah selisih antara nilai Penutupan harga saham pada hari tertentu dan hari berikutnya.

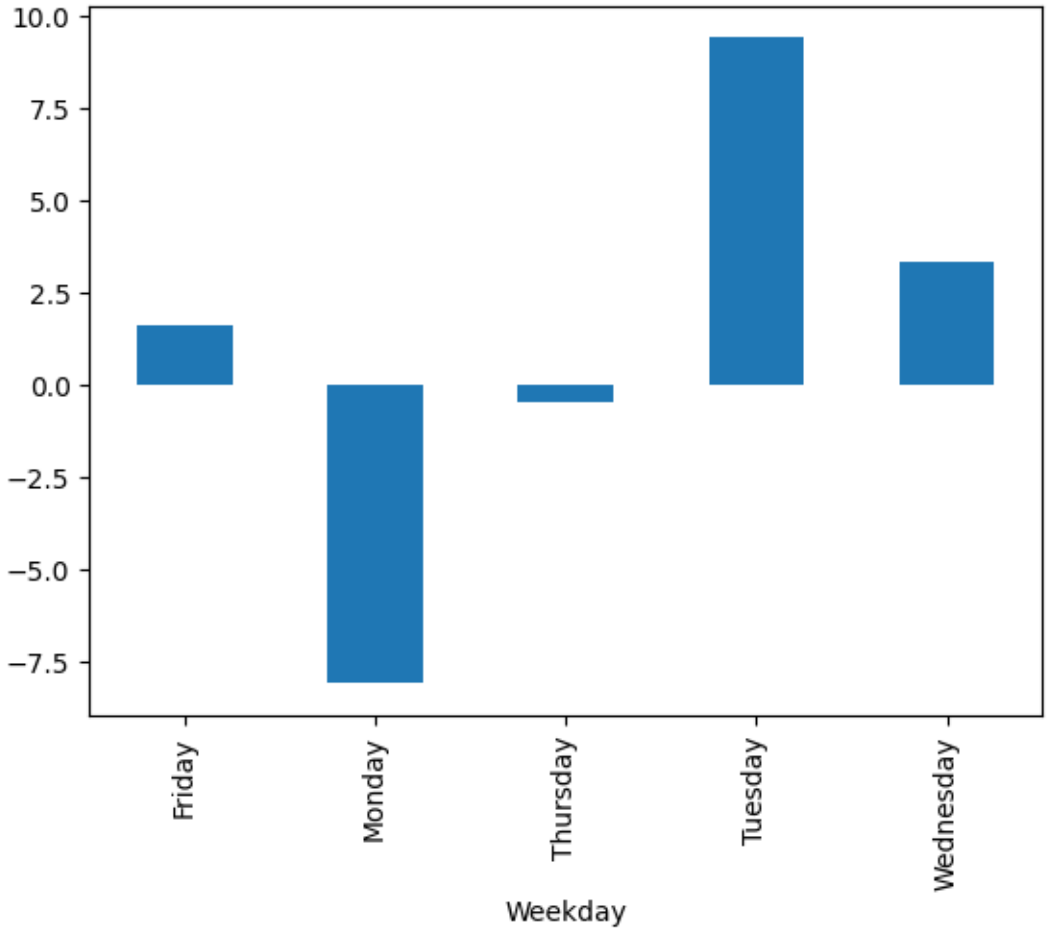
Misalkan kita ingin membandingkan `Close_Diff` antara setiap Hari Kerja; Rata-rata, apakah hari Selasa mencatat perbedaan yang lebih tinggi antara harga Penutupan saham BRI dibandingkan hari Kamis?

[29]:
```
bbri.groupby('Weekday').mean(numeric_only=True)
```

[29]:
```
Attributes  Adj Close     Close      High       Low      Open          Volume
Weekday
Friday      4,049.947 4,348.636 4,400.580 4,298.007 4,351.735 170,257,775.441  \
Monday      4,029.579 4,330.059 4,382.903 4,283.034 4,341.923 147,363,182.086
Thursday    4,021.308 4,325.322 4,379.851 4,278.062 4,332.870 166,182,807.247
Tuesday     4,033.065 4,330.981 4,386.888 4,284.111 4,332.204 163,620,687.776
Wednesday   4,030.649 4,331.024 4,384.109 4,282.291 4,338.537 169,151,324.725

Attributes  Close_Diff
Weekday
Friday           1.634
Monday          -8.090
Thursday        -0.457
Tuesday          9.430
Wednesday        3.370
```

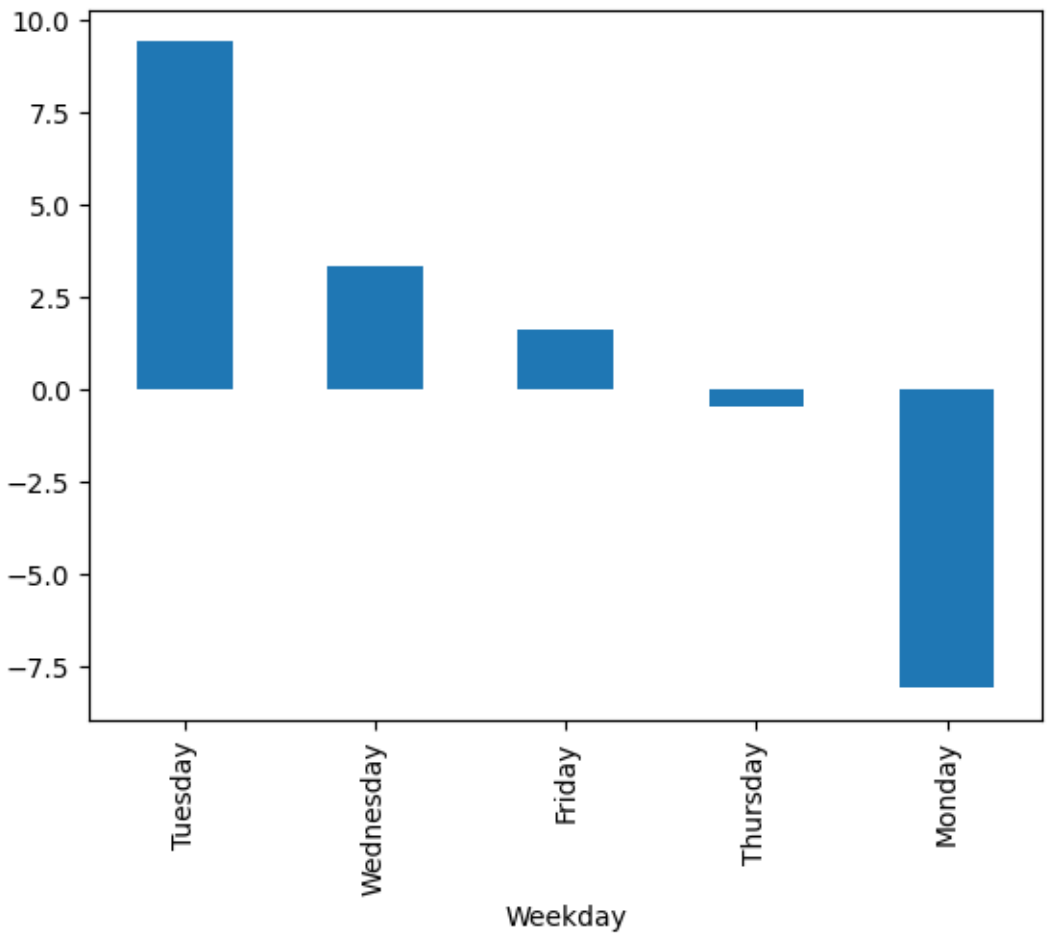
Now to create the same bar chart using `plot` function:

[30]:
```
bbri.groupby('Weekday').mean(numeric_only=True)['Close_Diff'].plot.bar();
```



Kita juga dapat meningkatkan efisiensi visualisasi dengan nilai rata-rata volume transaksi terlebih dahulu, sehingga batang dari plot kita akan disusun berdasarkan nilainya, bukan berdasarkan urutan abjad pada hari kerja.

[31]:
```
# berdasarkan urutan nilai terbesar - terkecil
bbri.groupby('Weekday').mean(numeric_only=True)['Close_Diff'].
↪sort_values(ascending=False).plot.bar();
```

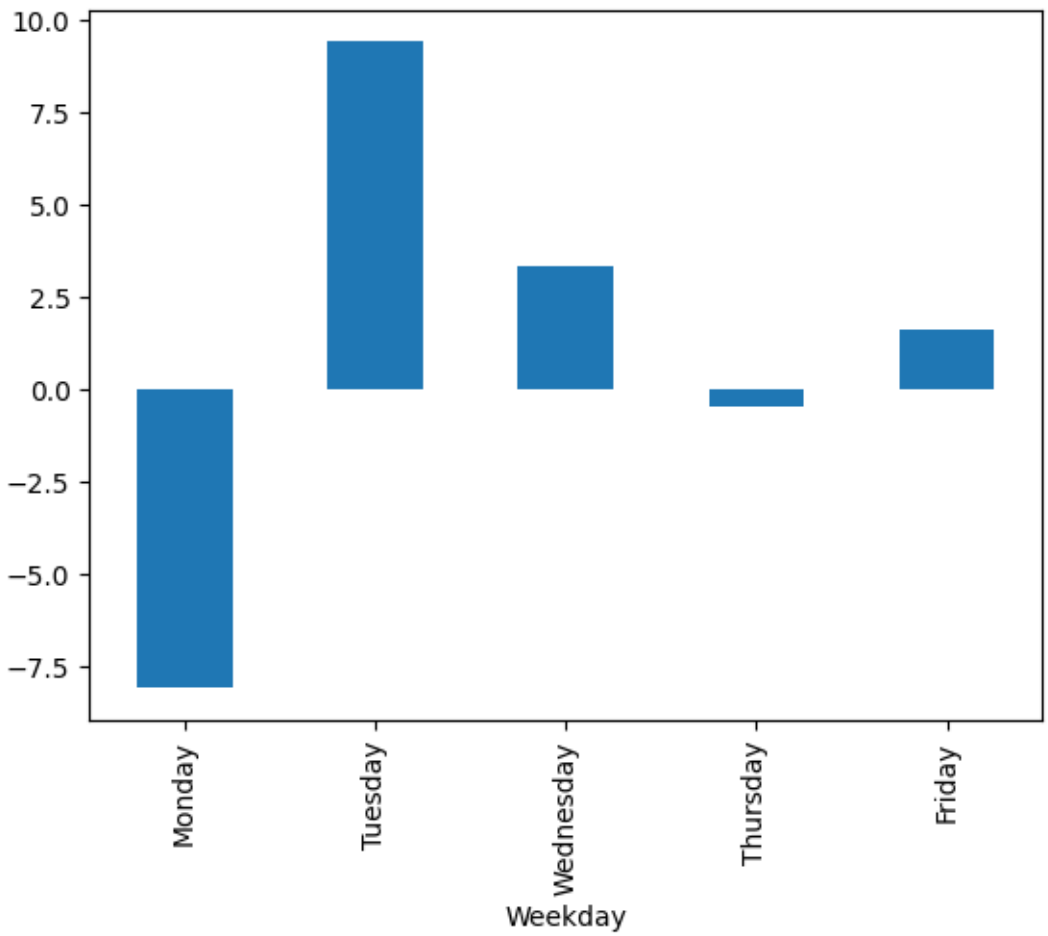


Kita juga dapat membuat indeks yang diurutkan secara manual dengan menentukan urutan hari itu.

[32]:
```
wday = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]

bbri_wday = bbri.groupby('Weekday').mean(numeric_only=True)['Close_Diff']
```

[33]:
```
bbri_wday.index = pd.CategoricalIndex(bbri_wday.index,\
                                      categories=wday,\
                                      ordered=True)
```

[34]:
```
bbri_wday.sort_index().plot.bar();
```



17.2 Menggunakan Barchart Group

Dengan menggunakan `closingprice`, kita dapat mencoba memvisualisasikan menggunakan barchart yang dikelompokkan untuk membandingkan harga penutupan setiap bulan untuk kuartal pertama tahun 2019 dan membandingkannya untuk 3 saham tersebut.

Pertama, lihat `harga penutupan` dan pastikan tidak ada nilai yang hilang pada data. Jika sudah, isi dengan cara yang sesuai

[35]:
```
closingprice = stock['Close']
closingprice.head()
```

[35]:
```
Symbols      BBRI.JK   BMRI.JK  BRIS.JK
Date
2020-01-02 4,410.000 3,875.000  332.000
2020-01-03 4,420.000 3,862.500  328.000
2020-01-06 4,370.000 3,800.000  324.000
2020-01-07 4,400.000 3,800.000  318.000
```

```
2020-01-08 4,380.000 3,750.000  312.000
```

[41]:
```
## Write your solution code here
# Buat kolom baru bernama 'Month', yang menunjukkan nama bulan pada tanggal␣
↪tersebut

closingprice.loc[:, 'Month'] = closingprice.index.month_name()
```

[40]:
```
closingprice
```

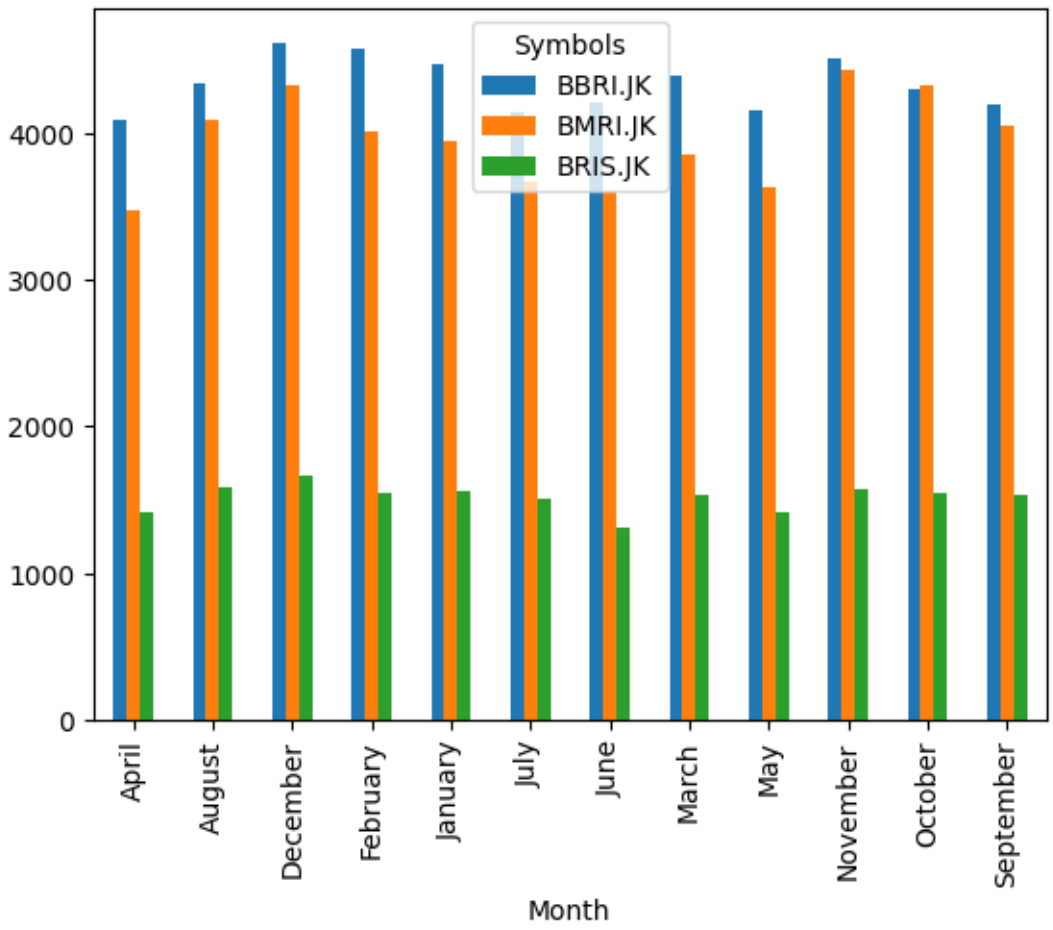
[40]:
```
Symbols      BBRI.JK   BMRI.JK   BRIS.JK     Month
Date
2020-01-02 4,410.000 3,875.000   332.000   January
2020-01-03 4,420.000 3,862.500   328.000   January
2020-01-06 4,370.000 3,800.000   324.000   January
2020-01-07 4,400.000 3,800.000   318.000   January
2020-01-08 4,380.000 3,750.000   312.000   January
…               …        …        …        …
2023-12-15 5,550.000 5,900.000 1,700.000  December
2023-12-18 5,500.000 5,925.000 1,695.000  December
2023-12-19 5,550.000 5,975.000 1,705.000  December
2023-12-20 5,550.000 5,925.000 1,755.000  December
2023-12-21 5,575.000 5,975.000 1,690.000  December

[970 rows x 4 columns]
```

Setelah kita memiliki kolom `Month`, mari kelompokkan berdasarkan Bulan dan lihat DataFrame yang dihasilkan

[34]:
```
average_closing = closingprice.groupby('Month').mean()
average_closing
```

[34]:
```
Symbols     BBRI.JK   BMRI.JK   BRIS.JK
Month
April     4,081.733 3,475.367 1,419.053
August    4,334.204 4,088.415 1,581.646
December  4,610.789 4,328.618 1,662.105
February  4,572.727 4,013.961 1,540.818
January   4,467.738 3,941.964 1,563.738
July      4,135.952 3,672.857 1,502.476
June      4,199.750 3,603.344 1,307.487
March     4,385.581 3,847.384 1,532.256
May       4,156.159 3,627.935 1,408.116
November  4,505.000 4,430.172 1,576.552
October   4,300.305 4,316.921 1,546.829
September 4,194.843 4,051.366 1,528.023
```

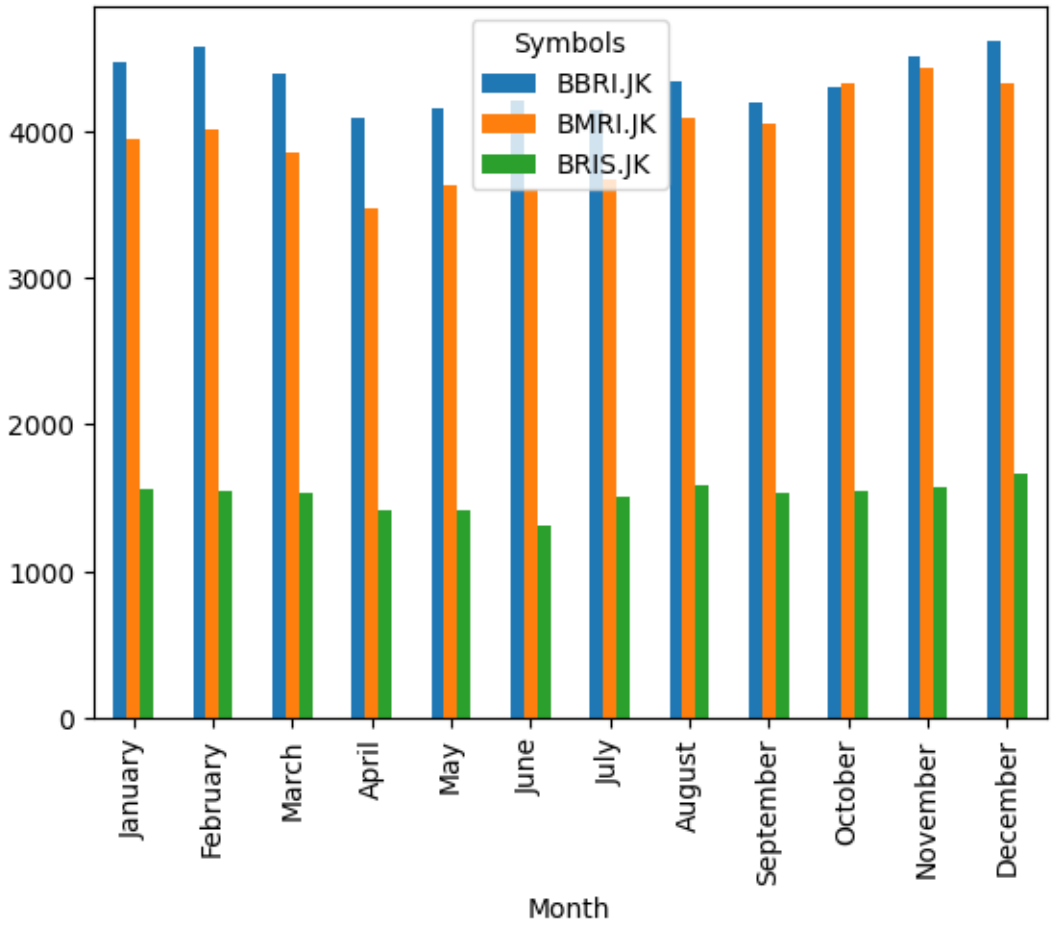
[35]:
```
average_closing.sort_index().plot.bar();
```



Namun, jika ingin menyusun ulang bulannya, kita harus menetapkan indeks sebagai nilai kategorikal yang diurutkan (Lihat materi Analisis Data Eksplorasi jika Anda perlu mengingatnya).

[36]:
```
months= ['January', 'February', 'March', 'April', 'May', 'June', 'July',␣
↪'August', 'September', 'October', 'November', 'December']
```

[37]:
```
average_closing.index = pd.CategoricalIndex(average_closing.index,\
                                            categories=months,\
                                            ordered=True)

average_closing.sort_index().plot.bar();
```



Referensi lengkap ke dokumentasi resmi tentang metode ini berada di luar cakupan buku pelajaran ini, tapi layak untuk dibaca.

17.3 Menggabungkan `agg` dan `groupby`

Sejauh ini, kami telah menjelajahi beberapa perangkat agregasi panda, seperti:

- `pd.crosstab()`
- `pd.pivot_table()`

Dalam bab ini, kita akan menjelajahi alat agregasi panda lainnya:

- agregasi `groupby`.

Pembahasan:

(`pivot_table` & `pd.crosstab` kesetaraan)

Metode `pivot_table` dan fungsi `crosstab` keduanya dapat menghasilkan hasil yang sama persis dengan bentuk yang sama. Keduanya berbagi parameter; `indeks`, `kolom`, `nilai`, dan `aggfunc`.

Perbedaan utama yang terlihat adalah `crosstab` adalah sebuah fungsi dan bukan metode

DataFrame. Ini memaksa Anda untuk menggunakan kolom sebagai Seri dan bukan nama string untuk parameternya.

1. Misalkan Anda ingin membandingkan jumlah total transaksi selama Hari Kerja setiap periode kuartal. Buat `pivot_table` yang menyelesaikan masalah!
2. Cobalah untuk mereproduksi hasil yang sama dengan menggunakan `crosstab`
3. Bagaimana jika, alih-alih membandingkan total transaksi, Anda ingin membandingkan total pendapatan dari periode yang sama? Gunakan `pivot_table` dan `crosstab` sebagai solusinya. Diskusikan dengan teman anda, metode mana yang lebih relevan dalam kasus ini?

Perhatikan kelompok berikut berdasarkan operasi:

[38]:
```
stock.stack().reset_index().groupby('Symbols').mean(numeric_only=True)
```

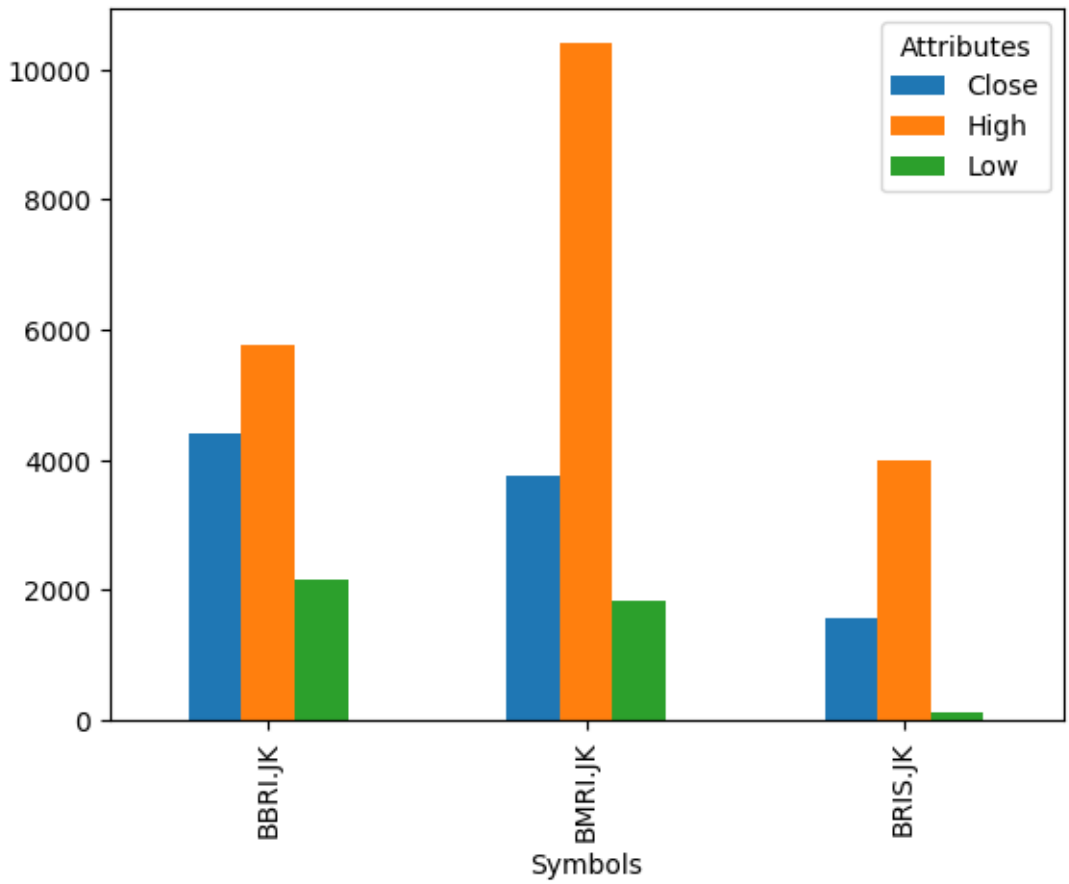
[38]:
```
Attributes  Adj Close     Close      High       Low      Open          Volume
Symbols
BBRI.JK     4,029.681 4,330.575 4,384.187 4,282.450 4,336.708 163,307,060.979
BMRI.JK     3,704.355 3,957.539 4,007.769 3,908.425 3,960.150 106,337,189.669
BRIS.JK     1,498.105 1,516.107 1,553.779 1,487.457 1,518.201  66,685,960.537
```

[39]:
```
stock.stack().reset_index().groupby('Symbols').agg({
    'Close': 'mean',
    'High': 'max',
    'Low': 'min'
})
```

[39]:
```
Attributes     Close       High       Low
Symbols
BBRI.JK    4,330.575  5,750.000 2,160.000
BMRI.JK    3,957.539 10,400.000 1,830.000
BRIS.JK    1,516.107  3,980.000   135.000
```

Katakanlah kita ingin mengetahui sekilas harga saham maksimum, harga saham minimum, dan rata-rata harga penutupan dari ketiga perusahaan tersebut. Untuk melakukannya, kita perlu menggabungkan `groupby` dengan `agg` dan memetakan setiap kolom dengan fungsi agregasinya.

[41]:
```
stock.stack().reset_index().groupby('Symbols').agg({
    'Close': 'median',
    'High': 'max',
    'Low': 'min'
}).plot.bar();
```



Knowledge Check: Menggunakan `plot`

Pertimbangkan kerangka data berikut:

[42]:
```
import datetime

stock['YearMonth'] = pd.to_datetime(stock.index.date).to_period('M')
monthly_closing = stock.groupby('YearMonth').mean().loc[:,['Close','Low',␣
↪'High']]
monthly_closing.head()
```

[42]:
```
Attributes     Close                          Low                          High
Symbols      BBRI.JK   BMRI.JK BRIS.JK   BBRI.JK   BMRI.JK BRIS.JK   BBRI.JK
YearMonth
2020-01    4,550.000 3,847.727 317.909 4,513.182 3,805.114 314.727 4,576.818  \
2020-02    4,477.500 3,862.500 296.900 4,439.500 3,807.500 293.000 4,528.500
2020-03    3,440.952 2,916.667 196.857 3,362.857 2,847.976 187.286 3,615.714
2020-04    2,785.714 2,260.238 194.714 2,727.143 2,196.190 191.095 2,878.095
2020-05    2,530.625 2,045.312 267.500 2,468.750 2,004.062 254.625 2,590.000
```

```
Attributes
Symbols      BMRI.JK BRIS.JK
YearMonth
2020-01    3,892.045 322.909
2020-02    3,900.000 303.800
2020-03    3,025.238 212.810
2020-04    2,337.738 202.286
2020-05    2,096.875 284.500
```

Manakah dari berikut ini yang merupakan plot yang sesuai untuk digunakan?

- ☐ Line plot -> .plot()
- ☐ Scatter plot -> .plot.scatter(x=? , y=?)
- ☐ Bar plot -> .plot.bar()
- ☐ Box plot -> .plot.box()

[51]:
```
## Your code below
```

Coursebook: Regression Model

- Bagian 4 Audit Analytics untuk Bank Rakyat Indonesia
- Durasi: 7 Jam
- *Last Updated*: December 2023

- *Coursebook* ini disusun dan dikurasi oleh tim produk dan instruktur dari Algoritma Data Science School

18 Background

Coursebook ini merupakan bagian dari **BRI Audit Analytics** yang disiapkan oleh Algoritma. *Coursebook* ini ditujukan hanya untuk khalayak terbatas, yaitu individu dan organisasi yang menerima *coursebook* ini langsung dari organisasi pelatihan. Tidak boleh direproduksi, didistribusikan, diterjemahkan, atau diadaptasi dalam bentuk apapun di luar individu dan organisasi ini tanpa izin. Algoritma adalah pusat pendidikan *data science* yang berbasis di Jakarta. Kami menyelenggarakan *workshop* dan program pelatihan untuk membantu para profesional dan mahasiswa dalam menguasai berbagai sub-bidang *data science* yaitu: *data visualization, machine learning,* statistik, dan lain sebagainya.

18.1 Training Objectives

Kita akan memulai dengan mempelajari apa itu *machine learning* secara umum dan mempelajari model regresi pertama Anda. Pada *coursebook* ini kita akan membahas:

- **Data Preparation & Exploratory Data**
  - Data types inspection & preparation
  - Statistics summary
  - Correlation and distribution from several variables
- **Data Pre-Processing**
  - Target & predictors separation
  - Dummy Variables for Categorical Predictor
- **Linear Model**
  - Ordinary Least Square
  - Linear Regression
  - Linear Model Prediction
  - Plotting Regression
- **Interpreting Linear Models**
  - Understanding Coefficients
  - Estimates and standard error
  - Confidence interval
- **Model Assumptions and Evaluation**
  - Model assumptions

– Model evaluation

19 Data Preparation

Sebelum masuk ke topik utama pada coursebook ini, mari kita import terlebih dahulu beberapa library yang akan digunakan:

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# suppress scientific notation
np.set_printoptions(suppress=True) # numpy output
pd.set_option('display.float_format', lambda x: '%.5f' % x) # pandas output

# suppress warning
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

Setelah menentukan library yang akan digunakan maka kita akan melakukan persiapan data terlebih dahulu.

[2]:
```
copiers = pd.read_csv("data_input/copiers.csv")
copiers.head()
```

[2]:
```
         Order.ID Order.Date Ship.Date       Ship.Mode Customer.ID  \
0  CA-2015-137946     9/1/15    9/4/15    Second Class    DB-13615
1  US-2014-135972    9/21/14   9/23/14    Second Class    JG-15115
2  CA-2017-117457    12/8/17  12/12/17  Standard Class    KH-16510
3  CA-2017-127432    1/22/17   1/27/17  Standard Class    AD-10180
4  CA-2014-131450     8/8/14   8/15/14  Standard Class    LR-16915

       Segment       Product.ID    Category Sub.Category  \
0     Consumer  TEC-CO-10001449  Technology      Copiers
1     Consumer  TEC-CO-10002313  Technology      Copiers
2     Consumer  TEC-CO-10004115  Technology      Copiers
3  Home Office  TEC-CO-10003236  Technology      Copiers
4     Consumer  TEC-CO-10004115  Technology      Copiers

                          Product.Name      Sales  Quantity  Discount  \
0  Hewlett Packard LaserJet 3310 Copier  959.98400         2   0.20000
1         Canon PC1080F Personal Copier 1799.97000         3   0.00000
2        Sharp AL-1530CS Digital Copier 1199.97600         3   0.20000
3         Canon Image Class D660 Copier 2999.95000         5   0.00000
4        Sharp AL-1530CS Digital Copier 1199.97600         3   0.20000
```

```
      Profit
0  335.99440
1  701.98830
2  434.99130
3 1379.97700
4  434.99130
```

[3]:
```
copiers.shape
```

[3]:
```
(62, 14)
```

Data di atas merupakan data historis transaksi dari suatu online retail, berikut deskripsi variabel dari data tersebut:

- `Order.ID` : ID transaksi
- `Order.Date` : Tanggal transaksi
- `Ship.Date` : Tanggal pengiriman barang
- `Ship.Mode` : Jenis pengiriman yang digunakan (Standard Class, Second Class, First Class, Same Day)
- `Customer.ID` : ID pelanggan
- `Segment` : Segmen pelanggan (Consumer, Corporate, Home Office)
- `Product.ID` : ID barang
- `Category` : Kategori barang
- `Sub.Categoory`: Sub kategori barang
- `Product.Name` : Nama barang
- `Sales` : Harga barang (USD)
- `Quantity` : Jumlah barang yang dibeli
- `Discount` : Potongan harga (%)
- `Profit` : Keuntungan yang diperoleh (USD)

Berdasarkan informasi di atas, kita akan melihat tipe data pada seluruh variabel yang kita miliki:

[4]:
```
copiers.dtypes
```

[4]:
```
Order.ID         object
Order.Date       object
Ship.Date        object
Ship.Mode        object
Customer.ID      object
Segment          object
Product.ID       object
Category         object
Sub.Category     object
Product.Name     object
Sales           float64
Quantity          int64
Discount        float64
Profit          float64
dtype: object
```

Beberapa variabel yang nilainya berulang sebaiknya kita ubah terlebih dahulu ke dalam tipe kategori. Variabel-variabel ini diketahui berdasarkan keterangan data sebelumnya.

[5]:
```
cat_var = ["Ship.Mode", "Segment", "Category", "Sub.Category"]

copiers[cat_var] = copiers[cat_var].astype("category")

copiers.dtypes
```

[5]:
```
Order.ID          object
Order.Date        object
Ship.Date         object
Ship.Mode       category
Customer.ID       object
Segment         category
Product.ID        object
Category        category
Sub.Category    category
Product.Name      object
Sales            float64
Quantity           int64
Discount         float64
Profit           float64
dtype: object
```

Setelah tipe datanya berubah, kita dapat melihat secara lebih mendetail informasi pada data kita. Hal pertama yang akan kita tinjau adalah deskripsi statistik dari data kita.

[6]:
```
copiers.describe()
```

[6]:

	Sales	Quantity	Discount	Profit
count	62.00000	62.00000	62.00000	62.00000
mean	1331.10155	3.30645	0.16774	448.80549
std	966.49320	1.76104	0.12646	434.97181
min	299.99000	1.00000	0.00000	59.99800
25%	599.97250	2.00000	0.00000	161.99190
50%	979.98200	3.00000	0.20000	313.49215
75%	1749.95750	5.00000	0.20000	611.23825
max	4899.93000	8.00000	0.40000	2302.96710

Secara default, metode `describe()` hanya akan menampilkan data-data dengan tipe numerik. Agar kita dapat melihat data dengan tipe kategorik maka kita gunakan parameter include.

[7]:
```
copiers.describe(include="category")
```

[7]:

	Ship.Mode	Segment	Category	Sub.Category
count	62	62	62	62
unique	4	3	1	1
top	Standard Class	Consumer	Technology	Copiers

```
freq                32        32          62          62
```

Sampai tahapan ini, kita telah mengubah tipe data dan melihat secara singkat deskripsi statistik dari data kita. Hal-hal tersebut merupakan tahapan penting sebelum kita dapat mengolah data kita ke tahap selanjutnya yaitu *machine learning*.

20 Introduction to Machine Learning

Machine learning pada dasarnya mengacu pada sub-bidang ilmu komputer yang "memberikan kemampuan kepada komputer untuk belajar tanpa harus diprogram secara eksplisit". Lebih sederhananya, hal ini berkaitan dengan teori dan aplikasi metode statistik dan matematika untuk mencapai tujuan tertentu tanpa mengikuti seperangkat aturan yang ditentukan secara kaku dan ketat.

Machine learning yang paling umum dibagi menjadi dua jenis, yaitu ***supervised learning*** dan ***unsupervised learning***. Perbedaan antara keduanya adalah ketika kita membahas ***supervised learning***, maka kita berusaha melakukan prediksi dari suatu nilai karena dalam supervised learning data yang akan kita analisis memiliki target variabel.

Data kita (yang umumnya berbentuk tabular) akan dipisahkan pada kasus supervised learning, menjadi target variable dan predictor variable:

- ***Target Variable***: Satu buah variabel/kolom yang ingin kita prediksi nilainya, biasa disebut juga sebagai dependent variable.
- ***Predictor Variable***: Variabel-variabel yang mempengaruhi target variabel atau nilai yang akan kita prediksi, biasa disebut juga sebagai independent variable.

Ketika target variable bersifat numerik (seperti harga minyak, curah hujan, penjualan, tekanan darah, dll), umumnya kasus ini disebut sebagai "regresi". Hal ini berbeda dengan masalah "klasifikasi", istilah umum untuk ketika nilai target variable kita bersifat kategorikal (*loan defaults*, deteksi *email spam, handwriting recognition*, dll).

Sementara itu, jika kita membahas tentang ***unsupervised learning***, maka kita berusaha melakukan eksplorasi data karena data yang ingin kita proses tidak memiliki target variabel. Metode yang biasanya digunakan untuk menganalisis data menggunakan unsupervised learning adalah clustering atau data grouping.

21 Regression Model

Metode *machine learning* pertama yang akan kita pelajari yaitu model regresi.

Penting untuk diingat bahwa model regresi tidak hanya digunakan dalam konteks *machine learning* untuk prediksi numerik. Regresi, sebenarnya, merupakan "workhorse of data science" dan termasuk model yang paling praktis dan teoretis untuk dipahami dalam statistik. Model regresi linear merupakan metode statistik yang membentuk sebuah garis lurus berdasarkan data kita. Persamaan umumnya yaitu:

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n$$

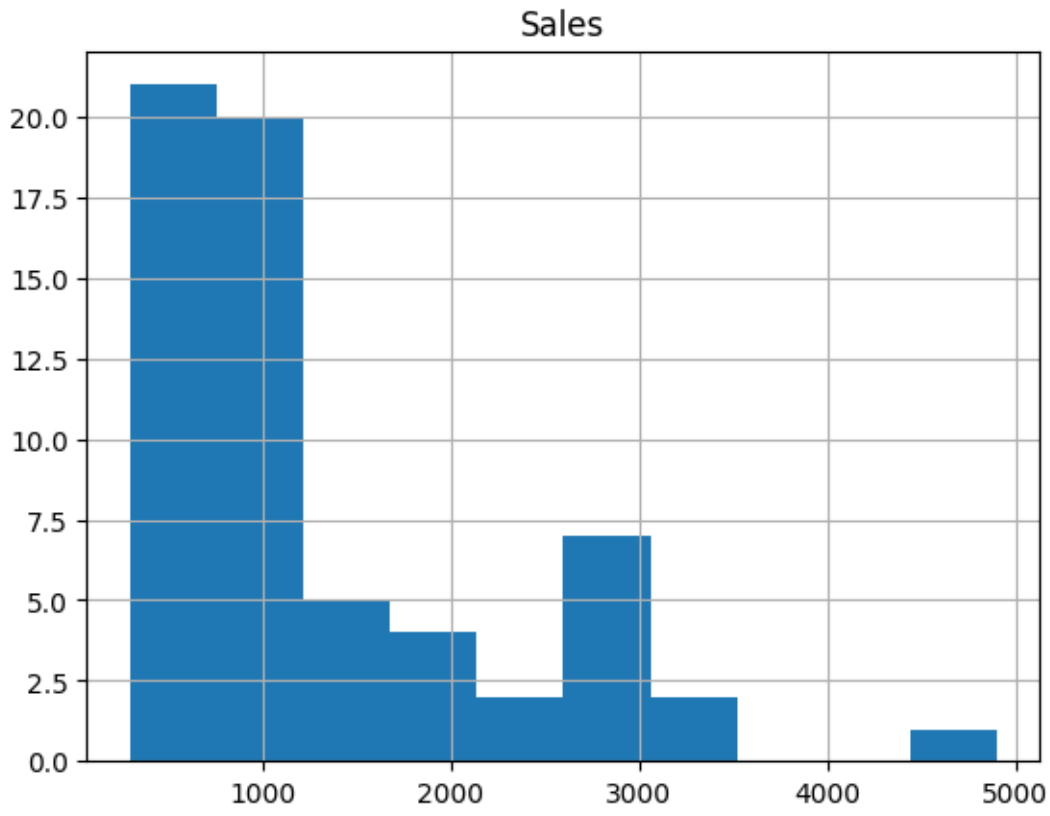
Data scientist yang terlatih dengan baik dengan ilmu dasar ini akan mampu mengatasi berbagai masalah karena model regresi sering menghasilkan model yang sangat dapat diinterpretasi.

Kita dapat (dan seharusnya) mempertimbangkannya sebagai metode statistik yang berguna pada kebanyakan kasus *data science* yang paling umum.

21.1 Least Squares Regression

Salah satu istilah yang akan Anda dengar dalam course ini adalah **least squares** (kuadrat terkecil). Jadi mari kita coba dan memahami intuisi tentang konsep penting ini melalui ilustrasi berikut. Kita akan melanjutkan dan membaca suatu data dan membuat histogram dari data tersebut.

[8]:
```
copiers.hist(column=['Sales']);
```



Ketika kita memiliki sebuah data (contoh: Profit mesin fotokopi) dengan distribusi penjualan seperti di atas dan kita ingin membuat perkiraan nilai yang paling representatif maka kita akan menghitung jumlah jarak kuadrat antara data yang kita perkirakan dengan data aslinya (*error*). Inilah konsep kuadrat terkecil atau *least square*. Tanpa informasi lain (variabel prediktor), *least square* akan memprediksi nilai sebuah data dengan nilai rata-rata sampelnya.

Karena hasil estimasi model kita tidak akan benar-benar tepat jatuh di nilai asli data kita, maka dengan meminimalisir rata-rata kuadrat dari *error* adalah pendekatan yang masuk akal dilakukan.

Pendekatan *least square* ini merupakan dasar dari metode regresi linear. Regresi linear akan membuat satu buah garis yang paling representatif terhadap data kita, kondisi garis paling representatif ini berdasarkan jarak minimum seperti yang sudah dijelaskan di atas.

21.2 Simple Linear Regression

Secara sederhana, model regresi akan membuat satu buah garis linear yang berdasarkan gambaran data kita. Garis linear ini dituliskan pada persamaan $\hat{Y} = \beta_0 + \beta_1 X_1$.

Berdasarkan persamaan di atas, kita harus menentukan "Y" sebagai target variabel dan "X" sebagai prediktor. Pemilihan "X" atau prediktor ini umumnya didasari oleh hubungan linear antara prediktor dengan target ("Y"). Hal ini dapat kita tinjau dengan melihat nilai korelasi. Berikut adalah cara pengecekan korelasi menggunakan metode `.corr()`.

[9]:
```
copiers.corr(numeric_only=True).sort_values(
    'Profit', ascending=False
).loc[:,'Profit']
```

[9]:
```
Profit      1.00000
Sales       0.93958
Quantity    0.68599
Discount   -0.43234
Name: Profit, dtype: float64
```

Pada kasus dalam pembelajaran regresi linear ini, kita akan berusaha memprediksi profit penjualan mesin fotokopi (`Profit`) dan variabel dengan korelasi paling tinggi berdasarkan hasil eksplorasi korelasi di atas adalah `Sales` yang merepresentasikan harga barang. Nilai korelasi yang tinggi dapat membantu metode kita untuk mendapatkan hasil yang baik karena regresi linear didasari hubungan linear prediktor dengan target.

Oleh karena itu, pada kasus pertama kita yaitu regresi linear sederhana kita akan memprediksi `Profit` berdasarkan `Sales`.

Pembuatan model regresi linear di Python merupakan proses yang mudah. Kita akan memanggil fungsi `OLS()` dari library `statsmodels` dan menentukan dua parameter utama: "Y" atau target variabel dan "X" atau prediktor. Kedua parameter ini harus dipisahkan dan sebaiknya didefinisikan sebelum membuat model.

Notes: Nama variabel "Y" dan "X" tidaklah baku, kita dapat menentukan sesuai kebutuhan kita.*

[10]:
```
import statsmodels.api as sm

Y = copiers['Profit']
X_simple = sm.add_constant(copiers['Sales'])

# fit the model
lm_simple = sm.OLS(Y, X_simple).fit()
lm_simple.params
```

[10]:
```
const   -114.06251
Sales      0.42286
dtype: float64
```

Note: Tanpa sm.add_constant, `statsmodels` akan menggunakan asumsi $\beta_0 = 0$ pada persamaan $\hat{Y} = \beta_0 + \beta_1 X_1$ sehingga nilai β_0 yang seharusnya dihitung berdasarkan data kita akan bernilai 0.*

Nilai yang dikeluarkan oleh `lm.params` adalah koefisien dari model linear yang dibuat. Nilai koefisien ini menggambarkan seberapa banyak pengaruh sebuah variabel terhadap target variabel kita.

Jika kita lihat pada contoh di atas, kita memiliki nilai koefisien untuk variabel Tenure sebesar **0.423** dan karena nilai tersebut bertanda positif maka berdasarkan hubungan linear dari model kita dapat dikatakan bahwa dampak variabel tersebut terhadap Profit (target variabel) juga positif: semakin tinggi Sales maka semakin tinggi juga Profit. Sebaliknya, nilai koefisien negatif artinya kenaikan sebuah variabel akan menurunkan nilai targetnya.

Selain informasi di atas, kita juga dapat menyimpulkan persamaan regresi linear untuk penggunaan bandwidth tahunan akan mengikuti persamaan umum $\hat{Y} = \beta_0 + \beta_1 X_1$.

Jika persamaan di atas dituliskan dengan bahasa yang lebih sederhana menjadi: `Estimated Profit = Intercept + Slope * Sales` dan jika nilai koefisien betanya kita masukkan ke dalam formula tersebut maka menjadi: `Estimated Profit = -114.06 + 0.423 * Sales`

Dari persamaan tersebut kita dapat menyimpulkan bahwa **keuntungan diekspektasikan meningkat sebesar** 0.423 **ketika sales meningkat** 1 **satuan**. Jika nilai sales 0 maka keuntungan diprediksi bernilai negatif (profit tanpa sales diperkirakan sebesar −114.06 USD).

Apabila kita memiliki pelanggan dengan nilai Sales 910, berapakah prediksi Profit berdasarkan model linear kita?

[11]:
```
-114.06 + 0.423 * 910
```

[11]:
```
270.87
```

Ternyata berdasarkan formula model linear, kita ketahui bahwa penggunaan Profit kita senilai 270.87 USD. Hal ini cukup mudah kita lakukan ketika hanya memiliki satu variabel dan menebak satu pelanggan, akan tetapi akan sulit ketika keduanya sudah lebih dari satu. Oleh karena itu, `statsmodels` memiliki fungsi `*.predict()` yang dapat melakukan prediksi berdasarkan input data kita. Fungsi predict membutuhkan informasi yang sama dengan X yang masuk pada model machine learning kita sebagai parameter pertamanya. Pada kasus kita artinya dataframe dengan sebuah nilai konstanta dan nilai tenure.

[12]:
```
c=[1, 910]
lm_simple.predict(c)
```

[12]:
```
array([270.73899142])
```

Berikut ini adalah contoh penggunaan `*.predict()` pada dataframe yang barisnya lebih dari satu.

[13]:
```
lm_simple.predict(X_simple)[:5]
```

[13]:
```
0    291.87517
1    647.07063
2    393.35789
3   1154.49273
4    393.35789
dtype: float64
```

Hasil prediksi menggunakan `*.predict()` kepada data X kita akan menghasilkan nilai yang sama dengan nilai `*.fittedvalues` pada model kita.

[14]:
```
lm_simple.fittedvalues[:5]
```

[14]:
```
0    291.87517
1    647.07063
2    393.35789
3   1154.49273
4    393.35789
dtype: float64
```

Dan berdasarkan informasi tersebut, marilah kita buat plot hubungan antara Sales dengan Profit tetapi dengan tambahan sebuah garis yang dihasilkan dari prediksi model linear kita:

[15]:
```
plt.scatter(copiers['Sales'],  copiers['Profit']);
plt.plot(copiers['Sales'], lm_simple.fittedvalues, c='r');
plt.xlabel('Sales');
plt.ylabel('Profit');
plt.title('Sales vs. Profit');
```

Ternyata, garis tersebut adalah garis regresi terbaik untuk data asli kita. Garis ini telah disesuaikan dengan kondisi error karena titik-titik observasinya tidak diprediksi dengan sempurna oleh garis (meskipun mereka berada di sekitar garis), dan model linear kita telah memperkirakan garis ini untuk meminimalisir kesalahan kuadrat terkecil (**least square error**).

Perhitungan error pada metode least square didasari oleh formula `(actual-estimation)**2).sum()` yang umumnya disebut sebagai **sum of squared errors** atau SSE. Kita dapat melihat nilai `actual-estimation` (error) ini menggunakan attribute `*.resid` pada model kita dan jika dihitung akan mendapatkan nilai SSE sebesar:

[16]:
```
((lm_simple.resid)**2).sum()
```

[16]:
```
1352542.6932553158
```

Garis lurus yang dibentuk oleh model regresi linear merupakan hasil estimasi **dua parameter** yaitu titik ketika garis kita memotong sumbu y (Intercept) dan derajat kemiringan (Slope). Kedua nilai inilah yang ditunjukkan oleh `lm.params` atau nantinya muncul pada bagian **coef** yang merujuk pada "beta coefficients".

Nilai tersebut akan muncul ketika kita melihat fungsi `*.summary()` dari model kita. Fungsi ini umumnya terdapat pada seluruh model yang dibentuk oleh library **statsmodels**.

[17]:
```
lm_simple.summary()
```

[17]:

Dep. Variable:	Profit	**R-squared:**	0.883
Model:	OLS	**Adj. R-squared:**	0.881
Method:	Least Squares	**F-statistic:**	452.0
Date:	Thu, 21 Dec 2023	**Prob (F-statistic):**	1.27e-29
Time:	18:01:02	**Log-Likelihood:**	-397.68
No. Observations:	62	**AIC:**	799.4
Df Residuals:	60	**BIC:**	803.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	-114.0625	32.627	-3.496	0.001	-179.327	-48.798
Sales	0.4229	0.020	21.260	0.000	0.383	0.463

Omnibus:	2.345	**Durbin-Watson:**	2.295
Prob(Omnibus):	0.310	**Jarque-Bera (JB):**	1.652
Skew:	-0.380	**Prob(JB):**	0.438
Kurtosis:	3.250	**Cond. No.**	2.81e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.81e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Untuk saat ini, mari kita fokus pada area berikut:

[18]:
```
lm_simple.summary().tables[1]
```

[18]:

	coef	std err	t	P> |t|	[0.025	0.975]
const	-114.0625	32.627	-3.496	0.001	-179.327	-48.798
Sales	0.4229	0.020	21.260	0.000	0.383	0.463

Berikut adalah penjelasan mendetail untuk setiap kolom pada output tersebut:

- Pada kolom pertama output tersebut, kita dapat melihat terdapat dua baris yang merepresentasikan koefisien pada garis regresi kita. Nilai `coef` pada **const** merupakan intercept y dan pada **Sales** merupakan nilai slope. Titik ketika garis kita memotong sumbu y adalah sebesar -114.0625 dan derajat kemiringannya sebesar 0.4229. Koefisien derajat kemiringan diestimasi agar jarak vertikal antara titik asli data dan prediksi bernilai minimal.
- Kolom kedua yaitu standard error (`std err`) merupakan estimasi standar deviasi dari masing-masing koefisien. Nilai ini mengindikasikan seberapa besar prediksi kita bervariasi pada seluruh sampel. Semakin kecil nilainya, maka dapat dikatakan nilai koefisien kita semakin akurat.
- Nilai t-value (`t`) merupakan hasil perhitungan `coef` dibagi dengan `std err`.

[19]:
```
# t-value pada const
-114.0625/32.627
```

[19]:
```
-3.4959542710025437
```

- P-value (`P>|t|`) menunjukkan peluang untuk mendapatkan nilai t-value yang nilainya ekstrim. Nilai p-value yang lebih kecil dari 0.05 menunjukkan bahwa prediktor pada baris tersebut secara statistik berpengaruh signifikan terhadap model.
- Confidence interval (`[0.025  0.975]`) mewakili rentang di mana koefisien kita kemungkinan besar akan berada (dengan kemungkinan 95%).

Alasan kenapa kita sangat peduli terhadap pengujian statistik ini adalah karena dalam pemodelan regresi, kita berusaha menyelidiki apakah ada bukti yang cukup bahwa suatu variabel independent (atau "variabel prediktor") berbeda dengan 0. Artinya, kita ingin tahu apakah variabel seperti `Quantity`, `Discount`, dan jumlah `Ship.Mode` memiliki efek terhadap Profit.

Menggunakan model regresi, kita ingin melihat apakah variabel-variabel tadi benar-benar mempengaruhi variabel target (Profit) atau apakah perbedaan yang muncul pada Profit hanya disebabkan oleh kebetulan dari random sampling.

Hipotesis nol merupakan alasan kita mengamati perbedaan penggunaan bandwidth antar area, misalnya. Hal ini mungkin terjadi akibat kebetulan dari random sampling saja. Artinya jika kita mengambil sample dengan jumlah yang berbeda pada waktu yang berbeda, kita mungkin saja mendapatkan skenario berbeda atau bahkan kontradiktif dengan kondisi saat ini.

Jika Anda telah mempelajari statistika inferential, kita ketahui bahwa *rule of thumb* untuk menolak hipotesis nol (H0) dan mendukung hipotesis alternatif (H1) memiliki threshold 0.05.

***Note**: Apabila Anda tertarik dengan konsep statistik di atas secara lebih mendalam, kami merekomendasikan sebuah copy buku PDF berjudul The Elements of Statistical Learning: Data Mining, Inference, and Prediction*

21.3 R-Squared

Sekarang, mari kita kembali melihat output `summary()` dari model kita.

[20]:
```
lm_simple.summary()
```

[20]:

Dep. Variable:	Profit	**R-squared:**	0.883
Model:	OLS	**Adj. R-squared:**	0.881
Method:	Least Squares	**F-statistic:**	452.0
Date:	Thu, 21 Dec 2023	**Prob (F-statistic):**	1.27e-29
Time:	18:01:02	**Log-Likelihood:**	-397.68
No. Observations:	62	**AIC:**	799.4
Df Residuals:	60	**BIC:**	803.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	-114.0625	32.627	-3.496	0.001	-179.327	-48.798
Sales	0.4229	0.020	21.260	0.000	0.383	0.463

Omnibus:	2.345	**Durbin-Watson:**	2.295
Prob(Omnibus):	0.310	**Jarque-Bera (JB):**	1.652
Skew:	-0.380	**Prob(JB):**	0.438
Kurtosis:	3.250	**Cond. No.**	2.81e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.81e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Perhatikan pada bagian kanan atas, Anda dapat melihat sebuah informasi bernama **R-squared**. R-squared didefinisikan sebagai persentase dari varians total yang dijelaskan oleh hubungan linear dengan prediktor (variansi regresi / variansi total):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

Dengan kata lain, R-squared dapat dianggap sebagai suatu nilai yang merepresentasikan total variansi data yang diwakili oleh model. Dalam kasus ini, nilai r-squared adalah persentase dari variasi Profit yang dijelaskan oleh hubungan regresi linear dengan Sales. Karena merupakan nilai persentase, artinya nilai R-squared berada di antara 0 dan 1.

Karena R-squared merupakan pengukuran statistik seberapa dekat data ke garis yang di-fit, kita ingin agar model kita memiliki nilai R-squared setinggi mungkin. Artinya model telah mempelajari data dengan baik (walaupun tidak selalu terjadi, tapi hal ini akan dibahas nanti). Kita juga dapat mengambil informasi R-squared dengan mengakses attribute `rsquared` pada model.

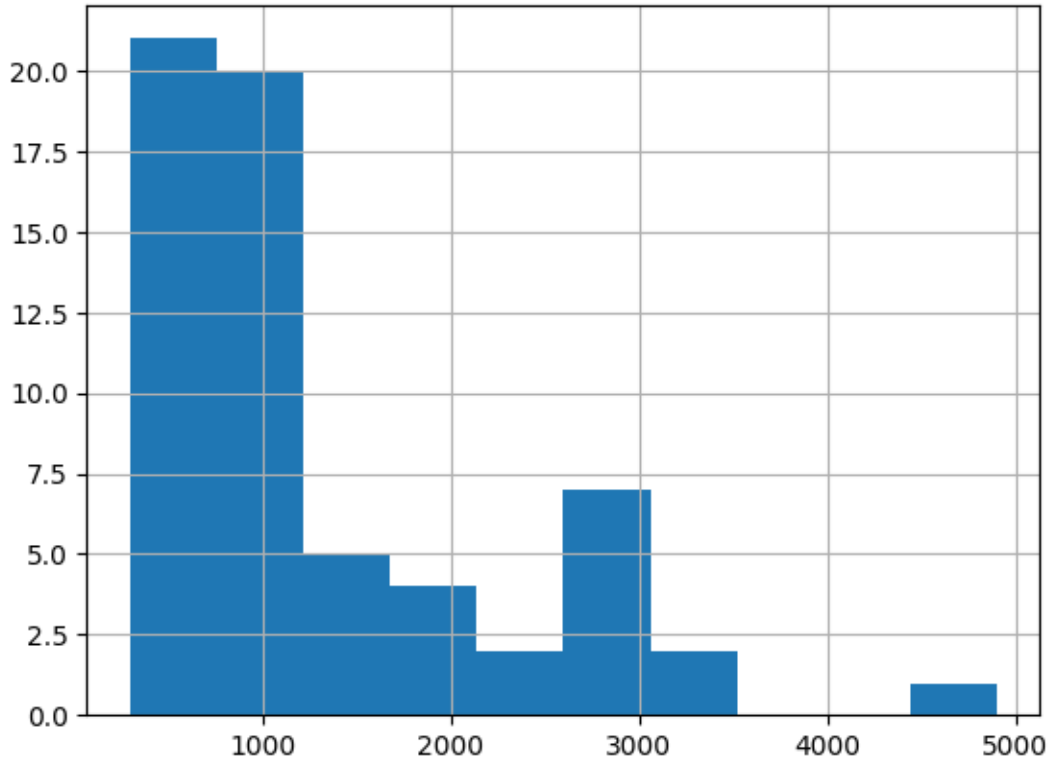
[21]:
```
lm_simple.rsquared
```

[21]: 0.8828077388598752

21.4 [Additional] Leverage vs. Influence

Pada kebanyakan kasus, kita akan berhadapan dengan kondisi data yang memiliki nilai ekstrim atau outliers. Pada data kita kali ini kita akan meninjau kolom `Sales`:

[22]:
```
copiers = pd.read_csv('data_input/copiers.csv')
copiers['Sales'].hist();
```



Sekarang, mari kita buat sebuah model regresi linear sederhana untuk memprediksi `Profit` berdasarkan jumlah `Sales` data kita.

[23]:
```
X = sm.add_constant(copiers['Sales'])
Y = copiers['Profit']

lm_outliers = sm.OLS(Y, X).fit()
```

Sebagai perbandingan, mari kita buat model tanpa outlier dengan mengambil data yang memiliki `Sales` di bawah 4000 (angka kasar yang diambil berdasarkan histogram sebelumnya).

[24]:
```
copiers_no = copiers[copiers["Sales"] < 4000]
```

```
X = sm.add_constant(copiers_no['Sales'])
Y = copiers_no['Profit']

lm_no = sm.OLS(Y, X).fit()
```

Agar terlihat lebih jelas, kita akan membuat sebuah plot berdasarkan hasil kedua model di atas, berikut kodenya:

[25]:
```
plt.scatter(copiers['Sales'], copiers['Profit'])
plt.plot(copiers['Sales'], lm_outliers.fittedvalues, c='purple')
plt.plot(copiers_no['Sales'], lm_no.fittedvalues, c='orange')

plt.xlabel('Sales')
plt.ylabel('Profit')
plt.title('Sales vs. Profit');
```

Perhatikan bahwa keberadaan outlier telah memengaruhi garis regresi kita, setidaknya terdapat pergeseran sudut antara garis berwarna ungu dengan garis berwarna jingga. Hal ini tentunya berdampak terhadap persamaan garis yang dibentuk dan kita bisa melihatnya dengan menampilkan kondisi konstanta kita serta R-squared kedua model untuk mengetahui kondisi mana yang lebih baik:

[26]:
```
pd.DataFrame(
    {"condition": ["outlier", "without outlier"],
     "const": [lm_outliers.params[0], lm_no.params[0]],
     "Sales slope": [lm_outliers.params[1], lm_no.params[1]],
     "r-squared": [lm_outliers.rsquared, lm_no.rsquared]}
).set_index("condition")
```

[26]:

condition	const	Sales slope	r-squared
outlier	-114.06251	0.42286	0.88281
without outlier	-83.54810	0.39444	0.85138

Outlier dikatakan memiliki ***high leverage*** jika nilai prediktornya (sumbu x) jauh dari rata-rata nilai prediktor tersebut. Dengan kata lain, jika sebuah nilai jauh dari kumpulan data lainnya, kita dapat menyebut prediktor tersebut memiliki *high leverage*. Keberadaan sebuah *high leverage* memiliki potensi untuk mempengaruhi koefisien model regresi kita, seperti yang ditunjukkan pada tabel di atas.

Katakanlah terdapat sebuah titik outlier yang sesuai dengan garis regresi yang dibentuk berdasarkan data lainnya. Pada kondisi ini dapat dikatakan outlier tersebut memiliki ***low influence***.

Namun jika terdapat titik outlier lain yang memiliki *leverage* yang sama (nilai x-nya sama jauhnya dari rata-rata pengamatan lainnya) dan menurunkan nilai R-squared secara signifikan serta merubah garis regresi dengan cukup jauh (ditunjukkan dengan perubahan nilai koefisien yang jauh), kita dapat menyebut kondisi ini sebagai ***high influence***.

Apabila outlier yang kita miliki memiliki *low influence* dan meningkatkan nilai R-squared maka sebaiknya outlier ini tetap digunakan dalam analisis kita. Sementara jika terdapat kondisi *high influence* maka sebaiknya kita tidak menggunakan outlier pada analisis regresi kita.

Berdasarkan kedua kondisi di atas, tugas kitalah untuk mengidentifikasi kapan model regresi kita dipengaruhi oleh outlier yang secara buruk maupun baik. Kondisi ini dapat mengubah kemiringan garis yang dibentuk dan membuat analisis kita menjadi tidak berlaku.

22 Multiple Linear Regression

Pada kasus sebelumnya, kita telah melakukan prediksi nilai y dengan input satu buah variabel x. Ingat kembali bahwa pada kasus tersebut y adalah `Profit` dan x adalah `Sales`. Model regresi ternyata tidak hanya bisa dibangun dari satu variabel prediktor saja, melainkan bisa memiliki banyak prediktor atau biasa disebut Multiple/Multivariate Linear Regression.

Kita akan menggunakan data lain yaitu `crime.csv` pada kasus ini. Data ini merupakan data yang digunakan kriminologis untuk mempelajari tingkat kriminalitas berdasarkan berbagai macam faktor. Kita akan membaca data dan mengganti nama-nama kolomnya agar lebih jelas.

[27]:
```
crime = pd.read_csv("data_input/crime.csv", index_col=0)

crime.columns = ['percent_m', 'is_south', 'mean_education',
                 'police_exp60', 'police_exp59', 'labour_participation',
```

```
                 'm_per1000f', 'state_pop', 'nonwhites_per1000', 'unemploy_m24',
                 'unemploy_m39', 'gdp', 'inequality', 'prob_prison',␣
↪'time_prison',
                 'crime_rate']

crime.head()
```

[27]:

```
   percent_m  is_south  mean_education  police_exp60  police_exp59  \
1        151         1              91            58            56
2        143         0             113           103            95
3        142         1              89            45            44
4        136         0             121           149           141
5        141         0             121           109           101

   labour_participation  m_per1000f  state_pop  nonwhites_per1000  \
1                   510         950         33                301
2                   583        1012         13                102
3                   533         969         18                219
4                   577         994        157                 80
5                   591         985         18                 30

   unemploy_m24  unemploy_m39  gdp  inequality  prob_prison  time_prison  \
1           108            41  394         261      0.08460     26.20110
2            96            36  557         194      0.02960     25.29990
3            94            33  318         250      0.08340     24.30060
4           102            39  673         167      0.01580     29.90120
5            91            20  578         174      0.04140     21.29980

   crime_rate
1         791
2        1635
3         578
4        1969
5        1234
```

Data ini dikumpulkan pada tahun 1960 dan deskripsi lengkapnya tidak benar-benar tersedia. Kita akan menggunakan deskripsi dari penulis library MASS. Variabel yang ada yaitu:

- `percent_m`: percentage of males aged 14-24
- `is_south`: whether it is in a Southern state. 1 for Yes, 0 for No.
- `mean_education`: mean years of schooling
- `police_exp60`: police expenditure in 1960
- `police_exp59`: police expenditure in 1959 - labour_participation: labour force participation rate
- `m_per1000f`: number of males per 1000 females
- `state_pop`: state population
- `nonwhites_per1000`: number of non-whites resident per 1000 people
- `unemploy_m24`: unemployment rate of urban males aged 14-24

- `unemploy_m39`: unemployment rate of urban males aged 35-39
- `gdp`: gross domestic product per head
- `inequality`: income inequality
- `prob_prison`: probability of imprisonment
- `time_prison`: avg time served in prisons
- `crime_rate`: crime rate in an unspecified category

Pada kasus regresi linear, salah satu cara penentuan variabel yang baik digunakan adalah dengan melihat nilai korelasi antara setiap prediktor dengan target. Mari kita tinjau kondisi korelasi pada data kita.

[28]:
```
crime.corr(numeric_only=True).sort_values(
    'crime_rate', ascending=False
).loc[:,'crime_rate']
```

[28]:
```
crime_rate              1.00000
police_exp60            0.68760
police_exp59            0.66671
gdp                     0.44132
state_pop               0.33747
mean_education          0.32283
m_per1000f              0.21391
labour_participation    0.18887
unemploy_m39            0.17732
time_prison             0.14987
nonwhites_per1000       0.03260
unemploy_m24           -0.05048
percent_m              -0.08947
is_south               -0.09064
inequality             -0.17902
prob_prison            -0.42742
Name: crime_rate, dtype: float64
```

Pada hasil pengecekan korelasi di atas, dapat kita lihat bahwa nilai korelasi antara police_exp59 dan 60 dengan crime_rate cukup tinggi. Lalu variabel lain yang bisa dikatakan cukup kuat adalah gdp dan prob_prison. Ingat bahwa nilai korelasi dikatakan kuat berdasarkan besarnya. Sementara, tanda negatif atau positif di awal nilai menunjukkan arah korelasinya.

Di luar variabel yang disebutkan di atas, mari kita coba gunakan beberapa variabel yang secara logika akan berdampak terhadap crime_rate seperti mean_education dan gdp.

[29]:
```
Y = crime['crime_rate']
X = sm.add_constant(crime[['mean_education', 'gdp']])

lm_logic = sm.OLS(Y, X).fit()
lm_logic.params
```

[29]:
```
const            -15.22189
mean_education    -0.14901
```

```
gdp                 1.78165
dtype: float64
```

Berdasarkan model multivariate di atas, persamaan regresi yang dibentuk dengan kondisi ketiga variabel berhubungan linear, estimasi penggunaan bandwidth dapat dituliskan sebagai:

`Profit = -15.22 - 0.149 * mean_education + 1.78 * gdp`

Jika kita memiliki informasi kondisi suatu lokasi dengan mean_education sebesar 100 dan gdp 600, maka estimasi crime_rate kita yaitu:

[30]:
```
-15.22 - 0.149 * 100 + 1.78 * 600
```

[30]:
```
1037.88
```

[31]:
```
lm_logic.predict([1, 100, 600])
```

[31]:
```
array([1038.86656806])
```

Berdasarkan informasi yang sudah dipelajari model, jika kita mengetahui kondisi suatu lokasi dengan kriteria di atas maka kita dapatkan estimasi crime_rate sebesar 1038.86.

22.1 Adjusted R-Squared

Penggunaan R-squared pada multivariate linear regression untuk melihat kesesuaian model dapat menyesatkan. Hal ini karena keterbatasan dari metrik R-squared yang nilainya akan selalu meningkat setiap bertambahnya jumlah prediktor. Apabila kita menambahkan variabel yang dengan nilai angka random, maka R-squared tidak akan berkurang. Akibatnya model dengan lebih banyak variabel prediktor mungkin terlihat lebih sesuai walaupun sebenarnya tidak demikian.

Kelemahan R-squared tersebut diatasi dengan menggunakan **Adjusted R-squared**. Adjusted R-squared memungkinkan kita untuk membandingkan model regresi data crime dengan 4 variabel dan 3 variabel untuk mencari tahu apakah model 4 variabel mencapai R-squared yang lebih tinggi hanya karena memiliki lebih banyak prediktor atau apakah memang hasilnya lebih sesuai.

Kita akan mencobanya dengan contoh berikut ini, yaitu dengan menggunakan tiga model yang prediktornya masing-masing berbeda jumlahnya. Nilai yang akan kita lihat nantinya adalah R-squared dan adjusted R-squared.

[32]:
```
Y = crime['inequality']

# model 1
X_model1 = sm.add_constant(crime['gdp'])
model1 = sm.OLS(Y, X_model1).fit()

# model 2
X_model2 = sm.add_constant(crime[['gdp','mean_education']])
model2 = sm.OLS(Y, X_model2).fit()

# model 3
```

```
X_model3 = sm.
 ↪add_constant(crime[['gdp','labour_participation','m_per1000f','time_prison']])
model3 = sm.OLS(Y, X_model3).fit()

# compare result
pd.DataFrame({
    'model': ['model1', 'model2', 'model3'],
    'r-squared': [model1.rsquared, model2.rsquared, model3.rsquared],
    'adj. r-squared': [model1.rsquared_adj, model2.rsquared_adj, model3.
 ↪rsquared_adj]
})
```

[32]:

	model	r-squared	adj. r-squared
0	model1	0.78145	0.77659
1	model2	0.81185	0.80330
2	model3	0.79388	0.77425

`model1` memiliki satu buah variabel prediktor (`gdp`) dan nilai R-squared sebesar 0.78145 (adjusted R-squared of 0.77659). Jika kita menambahkan tiga prediktor lain (`labour_participation`, `m_per1000f`, `time_prison`) yaitu pada `model3`, nilai R-squared model kita meningkat menjadi 0.79388. Katakanlah R-squared mengindikasikan kualitas kesesuaian model, apakah hal ini berarti model3 merupakan model yang lebih baik dari model1? Jawabannya adalah tidak, karena dengan menambahkan tiga buah variabel, nilai adjusted R-squared kita berkurang menjadi 0.77425.

Sebagai rangkuman, nilai R-squared menandakan seberapa baik model kita menggambarkan data asli. Nilai ini mengukur sejauh mana varian dalam variabel dependent kita (inequality) dapat dijelaskan oleh variabel independent (gdp, dll). Namun seiring dengan peningkatan jumlah variabel independent yang digunakan, nilai R-squared model kita juga akan meningkat karena gabungan informasi penting serta noise dari seluruh variabel.

Di sisi lain, adjusted R-squared nilainya tidak meningkat seperti R-squared karena adanya faktor penyesuaian berdasarkan jumlah variabel prediktor dalam model kita. Nilai ini hanya akan meningkat ketika variabel baru benar-benar menghasilkan prediksi yang lebih baik. Meskipun detail matematis dari rumus adjusted R-squared di luar ranah *course* ini, kita akan membahas bukti singkat bahwa nilai Adjusted R-squared dikenakan penalti berdasarkan jumlah prediktornya.

Berikut ini adalah formula dari perhitungan adjusted R-squared:

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

Pada formula tersebut, diketahui bahwa n adalah jumlah observasi dan p adalah jumlah prediktor. Perhatikan bahwa jika nilai p meningkat maka nilai di bagian pecahan akan menjadi lebih besar dan menurunkan nilai keseluruhan dari adjusted R-squared.

Jika penjelasan di atas kurang jelas, kami telah menyediakan kode yang dapat diubah-ubah nilai jumlah prediktornya.

[33]:
```
# jumlah prediktor
no_of_preds = 4

# perhitungan adjusted r-squared
adjusted_model3 = 1 - (1-model3.rsquared) * (crime.shape[0]-1) / (crime.
 ↪shape[0]-1-no_of_preds)

adjusted_model3
```

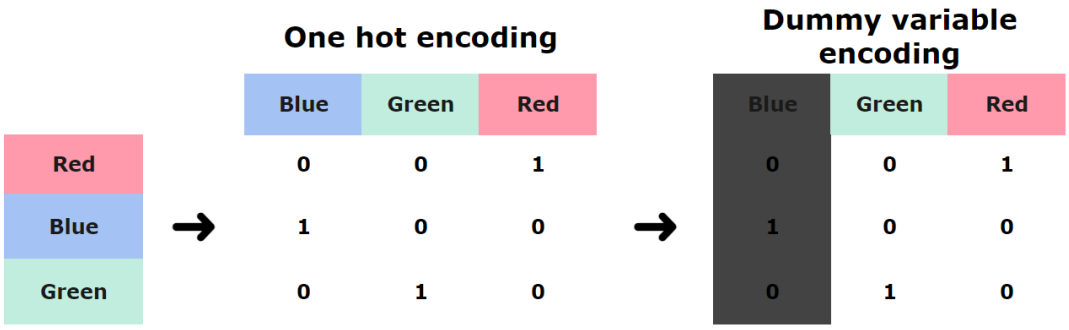
[33]: 0.7742470048366594

23 Encoding

Sampai tahap ini, kita hanya bisa menggunakan variabel bertipe numerik sebagai prediktor pada model kita. Hal ini dikarenakan Python hanya mampu memproses informasi numerik sementara variabel bertipe kategorikal atau *object* harus melalui tahap *pre-processing* yang disebut dengan ***Encoding***.

Terdapat beberapa jenis *encoding* yang dapat digunakan, tetapi pada *course* ini kita hanya akan mempelajari satu jenis yaitu ***dummy variable encoding***. *Dummy variable encoding* adalah metode lanjutan dari jenis encoding lain bernama *one-hot encoding* yang melakukan transformasi pada variabel kategorikal kita dan mengubahnya menjadi beberapa kolom sesuai jumlah kategori kita dengan informasi biner.

Karena informasi dari *one-hot encoding* ini bersifat redundan (berulang), maka salah satu kolomnya akan dibuang. Inilah yang disebut sebagai ***dummy variable encoding***. Berikut adalah penggambaran dari penjelasan di atas.



Untuk melakukan ***dummy variable encoding*** kita menggunakan fungsi `.get_dummies()` dari library pandas. Kali ini kita akan kembali menggunakan data copiers kita dengan membuang beberapa variabel yang tidak terlalu penting untuk saat ini.

[35]:
```
copiers_clean = copiers.drop(columns=["Order.ID", "Order.Date", "Ship.Date",
                              "Customer.ID", "Product.ID", "Category",
                              "Sub.Category", "Product.Name"])
cat_var = copiers_clean.select_dtypes(exclude='number').columns.to_list()

copiers_enc = pd.get_dummies(copiers_clean,
```

```
                                columns = cat_var,
                                drop_first = True,
                                dtype='int64')

copiers_enc.head()
```

[35]:

```
       Sales  Quantity  Discount     Profit  Ship.Mode_Same Day  \
0  959.98400         2   0.20000  335.99440                   0
1 1799.97000         3   0.00000  701.98830                   0
2 1199.97600         3   0.20000  434.99130                   0
3 2999.95000         5   0.00000 1379.97700                   0
4 1199.97600         3   0.20000  434.99130                   0

   Ship.Mode_Second Class  Ship.Mode_Standard Class  Segment_Corporate  \
0                       1                         0                  0
1                       1                         0                  0
2                       0                         1                  0
3                       0                         1                  0
4                       0                         1                  0

   Segment_Home Office
0                    0
1                    0
2                    0
3                    1
4                    0
```

Setelah melalui tahap *encoding* maka data kita sudah dapat digunakan untuk pembuatan model regresi. Pada kasus selanjutnya kita akan membuat model regresi menggunakan seluruh prediktor yang dimiliki oleh data kita.

[36]:

```
Y = copiers_enc['Profit']
X_enc = sm.add_constant(copiers_enc.drop(columns=['Profit']))

lm_all = sm.OLS(Y, X_enc).fit()

lm_all.summary()
```

[36]:

Dep. Variable:	Profit	**R-squared:**	0.952
Model:	OLS	**Adj. R-squared:**	0.944
Method:	Least Squares	**F-statistic:**	130.1
Date:	Thu, 21 Dec 2023	**Prob (F-statistic):**	4.85e-32
Time:	18:01:58	**Log-Likelihood:**	-370.30
No. Observations:	62	**AIC:**	758.6
Df Residuals:	53	**BIC:**	777.7
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P> |t|	[0.025	0.975]
const	63.4520	55.064	1.152	0.254	-46.992	173.896
Sales	0.4212	0.025	16.701	0.000	0.371	0.472
Quantity	-13.4098	13.545	-0.990	0.327	-40.578	13.758
Discount	-874.8310	117.023	-7.476	0.000	-1109.550	-640.112
Ship.Mode_Same Day	9.1379	57.485	0.159	0.874	-106.163	124.439
Ship.Mode_Second Class	42.9716	43.926	0.978	0.332	-45.132	131.075
Ship.Mode_Standard Class	13.9212	38.335	0.363	0.718	-62.969	90.812
Segment_Corporate	9.3012	31.294	0.297	0.767	-53.467	72.070
Segment_Home Office	-30.7099	39.860	-0.770	0.444	-110.660	49.240

Omnibus:	1.946	**Durbin-Watson:**	2.172
Prob(Omnibus):	0.378	**Jarque-Bera (JB):**	1.192
Skew:	-0.271	**Prob(JB):**	0.551
Kurtosis:	3.410	**Cond. No.**	1.55e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.55e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Ingat kembali bahwa kita dapat melakukan interpretasi pada nilai **coef** atau koefisien. Untuk variabel dengan tipe awal numerik, kita dapat menyimpulkan suatu koefisien sebagai berikut:

- Profit diekspektasikan **meningkat** sebesar 0.4212 USD setiap kenaikan `Sales`sebesar 1 satuan.
- Profit diekspektasikan **menurun** sebesar 874.8310 USD ketika `Discount` yang diberikan naik 1 tahun.

Sementara untuk variabel dengan tipe awal kategorik, kita akan melakukan interpretasi yang berbeda dan ini berhubungan dengan *dummy variable encoding* yang telah kita lakukan. Ingat kembali bahwa pada *dummy variable encoding* kita membuang satu buah kategori untuk setiap kolomnya. Kolom yang terbuang ini secara default merupakan kategori dengan huruf alfabet yang paling awal muncul. Contohnya jika terdapat satu variabel dengan kategori High, Normal, dan Low maka kategori yang akan dihilangkan adalah High karena huruf H muncul lebih dahulu dibanding N dan L.

Sebagai contoh kasus di data kita, variabel `Gender` memiliki tiga kategori yaitu Female, Male, dan Nonbinary. Kategori yang hilang pada variabel ini adalah Female tetapi informasinya sebenarnya ditangkap oleh koefisien **const**. Oleh karena itu, cara kita menginterpretasikan suatu variabel kategorik adalah dengan membandingkannya dengan kategori yang ditangkap oleh koefisien **const**:

- Profit diekspektasikan **meningkat** sebesar 9.3 USD ketika Segment bertipe `Home Office` dibandingkan dengan Segment bertipe `Consumer` dengan catatan variabel lain bernilai tetap.
- Profit diekspektasikan **menurun** sebesar 30.71 USD ketika Segment bertipe `Corporate` dibandingkan dengan Segment bertipe `Consumer` dengan catatan variabel lain bernilai tetap.
- Profit diekspektasikan **meningkat** sebesar 42.97 USD ketika Ship.Mode bertipe `Second Class` dibandingkan dengan Ship.Mode bertipe `First Class` dengan catatan variabel lain bernilai tetap.

24 Regression Model Evaluation

Pada beberapa kasus kita akan memilih beberapa model dengan skema yang berbeda, perbedaan yang normal terjadi adalah penggunaan jumlah variabel yang berbeda. Penting bagi kita untuk melakukan evaluasi berbagai performa model tersebut untuk menentukan model dengan kesalahan (*error*) terkecil. Berikut ini beberapa metrik untuk kita menghitung kesalahan/*error*:

- ***Mean Absolute Error*** (MAE): Memperlakukan error dengan lebih ringan.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

- ***Root Mean Squared Error*** (RMSE): Memperlakukan error dengan lebih sensitif. Ketika nilai error besar, maka nilai RMSE akan semakin besar dan sebaliknya.

$$RMSE = \sqrt{\frac{\sum_i^n (Predicted_i - Actual_i)^2}{n}}$$

- ***Mean Absolute Percentage Error*** (MAPE): Menunjukan seberapa besar penyimpangan error dalam bentuk persentase.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}|}{y}$$

Note: RMSE digunakan ketika model yang dibuat memuat observasi outlier. Sedangkan, MAE digunakan ketika model yang dibuat tidak memuat observasi outlier. MAPE adalah metrik yang baik untuk interpretasi karena mudah dipahami. Ketiganya bisa diinterpretasikan dan metrik paling aman digunakan jika kita mengerti formula perhitungannya adalah RMSE.

Pada kasus ini, kita akan meninjau model kita berdasarkan nilai RMSE-nya. Library `statsmodels` memiliki fungsi `rmse` yang dapat digunakan untuk menghitung nilai RMSE:

[39]:
```
from statsmodels.tools.eval_measures import rmse

# predict data with different models
y_simple = lm_simple.predict(X_simple)
y_all = lm_all.predict(X_enc)

# compare the rmse
pd.DataFrame({'RMSE': [rmse(Y, y_simple), rmse(Y, y_all)]},
             index= ['Model 1', 'Model 2'])
```

[39]:

	RMSE
Model 1	147.69971
Model 2	94.97767

25 Limitations of Regression Models

Walaupun model regresi linear merupakan metode yang sangat *powerful* untuk menyelesaikan berbagai masalah data science, metode ini masih memiliki batasan secara statistik.

- Regresi linear baik untuk data yang memiliki hubungan linear antara prediktor dengan target.
- Model regresi simple/multiple mungkin sensitif terhadap data outliers.
- Model regresi simple/multiple mengasumsikan bahwa variabel prediktor tidak berkorelasi tinggi satu sama lain

Oleh karena batasan tersebut, ketika membuat prediksi variabel target menggunakan regresi linear **kita harus memenuhi beberapa asumsi sebagai syarat agar model regresi yang kita tepat secara statistik**. Beberapa asumsi yang harus dipenuhi dalam model regresi linear adalah:

1. *Linear prediction variables for target variables*
2. *Normality of residuals*
3. *Residuals have no heteroscedasticity*
4. *Little to no multicollinearity*

25.1 Linearity

Dilakukan sebelum membuat model. Untuk menguji apakah variabel target dan prediktor memiliki hubungan linear. Dapat dilihat dengan nilai **korelasi**.

Linearity artinya target variabel dengan prediktornya memiliki hubungan yang linear atau hubungannya bersifat garis lurus. Selain itu, efek atau nilai koefisien antar variabel bersifat additive. Jika *linearity* ini tidak terpenuhi, maka otomatis semua nilai koefisien yang kita dapatkan tidak valid karena model berasumsi bahwa pola yang akan kita buat adalah linear.

25.2 Normality of Residuals

Ingat kembali bahwa kita akan selalu melihat perbedaan antara data target asli dengan hasil prediksi/*fitted values* kita. Perbedaan ini sudah dibahas di atas dengan sebutan *error* atau nama lainnya adalah residual. Asumsi *normality* menyebutkan bahwa residual dari model regresi linear sebaiknya berdistribusi normal. Artinya residual banyak berkumpul di sekitar angka 0. Untuk mengecek apakah residual model kita berdistribusi normal terdapat beberapa uji statistik, salah satunya adalah Shapiro-Wilk test. Pengujian statistik ini memiliki uji hipotesis sebagai berikut:

- H_0: Residual berdistribusi normal
- H_1: Residual tidak berdistribusi normal

Dalam melakukan pengujiannya kita akan dibantu library `scipy` dan memanfaatkan fungsi `shapiro()`. Untuk memanfaatkan fungsi tersebut, kita akan mengeluarkan nilai residu dari model yang sudah dibuat dengan menambahkan `.resid` pada objek model yang dibuat.

[40]:
```
from scipy.stats import shapiro

name = {0: 'Statistics',1: 'p-value'}
pd.DataFrame({shapiro(lm_all.resid)}).rename(columns=name)
```

[40]:
```
   statistic  pvalue
0    0.98628 0.71771
```

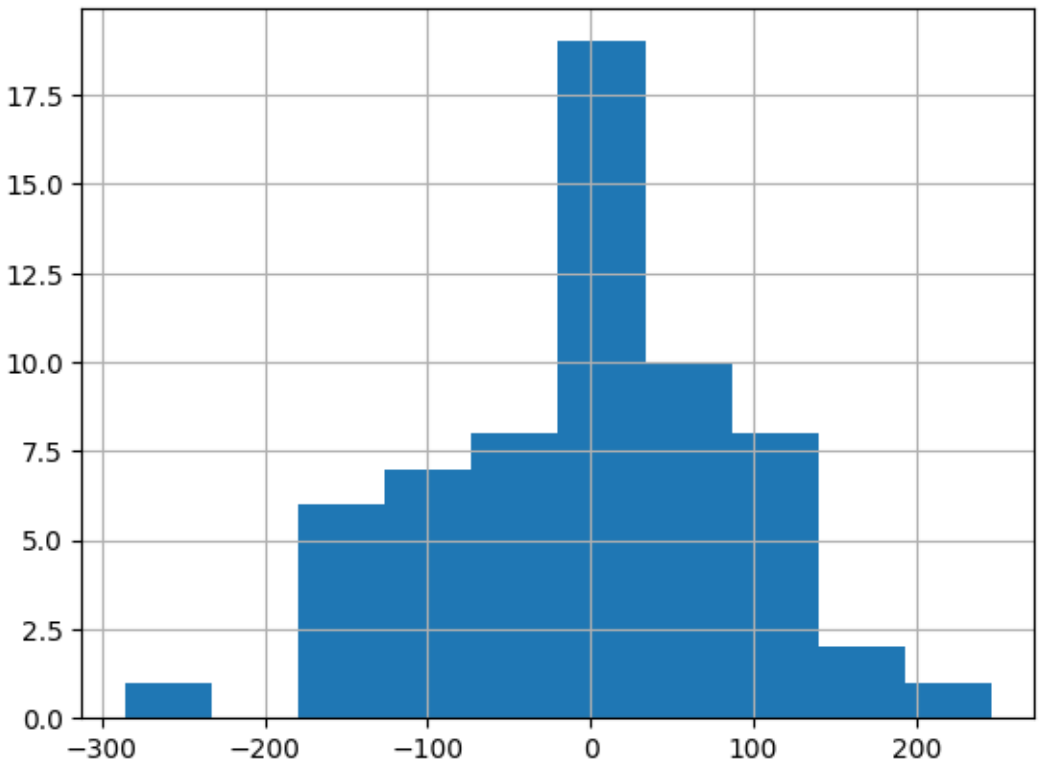
Pada uji statistik ini nilai yang perlu kita perhatikan adalah p-value sehingga dan nilai yang kita harapkan adalah p-value lebih besar dari alpha (0.05). Ketika kita mendapatkan nilai p-value > 0.05 maka uji hipotesis kita akan gagal menolak H0.

Berdasarkan hasil dari Shapiro-Wilk test kita didapatkan nilai p-value > 0.05 artinya kita gagal tolak H0 yang menandakan residual model kita berdistribusi normal.

Tetapi, terdapat sebuah catatan dari dokumentasi official fungsi ini yaitu jika data kita berjumlah lebih dari 5000 nilai statistik yang dihasilkan masih akurat tetapi tidak dengan nilai p-valuenya.

Oleh karena itu, kita harus sadar akan keterbatasan metode ini dan menggunakan metode tambahan untuk memverifikasi hasil kita, seperti memeriksa distribusi data secara visual.

[41]:
```
lm_all.resid.hist();
```



25.2.1 [Optional] Interpret the Residual Normality Indicators from `sm` results

Selain menggunakan Shapiro-Wilk test kita dapat juga melihat summary model untuk melihat normalitas dari residual data kita, yaitu dengan meninjau:

- **Omnibus/Prob(Omnibus)** - Tes skewness dan kurtosis dari residual. Nilai yang mendekati 0 menunjukkan normalitas. Prob (Omnibus) melakukan uji statistik pada probabilitas resid-

ual yang terdistribusi normal. Nilai yang mendekati 1 menunjukkan probabilitas bahwa residual berdistribusi normal.

- **Skew** – Mengukur kesimetrian data. Nilai yang mendekati 0 menunjukkan residual berdistribusi normal.
- **Jarque-Bera/Prob(JB)** – Nilar 6 atau lebii menunjukkan bahwaresidualn tidakbterdistribusi normal dan hipotesis nol normalitas telah ditolak pada tingkat signifikansi 5 (alpha 0.05)%. Nilai yang mendekati 0 menunjukkan bahwa data terdistribusi normal.

[42]:
```
lm_all.summary().tables[2]
```

[42]:

Omnibus:	1.946	**Durbin-Watson:**	2.172
Prob(Omnibus):	0.378	**Jarque-Bera (JB):**	1.192
Skew:	-0.271	**Prob(JB):**	0.551
Kurtosis:	3.410	**Cond. No.**	1.55e+04

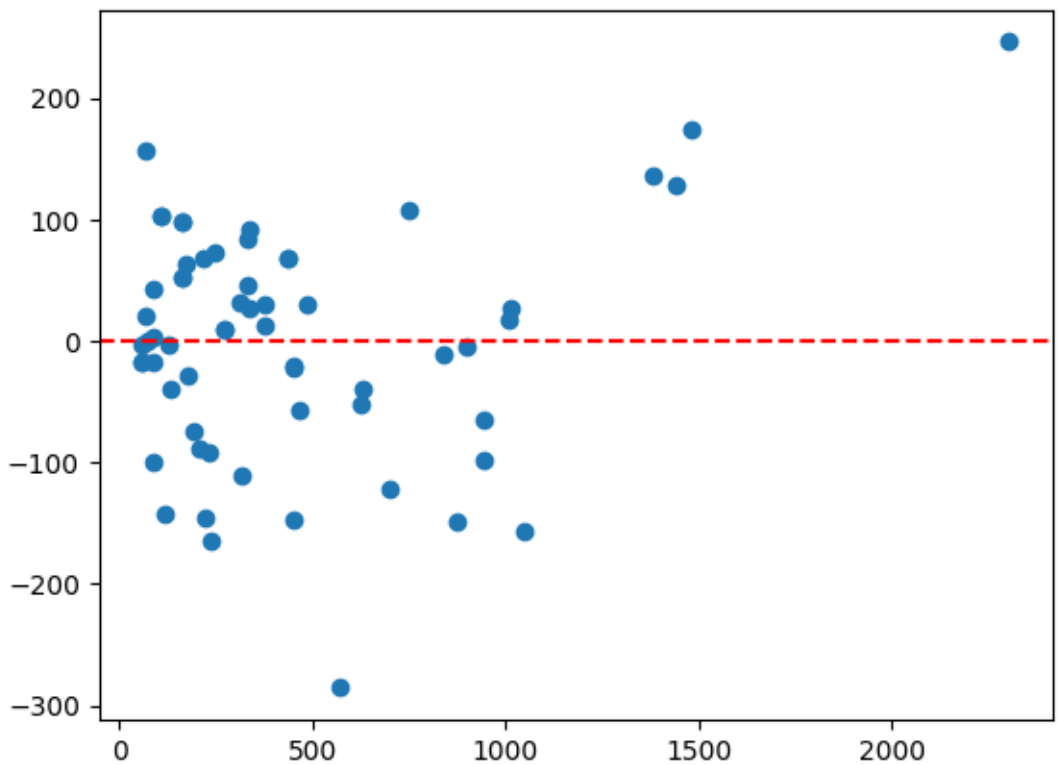
25.3 No-Heteroscedasticity

Selain harus berdistribusi normal, residual kita juga harus bersifat homogen yang artinya tersebar tanpa pola. Jika data residual kita tidak membentuk pola, artinya model regresi linear kita sudah cukup menjelaskan data kita.

Plot residual adalah sebuah plot yang menampilkan nilai residual pada sumbu y dan nilai *fitted values* pada sumbu x. Pada plot tersebut kita dapat melihat apakah terdapat pola yang mengindikasikan pelanggaran pada asumsi model kita dan berpengaruh pada nilai standar error pada estimasi/koefisien prediktor.

Plot yang diharapkan adalah plot yang memiliki titik-titik data yang menyebar secara acak di sekitar sumbu x. Hal ini tidak berarti model kita sempurna, tetapi model regresi yang kita pakai secara tepat menggambarkan variasi dalam variabel prediktor kita. Apabila nantinya terdapat pola tertentu maka model ini sebaiknya diperbaiki atau bisa dikatakan tidak memenuhi hubungan linear prediktor dengan target.

[43]:
```
plt.scatter(x = copiers['Profit'], y = lm_all.resid);
plt.axhline(y=0, color='r', linestyle='--');
```



Karena jika melihat visualnya saja terkadang terdapat subjektivitas, kita juga bisa melakukan pengecekan menggunakan uji Breuch-Pagan. Pada uji Breuch-Pagan ini kita memiliki uji hipotesis berikut:

- H_0: residual homogen(tidak membentuk sebuah pola/acak)
- H_1: residual heteros (membentuk sebuah pola)

Kita bisa menggunakan method `het_breuschpagan()` dari library `statsmodels`.

[44]:
```
import statsmodels.stats.api as sms

name = {0: 'Lagrange multiplier statistic',
        1: 'p-value',
        2: 'f-value',
        3: 'f p-value'}

pd.DataFrame({
    sms.het_breuschpagan(lm_all.resid, lm_all.model.exog)
}).rename(columns= name)
```

[44]:

	Lagrange multiplier statistic	p-value	f-value	f p-value
0	21.53229	0.00586	3.52507	0.00245

Berdasarkan hasil dari uji Breuch-Pagan kita mendapatkan nilai p-value < 0.05 artinya kita tolak

H0 yang menandakan residual model tidak berpola acak atau disebut sebagai *heteroscedastisity*.

25.4 No Multicolinearity

Dalam formula regresi linear, kita mengasumsikan bahwa tidak ada korelasi antara variabe prediktor agar dapat memiliki efek independen dari setiap variabel. Salah satu metode statistik yang dapat digunakan saat menilai *multicollinearity* adalah Variance Inflation Factor (VIF). Secara sederhana, VIF adalah cara untuk mengukur efek *multicollinearity* di antara prediktor dalam model kita.

[45]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor

x = copiers.select_dtypes(include="number").drop(columns=["Profit"])
xc = sm.add_constant(x)

vif = [variance_inflation_factor(xc.values, i) for i in range(len(xc.columns))]
pd.Series(data=vif, index=xc.columns).sort_values(ascending=False)
```

[45]:
```
const      6.36132
Sales      3.11759
Quantity   3.00142
Discount   1.13280
dtype: float64
```

Keterbatasan dari berbagai model *machine learning* adalah hal yang akan kita tinjau lagi seiring perjalanan pembelajaran kita. Kita tidak hanya akan belajar bagaimana mengidentifikasinya tetapi kita juga akan mempelajari berbagai teknik untuk mengatasinya, mencegah *overfitting/underfitting*, dan membuat diagnosis model menjadi bagian penting dari pembelajaran metode *machine learning* ini.

Coursebook: Classification in Machine Learning

- Bagian 5 Audit Analytics untuk Bank Rakyat Indonesia
- Durasi: 7 Jam
- *Last Updated*: December 2023

- *Coursebook* ini disusun dan dikurasi oleh tim produk dan instruktur dari Algoritma Data Science School

26 Background

Coursebook ini merupakan bagian dari **BRI Audit Analytics** yang disiapkan oleh Algoritma. *Coursebook* ini ditujukan hanya untuk khalayak terbatas, yaitu individu dan organisasi yang menerima *coursebook* ini langsung dari organisasi pelatihan. Tidak boleh direproduksi, didistribusikan, diterjemahkan, atau diadaptasi dalam bentuk apapun di luar individu dan organisasi ini tanpa izin. Algoritma adalah pusat pendidikan *data science* yang berbasis di Jakarta. Kami menyelenggarakan *workshop* dan program pelatihan untuk membantu para profesional dan mahasiswa dalam menguasai berbagai sub-bidang *data science* yaitu: *data visualization, machine learning,* statistik, dan lain sebagainya.

27 Classification in Machine Learning

27.1 Training Objectives

Pada *workshop* ini, Anda akan mempelajari salah satu *case* yang cukup umum diselesaikan dengan *machine learning*, yaitu klasifikasi. Klasifikasi merupakan aplikasi dari *supervised learning* (cabang dari *machine learning*), dengan model diharapkan dapat memberikan *output* berdasarkan kategori yang sudah ditentukan. Misalnya:

- Mengidentifikasi apakah suatu *email* merupakan *spam* atau tidak.
- Mengidentifikasi apakah pelanggan akan membeli suatu produk atau tidak.
- Mengidentifikasi objek buah-buahan pada suatu gambar, apakah objek tersebut merupakan apel, pisang, atau mangga.

Anda akan mempelajari siklus proses dalam membangun model *machine learning* untuk klasifikasi dan konsep matematika di belakangnya. *Workshop* ini akan berfokus pada 3 model *machine learning*: *logistic regression*, *decision tree*, dan *random torest*. Dengan durasi selama 7 jam, pembelajaran pada *workshop* ini terbagi dalam beberapa modul:

- ***Data Preprocessing***
  - *Target and Predictors Splitting*
  - *Dummy Variables for Categorical Predictor*
  - *Cross Validation*
- ***Classification Concepts for Logistic Regression***
  - *Probability concept*
  - *Understanding log of odds*
  - *Understanding logit function*
- ***Logistic Regression Implementation***

    - *Logistic regression with discrete predictor variables*
    - *Logistic regression with one continuous predictor variables*
    - *Logistic regression with multiple predictors variable*
    - *Assumption of logistic regression*
- ***Performance Evaluation and Model Selection***
    - *Confusion Matrix*
        * *Accuracy*
        * *Sensitivity*
        * *Recall*
        * *Specificity*
- ***K - Nearest Neighbors***
    - *Euclidean distance*
    - *Choosing an appropriate K*
    - *Features rescaling*
        * *Min-max normalization*
        * *Z-score standardization*
    - *Characteristics of K-NN*

[1]:
```
# Package
'''
Package dapat dibayangkan sebagai sekumpulan program yang telah ditulis␣
↪seseorang sehingga
dapat digunakan oleh orang lain yang ingin menyelesaikan permasalahan yang␣
↪serupa.
'''

import warnings

# package untuk perhitungan matematika
import math
import numpy as np

# package untuk persiapan data
import pandas as pd

# package untuk visualisasi data
import matplotlib.pyplot as plt
import seaborn as sns

# package untuk keperluan modeling
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
```

[2]:
```
# pandas display setup
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)

# suppress scientific notation
np.set_printoptions(suppress=True) # numpy output
pd.set_option('display.float_format', lambda x: '%.5f' % x) # pandas output

# suppress warning
warnings.filterwarnings("ignore")
%matplotlib inline
```

28 Data Preparation and Exploratory Data

Di dunia nyata, sering kali terdapat permasalahan pada data mentah yang dapat mengganggu proses analisis, seperti kesalahan *input*, nilai yang hilang, atau format penulisan yang tidak konsisten. Untuk mengatasi permasalahan tersebut, dilakukan preparasi data.

> Preparasi data merupakan serangkaian proses mengubah data mentah menjadi format yang lebih siap dan sesuai untuk analisis lebih lanjut. Preparasi data merupakan proses yang penting dan harus dilakukan sebelum analisis dan pembuatan model *machine learning*.

Berikut beberapa *step* yang sering dilakukan pada tahap preparasi data:

1. Cek tipe data setiap kolom.
2. Cek data yang hilang/*missing values*.
3. *Feature selection*.

28.1 Read The Dataset

Pada *step* ini, kita akan memuat data mentah yang akan digunakan sepanjang proses analisis dan pembuatan model *machine learning*. *Package* `pandas` menyediakan banyak fungsi yang dapat kita manfaatkan untuk membaca data, tergantung format data yang kita miliki.

- Format `.xlsx` menggunakan `pd.read_excel()`.
- Format `.csv` menggunakan `pd.read_csv()`.
- Format data lainnya yang dapat dilihat di sini.

> ***Note***: Selama proses persiapan dan pemrosesan data, kita akan banyak menggunakan fungsi yang terdapat pada *package* `pandas`.

Kita akan menggunakan *dataset* `credit100.csv`. Data ini merupakan data akun bank dari research paper metode evaluasi machine learning berikut ini. Secara detailnya data ini bisa dilihat lebih lanjut pada link GitHub ini.

Karena *dataset* ini memiliki format `.csv`, kita akan menggunakan fungsi `pd.read_csv()`.

[3]:
```
credit = pd.read_csv("data_input/credit100.csv")
credit = credit.drop(columns=["device_fraud_count"])

# menampilkan 5 baris teratas
credit.head()
```

[3]:
```
   fraud_bool  income  name_email_similarity  prev_address_months_count
current_address_months_count  customer_age  days_since_request
intended_balcon_amount payment_type  zip_count_4w  velocity_6h  velocity_24h
velocity_4w  bank_branch_count_8w  date_of_birth_distinct_emails_4w
employment_status  credit_risk_score  email_is_free housing_status
phone_home_valid  phone_mobile_valid  bank_months_count  has_other_cards
proposed_credit_limit  foreign_request    source  session_length_in_minutes
device_os  keep_alive_session  device_distinct_emails_8w  month
0           0 0.90000                0.61579                         -1
51            40             0.01034                -0.69955           AB
2659  11007.70883    7448.79742   5576.30356                  2030
7                CA                253              1             BA
1                   1                  9                1             1500.00000
0  INTERNET                   13.07981     linux                   0
1      1
1           0 0.60000                0.07246                         -1
125            20             0.00697                -0.95947           AC
5599   6304.26284    4373.09359   4721.10002                     1
13                CA                112              1             BE
1                   0                 -1                0             200.00000
0  INTERNET                    6.66655   windows                   0
1      4
2           1 0.50000                0.07659                         -1
37            50             0.00632                -1.21993           AB
1352   1357.41355    6667.02966   6318.87626                    15
6                CF                199              1             BC
1                   1                 28                0             1500.00000
0  INTERNET                    4.49267   windows                   1
2      0
3           0 0.30000                0.25543                         23
19            20             0.00927                 3.24224           AA
3743   5234.26230    6139.48958   4201.72592                     1
14                CA                 63              1             BC
0                   1                 16                0             200.00000
0  INTERNET                    2.32189     other                   1
1      3
4           0 0.90000                0.19236                         -1
34            30             0.01924                -1.13156           AB
546   3589.15345    4574.39557   4318.25891                    16
5                CA                275              1             BC
1                   1                  1                1             1000.00000
```

```
0  INTERNET                    4.84515     linux                    1
1      5
```

Penjelasan Dataset

Berikut adalah penjelasan setiap kolom yang terdapat pada *dataset*:

- `income` (numeric): *Annual income of the applicant (in decile form). Ranges between [0.1, 0.9].*
- `name_email_similarity` (numeric): *Metric of similarity between email and applicant's name. Higher values represent higher similarity. Ranges between [0, 1].*
- `prev_address_months_count` (numeric): *Number of months in previous registered address of the applicant, i.e. the applicant's previous residence, if applicable. Ranges between [−1, 380] months (-1 is a missing value).*
- `current_address_months_count` (numeric): *Months in currently registered address of the applicant. Ranges between [−1, 429] months (-1 is a missing value).*
- `customer_age` (numeric): *Applicant's age in years, rounded to the decade. Ranges between [10, 90] years.*
- `days_since_request` (numeric): *Number of days passed since application was done. Ranges between [0, 79] days.*
- `intended_balcon_amount` (numeric): *Initial transferred amount for application. Ranges between [−16, 114] (negatives are missing values).*
- `payment_type` (categorical): *Credit payment plan type. 5 possible (annonymized) values.*
- `zip_count_4w` (numeric): *Number of applications within same zip code in last 4 weeks. Ranges between [1, 6830].*
- `velocity_6h` (numeric): *Velocity of total applications made in last 6 hours i.e., average number of applications per hour in the last 6 hours. Ranges between [−175, 16818].*
- `velocity_24h` (numeric): *Velocity of total applications made in last 24 hours i.e., average number of applications per hour in the last 24 hours. Ranges between [1297, 9586]*
- `velocity_4w` (numeric): *Velocity of total applications made in last 4 weeks, i.e., average number of applications per hour in the last 4 weeks. Ranges between [2825, 7020].*
- `bank_branch_count_8w` (numeric): *Number of total applications in the selected bank branch in last 8 weeks. Ranges between [0, 2404].*
- `date_of_birth_distinct_emails_4w` (numeric): *Number of emails for applicants with same date of birth in last 4 weeks. Ranges between [0, 39].*
- `employment_status` (categorical): *Employment status of the applicant. 7 possible (annonymized) values.*
- `credit_risk_score` (numeric): *Internal score of application risk. Ranges between [−191, 389].*
- `email_is_free` (binary): *Domain of application email (either free or paid).*
- `housing_status` (categorical): *Current residential status for applicant. 7 possible (annonymized) values.*
- `phone_home_valid` (binary): *Validity of provided home phone.*
- `phone_mobile_valid` (binary): *Validity of provided mobile phone.*
- `bank_months_count` (numeric): *How old is previous account (if held) in months. Ranges between [−1, 32] months (-1 is a missing value).*
- `has_other_cards` (binary): *If applicant has other cards from the same banking company.*
- `proposed_credit_limit` (numeric): *Applicant's proposed credit limit. Ranges between [200, 2000].*

- `foreign_request` (binary): *If origin country of request is different from bank's country.*
- `source` (categorical): *Online source of application. Either browser (INTERNET) or app (TELEAPP).*
- `session_length_in_minutes` (numeric): *Length of user session in banking website in minutes. Ranges between [−1, 107] minutes (-1 is a missing value).*
- `device_os` (categorical): *Operative system of device that made request. Possible values are: Windows, macOS, Linux, X11, or other.*
- `keep_alive_session` (binary): *User option on session logout.*
- `device_distinct_emails` (numeric): *Number of distinct emails in banking website from the used device in last 8 weeks. Ranges between [−1, 2] emails (-1 is a missing value).*
- `device_fraud_count` (numeric): *Number of fraudulent applications with used device. Ranges between [0, 1].*
- `month` (numeric): *Month where the application was made. Ranges between [0, 7].*
- `fraud_bool` (binary): *If the application is fraudulent or not.*

28.2 Descriptive Statistics

Pemeriksaan statistik deskriptif merupakan analisis awal yang dilakukan untuk mendapatkan pemahaman tentang sifat dasar dari *dataset*. Seperti namanya, statistik deskriptif mencakup penghitungan ringkasan statistik yang menggambarkan ciri-ciri kumpulan data tersebut, seperti: ukuran pemusatan data (*mean* dan median) dan ukuran penyebaran data (standar deviasi).

Meskipun perhitungan yang ditampilkan cukup sederhana, statistik deskriptif bisa menjadi alternatif *tools* yang cukup *powerful*. Sebagai contoh, kita dapat mengetahui kecenderungan persebaran data dan melakukan deteksi awal apakah terdapat nilai pencilan (*outlier*) yang berbeda jauh dibandingkan nilai-nilai lainnya.

Untuk menampilkan statistik deskriptif pada data `credit`, kita dapat menggunakan fungsi `.describe()`.

[4]:
```
credit.describe()
```

[4]:
```
        fraud_bool       income  name_email_similarity
prev_address_months_count  current_address_months_count  customer_age
days_since_request  intended_balcon_amount  zip_count_4w  velocity_6h
velocity_24h  velocity_4w  bank_branch_count_8w
date_of_birth_distinct_emails_4w  credit_risk_score  email_is_free
phone_home_valid  phone_mobile_valid  bank_months_count  has_other_cards
proposed_credit_limit  foreign_request  session_length_in_minutes
keep_alive_session  device_distinct_emails_8w        month
count 100000.00000 100000.00000           100000.00000
100000.00000                   100000.00000  100000.00000        100000.00000
100000.00000  100000.00000 100000.00000  100000.00000 100000.00000
100000.00000                       100000.00000       100000.00000   100000.00000
100000.00000        100000.00000       100000.00000     100000.00000
100000.00000     100000.00000               100000.00000        100000.00000
100000.00000 100000.00000
mean       0.11029      0.57243                0.48334
15.67167                       89.09414      34.34520             1.02909
```

8.05770 1579.65136 5622.74948 4751.70038 4846.51788
178.94636 9.31030 135.47810 0.54335
0.39658 0.88576 10.76559 0.20875
545.64380 0.02764 7.64760 0.55407
1.02486 3.30663
std 0.31325 0.29097 0.29139
43.04344 88.27788 12.31084 5.39503
19.80150 1003.51929 3006.95237 1476.12014 926.53103
455.26869 5.05074 72.37795 0.49812
0.48919 0.31810 12.18756 0.40642
513.04025 0.16394 8.28959 0.49707
0.20006 2.22212
min 0.00000 0.10000 0.00006
-1.00000 -1.00000 10.00000 0.00000
-14.06498 1.00000 8.67828 1320.28399 2863.78334
0.00000 0.00000 -170.00000 0.00000
0.00000 0.00000 -1.00000 0.00000
190.00000 0.00000 -1.00000 0.00000
-1.00000 0.00000
25% 0.00000 0.30000 0.21060
-1.00000 22.00000 20.00000 0.00706
-1.18451 899.00000 3388.04839 3578.88819 4260.40529
1.00000 5.00000 85.00000 0.00000
0.00000 1.00000 -1.00000 0.00000
200.00000 0.00000 3.11938 0.00000
1.00000 1.00000
50% 0.00000 0.60000 0.47765
-1.00000 56.00000 30.00000 0.01492
-0.83992 1272.00000 5272.14754 4734.07444 4904.60572
9.00000 9.00000 126.00000 1.00000
0.00000 1.00000 5.00000 0.00000
200.00000 0.00000 5.09977 1.00000
1.00000 3.00000
75% 0.00000 0.80000 0.74887
11.00000 133.00000 40.00000 0.02607
-0.19746 1963.00000 7644.29673 5733.29879 5483.66273
23.00000 13.00000 185.00000 1.00000
1.00000 1.00000 25.00000 0.00000
1000.00000 0.00000 8.87309 1.00000
1.00000 5.00000
max 1.00000 0.90000 1.00000
365.00000 424.00000 90.00000 76.57727
112.25311 6521.00000 16665.35953 9506.89660 6994.76420
2322.00000 37.00000 378.00000 1.00000
1.00000 1.00000 32.00000 1.00000
2100.00000 1.00000 82.00473 1.00000
2.00000 7.00000

`.describe()` akan mengeluarkan perhitungan:

1. Banyaknya baris pada setiap kolom (`count`).
2. Rata-rata setiap kolom (`mean`).
3. Standar deviasi setiap kolom (`std`).
4. Nilai minimum (`min`) dan maksimum (`max`) setiap kolom.
5. Kuartil bawah (`25%`), median (`50%`), dan kuartil atas (`75%`) setiap kolom.

28.3 Data Types Inspection and Preparation

Pemeriksaan tipe data adalah proses memverifikasi jenis data dalam setiap kolom dari *dataset* untuk memastikan bahwa setiap kolom sesuai dengan jenis data yang diharapkan. Misalnya, kolom yang berisi tanggal harus diidentifikasi sebagai tipe data tanggal/waktu, sementara kolom yang berisi teks harus diidentifikasi sebagai *string*, dan kolom dengan angka harus diidentifikasi sebagai bilangan bulat (*integer*) atau bilangan berkoma *(floating point)* tergantung pada kebutuhan analisis.

Untuk memeriksa tipe setiap kolom, kita bisa menggunakan `.dtypes`.

[5]:
```
credit.dtypes
```

[5]:
```
fraud_bool                          int64
income                            float64
name_email_similarity             float64
prev_address_months_count           int64
current_address_months_count        int64
customer_age                        int64
days_since_request                float64
intended_balcon_amount            float64
payment_type                       object
zip_count_4w                        int64
velocity_6h                       float64
velocity_24h                      float64
velocity_4w                       float64
bank_branch_count_8w                int64
date_of_birth_distinct_emails_4w    int64
employment_status                  object
credit_risk_score                   int64
email_is_free                       int64
housing_status                     object
phone_home_valid                    int64
phone_mobile_valid                  int64
bank_months_count                   int64
has_other_cards                     int64
proposed_credit_limit             float64
foreign_request                     int64
source                             object
session_length_in_minutes         float64
device_os                          object
keep_alive_session                  int64
```

```
device_distinct_emails_8w             int64
month                                 int64
dtype: object
```

Beberapa kolom yang bertipe kategorikal berdasarkan deskripsi data sebelumnya perlu kita ubah terlebih dahulu. Pengubahan tipe data ini menggunakan fungsi `.astype()`. Detail lebih lanjut terkait fungsi `.astype()`, Anda dapat mengunjungi referensi berikut.

[6]:
```
# mengubah tipe data sebagai kategorikal

cat_cols =␣
↪['payment_type','employment_status','housing_status','source','device_os']

credit[cat_cols] = credit[cat_cols].astype('category')
```

[7]:
```
# cek tipe data setelah diubah

credit.dtypes
```

[7]:
```
fraud_bool                            int64
income                              float64
name_email_similarity               float64
prev_address_months_count             int64
current_address_months_count          int64
customer_age                          int64
days_since_request                  float64
intended_balcon_amount              float64
payment_type                       category
zip_count_4w                          int64
velocity_6h                         float64
velocity_24h                        float64
velocity_4w                         float64
bank_branch_count_8w                  int64
date_of_birth_distinct_emails_4w      int64
employment_status                  category
credit_risk_score                     int64
email_is_free                         int64
housing_status                     category
phone_home_valid                      int64
phone_mobile_valid                    int64
bank_months_count                     int64
has_other_cards                       int64
proposed_credit_limit               float64
foreign_request                       int64
source                             category
session_length_in_minutes           float64
device_os                          category
keep_alive_session                    int64
```

```
device_distinct_emails_8w              int64
month                                  int64
dtype: object
```

> ***Note***: perlu diingat kembali bahwa terdapat dua tipe data: numerikal dan kategorikal. Khusus kolom kategorik, terdapat pemrosesan tambahan yang akan dibahas lebih lanjut pada bagian *Data Preprocessing*.

28.4 Check Missing Values

Pemeriksaan nilai *null/missing value* adalah proses untuk mendeteksi keberadaan nilai *null* dalam sebuah *dataset*. Nilai *null* adalah entri yang tidak memiliki data di dalamnya. Hal ini bisa berarti bahwa informasi tersebut tidak diketahui, tidak ada, atau tidak berlaku. Dalam konteks pemrograman dan analisis data, nilai *null* ini sering kali diwakili dengan nilai khusus seperti `NULL`, `None`, `NaN` (*Not a Number*).

Untuk melihat apakah terdapat nilai *null* pada data, kita dapat menggunakan sintaks berikut.

[8]:
```
credit.isna().sum()
```

[8]:
```
fraud_bool                          0
income                              0
name_email_similarity               0
prev_address_months_count           0
current_address_months_count        0
customer_age                        0
days_since_request                  0
intended_balcon_amount              0
payment_type                        0
zip_count_4w                        0
velocity_6h                         0
velocity_24h                        0
velocity_4w                         0
bank_branch_count_8w                0
date_of_birth_distinct_emails_4w    0
employment_status                   0
credit_risk_score                   0
email_is_free                       0
housing_status                      0
phone_home_valid                    0
phone_mobile_valid                  0
bank_months_count                   0
has_other_cards                     0
proposed_credit_limit               0
foreign_request                     0
source                              0
session_length_in_minutes           0
device_os                           0
```

```
keep_alive_session                  0
device_distinct_emails_8w           0
month                               0
dtype: int64
```

Fungsi `.isna().sum()` digunakan untuk melihat banyaknya nilai *null* di setiap kolom. Terlihat pada *output* yang ditampilkan bahwa data kita tidak memiliki nilai *null* sehingga tidak perlu dilakukan *handling missing values*.

[***Optional***] Selain menggunakan `.dtypes()` untuk memeriksa tipe data dan `.isna().sum()` untuk melihat banyak *missing values*, kita dapat menggunakan `.info()` untuk menampilkan *summary* yang lebih lengkap.

[9]:
```
credit.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 31 columns):
 #   Column                            Non-Null Count   Dtype
---  ------                            --------------   -----
 0   fraud_bool                        100000 non-null  int64
 1   income                            100000 non-null  float64
 2   name_email_similarity             100000 non-null  float64
 3   prev_address_months_count         100000 non-null  int64
 4   current_address_months_count      100000 non-null  int64
 5   customer_age                      100000 non-null  int64
 6   days_since_request                100000 non-null  float64
 7   intended_balcon_amount            100000 non-null  float64
 8   payment_type                      100000 non-null  category
 9   zip_count_4w                      100000 non-null  int64
 10  velocity_6h                       100000 non-null  float64
 11  velocity_24h                      100000 non-null  float64
 12  velocity_4w                       100000 non-null  float64
 13  bank_branch_count_8w              100000 non-null  int64
 14  date_of_birth_distinct_emails_4w  100000 non-null  int64
 15  employment_status                 100000 non-null  category
 16  credit_risk_score                 100000 non-null  int64
 17  email_is_free                     100000 non-null  int64
 18  housing_status                    100000 non-null  category
 19  phone_home_valid                  100000 non-null  int64
 20  phone_mobile_valid                100000 non-null  int64
 21  bank_months_count                 100000 non-null  int64
 22  has_other_cards                   100000 non-null  int64
 23  proposed_credit_limit             100000 non-null  float64
 24  foreign_request                   100000 non-null  int64
 25  source                            100000 non-null  category
 26  session_length_in_minutes         100000 non-null  float64
 27  device_os                         100000 non-null  category
 28  keep_alive_session                100000 non-null  int64
```

```
 29  device_distinct_emails_8w         100000 non-null  int64
 30  month                             100000 non-null  int64
dtypes: category(5), float64(9), int64(17)
memory usage: 20.3 MB
```

Fungsi `.info()` mengeluarkan *output*:

- Banyak baris pada *dataset*: data `credit` terdiri dari 100000 baris.
- Banyak kolom pada *dataset*: data `credit` terdiri dari 31 kolom.
- Nama setiap kolom dan tipe datanya.
- Banyaknya observasi yang tidak *null*. Kita dapat mengidentifikasi kolom dengan nilai *null* dari output ini. Data `credit` tidak memiliki *missing values*.

28.5 Feature Selection

Feature selection atau seleksi fitur merupakan proses pemilihan *subset* fitur (variabel atau atribut) yang paling relevan untuk pembuatan model *machine learning*. *Feature selection* bertujuan untuk memilih fitur-fitur yang paling sesuai dan mengabaikan fitur-fitur yang tidak relevan atau redundan.

Secara umum, terdapat 2 langkah untuk melakukan *feature selection*:

1. *Feature selection* berdasarkan intuisi bisnis

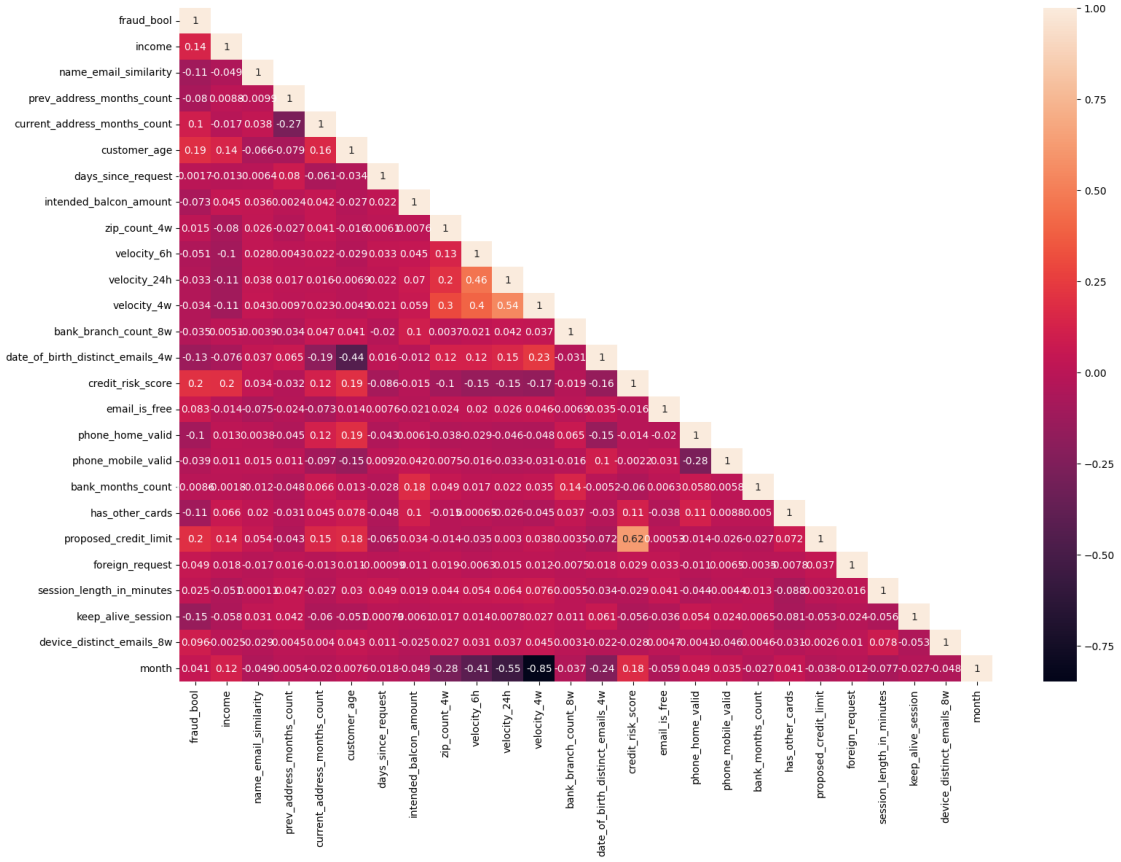
Artinya, kita memasukkan fitur/variabel/atribut yang menurut pandangan bisnis berpengaruh kuat dalam melakukan prediksi. Langkah ini dapat dilakukan apabila terdapat *domain knowledge*, pandangan ahli, atau kesepakatan yang dapat dijustifikasi kebenarannya.

2. Feature selection berdasarkan perhitungan matematika/statistika

Artinya, pemilihan fitur/variabel/atribut yang akan digunakan untuk membuat *machine learning* didasarkan pada perhitungan matematis. Sebagai contoh, *feature selection* yang berpedoman pada perhitungan korelasi setiap kolom.

Sekarang, kita akan berfokus kepada kolom-kolom numerikal data kita. Kita akan membuat sebuah *heatmap* yang menunjukkan korelasi antarkolom numerikal.

[10]:
```
# cek korelasi

corr = credit.select_dtypes("number").corr()
matrix = np.triu(corr, k=1)

plt.figure(figsize = (18,12));
sns.heatmap(corr, annot=True, mask=matrix);
```



Berdasarkan gambar di atas, terlihat bahwa sebagian besar variabel memiliki korelasi yang rendah dengan variabel lainnya. Korelasi yang cukup kuat (sebesar -0.85) terdapat antara `month` dan `velocity_24h`. Idealnya, kita mengharapkan antarkolom numerikal di atas memiliki korelasi yang rendah.

> Perlu diperhatikan bahwa beberapa model mengasumsikan antarprediktor tidak saling berkorelasi tinggi, seperti *logistic regression*. Akan tetapi, model yang cukup *robust*, seperti *decision tree* dan *random forest* tidak memiliki aturan yang berhubungan dengan korelasi.
>
> *Feature selection* berdasarkan korelasi hanya merupakan salah satu alternatif dalam melakukan *feature selection*. Metode *feature selection* lainnya dapat dilihat pada referensi berikut.

29 Data Preprocessing

Pada dasarnya, *data preprocessing* tidak berbeda jauh dengan *data preparation*. Keduanya sering digunakan secara bergantian dan sama-sama mengacu kepada serangkaian proses yang dilakukan untuk menghasilkan data dalam format yang siap digunakan untuk proses analisis lanjutan. Akan tetapi, biasanya *data preparation* memiliki cakupan yang lebih luas dan *data preprocessing* termasuk ke dalam cakupan tersebut.

Dalam konteks *course* ini, kita akan mengacu *data preprocessing* sebagai proses lanjutan yang secara spesifik mempersiapkan data untuk proses *modeling* menggunakan *machine learning*. Terdapat beberapa *step* yang umum dilakukan untuk *data preprocessing*:

1. Konversi variabel prediktor yang kategorikal menjadi *dummy variables*.
2. Pemisahan variabel prediktor dan variabel target.
3. *Train-test split*.

29.1 Dummy Variables for Categorical Predictor

Water	Temperature
A	Hot
B	Cold
C	Warm
D	Cold

Dummy Variables

Water	Temperature	var_hot	var_warm	var_cold
A	Hot	1	0	0
B	Cold	0	0	1
C	Warm	0	1	0
D	Cold	1	0	0

Adakalanya, data kita memiliki prediktor yang kategorikal, seperti yang terlihat pada gambar. Terdapat kolom `Temperature` dengan tiga variasi nilai: `Hot`, `Cold`, dan `Warm`. Data dalam bentuk seperti ini tidak bisa langsung diproses oleh mesin untuk membuat model *machine learning*. Untuk mengatasi hal tersebut, kita perlu melakukan *processing* tambahan, yaitu *one-hot encoding*.

One-hot encoding merupakan teknik yang digunakan untuk membuat data kategorikal ke dalam representasi yang dapat dimengerti oleh model *machine learning*. *One-hot encoding* bekerja dengan cara berikut:

- Mengidentifikasi nilai unik yang terdapat pada kolom kategorik.
- Membuat kolom baru (*dummy variable*) untuk setiap nilai unik.
- Memasukkan nilai biner (1 atau 0) ke setiap kolom baru tersebut.

Perhatikan kembali gambar di atas. Awalnya terdapat tiga variasi nilai: `Hot`, `Cold`, dan `Warm`. Selanjutnya, dibuat tiga kolom baru untuk setiap variasi nilai: `var_hot`, `var_cold`, dan `var_warm`. Setelah itu, nilai 1 dan 0 dimasukkan ke setiap kolom berdasarkan kolom aslinya.

- Jika `Temperature` awalnya adalah `Hot`, maka `var_hot = 1`, `var_cold = 0`, dan `var_warm = 0`.
- Jika `Temperature` awalnya adalah `Cold`, maka `var_hot = 0`, `var_cold = 1`, dan `var_warm = 0`.
- Jika `Temperature` awalnya adalah `Warm`, maka `var_hot = 0`, `var_cold = 0`, dan `var_warm = 1`.

Untuk melakukan *one-hot encoding*, kita dapat memanfaatkan fungsi `pd.get_dummies()` dari `pandas`.

[11]:
```
# contoh implementasi pd.get_dummies()

example_dummy = pd.DataFrame({
    "Daerah" : ['Jakarta','Bandung','Depok','Bekasi','Jakarta'],
    "Suhu" : [38, 36, 37.5, 38, 39],
})

example_dummy
```

[11]:
```
    Daerah     Suhu
0  Jakarta 38.00000
1  Bandung 36.00000
2    Depok 37.50000
3   Bekasi 38.00000
4  Jakarta 39.00000
```

[12]:
```
pd.get_dummies(example_dummy, dtype = int)
```

[12]:

	Suhu	Daerah_Bandung	Daerah_Bekasi	Daerah_Depok	Daerah_Jakarta
0	38.00000	0	0	0	1
1	36.00000	1	0	0	0
2	37.50000	0	0	1	0
3	38.00000	0	1	0	0

```
4 39.00000                0               0              0                1
```

Detail lebih lanjut bagaimana menggunakan fungsi `pd.get_dummies()` dapat dilihat pada referensi berikut.

Sekarang, kita akan melihat kembali bagaimana data `telco_clean` kita sebelumnya.

[13]:
```
credit.head()
```

[13]:
```
   fraud_bool  income  name_email_similarity  prev_address_months_count
current_address_months_count  customer_age  days_since_request
intended_balcon_amount payment_type  zip_count_4w  velocity_6h  velocity_24h
velocity_4w  bank_branch_count_8w  date_of_birth_distinct_emails_4w
employment_status  credit_risk_score  email_is_free housing_status
phone_home_valid  phone_mobile_valid  bank_months_count  has_other_cards
proposed_credit_limit  foreign_request    source  session_length_in_minutes
device_os  keep_alive_session  device_distinct_emails_8w  month
0           0 0.90000                0.61579                         -1
51            40             0.01034                -0.69955           AB
2659  11007.70883    7448.79742   5576.30356                  2030
7                CA                253              1             BA
1                   1                  9                1             1500.00000
0  INTERNET                   13.07981     linux                    0
1      1
1           0 0.60000                0.07246                         -1
125            20             0.00697                -0.95947           AC
5599   6304.26284    4373.09359   4721.10002                     1
13                CA                112              1             BE
1                   0                 -1                0             200.00000
0  INTERNET                    6.66655   windows                    0
1      4
2           1 0.50000                0.07659                         -1
37            50             0.00632                -1.21993           AB
1352   1357.41355    6667.02966   6318.87626                    15
6                CF                199              1             BC
1                   1                 28                0             1500.00000
0  INTERNET                    4.49267   windows                    1
2      0
3           0 0.30000                0.25543                         23
19            20             0.00927                 3.24224           AA
3743   5234.26230    6139.48958   4201.72592                     1
14                CA                 63              1             BC
0                   1                 16                0             200.00000
0  INTERNET                    2.32189     other                    1
1      3
4           0 0.90000                0.19236                         -1
34            30             0.01924                -1.13156           AB
546   3589.15345    4574.39557   4318.25891                    16
```

```
5                CA                275              1              BC
1                   1                  1                1             1000.00000
0  INTERNET                    4.84515     linux                    1
1      5
```

Berdasarkan deskripsi data sebelumnya, kita mengetahui bahwa kita memiliki beberapa kolom kategorikal.

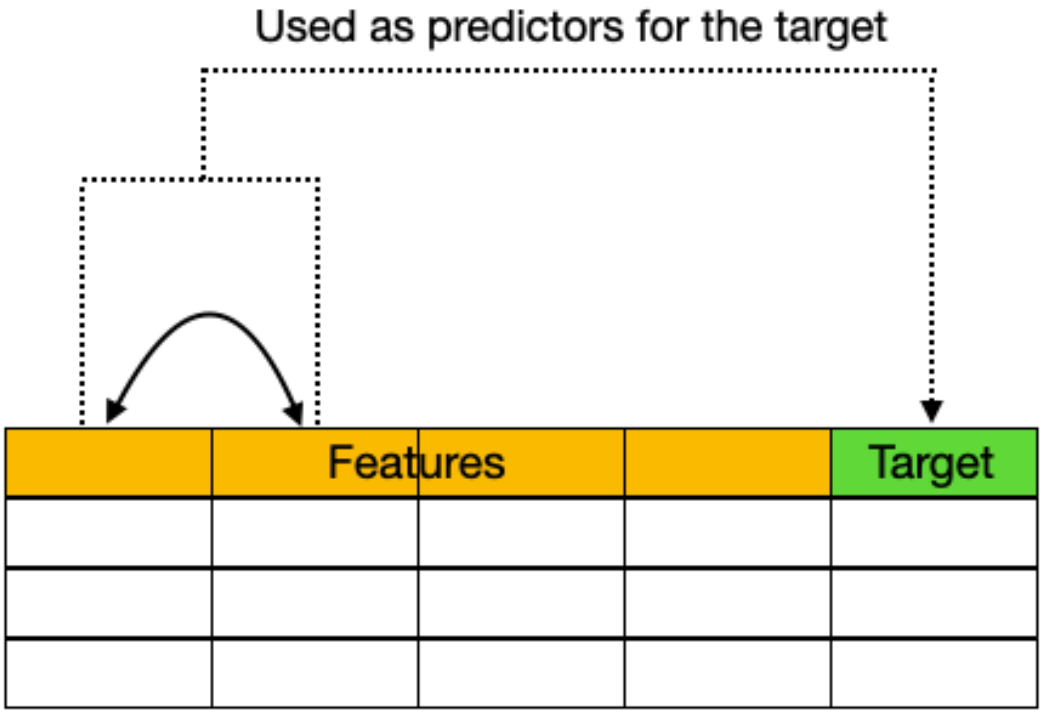
Mari kita ubah kolom-kolom tersebut menggunakan `pd.get_dummies()`.

[14]:
```
credit_enc = pd.get_dummies(credit,
                            columns = cat_cols,
                            drop_first = True,
                            dtype='int64')

credit_enc.head()
```

[14]:
```
   fraud_bool  income  name_email_similarity  prev_address_months_count
current_address_months_count  customer_age  days_since_request
intended_balcon_amount  zip_count_4w  velocity_6h  velocity_24h  velocity_4w
bank_branch_count_8w  date_of_birth_distinct_emails_4w  credit_risk_score
email_is_free  phone_home_valid  phone_mobile_valid  bank_months_count
has_other_cards  proposed_credit_limit  foreign_request
session_length_in_minutes  keep_alive_session  device_distinct_emails_8w  month
payment_type_AB  payment_type_AC  payment_type_AD  payment_type_AE
employment_status_CB  employment_status_CC  employment_status_CD
employment_status_CE  employment_status_CF  employment_status_CG
housing_status_BB  housing_status_BC  housing_status_BD  housing_status_BE
housing_status_BF  housing_status_BG  source_TELEAPP  device_os_macintosh
device_os_other  device_os_windows  device_os_x11
0           0 0.90000                0.61579                        -1
51            40             0.01034                -0.69955          2659
11007.70883    7448.79742   5576.30356                  2030
7                253              1                 1                   1
9                1             1500.00000                0
13.07981                    0                          1      1                1
0                0                0                    0                    0
0                    0                    0                    0
0                  0                  0                  0                  0
0               0                    0                0                  0
0
1           0 0.60000                0.07246                        -1
125            20             0.00697                -0.95947          5599
6304.26284    4373.09359   4721.10002                    1
13                112              1                 1                   0
-1                0              200.00000                0
6.66655                    0                          1      4                0
1                0                0                    0                    0
```

0 0 0 0
0 0 0 1 0
0 0 0 0 1
0
2 1 0.50000 0.07659 -1
37 50 0.00632 -1.21993 1352
1357.41355 6667.02966 6318.87626 15
6 199 1 1 1
28 0 1500.00000 0
4.49267 1 2 0 1
0 0 0 0 0
0 0 1 0
0 1 0 0 0
0 0 0 0 1
0
3 0 0.30000 0.25543 23
19 20 0.00927 3.24224 3743
5234.26230 6139.48958 4201.72592 1
14 63 1 0 1
16 0 200.00000 0
2.32189 1 1 3 0
0 0 0 0 0
0 0 0 0
0 1 0 0 0
0 0 0 1 0
0
4 0 0.90000 0.19236 -1
34 30 0.01924 -1.13156 546
3589.15345 4574.39557 4318.25891 16
5 275 1 1 1
1 1 1000.00000 0
4.84515 1 1 5 1
0 0 0 0 0
0 0 0 0
0 1 0 0 0
0 0 0 0 0
0

29.2 Target & Predictors Splitting



Di dunia *machine learning*, variabel yang kita gunakan dalam proses pemodelan terbagi menjadi dua:

1. Variabel Prediktor/Variabel Independen (X)

Variabel prediktor merupakan variabel-variabel yang digunakan untuk membuat prediksi. Hasil prediksi *machine learning* sangat bergantung pada nilai dari variabel-variabel prediktor. Variabel prediktor dapat berupa variabel numerikal maupun variabel kategorikal. Umumnya, variabel prediktor dinotasikan dengan X. Seperti yang terlihat pada gambar, variabel prediktor ditunjukkan oleh kolom-kolom yang berwarna oranye.

2. Variabel Target/Variabel Dependen (y)

Variabel target merupakan variabel yang akan diprediksi menggunakan prediktor. Seperti namanya, hasil prediksi dependen terhadap variabel prediktor. Pada kasus klasifikasi, variabel target merupakan variabel kategorikal. Berbeda dengan *case* regresi yang variabel targetnya merupakan variabel numerikal. Variabel target dinotasikan sebagai y. Pada gambar, variabel target ditunjukkan oleh kolom yang berwarna hijau.

Untuk memisahkan variabel prediktor dan variabel target, kita bisa memanfaatkan fungsi `.drop()`. Berdasarkan *problem statement* sebelumnya, variabel target kita adalah kolom `fraud_bool` dan sisanya merupakan variabel prediktor.

[15]:
```
# mendefinisikan variabel target
y = credit_enc['fraud_bool']

# mendefinisikan variabel prediktor
```

```
X = credit_enc.drop(columns = ['fraud_bool'])
```

[16]:
```
# melihat 5 baris pertama X
X.head()
```

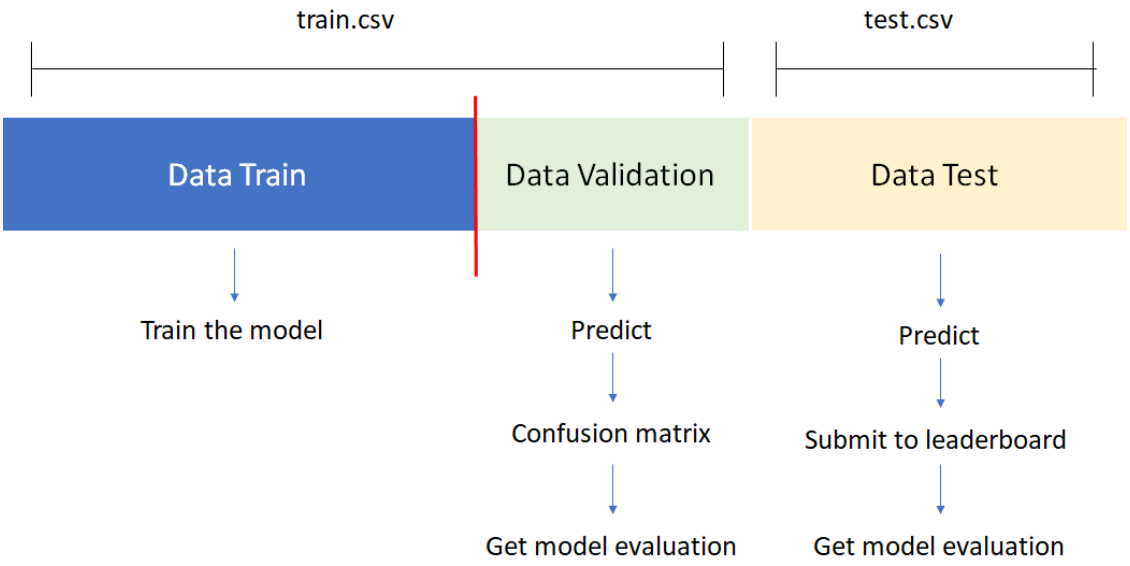
[16]:
```
   income  name_email_similarity  prev_address_months_count
current_address_months_count  customer_age  days_since_request
intended_balcon_amount  zip_count_4w  velocity_6h  velocity_24h  velocity_4w
bank_branch_count_8w  date_of_birth_distinct_emails_4w  credit_risk_score
email_is_free  phone_home_valid  phone_mobile_valid  bank_months_count
has_other_cards  proposed_credit_limit  foreign_request
session_length_in_minutes  keep_alive_session  device_distinct_emails_8w  month
payment_type_AB  payment_type_AC  payment_type_AD  payment_type_AE
employment_status_CB  employment_status_CC  employment_status_CD
employment_status_CE  employment_status_CF  employment_status_CG
housing_status_BB  housing_status_BC  housing_status_BD  housing_status_BE
housing_status_BF  housing_status_BG  source_TELEAPP  device_os_macintosh
device_os_other  device_os_windows  device_os_x11
0 0.90000                0.61579                         -1
51            40             0.01034                -0.69955          2659
11007.70883    7448.79742   5576.30356                  2030
7                253              1                 1                   1
9                1             1500.00000                0
13.07981                   0                          1      1                1
0                0                0                    0                    0
0                    0                     0                     0
0                  0                  0                  0                  0
0               0                    0                0                  0
0
1 0.60000                0.07246                         -1
125            20             0.00697                -0.95947          5599
6304.26284    4373.09359   4721.10002                    1
13                112              1                 1                   0
-1                0              200.00000                0
6.66655                   0                          1      4                0
1                0                0                    0                    0
0                    0                     0                     0
0                  0                  0                  1                  0
0               0                    0                0                  1
0
2 0.50000                0.07659                         -1
37            50             0.00632                -1.21993          1352
1357.41355    6667.02966   6318.87626                   15
6                199              1                 1                   1
28                0             1500.00000                0
4.49267                    1                          2      0                1
0                0                0                    0                    0
```

```
0                    0                    1                    0
0                  1                  0                  0                  0
0              0                  0                0                  1
0
3 0.30000                0.25543                        23
19            20             0.00927                3.24224          3743
5234.26230    6139.48958   4201.72592                    1
14                 63              1                0                  1
16               0              200.00000                0
2.32189                  1                          1      3                0
0                0                0                    0                    0
0                    0                    0                    0
0                  1                  0                  0                  0
0              0                  0                1                  0
0
4 0.90000                0.19236                        -1
34            30             0.01924               -1.13156           546
3589.15345    4574.39557   4318.25891                   16
5                275              1                1                  1
1                1             1000.00000               0
4.84515                  1                          1      5                1
0                0                0                    0                    0
0                    0                    0                    0
0                  1                  0                  0                  0
0              0                  0                0                  0
0
```

[17]:
```python
# melihat 5 baris pertama y

y.head()
```

[17]:
```
0    0
1    0
2    1
3    0
4    0
Name: fraud_bool, dtype: int64
```

29.3 Cross Validation



Sebelum membuat model *machine learning* untuk klasifikasi, kita akan berkenalan dengan konsep *cross validation* di mana kita:

1. Memisahkan *dataset* kita menjadi *training set*, *validation set*, dan *testing set*.
2. Melatih model kita hanya menggunakan *training set*.
3. Mengevaluasi model pada *validation set* dan kembali ke langkah sebelumnya jika perlu (misalnya, memilih variabel prediktor yang berbeda, menggunakan parameter yang berbeda, atau menyetel aspek lain dari spesifikasi model).
4. Memilih model final berdasarkan kriteria evaluasi, misalnya berdasarkan akurasi.
5. Mendapatkan hasil prediksi yang tidak bias saat melakukan prediksi pada *testing set*.

Kita dapat mengulangi langkah 2 dan langkah 3 sebanyak yang diperlukan. Misalnya, kita ingin menguji berbagai metode klasifikasi, mengganti parameter model, atau kombinasi variabel prediktor yang berbeda. Dari hasil percobaan tersebut, kita memilih model terbaik berdasarkan suatu kriteria evaluasi. Model terbaik ini yang akan digunakan untuk memprediksi data pada *testing set*. Perlu diperhatikan bahwa *testing set* tidak boleh digunakan selama proses pelatihan model. Dalam hal ini, *testing set* merupakan data baru yang tidak ditemui model selama proses pelatihan.

29.3.1 Train-Test Split

Untuk mengimplementasikan *cross validation*, kita dapat menggunakan fungsi `train_test_split()` dari *package* `sklearn`.

> Terdapat banyak praktik terkait berapa persentase dari data yang dialokasikan untuk *training set* dan *testing set*. Beberapa praktik yang cukup umum adalah 80% untuk *training set* dan 20% untuk *testing set*. Praktik lainnya menggunakan 75% dari data untuk *training set* dan 25% untuk *testing set*.
>
> ***Note***: perlu diingat bahwa proporsi untuk *training set* selalu lebih besar.

[18]:
```
# implementasi train_test_split dengan test_size sebesar 20% dari keseluruhan␣
↪data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,␣
↪random_state=42, stratify=y)
```

[19]:
```
# shape data awal

X.shape, y.shape
```

[19]: ((100000, 46), (100000,))

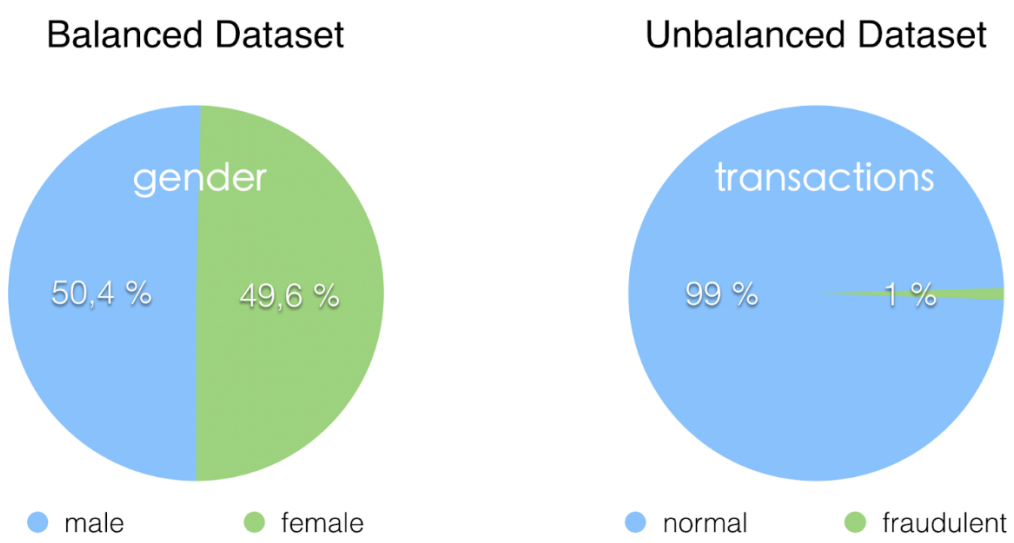
[20]:
```
# shape data untuk training set

X_train.shape, y_train.shape
```

[20]: ((80000, 46), (80000,))

[21]:
```
# shape data untuk testing set

X_test.shape, y_test.shape
```

[21]: ((20000, 46), (20000,))

29.3.2 Class Balance & Imbalance



Imbalance merupakan suatu kondisi di mana terdapat satu kelas yang proporsinya lebih banyak (mayoritas) dibandingkan kelas lainnya. *Imbalance* sangat sering ditemui di dunia nyata, termasuk pada *dataset* yang kita gunakan saat ini.

Sekarang, mari kita lihat bagaimana proporsi antara label 0 dan 1 pada `y_train` menggunakan fungsi `.value_counts()`.

[22]:
```
y_train.value_counts()
```

[22]:
```
fraud_bool
0    71177
1     8823
Name: count, dtype: int64
```

Terlihat bahwa kelas 0, yakni pembeli yang tidak membeli produk lebih banyak dari pada kelas 1, yakni pembeli yang membeli produk. Kasus *imbalance* ini perlu ditangani dengan benar untuk mencegah model bias ke salah satu kelas. Dalam hal ini, model lebih banyak mempelajari contoh kasus dari kelas 0 dan tidak dapat mengidentifikasi karakteristik dari kelas 1 dengan benar.

> Teknik yang bisa dimanfaatkan untuk mengatasi *imbalance dataset* adalah *oversampling* (*upsampling*) dan *undersampling* (*downsampling*). Referensi terkait metode *handling imbalance dataset* dapat dilihat di sini

30 Model Classification 1 - Logistic Regression

Sejauh ini, kita sudah melakukan persiapan untuk membuat model *machine learning* yang bisa melakukan klasifikasi. Mulai dari akses data yang akan digunakan sampai melakukan *sampling* untuk mengatasi data yang *imbalance*. Model pertama yang akan kita gunakan untuk melakukan klasifikasi adalah *logistic regression*.

Logistic regression merupakan metode klasifikasi yang menjadi pondasi untuk menentukan hasil prediksi y.

- Apabila y merupakan satu nilai di antara dua pilihan kategorikal (biner), *logistic regression* dapat juga disebut sebagai *binomial logistic regression*.
- Apabila y merupakan satu nilai di antara banyak pilihan kategorikal, *logistic regression* dapat juga disebut sebagai *multinomimial logistic regression*.

Logistic regression menggunakan konsep *probability*/peluang untuk mengeluarkan prediksi. Oleh karena itu, kita akan coba untuk meninjau kembali konsep peluang.

30.1 Probability

Secara sederhana, *probability*/peluang dapat dikatakan sebagai besarnya kemungkinan sesuatu akan terjadi. Apabila didefinisikan secara matematis, peluang p suatu kejadian dapat dirumuskan:

$$p = \frac{frekuensi\ kejadian\ yang\ diharapkan}{frekuensi\ seluruh\ percobaan}$$

Contoh:

- Jika terdapat 20 dari 100 pelanggan perusahaan telekomunikasi yang *churn* (hilang), maka peluang *churn* $= \frac{20}{100} = 0.2$.
- Jika 3 dari 5 *email* terdeteksi sebagai email yang *spam*, maka peluang *spam* $= \frac{3}{5} = 0.6$.
- Jika 4 dari 5 penerbangan dilakukan tepat waktu, maka peluang penerbangan dilakukan tepat waktu $= \frac{4}{5} = 0.8$.

Berikut merupakan beberapa poin penting yang perlu diingat mengenai peluang:

- *Range* peluang adalah 0 sampai 1. 0 artinya kejadian tersebut mustahil terjadi dan 1 artinya kejadian tersebut pasti terjadi.
- Bila p merupakan peluang suatu peristiwa untuk terjadi, maka q adalah peluang di mana peristiwa tidak terjadi (peluang komplemen). Secara matematis:

$$p + q = 1 \rightarrow q = 1 - p$$

30.2 Relating Probabilities to Odds

Selama ini, kita hanya mengenal p untuk menyatakan suatu kemungkinan/peluang. Sekarang, kita akan mendalami konsep lain yang juga digunakan untuk membahasakan peluang, yakni *odds*.

Odds secara sederhana merupakan rasio peluang suatu peristiwa terjadi (p) dan peluang suatu peristiwa tidak terjadi/peluang komplemen peristiwa tersebut (q).

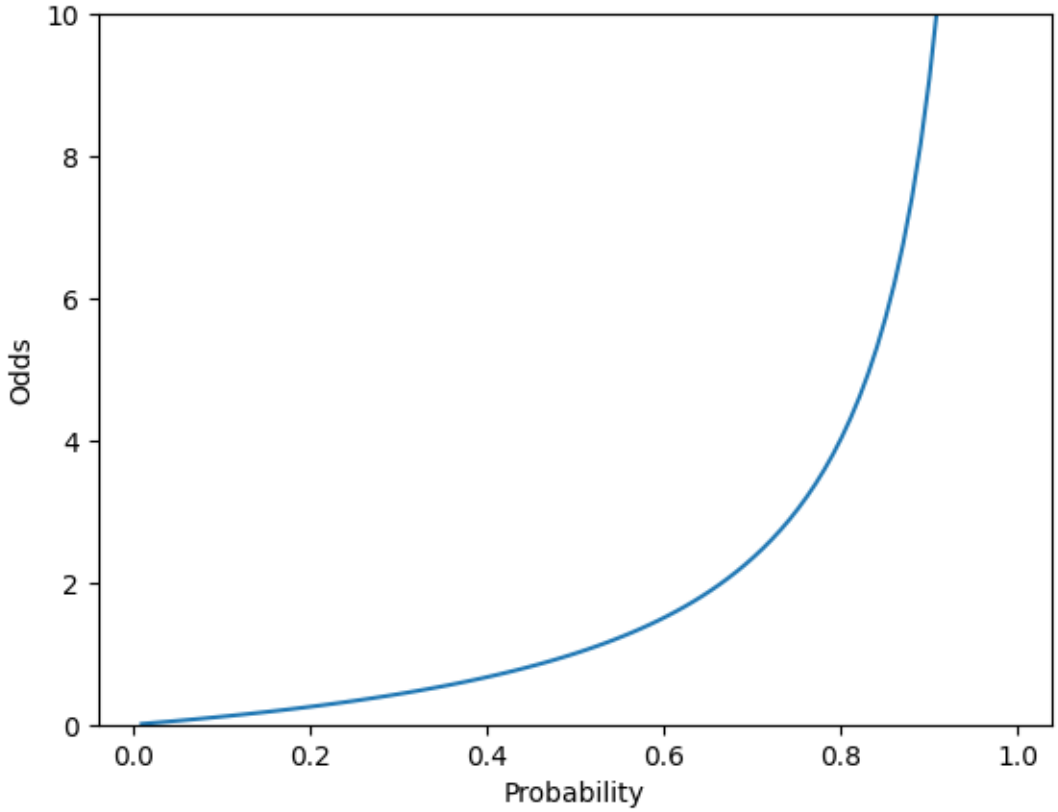
$$Odds = \frac{p}{q} = \frac{p}{1-p}$$

Apabila kita menggunakan kembali contoh pada bagian *probability*:

- *Odds* pelanggan yang *churn* dari perusahaan telekomunikasi $= \frac{0.2}{0.8} = 0.125$. Artinya, peluang pelanggan untuk kembali/berlangganan kembali 4 kali lebih besar dibandingkan dengan peluang pelanggan yang hilang.
- *Odds email* yang *spam* $= \frac{0.6}{0.4} = 1.5$. Artinya, peluang suatu email untuk *spam* 1.5 kali lebih tinggi daripada *email* tersebut tidak *spam*.
- *Odds* penerbangan tepat waktu $= \frac{0.8}{0.2} = 4$. Artinya, suatu penerbangan 4 kali lebih berkemungkinan untuk berangkat tepat waktu.

Apabila kita membuat suatu plot yang menunjukkan hubungan peluang p dan nilai *odds*-nya:

[23]:
```
x = np.linspace(0.01, 0.99, 100)
y = x/(1-x)

plt.plot(x,y);
plt.ylabel('Odds');
plt.xlabel('Probability');
plt.ylim(0,10);
```



Terlihat bahwa hubungan antara peluang p dan *odds*-nya adalah monotonik. Semakin besar nilai peluang p, semakin besar nilai *odds*.

Range nilai *odds* adalah 0 sampai `+Inf`

30.2.1 Log of Odds

Sebelumnya kita sudah mengenal *odds* sebagai perbandingan antara peluang p dan komplemen q. *Log of odds* tidak lain merupakan *odds* yang ditransformasi dengan fungsi `log()`. Untuk transformasi ini, kita bisa memanfaatkan fungsi `np.log()` dari *package* `numpy`.

$$log(odds) = log(\frac{p}{q}) = log(\frac{p}{1-p}) = logit(p)$$

Misalnya, kita memiliki peluang $p = 0.2$. Untuk mendapatkan *log of odds*-nya:

[24]:
```
# menentukan log of odds untuk peluang p = 0.2

p = 0.2
odds = p/(1-p)
log_odds = np.log(odds)
```

```
print(odds)
print(log_odds)
```

```
0.25
-1.3862943611198906
```

Untuk $p = 0.2$ didapatkan nilai *log of odds*-nya adalah -1.3862943611198906.

> *Range* nilai *log of odds* adalah `-Inf` sampai `+Inf`.

Apabila kita ingin mendapatkan kembali nilai *odds* dari nilai *log of odds* tersebut, kita bisa menggunakan fungsi `np.exp()` dari *package* `numpy`.

[25]:
```
# mendapatkan nilai odds dari nilai log of odds = -1.3862943611198906

np.exp(log_odds)
```

[25]: 0.25

Untuk mendapatkan nilai peluang dari *log of odds*:

[26]:
```
# mendapatkan nilai p dari lof of odds

1/(1 + np.exp(-log_odds))
```

[26]: 0.2

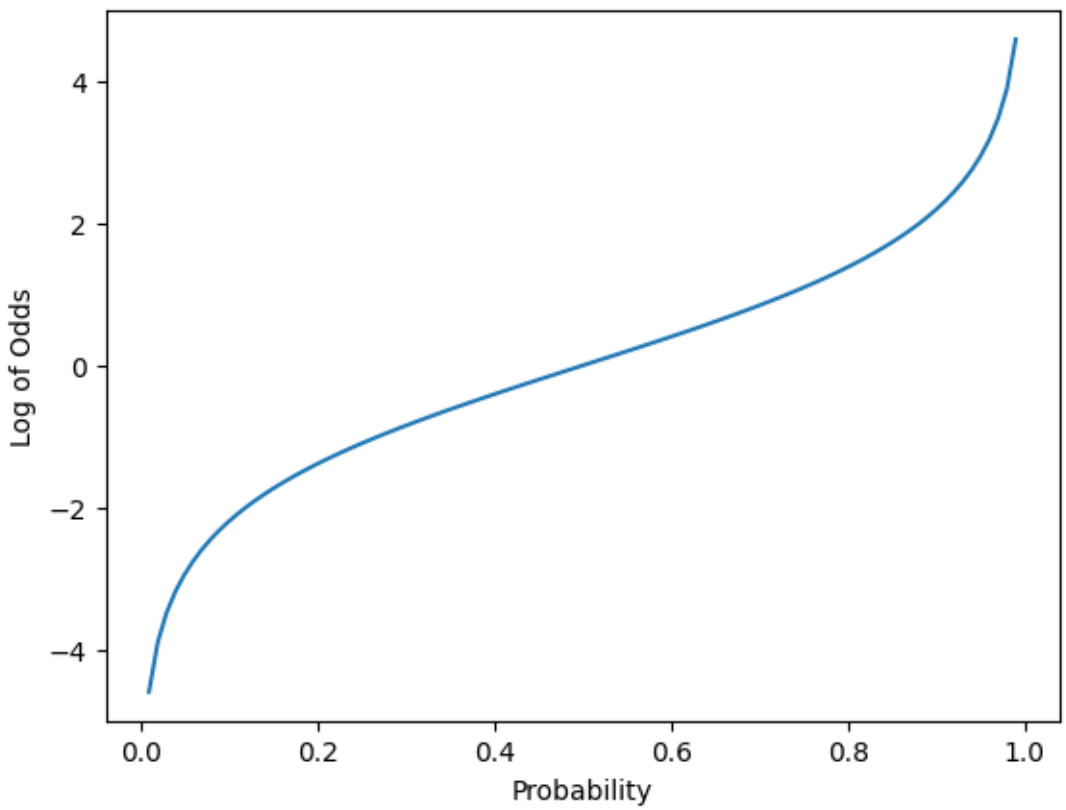
[***Key Points***]

- Proses transformasi: peluang - *odds* - *log of odds*.
- Mengubah peluang menjadi *odds*:
$$\frac{p}{1-p}$$
- Mengubah *odds* menjadi *log of odds*:
$$log(odds)$$
- Mengubah *log of odds* menjadi *odds*:
$$exp(log(odds))$$
- Mengubah *log of odds* menjadi peluang :
$$\frac{1}{(1+e^{(-log\ of\ odds)})}$$

Berikut merupakan plot antara peluang p dengan nilai *log of odds*-nya.

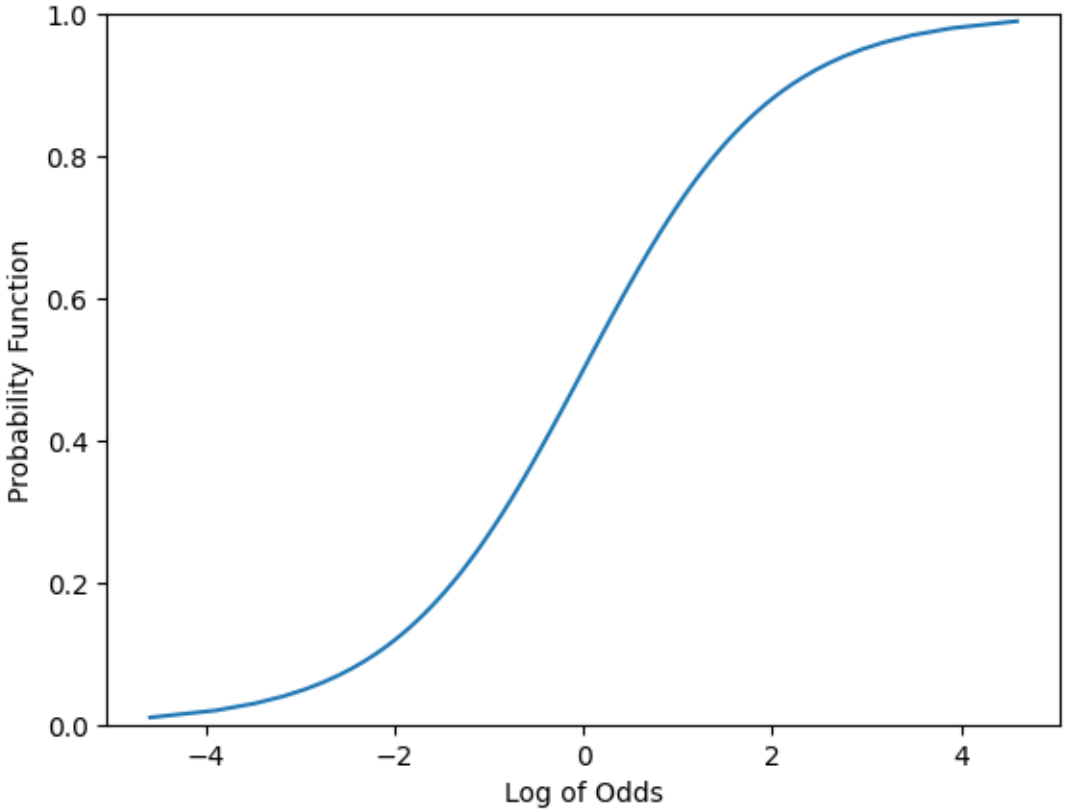
[27]:
```
x = np.linspace(0.01, 0.99, 100)
odds = x/(1-x)
y = np.log(odds)
```

```
plt.plot(x,y);
plt.ylabel('Log of Odds');
plt.xlabel('Probability');
plt.ylim(-5,5);
```



Pada plot di atas, kita melihat peluang p pada sumbu x dan *log of odds* pada sumbu y. Sekarang, kita lebih tertarik dengan nilai peluang. Bagaimana jika kita tukar posisi keduanya sehinga *log of odds* terletak pada sumbu x dan peluang p pada sumbu y?

[28]:
```
y = np.linspace(0.01, 0.99, 100)
odds = y/(1-y)
x = np.log(odds)

plt.plot(x,y);
plt.ylabel('Probability Function');
plt.xlabel('Log of Odds');
plt.ylim(0,1);
```



Ternyata kita mendapatkan grafik berbentuk huruf S. Plot di atas tidak lain merupakan plot dari fungsi sigmoid yang digunakan oleh model *logistic regression* untuk memprediksi peluang.

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Peluang inilah yang nantinya akan menjadi patokan apakah data kita termasuk dalam kelas 0 atau 1. Misalnya, dengan menetapkan *threshold* 0.5, apabila kita memiliki prediksi peluang sebesar 0.8, maka kita mengategorikan bahwa data termasuk ke dalam kelas 1.

30.3 Logistic Regression from First Principles

Meskipun namanya mengandung kata "*regression*", *logistic regression* tidak ditujukan untuk *case* regresi. *Nature* dari *logistic regression* yang melibatkan konsep peluang membuatnya lebih cocok untuk *case* klasifikasi. Apabila ditinjau lebih lanjut, *logistic regression* dapat dikatakan sebagai versi spesial dari regresi linear.

Recall kembali bahwa pada regresi linear, kita memprediksi variabel target yang kontinu. Hubungan antara prediktor x dan target y dapat dituliskan sebagai berikut.

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + ... + \beta_n \cdot x_n$$

Keterangan:

- n = banyak prediktor
- β_0 = *intercept*. *Intercept* menunjukkan nilai y saat semua prediktornya 0.
- β_1, β_2, ..., β_n = koefisien prediktor. Koefisien prediktor menunjukkan besarnya kenaikan variabel target saat variabel prediktor naik sebesar 1 satuan dengan catatan nilai semua prediktor lainnya tetap.

Dengan formulasi di atas, maka variabel target y untuk *case* regresi dapat memiliki nilai pada *range* `-Inf` sampai `+Inf`.

Apabila kita menggunakan formulasi yang sama untuk memprediksi peluang pada *case* klasifikasi, maka tidak akan sesuai. Ingat kembali bahwa peluang hanya memiliki *range* dari 0 sampai 1. Untuk menyiasati hal tersebut, kita perlu menggunakan "representasi lain/transformasi" dari peluang di mana "representasi lain/transformasi" tersebut memiliki nilai pada *range* `-Inf` sampai `+Inf`.

Masih ingat dengan *log of odds* yang kita bahas pada bagian sebelumnya? Kita akan menggunakan *log of odds* sebagai "representasi lain/transformasi" dari peluang. Dengan demikian, untuk *case* klasifikasi menggunakan *logistic regression*, formulasi sebelumnya dapat ditulis ulang sebagai berikut:

$$log(\frac{p}{1-p}) = logit(p) = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + ... + \beta_n \cdot x_n$$

Untuk mengubah *log of odds* menjadi peluang yang diinginkan, kita menggunakan fungsi sigmoid yang plot grafiknya sempat kita gambarkan pada bagian sebelumnya.

$$p(x) = Sigmoid(\beta_0 + \beta_1 \cdot x_1 + ... + \beta_n \cdot x_n) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot x_1 + ... + \beta_n \cdot x_n)}}$$

[***Key Points***]

- Pada *logistic regression*, kita memprediksi *log of odds* sebagai bentuk lain dari peluang.
- Fungsi sigmoid digunakan untuk *mapping log of odds* menjadi peluang.

30.4 Logistic Regression in Action

Pada bagian sebelumnya, kita telah mengetahui bahwa:

- *Logistic regression* mengeluarkan prediksi dalam bentuk *log of odds*.
- Untuk mengubah *log of odds* menjadi peluang, digunakan fungsi sigmoid.

Pada bagian ini, kita akan melihat bagaimana implementasi *logistic regression* pada Python menggunakan *package* `stats.model`. Untuk keperluan ini, kita coba untuk menggabungkan kembali `X_train` dan `y_train` yang sudah kita pisahkan sebelumnya.

[29]:
```
# menggabungkan variabel prediktor dan variabel target dalam satu tabel/
↪dataframe

train_data = X_train.copy()
train_data['fraud_bool'] = y_train
```

```
train_data = train_data.reset_index().drop(columns = "index")
```

[30]:
```
train_data.head()
```

[30]:
```
   income  name_email_similarity  prev_address_months_count
current_address_months_count  customer_age  days_since_request
intended_balcon_amount  zip_count_4w  velocity_6h  velocity_24h  velocity_4w
bank_branch_count_8w  date_of_birth_distinct_emails_4w  credit_risk_score
email_is_free  phone_home_valid  phone_mobile_valid  bank_months_count
has_other_cards  proposed_credit_limit  foreign_request
session_length_in_minutes  keep_alive_session  device_distinct_emails_8w  month
payment_type_AB  payment_type_AC  payment_type_AD  payment_type_AE
employment_status_CB  employment_status_CC  employment_status_CD
employment_status_CE  employment_status_CF  employment_status_CG
housing_status_BB  housing_status_BC  housing_status_BD  housing_status_BE
housing_status_BF  housing_status_BG  source_TELEAPP  device_os_macintosh
device_os_other  device_os_windows  device_os_x11  fraud_bool
0 0.50000                0.58414                        -1
354            60             0.02091                22.64003          1342
1097.47353    4135.31093   5993.67415                   547
4                142              1                 1                   1
28                1              200.00000                0
6.25370                    0                          1      0                0
0                0                0                    0                    1
0                    0                    0                    0
0                  0                  0                  0                  0
0               0                    0                0                  0
0           0
1 0.10000                0.88506                        -1
156            20             0.00458                14.54435           597
4891.02002    3214.06166   3144.47120                     1
6                223              1                 0                   1
28                0              990.00000                0
8.95264                    1                          1      7                0
0                0                0                    0                    0
0                    0                    1                    0
1                  0                  0                  0                  0
0               0                    0                1                  0
0           0
2 0.10000                0.69251                        -1
37            20             0.01531                -0.76549          2120
2480.21932    6969.44880   3710.56637                     0
7                128              1                 1                   1
-1                0              200.00000                1
9.67542                    1                          1      6                0
1                0                0                    0                    0
0                    0                    0                    0
```

```
0                  1                  0                  0                  0
0               0                  0                1                  0
0           0
3 0.90000                0.58853                        61
12            40             0.00384                15.55668           268
4593.44153    3050.17974   4256.22243                     46
11                108              1                 1                  1
21                1              500.00000                0
8.50664                   0                          1      6                0
0                0                0                    0                    0
0                    0                      1                      0
1                  0                  0                  0                  0
0               0                  0                0                  0
0           0
4 0.30000                0.81677                        -1
209            20             0.00454                -0.74999          1629
1620.07894    2234.76245   3111.30845                     12
6                130              1                 0                  1
1                0              200.00000                0
6.34498                   0                          1      7                1
0                0                0                    0                    0
0                    0                      0                      0
0                  0                  0                  1                  0
0               0                  0                0                  1
0           1
```

30.4.1 Logistic Regression with One Categorical (Discrete) Predictor

Untuk memahami bagaimana *logistic regression* bekerja, kita akan coba untuk menggunakan satu prediktor kategorikal untuk memprediksi variabel target kita. Untuk pembelajaran pada kasus ini, kita akan menggunakan:

- Prediktor: kolom `has_other_cards`.
- Target: kolom `fraud_bool`.

Dengan satu prediktor, yaitu kolom `has_other_cards`, maka formula untuk prediksi kita dapat ditulis:

$$logit(p) = \beta_0 + \beta_1 \cdot has_other_cards$$

Kita akan menggunakan fungsi `sm.Logit()` dari *package* `statsmodel` untuk melakukan *logistic regression*.

[31]:
```
train_data['intercept'] = 1

logit_model_tr = sm.Logit(train_data['fraud_bool'],␣
↪train_data[['intercept','has_other_cards']]).fit()
logit_model_tr.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.340175
         Iterations 7
```

[31]:

Dep. Variable:	fraud_bool	**No. Observations:**	80000
Model:	Logit	**Df Residuals:**	79998
Method:	MLE	**Df Model:**	1
Date:	Sun, 24 Dec 2023	**Pseudo R-squ.:**	0.02000
Time:	14:49:36	**Log-Likelihood:**	-27214.
converged:	True	**LL-Null:**	-27769.
Covariance Type:	nonrobust	**LLR p-value:**	1.672e-243

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
intercept	-1.9227	0.012	-161.481	0.000	-1.946	-1.899
has_other_cards	-1.1462	0.039	-29.021	0.000	-1.224	-1.069

Dengan hasil di atas, formula prediksi kita dapat ditulis ulang:

$$logit(p) = -1.9227 - 1.1462 \cdot has_other_cards$$

Mari kita bahas arti dari setiap koefisien yang telah kita dapatkan.

Intercept (β_0)

Intercept menunjukkan besarnya *log of odds* saat `has_other_cards = 0`.

Pada kasus kita maka *log of odds* yang didapatkan adalah -1.9227.

Untuk mendapatkan *odds* dan peluang dari *log of odds*:

[32]:
```
y_train.value_counts()
```

[32]:
```
fraud_bool
0    71177
1     8823
Name: count, dtype: int64
```

[33]:
```
# mendapatkan odds dan peluang dari log of odds = -1.7615

log_odds = -1.9227
odds = np.exp(log_odds)
p = 1 / (1 + np.exp(-log_odds)) # fungsi sigmoid

print(f'Log of odds akun bank fraud = {round(log_odds, 3)}.')
print(f'Odds akun bank fraud ketika tidak memiliki kartu lain = {round(odds,␣
↪3)}.')
print(f'Peluang akun bank fraud ketika tidak memiliki kartu lain = {round(p,␣
↪3)}.')
```

```
Log of odds akun bank fraud = -1.923.
Odds akun bank fraud ketika tidak memiliki kartu lain = 0.146.
Peluang akun bank fraud ketika tidak memiliki kartu lain = 0.128.
```

[34]:
```python
# crosscheck
pd.crosstab(train_data['fraud_bool'], train_data['has_other_cards'])
```

[34]:
```
has_other_cards      0      1
fraud_bool
0                55297  15880
1                 8085    738
```

[35]:
```python
# Peluang akun bank fraud ketika tidak memiliki kartu lain

round(8085/(55297 + 8085), 3)
```

[35]:
```
0.128
```

Saat memiliki kartu lain (`has_other_cards = 0`), peluang akun tersebut `Response = 1` adalah sebesar 12.8%.

Koefisien `has_other_cards` (β_1)

Koefisien `has_other_cards` menunjukkan *log* dari perbandingan *odds* akun merupakan fraud ketika tidak memiliki kartu lain dan ketika memiliki kartu lain.

$$Koefisien_{has_other_cards} = \log(\frac{odd(has_other_cards = 1, fraud_bool = 1)}{odd(has_other_cards = 0, fraud_bool = 1)})$$

Pada kasus kita, *log* perbandingan *odds* saat memiliki kartu lain dan saat tidak tidak memiliki kartu lain adalah -1.1462.

Dengan *log* perbandingan sebesar -1.1462, maka rasio *odds* yang sebenarnya adalah:

[36]:
```python
log_ratio = -1.1462
ratio = np.exp(log_ratio)

print(f'Perbandingan odds akun bank fraud jika akun memiliki kartu lain yaitu =␣
↪{round(ratio, 3)}.')
```

```
Perbandingan odds akun bank fraud jika akun memiliki kartu lain yaitu = 0.318.
```

Berdasarkan perbandingan *odds*-nya, *odds* untuk sebuah akun jika memliki kartu lain adalah (`has_other_cards = 1`) 31% lebih rendah daripada odds ketika akun tidak memiliki kartu lain (`has_other_cards = 0`).

Jika kita melihat kembali proporsi antara kolom `fraud_bool` dan `has_other_cards`:

[37]:
```python
# crosscheck: manual calculation

pd.crosstab(train_data['fraud_bool'], train_data['has_other_cards'])
```

[37]:
```
has_other_cards      0      1
fraud_bool
0                  55297  15880
1                   8085    738
```

- Untuk `has_other_cards = 0`
  - Peluang `fraud_bool = 0` = $\frac{55297}{(55297+8085)}$
  - Peluang `fraud_bool = 1` = $\frac{8085}{(55297+8085)}$
- Untuk `has_other_cards = 1`
  - Peluang `fraud_bool = 0` = $\frac{15880}{(15880+738)}$
  - Peluang `fraud_bool = 1` = $\frac{738}{(15880+738)}$
- Perhitungan *odds*
  - *Odds* `has_other_cards = 0` mendapatkan `fraud_bool = 1`

$$\frac{\frac{8085}{(55297+8085)}}{\frac{55297}{(55297+8085)}} = \frac{8085}{55297}$$

  - *Odds* `has_other_cards = 1` mendapatkan `fraud_bool = 1`

$$\frac{\frac{738}{(15880+738)}}{\frac{15880}{(15880+738)}} = \frac{738}{15880}$$

  - Perbandingan *odds*

$$\frac{\frac{738}{15880}}{\frac{8085}{55297}} = -1.146$$

[38]:
```python
# koefisien dari Treatment

odd_fraud1_card0 = (8085/55297)
odd_fraud1_card1 = (738/15880)

ratio_card0and1 = odd_fraud1_card0/odd_fraud1_card1

print(f'Rasio odds: {round(ratio_card0and1, 3)}')
print(f'Log rasio odds: {round(np.log(ratio_card0and1),3)}.')
```

```
Rasio odds: 3.146
Log rasio odds: 1.146.
```

Kita berhasil memverifikasi bahwa koefisien dari `has_other_cards` merupakan *log* dari perbandingan *odds* apakah akun bank fraud berdasarkan status kepemilikan kartu lain akun tersebut.

30.4.2 Logistic Regression with One Numerical (Continuous) Predictor

Sekarang, kita akan mencoba untuk menggunakan satu prediktor numerikal, yaitu `credit_risk_score`. Variabel targetnya masih menggunakan kolom `fraud_bool`. Dengan demikian, fungsi untuk melakukan prediksi dapat ditulis:

$$logit(p) = \beta_0 + \beta_1 \cdot credit_risk_score$$

[39]:
```
logit_model_crs = sm.Logit(train_data['fraud_bool'],␣
↪train_data[['intercept','credit_risk_score']]).fit()
logit_model_crs.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.326851
         Iterations 7
```

[39]:

Dep. Variable:	fraud_bool	**No. Observations:**	80000
Model:	Logit	**Df Residuals:**	79998
Method:	MLE	**Df Model:**	1
Date:	Sun, 24 Dec 2023	**Pseudo R-squ.:**	0.05838
Time:	14:49:37	**Log-Likelihood:**	-26148.
converged:	True	**LL-Null:**	-27769.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
intercept	-3.4334	0.029	-118.024	0.000	-3.490	-3.376
credit_risk_score	0.0088	0.000	55.977	0.000	0.008	0.009

Berdasarkan hasil di atas, kita dapat menulis ulang fungsi prediksi kita menjadi:

$$logit(p) = -3.4334 + 0.0088 \cdot credit_risk_score$$

Intercept (β_0)

Intercept menunjukkan besarnya *log of odds* saat `credit_risk_score = 0`.

Pada kasus kita, saat `credit_risk_score = 0`, maka besar *log of odds* akun bank tersebut fraud adalah -3.4334.

Untuk mendapatkan *odds* dan peluang dari *log of odds*:

[40]:
```
# mendapatkan odds dan peluang dari log of odds = -1.6828

log_odds = -3.4334
odds = np.exp(log_odds)
p = 1 / (1 + np.exp(-log_odds))

print(f'Log of odds akun bank fraud saat `credit_risk_score = 0` =␣
↪{round(log_odds, 3)}.')
print(f'Odds akun bank fraud saat `credit_risk_score = 0` = {round(odds, 3)}.')
print(f'Peluang akun bank fraud saat `credit_risk_score = 0` = {round(p, 3)}.')
```

```
Log of odds akun bank fraud saat `credit_risk_score = 0` = -3.433.
Odds akun bank fraud saat `credit_risk_score = 0` = 0.032.
Peluang akun bank fraud saat `credit_risk_score = 0` = 0.031.
```

Saat `credit_risk_score = 0`, peluang akun bank mendapatkan `fraud_bool = 1` adalah sebesar 3.1%.

Koefisien `credit_risk_score` (β_1)

Koefisien `credit_risk_score` menunjukkan kenaikan *log of odds* apabila nilai `credit_risk_score` naik sebesar 1 satuan.

Pada kasus kita, jika nilai `credit_risk_score` naik, maka nilai *log of odds* akan naik sebesar 0.0088.

Misalnya, nilai awal `credit_risk_score = 50`. Selanjutnya, nilai `credit_risk_score` berubah menjadi 51. Dengan demikian:

[41]:
```
# crosscheck

# log of odds saat credit_risk_score = 50
l1 = -3.4334 + 0.0088 * 50

# log of odds saat credit_risk_score = 51
l2 = -3.4334 + 0.0088 * 51

# selisih log of odds
print(f'Selisih log of odds = {round(l2-l1, 4)}.')
```

```
Selisih log of odds = 0.0088.
```

30.4.3 Logistic Regression with Multiple Predictor

Sejauh ini kita sudah mempelajari bagaimana cara mengartikan suatu koefisien jika prediktornya hanya satu kolom kategorikal saja atau satu kolom numerikal saja. Bagaimana jika kita ingin menggunakan banyak prediktor?

[42]:
```
train_data.phone_home_valid.unique()
```

[42]:
```
array([1, 0], dtype=int64)
```

[43]:
```
train_data.head(2)
```

[43]:
```
   income  name_email_similarity  prev_address_months_count
current_address_months_count  customer_age  days_since_request
intended_balcon_amount  zip_count_4w  velocity_6h  velocity_24h  velocity_4w
bank_branch_count_8w  date_of_birth_distinct_emails_4w  credit_risk_score
email_is_free  phone_home_valid  phone_mobile_valid  bank_months_count
has_other_cards  proposed_credit_limit  foreign_request
session_length_in_minutes  keep_alive_session  device_distinct_emails_8w  month
payment_type_AB  payment_type_AC  payment_type_AD  payment_type_AE
employment_status_CB  employment_status_CC  employment_status_CD
employment_status_CE  employment_status_CF  employment_status_CG
housing_status_BB  housing_status_BC  housing_status_BD  housing_status_BE
housing_status_BF  housing_status_BG  source_TELEAPP  device_os_macintosh
device_os_other  device_os_windows  device_os_x11  fraud_bool  intercept
0 0.50000                0.58414                          -1
```

```
354            60             0.02091                22.64003          1342
1097.47353    4135.31093   5993.67415                   547
4                142              1                 1                  1
28                1              200.00000                0
6.25370                    0                          1      0                0
0                0                0                    0                     1
0                     0                     0                     0
0                  0                  0                  0                  0
0               0                    0                 0                   0
0           0          1
1 0.10000                0.88506                        -1
156            20             0.00458                14.54435           597
4891.02002    3214.06166   3144.47120                    1
6                223              1                 0                  1
28                0              990.00000                0
8.95264                    1                          1      7                0
0                0                0                    0                     0
0                     0                     1                     0
1                  0                  0                  0                  0
0               0                    0                 1                   0
0           0          1
```

[44]:
```
model_logit_multi = sm.Logit(train_data['fraud_bool'],␣
↪train_data[['intercept','credit_risk_score','intended_balcon_amount',
                                                                    ␣
↪'has_other_cards', 'phone_home_valid']]).fit()
model_logit_multi.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.310968
         Iterations 7
```

[44]:

Dep. Variable:	fraud_bool	**No. Observations:**	80000
Model:	Logit	**Df Residuals:**	79995
Method:	MLE	**Df Model:**	4
Date:	Sun, 24 Dec 2023	**Pseudo R-squ.:**	0.1041
Time:	14:49:37	**Log-Likelihood:**	-24877.
converged:	True	**LL-Null:**	-27769.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
intercept	-3.0453	0.030	-100.579	0.000	-3.105	-2.986
credit_risk_score	0.0094	0.000	58.248	0.000	0.009	0.010
intended_balcon_amount	-0.0128	0.001	-16.504	0.000	-0.014	-0.011
has_other_cards	-1.2307	0.041	-30.332	0.000	-1.310	-1.151
phone_home_valid	-0.6579	0.026	-24.846	0.000	-0.710	-0.606

Dengan *multiple predictor*, fungsi logit untuk prediksi kita dapat ditulis:

$$logit(p) = -3.0453 + 0.0094 \cdot credit\ risk\ score - 0.0128 \cdot intended\ balcon\ amount - 1.2307 \cdot has\ other\ cards - 0.6579 \cdot ph$$

Cara kita menginterpretasi setiap koefisien masih sama. Hanya saja terdapat sedikit tambahan saat menginterpretasi koefiesien prediktor.

- *Intercept* menunjukkan *log of odds* pelanggan membeli produk saat semua prediktor = 0.
- Koefisien prediktor menunjukkan besarnya kenaikan *log of odds* saat semua prediktor lain nilainya tetap.

Sebagai contoh pada `intended_balcon_amount`:

[45]:
```
train_data.intended_balcon_amount.describe()
```

[45]:
```
count   80000.00000
mean        8.05265
std        19.76626
min       -14.06498
25%        -1.18521
50%        -0.84004
75%        -0.19351
max       111.69735
Name: intended_balcon_amount, dtype: float64
```

[46]:
```
crs = 50
balcon0 = 0.48
balcon1 = 1.48
cards = 0
phone = 1

# log of odds awal
l1 = -3.0453 + 0.0094*crs - 0.0128*balcon0 - 1.2307*cards - 0.6579*phone

# log of odds setelah var_2 bertambah 1 unit
l2 = -3.0453 + 0.0094*crs - 0.0128*balcon1 - 1.2307*cards - 0.6579*phone

# selisih
diff = l2-l1

print(f"Selisih log of odds setelah `intended_balcon_amount` naik sebesar 1␣
↪unit = {round(diff, 3)}.")
```

```
Selisih log of odds setelah `intended_balcon_amount` naik sebesar 1 unit =
-0.013.
```

Jika nilai variabel lain tidak berubah, maka dengan kenaikan `intended_balcon_amount` sebesar 1 unit, maka nilai *log of odds* akan berkurang sebesar 0.013.

30.5 Prediksi

Ketika kita sudah berhasil membuat model, normalnya kita akan mencoba melakukan prediksi terhadap data *test* yang sudah kita persiapkan pada tahap *train-test-splitting*. Dalam melakukan prediksi, kita bisa memanfaaatkan fungsi `predict()`. Dengan syntax sebagai berikut:

```
<nama_model>.predict(<var_prediktor>)
```

[47]:
```
X_test["intercept"] = 1

pred_credit = model_logit_multi.
 ↪predict(X_test[['intercept','credit_risk_score','intended_balcon_amount',
                                                 'has_other_cards',␣
 ↪'phone_home_valid']])

# ubah hasil prediksi dari peluang ke label
pred_label = pred_credit.apply(lambda x: 1 if x > 0.5 else 0)
pred_label.head()
```

[47]:
```
64955    0
53224    0
31080    0
91843    0
92873    0
dtype: int64
```

30.6 Evaluasi Model

Untuk mengetahui performa suatu model, kita perlu meninjau hasil prediksi model kita pada data test. Pada kasus ini kita menggunakan satu buah konsep terkait evaluasi kasus klasifikasi yaitu *confusion matrix*.

30.6.1 Confusion Matrix

Pada *confusion matrix* langkah awal yang perlu dilakukan adalah menentukan kelas positif. Ketika kita memiliki kasus klasifikasi, kelas positif adalah kelas yang lebih kita fokuskan atau secara sederhana yang ingin kita tinjau lebih dalam.

Pada kode berikut, kita akan meninjau hasil *confusion matrix* pada model kita:

[48]:
```
# buat confusion matrix
pd.crosstab(pred_label, y_test,
            rownames=["predicted label"],
            colnames=["true label"])
```

[48]:
```
true label           0     1
predicted label
0                17766  2174
1                   28    32
```

Pada data ini, kita akan berfokus kepada kelas 1 yang merupakan akun bank fraud. Perhatikan pada matrix di atas kita memprediksi data lalu membandingkannya dengan data aslinya yang berkelas 0 dan 1. Setiap pengukuran dari matrix ini sebenarnya memiliki nama, yaitu:

- *True Positive* (TP): banyaknya label positif yang diprediksi dengan benar (aktual +, prediksi +)
- *True Negative* (TN): banyaknya label negatif yang diprediksi dengan benar (aktual -, prediksi -)
- *False Positive* (FP): banyaknya label negatif yang diprediksi dengan salah (aktual -, prediksi +)
- *False Negative* (FN): banyaknya label negatif yang diprediksi dengan salah (aktual +, prediksi -)

		AKTUAL	
		0	1
PREDIKSI	0	TN	FN
	1	FP	TP

Dari keempat pengukuran tersebut, kita dapat menilai evaluasi model berdasarkan 4 metrik berikut:

- **Accuracy**: seberapa tepat model kita memprediksi kelas target (secara global).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Sensitivity/ Recall**: ukuran kebaikan model terhadap kelas `positif`.

$$Recall = \frac{TP}{TP + FN}$$

- Pos Pred Value/**Precision**: seberapa presisi model memprediksi kelas positif.

$$Precision = \frac{TP}{TP + FP}$$

- **Specificity**: ukuran kebaikan model terhadap kelas `negatif` (jarang digunakan karena merupakan kebalikan dari *recall*).

30.6.2 Accuracy

Kita dapat menghitung nilai akurasi dengan fungsi `accuracy_score()` dari `sklearn`.

[49]:
```
accuracy_score(y_test, pred_label)
```

[49]: 0.8899

30.6.3 Precision

Kita dapat menghitung nilai *precision* dengan fungsi `precision_score()` dari `sklearn`.

[50]:
```
precision_score(y_test, pred_label)
```

[50]: 0.5333333333333333

30.6.4 Recall/Sensitivity

Kita dapat menghitung nilai *recall* dengan fungsi `recall_score()` dari `sklearn`.

[51]:
```
recall_score(y_test, pred_label)
```

[51]: 0.014505893019038985

30.6.5 Specificity/True Negative Rate (TNR)

Kita dapat menghitung nilai *specificity* dengan fungsi `recall_score()` dan memasukkan parameter `pos_label = 0`.

[52]:
```
recall_score(y_test, pred_label, pos_label=0)
```

[52]: 0.998426435877262

Secara umum sebenarnya kita dapat menggunakan metrik *accuracy*, tetapi terdapat beberapa kondisi ketika kita harus memilih untuk meninjau ***recall/precision***:

- *Data imbalance* (proporsi target variabel tidak seimbang)
- Ada kelas yang lebih penting
- Ada resiko yang perlu dipertimbangkan

Tanpa ketiga syarat di atas, kita bisa dengan mudah meninjau nilai akurasi saja. Tetapi contohnya pada kasus kita sekarang, terdapat kelas yang lebih penting yaitu kelas 1 yang merupakan kondisi akun bank fraud. Ketika kita telah mengetahui kelas positifnya maka kita bisa meninjau resiko apa yang terjadi jika kita melakukan salah prediksi:

- FP: diprediksi fraud ternyata bukan fraud
  - resikonya adalah kehilangan nasabah yang mungkin menggunakan berbagai program bank.
- FN: diprediksi bukan fraud ternyata fraud
  - resikonya adalah kehilangan dana untuk menanggulangi kerugian akibat akun fraud ini.

Berdasarkan penjabaran di atas, kita sendiri yang akan menentukan resiko mana yang sebaiknya kita minimalisir. Apabila kita ingin meminimalisir FP maka kita akan menggunakan metrik ***precision***, sementara ketika kita ingin meminimalisir FN maka kita akan menggunakan metrik ***recall***.

30.7 Logistic Regression Key Assumptions

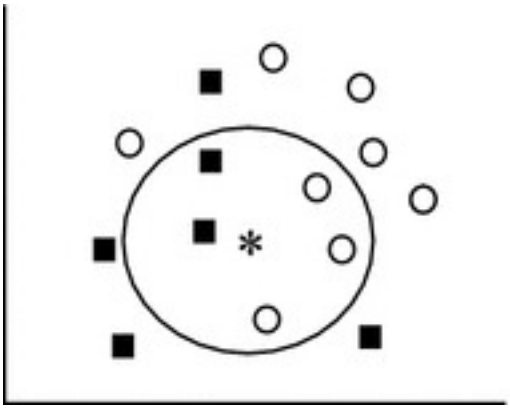
Seperti yang sudah kita pelajari di awal pembahasan mengenai *logistic regression*, *logistic regression* dapat dipandang sebagai *special case* dari regresi linear. Jika regresi linear memiliki asumsi yang harus dipenuhi, *logistic regression* juga memiliki beberapa asumsi:

- **Multikolinearitas**
  - Artinya, setiap prediktor tidak saling berkorelasi tinggi.
- **Obervasi yang Independen**
  - Artinya, setiap observasi tidak berasal dari pengukuran yang berulang dan harus independen satu sama lain.
- **Linearitas antara Prediktor dan *Log of Odds***
  - Artinya, terdapat hubungan linear antara prediktor dan *log of odds*.

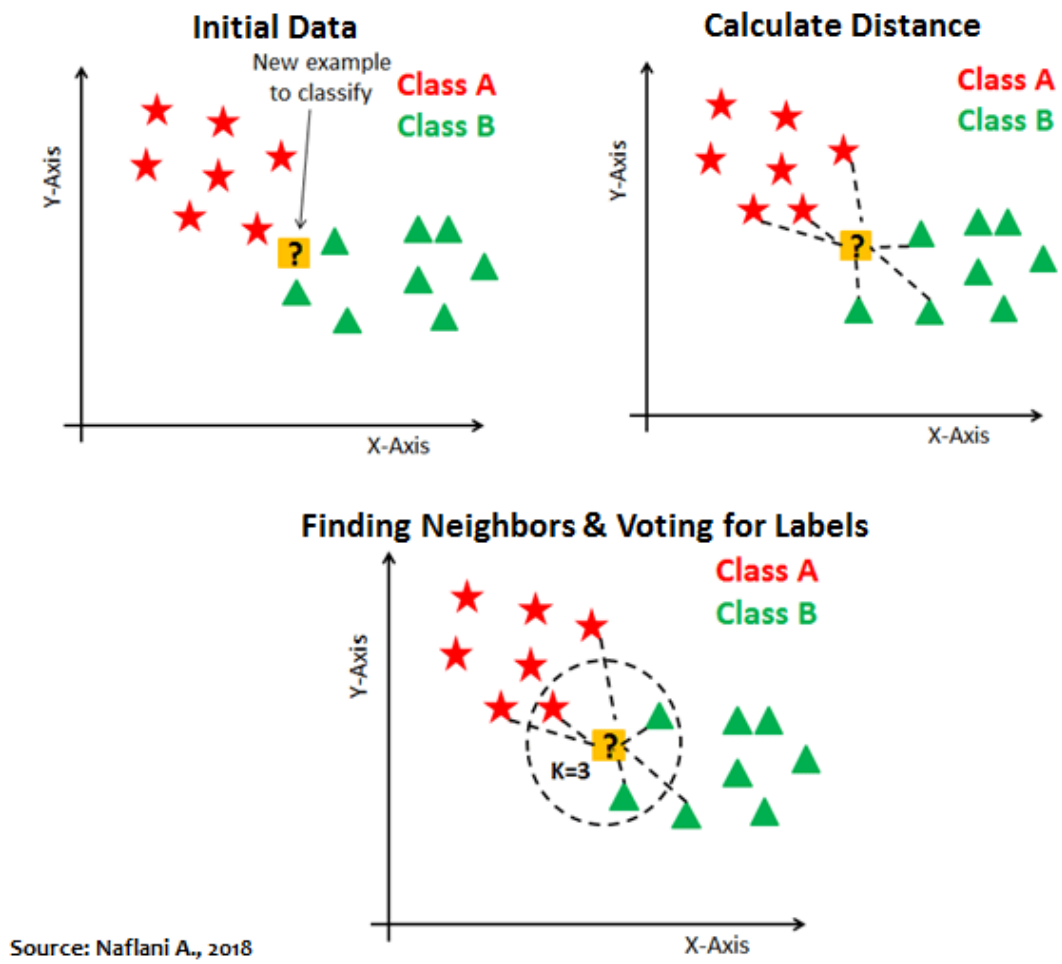
31 Model Classification 2 - K-Nearest Neighbour

Algoritma K-Nearest Neighbor merupakan metode yang menggunakan informasi `k` tetangga terdekat untuk mengklasifikasi data tanpa label. Ketika dilatih, model ini akan mempelajari informasi lokasi suatu data dan label data tersebut. Informasi lokasi dan label ini yang akan menjadi acuan ketika data tanpa label diidentifikasi. Dari jumlah tetangga terdekat suatu data tanpa label, kelas dominan yang akan dipilih sebagai kelas data tersebut.

Kita akan menggunakan contoh gambar di bawah sebagai ilustrasi. Anggaplah kita memilih nilai k = 1 maka tanda * akan menghitung lokasi ke setiap titik di sekitarnya dan mengambil 1 tetangga paling dekat sebagai kelas dirinya. Pada kasus ini artinya dia akan diklasifikasikan sebagai kelas kotak.



Tetapi apabila kita memilih k = 5, maka kelas dominan dari lima poin di sekitar tanda * ini adalah tanda bulat. Secara lebih jelasnya, berikut adalah proses pengambilan keputusan oleh k-NN.



31.1 Euclidean Distance

Metode k-NN ini menggunakan perhitungan jarak yang disebut *Euclidean distance* yaitu menghitung jarak paling dekat (seperti menggunakan penggaris untuk menghubungkan dua titik). Apabila kita memiliki dua buah data bernama A dan B dengan fitur seperti berikut:

A $= (x_1, x_2, ..., x_m)$

B $= (y_1, y_2, ..., y_m)$

Pada keterangan di atas, m merupakan dimensi fitur data kita. Untuk melakukan perhitungan jarak antara A dan B, formula *Euclidean distance* yang digunakan yaitu:

$$dist(A, B) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Jika kita aplikasikan formula di atas pada contoh *blind-tasting* suatu makanan, kita dapat menghitung jarak antara kedua makanan di bawah dengan fiturnya masing-masing sebagai berikut:

- tomato (sweet: 6, crunchy: 4)

- green bean (sweet: 3, crunchy: 7)

dist(tomato, green bean) = `sqrt((6-3)^2 + (4-7)^2))` yang menghasilkan angka 4.24.

31.2 Data Preparation

Pada kasus ini kita akan kembali menggunakan data credit kita sebelumnya. Pada metode k-NN karena kita memperhitungkan nilai berdasarkan jarak maka kita tidak akan menggunakan data kategorikal sama sekali. Mari kita lihat kembali data kita lalu kita ambil data numeriknya saja.

[53]:
```
credit_knn = credit.select_dtypes("number")
credit_knn.head(3)
```

[53]:
```
   fraud_bool  income  name_email_similarity  prev_address_months_count
current_address_months_count  customer_age  days_since_request
intended_balcon_amount  zip_count_4w  velocity_6h  velocity_24h  velocity_4w
bank_branch_count_8w  date_of_birth_distinct_emails_4w  credit_risk_score
email_is_free  phone_home_valid  phone_mobile_valid  bank_months_count
has_other_cards  proposed_credit_limit  foreign_request
session_length_in_minutes  keep_alive_session  device_distinct_emails_8w  month
0           0 0.90000                0.61579                          -1
51            40             0.01034                -0.69955          2659
11007.70883    7448.79742   5576.30356                  2030
7                253              1                 1                   1
9                 1             1500.00000                0
13.07981                    0                          1      1
1           0 0.60000                0.07246                          -1
125            20             0.00697                -0.95947          5599
6304.26284    4373.09359   4721.10002                     1
13                112              1                 1                   0
-1                 0              200.00000                0
6.66655                    0                          1      4
2           1 0.50000                0.07659                          -1
37            50             0.00632                -1.21993          1352
1357.41355    6667.02966   6318.87626                    15
6                199              1                 1                   1
28                 0             1500.00000                0
4.49267                    1                          2      0
```

Mari kita lanjutkan proses pembuatan model ini dengan membagi data kita menjadi data train dan data test.

[54]:
```
y = credit_knn["fraud_bool"]
X = credit_knn.drop(columns=["fraud_bool"])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,␣
 ↪random_state=42, stratify=y)
```

31.2.1 Features rescaling

Jika kita lihat di atas, data kita tersusun dari berbagai data dengan *range* nilai yang berbeda-beda. Hal ini akan menjadi masalah ketika kita menggunakan metode k-NN yang menghitung nilai berdasarkan jarak. Variabel dengan nilai di antara 1 sampai 500 akan jauh lebih signifikan dibanding variabel dengan nilai 0 sampai 1.

Kondisi ini dapat diselesaikan dengan melakukan *rescaling* keseluruhan variabel kita. Dengan *rescaling* kita dapat memampatkan atau melebarkan nilai data menjadi *range* yang kurang lebih sama. Beberapa metode scaling yang biasa digunakan yaitu:

- Mix-Max normalization
- z-score standardization

Min-max normalization Metode ini bekerja dengan melakukan transformasi nilai data agar berada di jarak 0 sampai 1. Formula yang digunakan:

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

z-score standardization Metode ini bekerja dengan melakukan transformasi berdasarkan informasi mean dan standar deviasi dari data. Formula yang digunakan:

$$x_{new} = \frac{(x - \bar{x})}{std(x)}$$

Metode z-score ini akan menghasilkan nilai dalam skala z-score yang tidak memiliki batas maksimum atau minimum dan dapat bernilai negatif maupun positif.

Pada kasus ini, kita akan menggunakan metode z-score pada data kita. Metode ini dapat diakses menggunakan fungsi `StandardScaler()` dari `sklearn.preprocessing`.

Kita akan mempelajari informasi dari data train dan scaling pada data test akan dilakukan berdasarkan informasi data train untuk mencegah *data leaking*.

[55]:
```
scaler = StandardScaler()

scaler.fit(X_train)

X_train_scale = scaler.transform(X_train)
X_test_scale = scaler.transform(X_test)
```

31.3 Modeling

31.3.1 Choosing an appropriate *k*

Penentuan berapa tetangga yang diperhitungkan oleh k-NN akan berdampak kepada seberapa mampu model men-generalisasi data. Keseimbangan antara overfitting dan underfitting data training umumnya dikenal sebagai **bias-variance tradeoff**. Penggunaan nilai k yang besar mengurangi

pengaruh variasi pada *noisy data*, tetapi dapat membuat pembelajaran model menjadi bias yang membuat model menghindari pola penting yang ditangkap dari informasi kecil/detail.

Jika kita menggunakan nilai k yang sangat besar, seperti sejumlah observasi pada data training kita. Hal ini akan menyebabkan model selalu memprediksi kelas dominan. Sebaliknya, jika kita menggunakan nilai k = 1 menyebabkan data outliers berpengaruh tidak semestinya terhadap klasifikasi. Jika salah satu data training secara tidak sengaja salah dilabeli dan terdapat satu data yang tepat di sebelahnya, pemilihan k = 1 akan menyebabkan misklasifikasi walaupun 10 data tetangga lain memberi vote kelas lain.

Pada prakteknya, salah satu strategi untuk menentukan nilai k yaitu dengan **menghitung akar kuadrat dari jumlah data training**. Strategi lain adalah menggunakan nilai k yang lebih besar tetapi memperhitungkan bobot berdasarkan jarak antar data. Jarak data yang lebih dekat berbobot lebih tinggi dibanding jarak data yang lebih jauh.

Selain itu pembulatan dari perhitungan akar kuadrat ini disesuaikan dengan beberapa syarat lain yaitu:

- k harus ganjil bila jumlah kelas target genap
- k harus genap bila jumlah kelas target ganjil
- k tidak boleh angka kelipatan jumlah kelas target

Hal ini dilakukan untuk menghindari seri ketika proses majority voting dilakukan. Apabila hasil majority voting ini seri maka kelas akan dipilih secara random. Mari kita hitung nilai k pada kasus kita sebelum membuat model.

[56]:
```
np.sqrt(X_train.shape[0])
```

[56]:
```
282.842712474619
```

Berdasarkan perhitungan di atas maka kita akan menggunakan nilai `k = 283`.

[57]:
```
model_knn = KNeighborsClassifier(n_neighbors = 283)
model_knn.fit(X_train_scale, y_train)
```

[57]:
```
KNeighborsClassifier(n_neighbors=283)
```

31.3.2 Characteristics of k-NN

Metode klasifikasi menggunakan k-NN biasa disebut sebagai '*lazy learners*' karena kita tidak benar-benar membuat model. Pada proses k-NN tidak terjadi proses generalisasi seperti pada model logistic regression. Secara teknikal dapat dikatakan bahwa k-NN hanya menghafal lokasi data tidak benar-benar mempelajari datanya.

Mari kita simpulkan proses yang perlu dilakukan k-NN hingga mendapatkan prediksi:

- Scaling (transformasi variabel agar berada pada *range* nilai yang kurang lebih sama agar tidak ada yang lebih dominan)
- Memilih nilai k
- Membuat model (memilih k tetangga terdekat untuk setiap sampel uji)
- Melakukan klasifikasi berdasarkan kelas dominannya

Karena berbagai alasan di atas, maka secara komputasi proses ini cukup berat apabila kita memiliki data dengan dimensi yang besar (membutuhkan memori yang besar karena melakukan perhitungan jarak secara berulang). Jika kita memilih k yang kecil, metode ini akan sensitif terhadap *noise* di data kita.

Meskipun terdapat berbagai limitasi, k-NN sangatlah *powerful* dan *versatile*. Beberapa kelemahan seperti kondisi outlier dapat diselesaikan dengan scaling yang sudah dipelajari. k-NN juga akan menjadi tidak sensitif terhadap outlier dan noise apabila kita memilih nilai k dengan tepat. Selain itu k-NN mampu bekerja baik pada data non-linear tidak seperti metode logistic regression.

31.4 Prediction

Mari kita kembali ke model kita dan coba melakukan prediksi pada data test kita.

[58]:
```
knn_pred = model_knn.predict(X_test_scale)
```

31.5 Evaluation

Seperti model sebelumnya, kita tetap harus meninjau performa model pada data test agar kita dapat menentukan apakah model ini dapat digunakan atau tidak. Mari kita lihat confusion matrix model kita pada data test.

[59]:
```
pd.crosstab(knn_pred, y_test,
            rownames=["predicted label"],
            colnames=["true label"])
```

[59]:
```
true label           0     1
predicted label
0                17750  2065
1                   44   141
```

Mari kita juga tinjau ketiga metrik evaluasi yaitu akurasi, recall, dan precision.

[60]:
```
print("Akurasi: ", accuracy_score(y_test, knn_pred))
print("Recall: ", recall_score(y_test, knn_pred, pos_label=1))
print("Precision: ", precision_score(y_test, knn_pred, pos_label=1))
```

```
Akurasi:  0.89455
Recall:  0.06391659111514053
Precision:  0.7621621621621621
```

32 Modeling Summary

Secara kesimpulan, sebenarnya kedua model yang telah kita buat belum baik karena metriks utama kita (recall) masih menghasilkan nilai yang sangat kecil. Artinya kita masih perlu melakukan eksplorasi terkait penanganan data maupun tuning model. Hal ini utamanya karena pada kasus ini kita berhadapan dengan data yang imbalance.

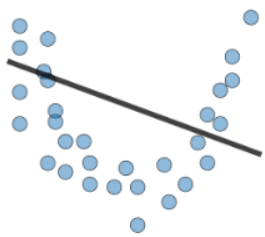
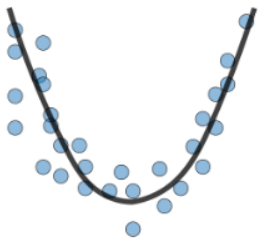
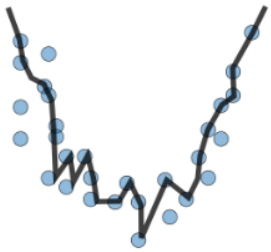
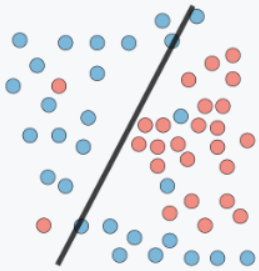
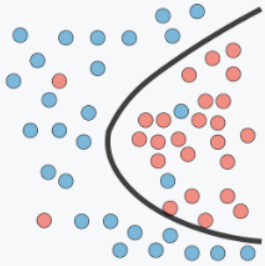
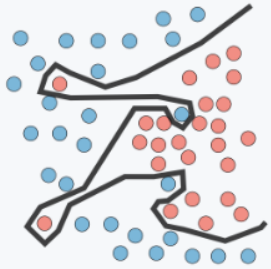
Seperti yang telah disebutkan sebelumnya, terdapat metode seperti oversampling dan undersampling yang mampu mengubah komposisi data kita. Selain itu kita juga dapat melakukan pemilihan

fitur yang digunakan pada model, melakukan *hyperparameter tuning*, dsb.

Berbagai metode di atas tidak akan dibahas di kelas ini dan dapat dieksplorasi mandiri di luar kelas.

32.1 [Additional] Classification Model Selection: Overfit, Underfit, Just Right

Pada bagian ini kita akan membahas istilah *overfitting* yang mengacu pada kondisi di mana model terlalu mempelajari pola pada data sehingga gagal memprediksi data baru. Perhatikan gambar di bawah ini!

	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			

Source: stanford.edu - Machine Learning tips and tricks cheatsheet (A. Amidi & S. Amidi)

32.1.1 Overfitting (High Variance)

Seperti yang sudah kita ketahui, bahwa *overfitting* merupakan kondisi di mana model sangat menghafal pola pada data *training*. Bayangkan bahwa data training memiliki *signal*/pola dan *noise*.

> Idealnya, kita hanya menginginkan model hanya mempelajari *signal*/pola.

Yang terjadi pada *overfitting*, model juga mempelajari *noise* pada data. Akibatnya, jika diberikan data baru, model gagal melakukan prediksi (gagal melakukan generalisasi).

Gejala *overfitting* dapat diketahui dari hal-hal berikut.

1. Model menunjukkan performa yang baik pada data *training* (tingkat *error* kecil).
2. Model menunjukkan performa yang rendah pada data *testing* (tingkat *error* besar). Perbedaan performa antara *training* dan *testing* sangat besar.

Penyebab *overfitting*:

1. Ukuran untuk data *training* sangat kecil dan tidak representatif.
2. Data yang digunakan untuk *training* banyak mengandung *noise*.
3. Prediktor yang digunakan pada data *training* tidak sesuai.
4. Model yang terlalu kompleks.

Cara mengatasi *overfitting* yang dapat dilakukan:

1. *Adjustment* dari segi model:
   - Menggunakan model yang lebih sederhana.
   - Melakukan *fine-tuning* model:
     - Melakukan *pruning/regularization*.
     - Melakukan *hyperparameter search*.
2. *Adjustment* dari segi data:
   - Menambah data yang digunakan untuk proses *training*.
   - Melakukan *feature selection*.
   - Melakukan *handling* untuk *imbalance dataset*.
3. *Adjustment* dari cara *training*:
   - Melakukan *cross validation*.

32.1.2 Underfitting/High Bias

Berkebalikan dengan *overfitting*, pada *underfitting*, model tidak dapat mendeteksi pola pada data. Akibatnya, model gagal untuk memprediksi data baru.

Gejala *underfitting* dapat diketahui dari hal berikut.

1. Model menunjukkan performa yang rendah pada data *training* (tingkat *error* besar).
2. Model menunjukkan performa yang rendah pada data *testing* (tingkat *error* besar). Perbedaan performa antara *training* dan *testing* tidak terlalu jauh.

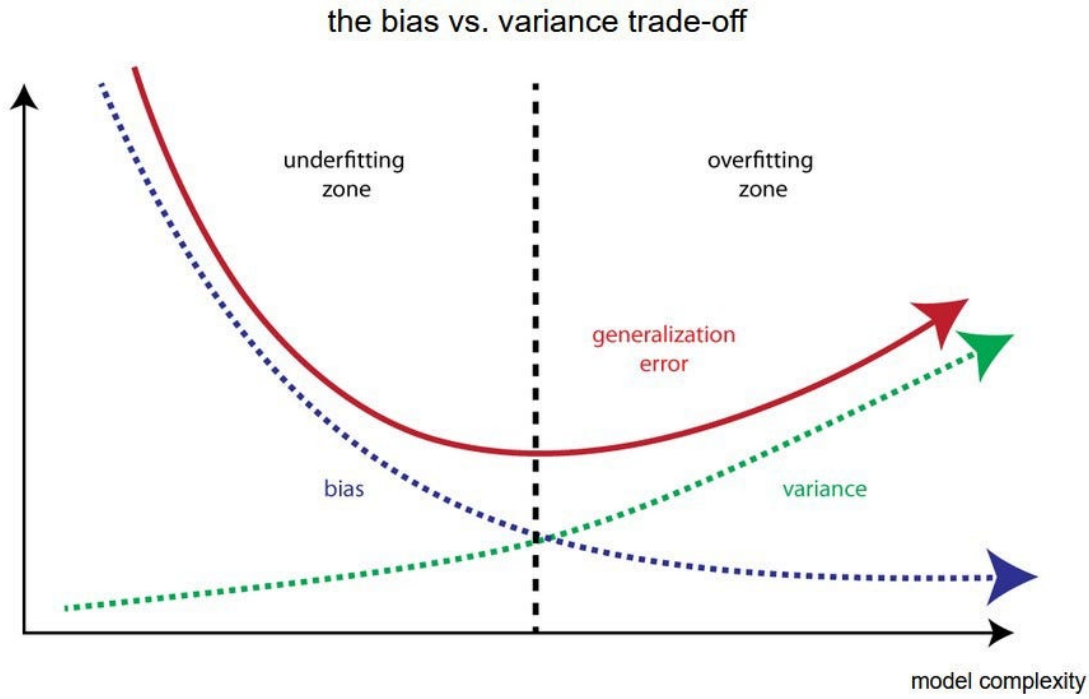
Penyebab *underfitting*:

1. Ukuran untuk data *training* sangat kecil dan tidak representatif.
2. Prediktor yang digunakan pada data *training* tidak sesuai.
3. Model yang terlalu sederhana.

Cara mengatasi *underfitting* yang dapat dilakukan:

1. *Adjustment* dari segi model:
   - Menggunakan model yang lebih kompleks.
   - Melakukan *fine-tuning* model.
2. *Adjustment* dari segi data:
   - Menambah data yang digunakan untuk proses *training*.
   - Mencoba mengganti prediktor yang dimasukkan ke data *training* (*feature selection*).
3. Adjustment dari cara *training*:
   - Melakukan *cross validation*.

32.1.3 Trade-Off Underfitting and Overfitting (Bias and Variance Trade-Off)

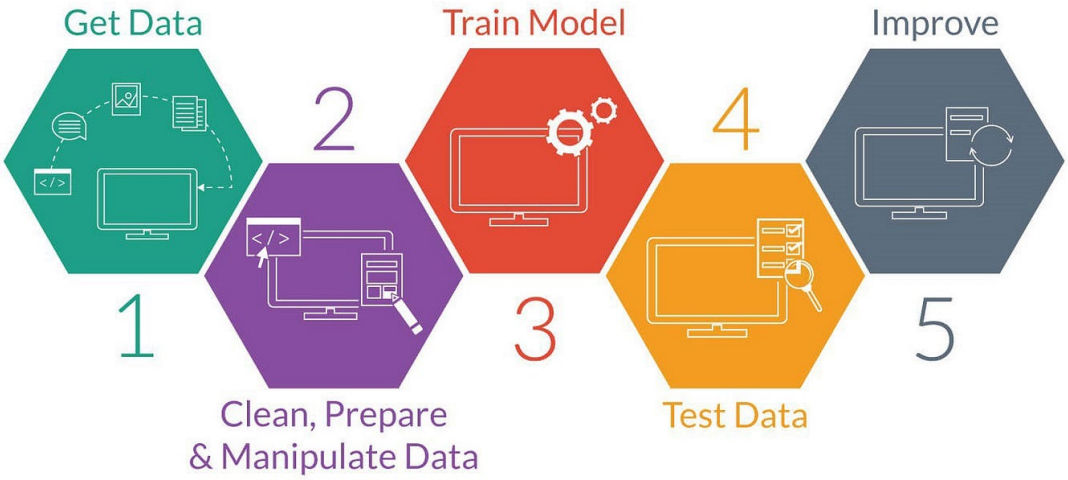


Challenge utama saat membangun *machine learning* adalah menghasilkan model yang merupakan perpotongan dari *underfitting* dan *overfitting*. Artinya, kita ingin membangun model yang tidak terlalu kompleks (tidak *overfit*), tetapi bisa menghasilkan prediksi dengan benar (tidak *underfit*).

> Model ideal saat melakukan *machine learning*: memiliki *variance* dan bias yang rendah (tingkat *overfitting* dan *underfitting* rendah).

33 Wrap Up

Pada *course* ini, kita sudah mempelajari banyak hal untuk membuat *machine learning*. Apabila dirunut kembali, apa yang sudah kita lakukan sejauh ini sebenarnya mengikuti *workflow machine learning* secara umum:



1. Mendapatkan data
2. Persiapan data: cek tipe data, cek *missing values*, *feature selection*, *target and predictors splitting*, *one-hot encoding*, *cross validation*, *sampling*, dll.
3. *Modeling*: membuat dan melatih model *machine learning*.
4. Evaluasi performa model: menggunakan *metrics evaluation*.

Proses ini biasanya tidak berjalan satu arah. Terkadang kita sering harus kembali ke *step* sebelumnya. Dalam hal ini, dari *step* 4, kadang kita bisa mengulang lagi dari *step* 2 atau bahkan *step* 1.

> Proses *machine learning* sering kali melibatkan serangkaian percobaan dan kesalahan. Pendekatan yang digunakan dalam menyelesaikan suatu masalah tertentu tidak selalu dapat diterapkan secara langsung untuk menyelesaikan masalah lain yang berbeda.

Coursebook: Unsupervised Learning

- Bagian 6 Audit Analytics untuk Bank Rakyat Indonesia
- Durasi: 7 Jam
- *Last Updated*: December 2023

- *Coursebook* ini disusun dan dikurasi oleh tim produk dan instruktur dari Algoritma Data Science School

34 Background

Coursebook ini merupakan bagian dari **BRI Audit Analytics** yang disiapkan oleh Algoritma. *Coursebook* ini ditujukan hanya untuk khalayak terbatas, yaitu individu dan organisasi yang menerima *coursebook* ini langsung dari organisasi pelatihan. Tidak boleh direproduksi, didistribusikan, diterjemahkan, atau diadaptasi dalam bentuk apapun di luar individu dan organisasi ini tanpa izin.

Algoritma adalah pusat pendidikan *data science* yang berbasis di Jakarta. Kami menyelenggarakan *workshop* dan program pelatihan untuk membantu para profesional dan mahasiswa dalam menguasai berbagai sub-bidang *data science* yaitu: *data visualization, machine learning,* statistik, dan lain sebagainya.

34.1 Training Objectives

Dalam lokakarya ini kami akan memfokuskan studi kami pada serangkaian metode pembelajaran tanpa pengawasan yang banyak digunakan mulai dari PCA (Principal Component Analysis), hingga Clustering, dan pendekatan penemuan pola lainnya di mana variabel target tidak diketahui atau ditentukan. Tujuan kami adalah mengembangkan intuisi yang kuat di balik masalah dimensi, mekanisme yang kami miliki, dan pada akhirnya memperkuat pemahaman kami dengan mengerjakan dua skenario bisnis kehidupan nyata yang paling umum.

Buku kursus ini ditujukan bagi peserta yang telah menyelesaikan kursus sebelumnya yang ditawarkan dalam **Spesialisasi Ilmu Data dalam Python**. Ini adalah kursus kesembilan, **Pembelajaran Tanpa Pengawasan**.

- **Data Preprocessing**
  - Principles and Motivation
  - Variance and matrix covariance
  - Eigenvalues and eigenvector
  - Principal Component Analysis (PCA)
- **Exploratory Data Analysis**
  - Biplot visualization
    - Individual factor map
    - Variable factor map
- **Anomaly Detection**
  - Understanding Local Outlier Factor (LOF)
  - Practicing Local Outlier Factor (LOF)

Bagian terakhir dari kursus ini adalah Tugas Bertingkat, di mana Anda diharapkan menerapkan semua yang telah Anda pelajari pada kumpulan data baru, dan mencoba menjawab pertanyaan yang diberikan.

35 Unsupervised Learning

Sepanjang Spesialisasi Machine Learning, kita telah mempelajari tentang algoritma yang sangat berguna dalam situasi regresi dan klasifikasi. Secara lebih umum, kita belajar mencari parameter untuk X1, X2…Xn untuk menjelaskan atau memprediksi respons "target" Y.

Dalam kasus pembelajaran tanpa pengawasan (unsupervised learning), situasinya berbeda karena tidak ada respons Y, melainkan kami tertarik untuk menemukan struktur antara X1, X2, hingga Xn - mungkin untuk mengidentifikasi peluang pengurangan dimensi atau pengelompokan. Beberapa orang menyamakan pembelajaran tanpa pengawasan dengan proses eksplorasi karena sulit atau tidak mungkin untuk mengetahui apakah model atau formulasi apa pun adalah yang "benar" karena kita tidak memiliki "kebenaran dasar" yang dapat kita gunakan sebagai tolok ukur. Teknik seperti validasi silang dan AUC tidak berlaku karena kurangnya label "kebenaran dasar".

Meskipun demikian, metode pembelajaran tanpa pengawasan masih bisa sangat berguna terutama di bidang pengelompokan dan reduksi dimensi. Dalam lokakarya ini, kita akan melihat secara mendalam algoritma tanpa pengawasan seperti PCA dan k-means - dan melihat mengapa metode tanpa pengawasan seperti ini merupakan alat hebat untuk ditambahkan ke kotak peralatan Anda.

36 Pengurangan Dimensi

36.1 Prinsip dan Motivasi

Pembelajaran mesin semakin banyak diadopsi di bidang yang berhubungan dengan data berdimensi tinggi seperti pengenalan angka tulisan tangan, internet-of-thing (IOT), dan pengenalan wajah. Akibatnya, ilmuwan data modern yang bekerja dengan teknologi ini dihadapkan pada masalah "dimensi" yang memerlukan solusi metodis, tidak hanya untuk mengurangi dimensi data tetapi juga meminimalkan hilangnya informasi.

Sebagai contoh motivasi, pikirkan kasus gambar beresolusi rendah. Lakukan penelusuran Google sederhana untuk wajah hitam putih dengan ukuran 40x40 piksel[^1]. Saat kami memperlakukan setiap gambar sebagai masukan, maka kumpulan data kami memiliki 1.600 dimensi.

Jika Anda memperbesar wajah, kami melihat 1.600 piksel individual. Saat menangani gambar skala abu-abu, strateginya dapat dengan memberikan nilai kecerahan pada skala 0 hingga 1 pada masing-masing 1.600 kolom, dengan 0 untuk "putih penuh" dan 1 untuk "hitam penuh", dan bergantung pada saturasi atau kecerahan setiap piksel - berikan nilai di antaranya.

Ada banyak makalah dan sumber bagus yang membahas topik penggunaan PCA dalam kompresi gambar, seperti yang ditulis oleh para peneliti di Institut Teknologi Kimia, Praha[^2], yang dibuat oleh Institut Informatika, Robotika, dan Sibernetika Ceko[^3], dan yang ini di sini[^4]. Di akhir bagian PCA ini, kami juga akan menerapkan PCA pada wajah manusia untuk melihat cara kerja kompresi gambar dalam praktiknya.

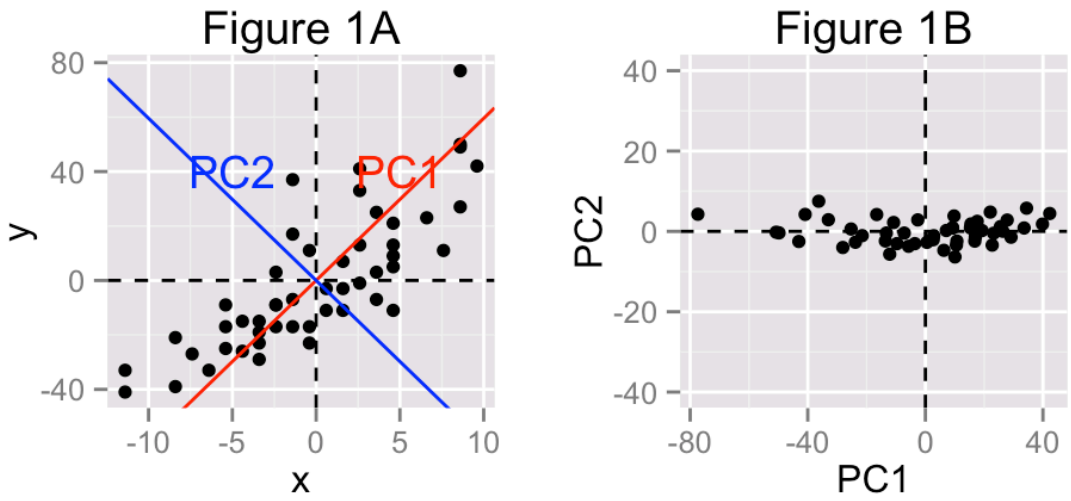
Gambar berukuran 40x40 mungkin tidak menarik. Jika Anda membuat pengklasifikasi gambar menggunakan foto dari iPhone 7 plus, resolusinya adalah 1.920 x 1.080 piksel (lebih dari 2 juta

dimensi). Dan itu untuk satu pengamatan. Ingat dari kursus Statistik Praktis Anda bahwa cara kita mengukur "informasi" adalah melalui varians, sehingga metode reduksi dimensi pada dasarnya berkaitan dengan mewakili sebanyak mungkin "varians" dalam dimensi sesedikit mungkin.

Hasil dari transformasi ini adalah data asli kita (matriks X) diwakili oleh matriks yang ditransformasikan secara linier, Z, dengan Z biasanya merupakan matriks dengan dimensi yang jauh lebih sedikit (biasanya <10) daripada $X $. Kolom pertama dari Z menjelaskan sebagian besar varians dalam X, dan kolom kedua menjelaskan jumlah varians yang lebih kecil dari kolom pertama, dan seterusnya hingga kolom terakhir.

Tujuan PCA adalah mencari Q, sehingga transformasi linier dapat dilakukan.

PCA akan mencari korelasi dalam data kita dan menggunakan redundansi tersebut untuk membuat matriks Z baru dengan dimensi yang cukup untuk menjelaskan sebagian besar varians dalam data asli. Variabel baru matriks Z disebut **komponen utama**.



Jika Anda melihat Gambar 1A, data asli kita berada pada bidang dengan koordinat `x` dan `y`. Dua dimensi (masing-masing x dan y) diperlukan pada Gambar 1A untuk menggambarkan varians dalam data kami sepenuhnya.

Namun, seandainya kita mengidentifikasi dua sumbu lain untuk mendeskripsikan data yang sama, dan salah satunya ortogonal langsung terhadap sumbu lainnya: sekarang kita dapat mengukur varians dalam data hanya dengan menggunakan sumbu ini (kita menyebutnya **komponen utama**) . Kami mengidentifikasi sumbu PC1 sebagai komponen utama pertama karena hanya menggunakan satu komponen utama, komponen inilah yang menjelaskan jumlah variasi paling banyak. Sumbu PC2 kemudian dipilih, sekali lagi dengan tujuan menjelaskan jumlah variasi terbanyak.

Jika kita menganggap PC2 konstan (katakanlah, 0) maka kita mengurangi dimensinya dari dua menjadi hanya satu, yaitu dengan memproyeksikan setiap titik data ke PC1. Kita memang kehilangan beberapa variasi karena observasi kita tidak tepat 0 pada sumbu PC2, namun karena hampir mendekati 0, variasi yang hilang karena pengurangan satu dimensi merupakan trade-off yang ingin kita terima.

(Ingat pelajaran kita dari Model Regresi, ketika kami memperkenalkan `VIF` untuk menunjukkan

bagaimana jika variabel "Pengeluaran Polisi tahun ini" dapat dijelaskan secara memadai oleh variabel "Pengeluaran Polisi tahun lalu" maka kita tidak memerlukan kedua variabel tersebut)

Aplikasi PCA lainnya: - Penemuan pola pada data berdimensi tinggi - Identifikasi variabel yang sangat berkorelasi dengan variabel lain - Memvisualisasikan data berdimensi tinggi

Dive Deeper: Manakah dari dua kumpulan data berikut yang paling terbantu oleh Analisis Komponen Utama (PCA)?

37 Import Library

[1]:
```
# try:
#     import pycaret
# except:
#     !pip install pycaret

import scipy.stats as stats
from scipy.stats import norm, skew
```

[2]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from pyod.models.lof import LOF
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from scipy.stats import norm, skew
import plotly.express as px
import numpy as np
from numpy.linalg import eig


from helper import biplot_pca
plt.style.use('seaborn-v0_8-bright')
```

[3]:
```
# Ignore warnings
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter(action='ignore', category=FutureWarning)
```

---

38 Langkah demi langkah: Analisis PCA

[4]:
```
credit_raw = pd.read_csv('credit100.csv')
credit_raw.head()
```

[4]:

```
   fraud_bool  income  name_email_similarity  prev_address_months_count  \
0           0     0.9               0.615786                         -1
1           0     0.6               0.072461                         -1
2           1     0.5               0.076592                         -1
3           0     0.3               0.255433                         23
4           0     0.9               0.192362                         -1

   current_address_months_count  customer_age  days_since_request  \
0                            51            40            0.010337
1                           125            20            0.006974
2                            37            50            0.006319
3                            19            20            0.009267
4                            34            30            0.019240

   intended_balcon_amount payment_type  zip_count_4w  …  has_other_cards  \
0               -0.699550           AB          2659  …                1
1               -0.959468           AC          5599  …                0
2               -1.219933           AB          1352  …                0
3                3.242242           AA          3743  …                0
4               -1.131560           AB           546  …                1

   proposed_credit_limit  foreign_request    source  \
0                 1500.0                0  INTERNET
1                  200.0                0  INTERNET
2                 1500.0                0  INTERNET
3                  200.0                0  INTERNET
4                 1000.0                0  INTERNET

   session_length_in_minutes device_os  keep_alive_session  \
0                  13.079806     linux                   0
1                   6.666551   windows                   0
2                   4.492668   windows                   1
3                   2.321885     other                   1
4                   4.845147     linux                   1

   device_distinct_emails_8w device_fraud_count  month
0                          1                  0      1
1                          1                  0      4
2                          2                  0      0
3                          1                  0      3
4                          1                  0      5

[5 rows x 32 columns]
```

Rangkuman Dataset:

The Bank Account Fraud (BAF) telah dipublikasikan di NeurIPS 2022 dan terdiri dari total 6 kumpulan data tabular penipuan rekening bank sintetis yang berbeda. BAF adalah tempat pengu-

jian yang realistis, lengkap, dan kuat untuk mengevaluasi metode baru dan yang sudah ada dalam ML dan ML yang adil, dan yang pertama dari jenisnya!

Rangkaian kumpulan data ini adalah: - Realistis, berdasarkan kumpulan data dunia nyata terkini untuk deteksi penipuan; - Bias, setiap kumpulan data memiliki tipe bias terkontrol yang berbeda; - Tidak seimbang, keadaan ini menunjukkan prevalensi kelas positif yang sangat rendah; - Dinamis, dengan data temporal dan pergeseran distribusi yang diamati; - Menjaga privasi, untuk melindungi identitas calon pelamar, kami telah menerapkan teknik privasi diferensial (penambahan noise), pengkodean fitur, dan melatih model generatif (CTGAN).

Berikut adalah penjelasan untuk setiap kolom dalam dataset:

1. **fraud_bool:** Label fraud (1 jika fraud, 0 jika no-fraud).
2. **income:** Pendapatan tahunan pelamar dalam kuartil. Rentang antara [0, 1].
3. **name_email_similarity:** Metrik kesamaan antara email dan nama pelamar. Nilai tinggi mengindikasikan kesamaan tinggi. Rentang antara [0, 1].
4. **prev_address_months_count:** Jumlah bulan di alamat terdaftar sebelumnya dari pelamar. Rentang antara [−1, 380] bulan (-1 adalah nilai yang hilang).
5. **current_address_months_count:** Bulan di alamat terdaftar saat ini dari pelamar. Rentang antara [−1, 406] bulan (-1 adalah nilai yang hilang).
6. **customer_age:** Usia pelamar dalam rentang per dekade (misalnya, 20-29 direpresentasikan sebagai 20).
7. **days_since_request:** umlah hari sejak aplikasi diajukan. Rentang antara [0, 78] hari.
8. **intended_balcon_amount:** Jumlah awal yang ditransfer untuk aplikasi. Rentang antara [−1, 108].
9. **payment_type:** Tipe rencana pembayaran kredit. 5 nilai mungkin (anonim).
10. **zip_count_4w:** Jumlah aplikasi dalam kode pos yang sama dalam 4 minggu terakhir. Rentang antara [1, 5767].
11. **velocity_6h:** Kecepatan total aplikasi yang dibuat dalam 6 jam terakhir, rata-rata jumlah aplikasi per jam dalam 6 jam terakhir. Rentang antara [−211, 24763].
12. **velocity_24h:** Kecepatan total aplikasi yang dibuat dalam 24 jam terakhir, rata-rata jumlah aplikasi per jam dalam 24 jam terakhir. Rentang antara [1329, 9527].
13. **velocity_4w:** Kecepatan total aplikasi yang dibuat dalam 4 minggu terakhir, rata-rata jumlah aplikasi per jam dalam 4 minggu terakhir. Rentang antara [2779, 7043].
14. **bank_branch_count_8w:** Jumlah total aplikasi di cabang bank terpilih dalam 8 minggu terakhir. Rentang antara [0, 2521].
15. **date_of_birth_distinct_emails_4w:** Jumlah email untuk pelamar dengan tanggal lahir yang sama dalam 4 minggu terakhir. Rentang antara [0, 42].
16. **employment_status:** Status pekerjaan pelamar. 7 nilai mungkin (anonim).
17. **credit_risk_score:** Skor internal risiko aplikasi. Rentang antara [−176, 387].
18. **email_is_free:** Domain email aplikasi (gratis atau berbayar).

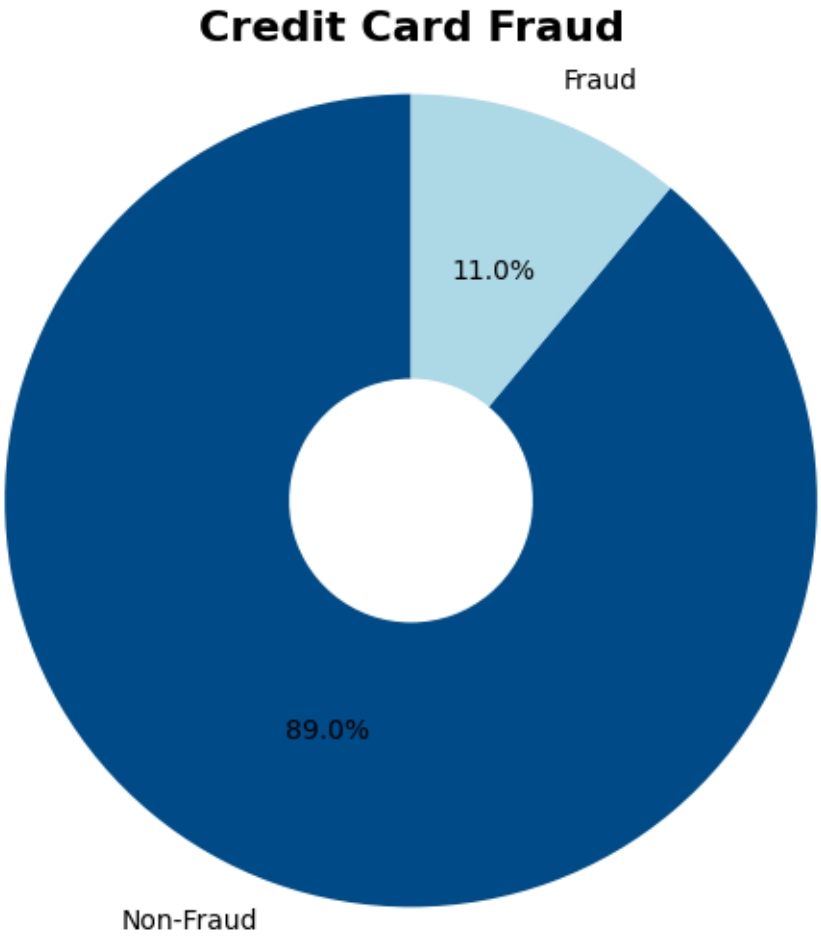
19. **housing_status:** Status tempat tinggal saat ini pelamar. 7 nilai mungkin (anonim).
20. **phone_home_valid:** Validitas nomor telepon rumah yang diberikan.
21. **phone_mobile_valid:** Validitas nomor telepon seluler yang diberikan.
22. **bank_months_count:** Berapa umur akun sebelumnya (jika ada) dalam bulan. Rentang antara [−1, 31] bulan (-1 adalah nilai yang hilang).
23. **has_other_cards:** Jika pelamar memiliki kartu lain dari perusahaan perbankan yang sama.
24. **proposed_credit_limit:** Batas kredit yang diajukan oleh pelamar. Rentang antara [200, 2000].
25. **foreign_request:** Jika negara asal permintaan berbeda dari negara bank.
26. **source:** Sumber aplikasi online. Baik browser (INTERNET) atau aplikasi seluler (APP).
27. **session_length_in_minutes:** Panjang sesi pengguna di situs web perbankan dalam menit. Rentang antara [−1, 107] menit.
28. **device_os:** Sistem operasi perangkat yang melakukan permintaan. Nilai mungkin: Windows, Macintosh, Linux, X11, atau lainnya.
29. **keep_alive_session:** Pilihan pengguna tentang logout sesi.
30. **device_distinct_emails_8w:** Jumlah email yang berbeda dalam situs web perbankan dari perangkat yang digunakan dalam 8 minggu terakhir. Rentang antara [0, 3].
31. **device_fraud_count:** Jumlah aplikasi fraud dengan perangkat yang digunakan. Rentang antara [0, 1]. Pada dataset hanya ada 0 sehingga tidak digunakan
32. **month:** Bulan di mana aplikasi diajukan. Rentang antara [0, 7].

38.1 Exploratory Data Analysis

38.1.1 Cek Jumlah Transaksi

[5]:
```
# Hitung proporsi non-fraud dan fraud
values = credit_raw['fraud_bool'].value_counts(normalize=True)
total_normal = credit_raw.loc[credit_raw['fraud_bool'] == 0].shape[0]
total_fraudulent = credit_raw.loc[credit_raw['fraud_bool'] == 1].shape[0]

# Persiapkan data untuk plot
labels = ['Non-Fraud', 'Fraud']
colors = ['#004B87', 'LightBlue']

# Buat plot pie
plt.figure(figsize=(6, 6))
plt.pie(values, labels=labels, autopct='%1.1f%%', colors=colors, startangle=90,␣
 ↪wedgeprops=dict(width=0.7))

# Atur proporsi lingkaran agar berbentuk lingkaran
```

```
plt.axis('equal')

# Atur judul dan tampilan plot
plt.title('Credit Card Fraud', fontsize=16, fontweight='bold', color='black')
plt.show()
```



NOTE:

> Dataset yang kita gunakan dalam course ini menyertakan kolom label yang mencerminkan kelas transaksi fraud dan no-fraud. Meskipun demikian, fokus utama analisis kita adalah pada eksplorasi struktur data dan pengidentifikasian pola tanpa memperhatikan label kelas. Tujuan utama course ini adalah memahami teknik `unsupervised learning`, anomaly detection, serta bagaimana menerapkannya untuk mendapatkan wawasan yang berharga dari dataset, terlepas dari ketidakseimbangan kelas. Dengan demikian, kita akan fokus pada teknik unsupervised learning dan pemahaman yang mendalam tentang struktur data.

38.1.2 Screening dataset secara cepat

[6]:
```
credit_raw.head(5).T
```

[6]:
```
                                        0            1            2  \
fraud_bool                              0            0            1
income                                0.9          0.6          0.5
name_email_similarity            0.615786     0.072461     0.076592
prev_address_months_count              -1           -1           -1
current_address_months_count           51          125           37
customer_age                           40           20           50
days_since_request               0.010337     0.006974     0.006319
intended_balcon_amount           -0.69955    -0.959468    -1.219933
payment_type                           AB           AC           AB
zip_count_4w                         2659         5599         1352
velocity_6h                     11007.70883  6304.262836  1357.413551
velocity_24h                    7448.797416  4373.093592  6667.029662
velocity_4w                     5576.303564  4721.100025  6318.876263
bank_branch_count_8w                 2030            1           15
date_of_birth_distinct_emails_4w        7           13            6
employment_status                      CA           CA           CF
credit_risk_score                     253          112          199
email_is_free                           1            1            1
housing_status                         BA           BE           BC
phone_home_valid                        1            1            1
phone_mobile_valid                      1            0            1
bank_months_count                       9           -1           28
has_other_cards                         1            0            0
proposed_credit_limit              1500.0        200.0       1500.0
foreign_request                         0            0            0
source                           INTERNET     INTERNET     INTERNET
session_length_in_minutes       13.079806     6.666551     4.492668
device_os                           linux      windows      windows
keep_alive_session                      0            0            1
device_distinct_emails_8w               1            1            2
device_fraud_count                      0            0            0
month                                   1            4            0

                                        3            4
fraud_bool                              0            0
income                                0.3          0.9
name_email_similarity            0.255433     0.192362
prev_address_months_count              23           -1
current_address_months_count           19           34
customer_age                           20           30
days_since_request               0.009267      0.01924
intended_balcon_amount           3.242242     -1.13156
```

```
payment_type                              AA           AB
zip_count_4w                            3743          546
velocity_6h                       5234.262304  3589.153453
velocity_24h                      6139.489579  4574.395571
velocity_4w                       4201.725919  4318.258913
bank_branch_count_8w                        1           16
date_of_birth_distinct_emails_4w           14            5
employment_status                          CA           CA
credit_risk_score                          63          275
email_is_free                               1            1
housing_status                             BC           BC
phone_home_valid                            0            1
phone_mobile_valid                          1            1
bank_months_count                          16            1
has_other_cards                             0            1
proposed_credit_limit                   200.0       1000.0
foreign_request                             0            0
source                                INTERNET     INTERNET
session_length_in_minutes             2.321885     4.845147
device_os                                 other        linux
keep_alive_session                          1            1
device_distinct_emails_8w                   1            1
device_fraud_count                          0            0
month                                       3            5
```

38.1.3 Melihat statistik dataset

[7]:
```
credit_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   fraud_bool                      100000 non-null  int64
 1   income                          100000 non-null  float64
 2   name_email_similarity           100000 non-null  float64
 3   prev_address_months_count       100000 non-null  int64
 4   current_address_months_count    100000 non-null  int64
 5   customer_age                    100000 non-null  int64
 6   days_since_request              100000 non-null  float64
 7   intended_balcon_amount          100000 non-null  float64
 8   payment_type                    100000 non-null  object
 9   zip_count_4w                    100000 non-null  int64
 10  velocity_6h                     100000 non-null  float64
 11  velocity_24h                    100000 non-null  float64
 12  velocity_4w                     100000 non-null  float64
 13  bank_branch_count_8w            100000 non-null  int64
```

```
 14  date_of_birth_distinct_emails_4w  100000 non-null  int64
 15  employment_status                 100000 non-null  object
 16  credit_risk_score                 100000 non-null  int64
 17  email_is_free                     100000 non-null  int64
 18  housing_status                    100000 non-null  object
 19  phone_home_valid                  100000 non-null  int64
 20  phone_mobile_valid                100000 non-null  int64
 21  bank_months_count                 100000 non-null  int64
 22  has_other_cards                   100000 non-null  int64
 23  proposed_credit_limit             100000 non-null  float64
 24  foreign_request                   100000 non-null  int64
 25  source                            100000 non-null  object
 26  session_length_in_minutes         100000 non-null  float64
 27  device_os                         100000 non-null  object
 28  keep_alive_session                100000 non-null  int64
 29  device_distinct_emails_8w         100000 non-null  int64
 30  device_fraud_count                100000 non-null  int64
 31  month                             100000 non-null  int64
dtypes: float64(9), int64(18), object(5)
memory usage: 24.4+ MB
```

[8]:
```
credit_raw.nunique()
```

[8]:
```
fraud_bool                            2
income                                9
name_email_similarity             99989
prev_address_months_count           357
current_address_months_count        404
customer_age                          9
days_since_request                99886
intended_balcon_amount            99943
payment_type                          5
zip_count_4w                       5455
velocity_6h                       99983
velocity_24h                      99993
velocity_4w                       99972
bank_branch_count_8w               2140
date_of_birth_distinct_emails_4w     38
employment_status                     7
credit_risk_score                   509
email_is_free                         2
housing_status                        7
phone_home_valid                      2
phone_mobile_valid                    2
bank_months_count                    32
has_other_cards                       2
proposed_credit_limit                12
```

```
foreign_request                        2
source                                 2
session_length_in_minutes          99780
device_os                              5
keep_alive_session                     2
device_distinct_emails_8w              4
device_fraud_count                     1
month                                  8
dtype: int64
```

Observasi:

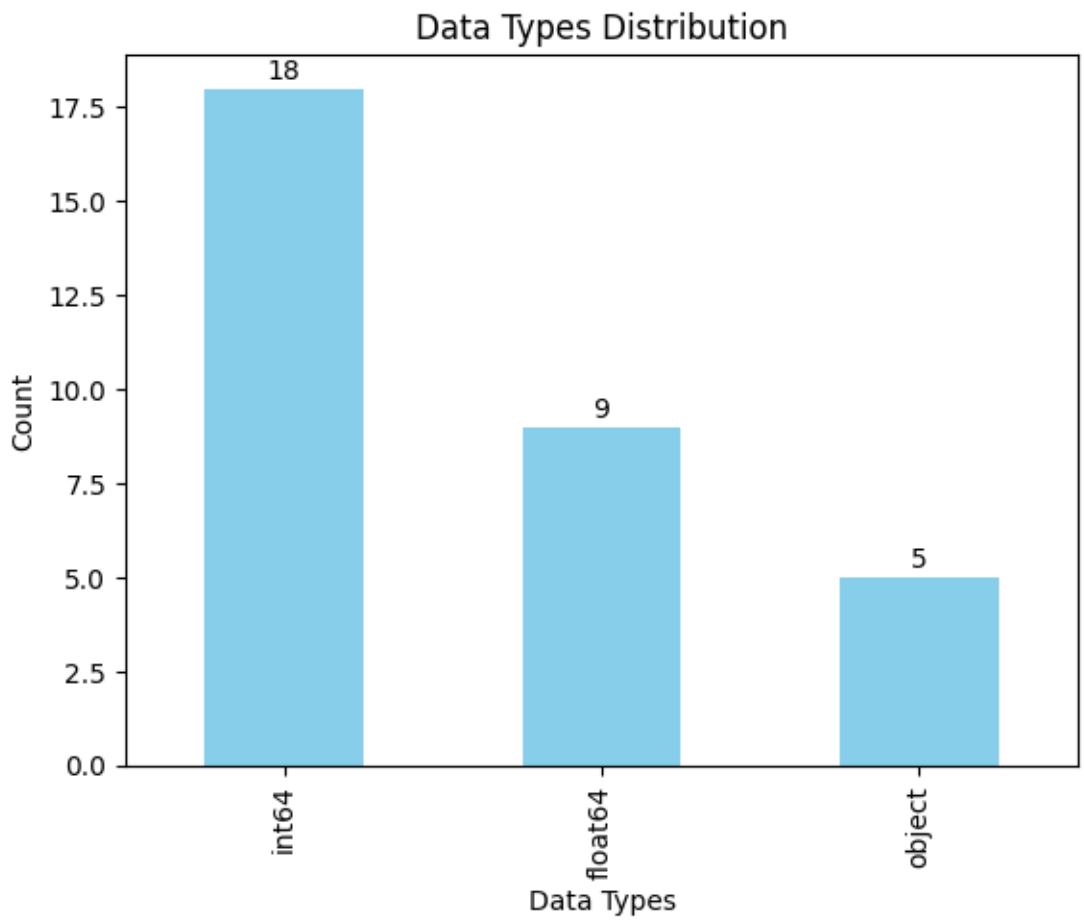
Berdasarkan dataset yang kita gunakan, untuk menggunakan algoritma unsupervised learning seperti Local Outlier Factor (LOF), KMeans, dll., Anda perlu memastikan bahwa tipe data setiap kolom sesuai dengan kebutuhan algoritma tersebut. Berikut adalah beberapa langkah yang bisa Anda lakukan untuk membenarkan tipe data:

1. **Pengecekan dan Koreksi Tipe Data:**
   - Pastikan bahwa kolom-kolom dengan nilai unik yang rendah dan bertipe integer yang seharusnya bersifat kategorikal diubah menjadi tipe data kategorikal. Contohnya: `fraud_bool`, `employment_status`, `housing_status`, `phone_home_valid`, `phone_mobile_valid`, `has_other_cards`, `foreign_request`, dan `source`.
   - Pastikan bahwa kolom dengan nilai unik yang rendah dan bertipe float.
2. **Pengecekan dan Koreksi Nilai yang Anomali:**
   - Periksa nilai-nilai unik dalam kolom-kolom dengan nilai unik yang rendah untuk memastikan tidak ada nilai yang tidak valid atau anomali. Jika ditemukan, perbaiki atau hapus nilai tersebut.
   - Periksa kolom dengan nilai float atau integer apakah ada nilai yang diluar rentang yang masuk akal.

Pastikan Anda menyesuaikan langkah-langkah di atas sesuai dengan karakteristik dan kebutuhan data Anda. Jika Anda memiliki kolom lain yang memerlukan penanganan khusus, pastikan untuk memeriksa dan menyesuaikan tipe datanya. Setelah ini, Anda dapat melanjutkan dengan langkah-langkah unsupervised learning seperti PCA, LOF, dan KMeans seperti yang sudah dijelaskan sebelumnya.

[9]:
```python
# Menghitung distribusi tipe data
data_type_counts = credit_raw.dtypes.astype(str).value_counts()

# Membuat plot
ax = data_type_counts.plot(kind='bar', color='skyblue')

# Menambahkan label pada setiap bar
for i, count in enumerate(data_type_counts):
    ax.text(i, count + 0.1, str(count), ha='center', va='bottom')

# Menambahkan judul dan label pada sumbu
plt.title("Data Types Distribution")
plt.xlabel("Data Types")
```

```
plt.ylabel("Count")

# Menampilkan plot
plt.show()
```



38.2 Data Perparation

[10]:
```
credit_clean = credit_raw.copy()
```

1. Cek Missing Value

[11]:
```
credit_clean.isna().sum()
```

[11]:
```
fraud_bool                      0
income                          0
name_email_similarity           0
prev_address_months_count       0
current_address_months_count    0
```

```
customer_age                        0
days_since_request                  0
intended_balcon_amount              0
payment_type                        0
zip_count_4w                        0
velocity_6h                         0
velocity_24h                        0
velocity_4w                         0
bank_branch_count_8w                0
date_of_birth_distinct_emails_4w    0
employment_status                   0
credit_risk_score                   0
email_is_free                       0
housing_status                      0
phone_home_valid                    0
phone_mobile_valid                  0
bank_months_count                   0
has_other_cards                     0
proposed_credit_limit               0
foreign_request                     0
source                              0
session_length_in_minutes           0
device_os                           0
keep_alive_session                  0
device_distinct_emails_8w           0
device_fraud_count                  0
month                               0
dtype: int64
```

2. Cek Duplikat Data

[12]:
```
# cek duplicated value
credit_clean.duplicated().sum()
```

[12]: 0

3. Buang kolom yang tidak digunakan

[13]:
```
# device_fraud_count : karena isinya 0 semua
credit_clean.drop('device_fraud_count', axis = 1, inplace=True)
```

4. Data wrangling

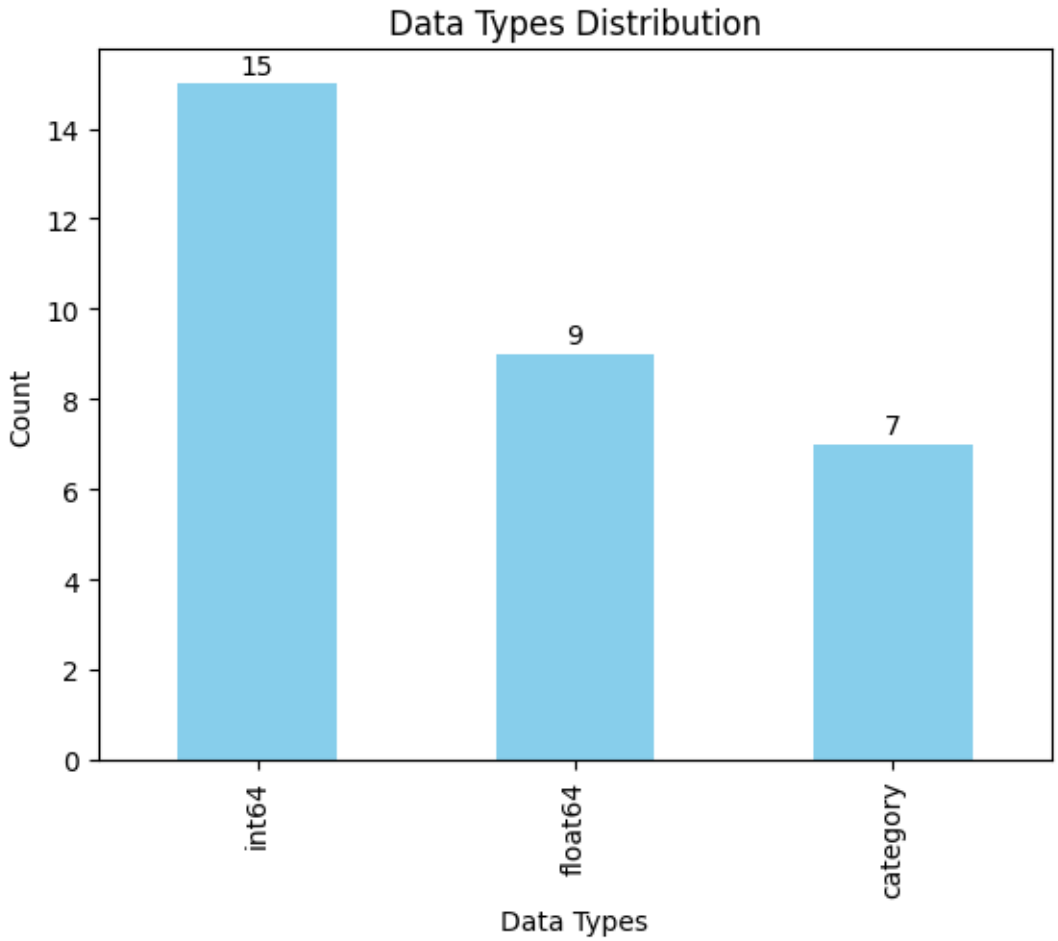
[14]:
```
# Contoh untuk beberapa kolom yang bersifat kategorikal
categorical_columns = ['payment_type','employment_status',
                       'housing_status', 'has_other_cards',
                       'foreign_request','source', 'device_os']

# Ubah tipe data menjadi kategorikal
```

```
credit_clean[categorical_columns] = credit_clean[categorical_columns].
 ↪astype('category')
credit_clean.dtypes
```

[14]:
```
fraud_bool                            int64
income                              float64
name_email_similarity               float64
prev_address_months_count             int64
current_address_months_count          int64
customer_age                          int64
days_since_request                  float64
intended_balcon_amount              float64
payment_type                       category
zip_count_4w                          int64
velocity_6h                         float64
velocity_24h                        float64
velocity_4w                         float64
bank_branch_count_8w                  int64
date_of_birth_distinct_emails_4w      int64
employment_status                  category
credit_risk_score                     int64
email_is_free                         int64
housing_status                     category
phone_home_valid                      int64
phone_mobile_valid                    int64
bank_months_count                     int64
has_other_cards                    category
proposed_credit_limit               float64
foreign_request                    category
source                             category
session_length_in_minutes           float64
device_os                          category
keep_alive_session                    int64
device_distinct_emails_8w             int64
month                                 int64
dtype: object
```

[15]:
```
# Menghitung distribusi tipe data
data_type_counts = credit_clean.dtypes.astype(str).value_counts()

# Membuat plot
ax = data_type_counts.plot(kind='bar', color='skyblue')

# Menambahkan label pada setiap bar
for i, count in enumerate(data_type_counts):
    ax.text(i, count + 0.1, str(count), ha='center', va='bottom')
```

```
# Menambahkan judul dan label pada sumbu
plt.title("Data Types Distribution")
plt.xlabel("Data Types")
plt.ylabel("Count")

# Menampilkan plot
plt.show()
```



Observasi

Terdapat kumpulan data berisi fitur kategorikal. Meskipun ada berbagai metode yang tersedia untuk menangani data kategorikal, demi mencapai tujuan kami membandingkan regresi logistik dan mesin peningkatan gradien, kami telah memutuskan untuk mengubah fitur ini menjadi nilai numerik.

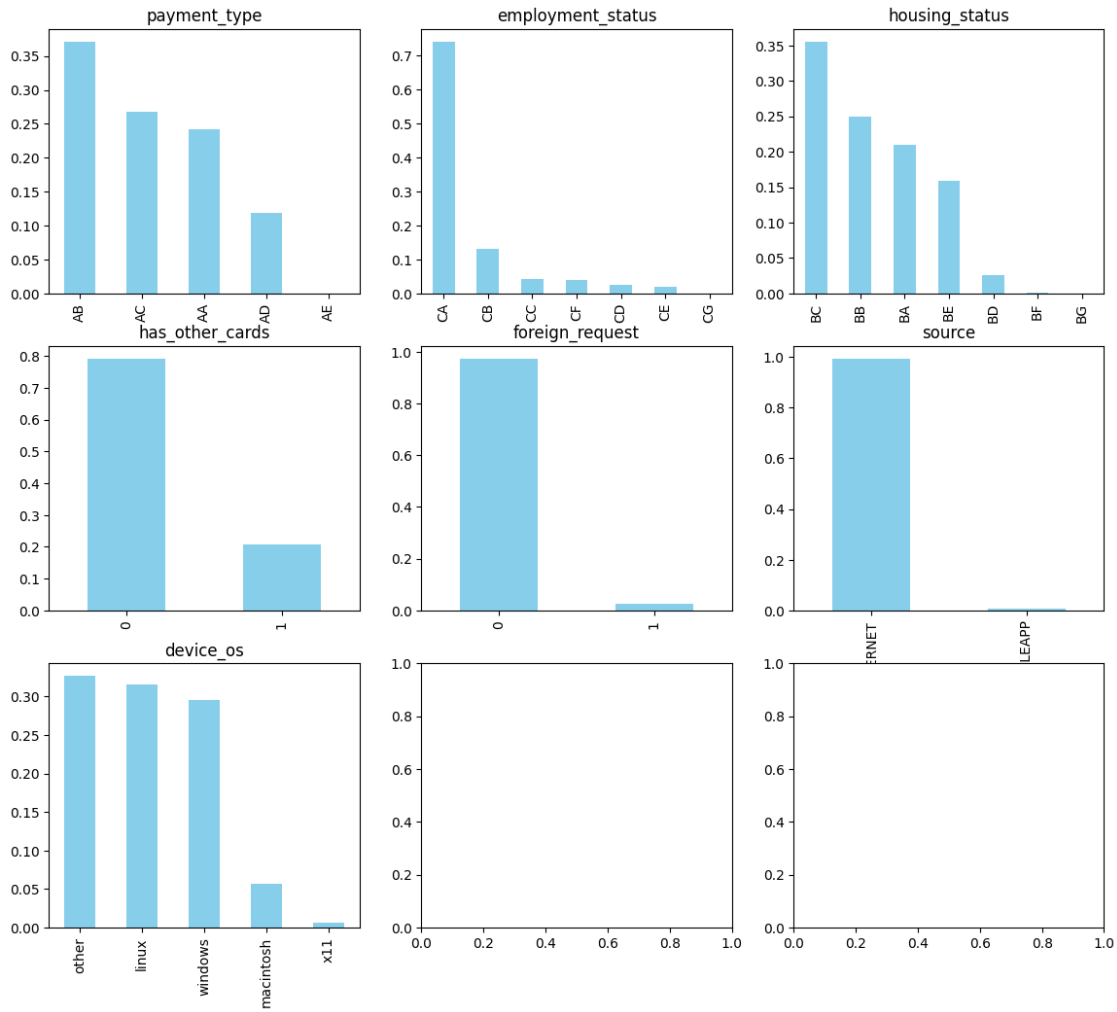
5. Ambil variabel dataset numerik

[16]:
```
credit_clean.dtypes
```

[16]:
```
fraud_bool                            int64
income                              float64
name_email_similarity               float64
prev_address_months_count             int64
current_address_months_count          int64
customer_age                          int64
days_since_request                  float64
intended_balcon_amount              float64
payment_type                       category
zip_count_4w                          int64
velocity_6h                         float64
velocity_24h                        float64
velocity_4w                         float64
bank_branch_count_8w                  int64
date_of_birth_distinct_emails_4w      int64
employment_status                  category
credit_risk_score                     int64
email_is_free                         int64
housing_status                     category
phone_home_valid                      int64
phone_mobile_valid                    int64
bank_months_count                     int64
has_other_cards                    category
proposed_credit_limit               float64
foreign_request                    category
source                             category
session_length_in_minutes           float64
device_os                          category
keep_alive_session                    int64
device_distinct_emails_8w             int64
month                                 int64
dtype: object
```

[17]:
```
import math

# Memilih hanya 7 kolom kategori
categories = credit_clean.columns[credit_clean.dtypes == 'category']

# Menentukan jumlah baris dan kolom untuk subplot
num_plots = len(categories)
num_rows = 3
num_cols = math.ceil(num_plots / num_rows)

# Membuat subplot
fig, axes = plt.subplots(num_rows, num_cols, figsize=(14, 12))  # Sesuaikan␣
↪ukuran subplot
```

```
# Iterasi melalui kolom kategori dan membuat subplot
for i, x in enumerate(categories):
    row = i // num_cols
    col = i % num_cols

    # Memeriksa apakah kolom memiliki data sebelum membuat plot
    if not credit_clean[x].isnull().all():
        credit_clean[x].value_counts(normalize=True).plot(kind='bar',␣
↪color='skyblue', ax=axes[row, col])
        axes[row, col].set_title(x)

# Menampilkan subplot
plt.show()
```



6. Data Scaling

Visualisasi Plot Skewness:

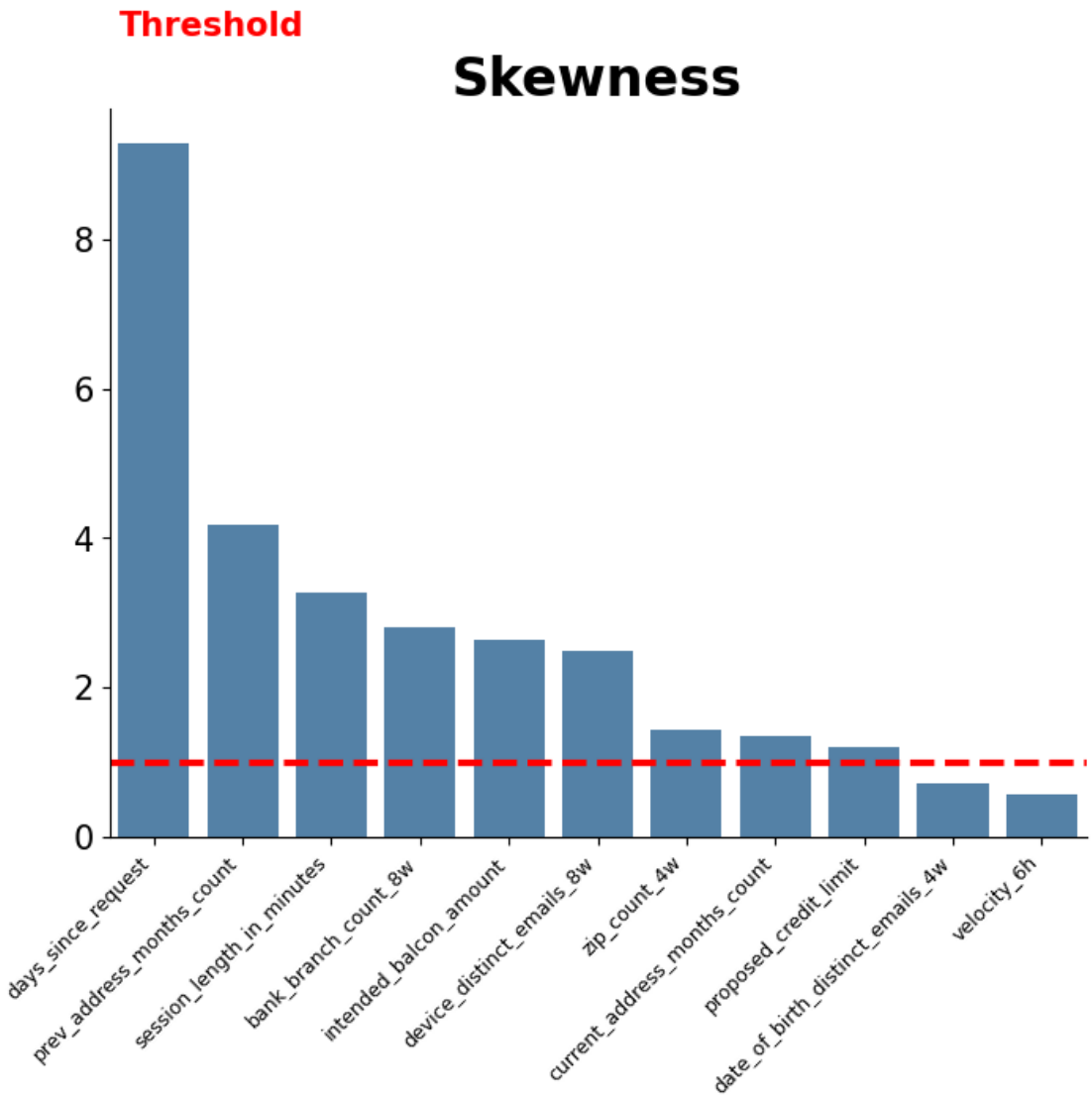
Sebelum melakukan scaling data, kita pertama-tama mengevaluasi skewness dari kolom-kolom numerik dalam dataset.

[18]:
```
# Pilih hanya kolom numerik
numerical_cols = credit_clean.select_dtypes(include=['int64', 'float64']).
 ↪columns

# Sisihkan kolom 'fraud_bool'
numerical_cols = numerical_cols[numerical_cols != 'fraud_bool']
```

[19]:
```
numerical_cols
```

[19]:
```
Index(['income', 'name_email_similarity', 'prev_address_months_count',
       'current_address_months_count', 'customer_age', 'days_since_request',
       'intended_balcon_amount', 'zip_count_4w', 'velocity_6h', 'velocity_24h',
       'velocity_4w', 'bank_branch_count_8w',
       'date_of_birth_distinct_emails_4w', 'credit_risk_score',
       'email_is_free', 'phone_home_valid', 'phone_mobile_valid',
       'bank_months_count', 'proposed_credit_limit',
       'session_length_in_minutes', 'keep_alive_session',
       'device_distinct_emails_8w', 'month'],
      dtype='object')
```

[20]:
```
plt.figure(figsize=(8, 6))
skew_features = credit_clean[numerical_cols].apply(lambda x: skew(x))
skew_features = skew_features[skew_features > 0.5].sort_values(ascending=False)␣
 ↪# treshold 0.5
ax = sns.barplot(x=skew_features.index, y=skew_features.values,␣
 ↪color='SteelBlue')
ax.set_ylabel('', fontsize=20)
ax.set_xlabel('', fontsize=20)
ax.tick_params(axis='both', labelsize=15)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right', fontsize=9)
ax.axhline(y=1, color='red', linestyle='--', linewidth=3)
ax.set_title('Skewness', ha = 'center', weight='bold', fontsize=24)
ax.text(0.01, 1.1, 'Threshold', color='red', transform=ax.transAxes,␣
 ↪fontsize=15, weight='bold')
sns.despine()
plt.gca().set_facecolor('white')
plt.show()
```



Observasi:

- Skewness mengukur tingkat asimetri distribusi data. Nilai skewness di atas 0.5 menunjukkan adanya kemiringan (asimetri) dalam distribusi data.
- Plot bar di atas menampilkan nilai skewness untuk setiap kolom numerik yang memiliki skewness di atas threshold 0.5.
- Garis merah putus-putus pada nilai skewness 1 berfungsi sebagai batas threshold; kolom-kolom di atas batas ini dianggap memiliki skewness tinggi. Kolom-kolom dengan skewness di atas threshold dapat mengindikasikan adanya ketidaknormalan dalam distribusi data. Kolom-kolom dengan skewness tinggi mungkin memerlukan teknik scaling atau transformasi data untuk meningkatkan kinerja model.
- Nilai skewness positif menunjukkan ekor distribusi yang lebih panjang di sisi kanan, sementara

nilai negatif menunjukkan ekor distribusi yang lebih panjang di sisi kiri.

Melakukan scaling data

Setelah memahami skewness data, kita dapat melanjutkan dengan tahap kedua, yaitu melakukan scaling data menggunakan z-score normalization. Hal ini dilakukan untuk menghilangkan penyimpangan skewness dan memastikan distribusi data mendekati distribusi normal standar. Rangkuman kalimat untuk tahap kedua dapat mencakup proses scaling data menggunakan z-score normalization untuk memastikan konsistensi dalam rentang nilai dan meningkatkan keterbacaan model.

[Optional] Math Behind Z-Score Normalization (StandardScaler) in sklearn:

Z-Score Normalization (StandardScaler) diimplementasikan dalam scikit-learn dengan menggunakan kelas `StandardScaler`. Formula matematis dari Z-Score Normalization adalah sebagai berikut:

$$Z = \frac{X - \mu}{\sigma}$$

- (Z) adalah nilai hasil normalisasi (z-score) dari suatu observasi.
- (X) adalah nilai observasi asli.
- μ adalah rata-rata dari seluruh nilai observasi dalam suatu fitur.
- σ adalah deviasi standar dari seluruh nilai observasi dalam suatu fitur.

Penjelasan: 1. **Mean (μ):** - Rata-rata dari seluruh nilai dalam suatu fitur. - Dihitung dengan menjumlahkan semua nilai dan membaginya dengan jumlah total nilai.

2. **Standard Deviation (σ):**
   - Deviasi standar mengukur sejauh mana nilai-nilai dalam suatu fitur tersebar dari rata-rata.
   - Dihitung dengan menghitung akar kuadrat dari varians.
3. **Z-Score ((Z)):**
   - Memberikan informasi tentang seberapa jauh nilai observasi berada dari rata-rata dalam satuan deviasi standar.
   - Jika (Z) positif, itu berarti nilai observasi berada di atas rata-rata. Jika (Z) negatif, itu berarti nilai observasi berada di bawah rata-rata.

Alasan Penggunaan Scaling pada Reduksi Dimensi: - Konsistensi Skala: - Algoritma reduksi dimensi, seperti PCA (Principal Component Analysis), sensitif terhadap skala variabel. Scaling memastikan bahwa semua variabel berada pada rentang yang serupa, sehingga tidak ada variabel yang mendominasi proses reduksi dimensi.

- **Interpretasi yang Mudah:**
  - Skala yang seragam memudahkan interpretasi bobot relatif dari masing-masing komponen utama setelah reduksi dimensi.
- **Konvergensi Algoritma:**
  - Beberapa algoritma reduksi dimensi, seperti gradient descent pada tugas t-SNE, dapat lebih cepat mencapai konvergensi dengan data yang telah di-scaled.
- **Mengatasi Skewness dan Outlier:**
  - Scaling dapat membantu mengurangi dampak skewness dan nilai ekstrem (outlier) dalam data, meningkatkan keandalan analisis reduksi dimensi.

Dengan melakukan Z-Score Normalization sebelum menerapkan metode reduksi dimensi, kita memastikan bahwa seluruh variabel berada pada skala yang seragam, memfasilitasi interpretasi dan meningkatkan performa algoritma reduksi dimensi.

[21]:
```
credit_num = credit_clean[numerical_cols]
```

[22]:
```
credit_num.head()
```

[22]:
```
   income  name_email_similarity  prev_address_months_count  \
0     0.9               0.615786                         -1
1     0.6               0.072461                         -1
2     0.5               0.076592                         -1
3     0.3               0.255433                         23
4     0.9               0.192362                         -1

   current_address_months_count  customer_age  days_since_request  \
0                            51            40            0.010337
1                           125            20            0.006974
2                            37            50            0.006319
3                            19            20            0.009267
4                            34            30            0.019240

   intended_balcon_amount  zip_count_4w   velocity_6h  velocity_24h  …  \
0               -0.699550          2659  11007.708830   7448.797416  …
1               -0.959468          5599   6304.262836   4373.093592  …
2               -1.219933          1352   1357.413551   6667.029662  …
3                3.242242          3743   5234.262304   6139.489579  …
4               -1.131560           546   3589.153453   4574.395571  …

   credit_risk_score  email_is_free  phone_home_valid  phone_mobile_valid  \
0                253              1                 1                   1
1                112              1                 1                   0
2                199              1                 1                   1
3                 63              1                 0                   1
4                275              1                 1                   1

   bank_months_count  proposed_credit_limit  session_length_in_minutes  \
0                  9                 1500.0                  13.079806
1                 -1                  200.0                   6.666551
2                 28                 1500.0                   4.492668
3                 16                  200.0                   2.321885
4                  1                 1000.0                   4.845147

   keep_alive_session  device_distinct_emails_8w  month
0                   0                          1      1
1                   0                          1      4
2                   1                          2      0
```

```
3                    1                            1      3
4                    1                            1      5

[5 rows x 23 columns]
```

Menggunakan Z-score standardization untuk scaling dataset numerik dengan fungsi StandardScaler() pada library sklearn:

[23]:
```
credit_scaled = StandardScaler().fit_transform(credit_num.values)
credit_scaled
```

[23]:
```
array([[ 1.12580168,  0.45453446, -0.38732389, …, -1.11467683,
        -0.12426586, -1.03803432],
       [ 0.09476593, -1.41007296, -0.38732389, …, -1.11467683,
        -0.12426586,  0.31203178],
       [-0.24891265, -1.39589477, -0.38732389, …,  0.89712101,
         4.87436093, -1.48805636],
       …,
       [ 0.43844452, -1.54122323, -0.38732389, …,  0.89712101,
        -0.12426586,  0.76205381],
       [-0.24891265, -1.29201166, -0.38732389, …, -1.11467683,
         4.87436093, -1.48805636],
       [ 0.7821231 , -0.40150292,  0.37934726, …,  0.89712101,
        -0.12426586, -0.58801229]])
```

[24]:
```
credit_scaled = pd.DataFrame(credit_scaled, columns=[numerical_cols])
```

[25]:
```
credit_scaled.head()
```

[25]:
```
     income name_email_similarity prev_address_months_count  \
0  1.125802              0.454534                 -0.387324
1  0.094766             -1.410073                 -0.387324
2 -0.248913             -1.395895                 -0.387324
3 -0.936270             -0.782141                  0.170255
4  1.125802             -0.998590                 -0.387324

  current_address_months_count customer_age days_since_request  \
0                    -0.431527     0.459337          -0.188833
1                     0.406739    -1.165255          -0.189457
2                    -0.590118     1.271634          -0.189578
3                    -0.794021    -1.165255          -0.189032
4                    -0.624102    -0.352959          -0.187183

  intended_balcon_amount zip_count_4w velocity_6h velocity_24h  …  \
0              -0.442254     1.075569    1.790845     1.827162  …
1              -0.455380     4.005273    0.226647    -0.256489  …
2              -0.468534    -0.226854   -1.418498     1.297549  …
3              -0.243188     2.155773   -0.129197     0.940165  …
```

```
4              -0.464071    -1.030032   -0.676301    -0.120116  …

  credit_risk_score email_is_free phone_home_valid phone_mobile_valid  \
0          1.623733      0.916752         1.233515           0.359129
1         -0.324384      0.916752         1.233515          -2.784511
2          0.877646      0.916752         1.233515           0.359129
3         -1.001389      0.916752        -0.810691           0.359129
4          1.927695      0.916752         1.233515           0.359129

  bank_months_count proposed_credit_limit session_length_in_minutes  \
0         -0.144869              1.860207                  0.655308
1         -0.965382             -0.673720                 -0.118348
2          1.414106              1.860207                 -0.380592
3          0.429490             -0.673720                 -0.642461
4         -0.801279              0.885620                 -0.338071

  keep_alive_session device_distinct_emails_8w     month
0          -1.114677                 -0.124266 -1.038034
1          -1.114677                 -0.124266  0.312032
2           0.897121                  4.874361 -1.488056
3           0.897121                 -0.124266 -0.137990
4           0.897121                 -0.124266  0.762054

[5 rows x 23 columns]
```

Kita telah mengetahui sebelumnya bahwa korelasi dalam data membuat beberapa variabel menjadi mubazir. Pasca-PCA, matriks Z harus berupa matriks yang setiap dimensinya tidak berkorelasi.

Diskusi: Manakah dari plot berikut yang menunjukkan variabel yang tidak berkorelasi?

Variabel Usia dan Harga pada `Sale Price of Vehicle` tidak berkorelasi. Jika Anda menggambar garis regresi di atasnya, garis tersebut menunjukkan kemiringan ke bawah, dan korelasi antara Usia dan Harga Jual akan bernilai mendekati -1.

Variabel Suhu dan Tekanan pada `Logistic Machinery` juga tidak berkorelasi. Jika Anda memplot garis regresi di atasnya, garis tersebut menunjukkan kemiringan ke atas, dan korelasi antara Suhu dan Tekanan akan bernilai mendekati 1.

Secara intuitif, kita secara visual menyimpulkan bahwa variabel "Blind Tasting" tidak berkorelasi karena mengetahui nilai x tidak memberi kita indikasi berapa nilai y.

Sekarang jika kita mengikuti contoh di bagian **pca1.png** (dari Prinsip dan Motivasi) dan memilih kombinasi sumbu baru untuk `Logistic Machinery` (yang merupakan garis ungu berlabel `v1` dan `v2` di bawah ini gambar), maka distribusi suhu dan tekanan menjadi tidak berkorelasi berdasarkan dua sumbu baru:

38.2.1 [Opsional] Nilai Eigen dan Vektor Eigen

Jika Anda memiliki paparan terbatas pada Aljabar Matriks - silakan lewati sub-bagian ini. Konsep yang disajikan dalam sub-bagian ini tidak akan dinilai, pada kenyataannya R memiliki fungsi

bawaan yang sangat berguna yang membantu Anda menghitung vektor eigen dan nilai eigen sehingga pengerjaan bagian dalam sebagian besar bersifat opsional bagi praktisi mana pun.

Dalam teori aljabar linier dan matriks, kita mempelajari bahwa sebuah vektor jika dikalikan dengan matriks, akan mengubah arahnya. Ketika kita mengambil sebuah vektor (2,3) dan mengalikannya dengan skala, katakanlah 2, maka vektor yang dihasilkan adalah (4,6) - skalar menskalakan vektor tersebut dengan besaran 2.

Sekarang jika kita mengambil vektor yang sama dan mengalikannya dengan matriks, vektor yang dikerjakan akan berubah arah:

$$\begin{pmatrix} 11 \\ 16 \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Secara sederhana, kita dapat memandang operasi di atas sebagai suatu matriks yang mentransformasi vektor menjadi vektor lain: vektor hasilnya (umumnya) memiliki arah dan panjang yang berbeda dari vektor asli. Namun, terdapat beberapa pengecualian penting terhadap hal ini:

- Matriks yang bertindak pada suatu vektor tanpa mengubahnya sama sekali disebut **matriks identitas**. Matriks identitas adalah matriks persegi di mana semua elemen diagonal utama bernilai satu dan semua elemen lainnya bernilai nol.

  Matriks (2,3) yang dikalikan dengan matriks identitas berukuran 2 akan menghasilkan matriks (2,3); baik arah maupun besarnya tidak berubah.

- Matriks yang memutar setiap vektor sejauh sudut tetap disebut **matriks rotasi**. Arah vektor berubah, tetapi tidak besarnya. Ambil matriks A di bawah ini dan kalikan dengan (-3, 2) untuk mendapatkan (3, -2). Jika Anda menggambar kedua vektor tersebut, Anda akan melihat bahwa keduanya memiliki panjang yang sama dengan vektor kedua diputar sejauh 180 derajat dari yang pertama. Kalikan A dengan (-20, 18) dan vektor akan berubah menjadi (20, -18), lagi-lagi diputar sejauh 180 derajat. Ternyata, A adalah matriks rotasi yang akan mengubah suatu vektor sejauh 180 derajat tanpa mengubah besarnya.

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} -3 \\ 2 \end{pmatrix}$$

- Untuk sebagian besar matriks, terdapat vektor tertentu yang disebut **vektor eigen** yang arahnya tidak berubah (kecuali jika dikalikan dengan -1, maka arahnya terbalik) ketika dioperasikan oleh matriks. Hasil perkalian matriks dengan vektor tersebut, vektor eigen, memiliki efek yang sama dengan mengalikan vektor tersebut dengan suatu skalar. Skalar ini adalah konstanta dan secara resmi disebut sebagai **nilai eigen**. Ketika kita mengalikan matriks A dengan vektor eigen x, hasilnya akan berupa konstanta λ dikalikan dengan x. Persamaannya adalah:

  $Ax = \lambda x$, di mana angka λ adalah nilai eigen dari A.

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \begin{pmatrix} 3 \\ 2 \end{pmatrix} 2 \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix} \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

Perhatikan bagaimana vektor eigen yang diperbesar tetap menjadi vektor eigen.

38.2.2 Intuisi dan Praktik: PCA

Ingat bahwa sebelumnya kita menyebutkan secara intuitif, tujuan dari PCA adalah memilih sumbu baru (dimensi baru) untuk plot kita, dan dimensi baru ini dari data kita disebut sebagai **komponen utama**? Dimensi baru ini seharusnya tidak memiliki korelasi satu sama lain (dalam konteks dimensionalitas, korelasi menyebabkan redundansi!). Jika kita memiliki matriks kovariansi:

[26]:
```
a = np.matrix('1.04, 0.77; 0.77, 0.68')
a
```

[26]:
```
matrix([[1.04, 0.77],
        [0.77, 0.68]])
```

Ingat dari course Practical Statistics bahwa ini berarti varians dari variabel pertama adalah 1,04, varians dari variabel kedua adalah 0,68, dan kovarians antara keduanya adalah 0,77.

Sekarang `A` adalah matriks simetris persegi, sehingga dapat didiagonalisasi dengan memilih sistem koordinat baru yang diberikan oleh eigenvector-nya, eigenvalue yang sesuai kemudian akan menggantikan nilai-nilai di diagonal. Dengan menggunakan sistem koordinat baru ini, matriks kovarians A kita akan terlihat seperti:

[27]:
```
np.cov(a)
```

[27]:
```
array([[0.03645, 0.01215],
       [0.01215, 0.00405]])
```

[28]:
```
w,v = eig(a)
print('E-value:', w)
print('E-vector', v)
```

```
E-value: [1.65075913 0.06924087]
E-vector [[ 0.78346326 -0.6214381 ]
 [ 0.6214381   0.78346326]]
```

38.2.3 Aplikasi PCA: Reduksi Dimensi pada Data credit

Pertama, mari kita lihat matriks kovariansi dari dataset `credit_scaled`:

7. Cek korelasi data

[29]:
```
# cek kembali data yang sudah di pca
credit_scaled.cov()
```

[29]:
```
                                  income name_email_similarity  \
income                          1.000010             -0.048731
name_email_similarity          -0.048731              1.000010
prev_address_months_count       0.008809             -0.009854
current_address_months_count   -0.017391              0.038234
customer_age                    0.137962             -0.066315
days_since_request             -0.012759             -0.006397
intended_balcon_amount          0.045168              0.035757
```

zip_count_4w	-0.079997	0.025874
velocity_6h	-0.100350	0.027620
velocity_24h	-0.106405	0.038162
velocity_4w	-0.108241	0.042983
bank_branch_count_8w	0.005063	-0.003919
date_of_birth_distinct_emails_4w	-0.075545	0.036560
credit_risk_score	0.195343	0.033618
email_is_free	-0.013721	-0.075405
phone_home_valid	0.012676	0.003814
phone_mobile_valid	0.010555	0.015459
bank_months_count	-0.001819	-0.011884
proposed_credit_limit	0.135009	0.053971
session_length_in_minutes	-0.051490	0.000112
keep_alive_session	-0.058315	0.031243
device_distinct_emails_8w	-0.002465	-0.029268
month	0.119413	-0.048978

	prev_address_months_count \
income	0.008809
name_email_similarity	-0.009854
prev_address_months_count	1.000010
current_address_months_count	-0.270593
customer_age	-0.078717
days_since_request	0.080489
intended_balcon_amount	0.002354
zip_count_4w	-0.027014
velocity_6h	0.004293
velocity_24h	0.017160
velocity_4w	0.009710
bank_branch_count_8w	-0.033813
date_of_birth_distinct_emails_4w	0.064815
credit_risk_score	-0.032034
email_is_free	-0.023588
phone_home_valid	-0.045306
phone_mobile_valid	0.010982
bank_months_count	-0.048139
proposed_credit_limit	-0.042681
session_length_in_minutes	0.046938
keep_alive_session	0.041924
device_distinct_emails_8w	0.004498
month	-0.005397

	current_address_months_count	customer_age \
income	-0.017391	0.137962
name_email_similarity	0.038234	-0.066315
prev_address_months_count	-0.270593	-0.078717
current_address_months_count	1.000010	0.159740

customer_age	0.159740	1.000010
days_since_request	-0.061145	-0.033966
intended_balcon_amount	0.042259	-0.027465
zip_count_4w	0.040709	-0.016129
velocity_6h	0.021910	-0.028680
velocity_24h	0.015533	-0.006880
velocity_4w	0.022781	-0.004876
bank_branch_count_8w	0.046643	0.041201
date_of_birth_distinct_emails_4w	-0.185124	-0.437193
credit_risk_score	0.115030	0.192823
email_is_free	-0.073249	0.014226
phone_home_valid	0.115907	0.188598
phone_mobile_valid	-0.097486	-0.150076
bank_months_count	0.066281	0.012682
proposed_credit_limit	0.149883	0.175815
session_length_in_minutes	-0.027386	0.029887
keep_alive_session	-0.059696	-0.051321
device_distinct_emails_8w	0.004002	0.042828
month	-0.019638	0.007649

	days_since_request	intended_balcon_amount	\
income	-0.012759	0.045168	
name_email_similarity	-0.006397	0.035757	
prev_address_months_count	0.080489	0.002354	
current_address_months_count	-0.061145	0.042259	
customer_age	-0.033966	-0.027465	
days_since_request	1.000010	0.022183	
intended_balcon_amount	0.022183	1.000010	
zip_count_4w	0.006127	0.007608	
velocity_6h	0.033093	0.045158	
velocity_24h	0.021932	0.070400	
velocity_4w	0.020836	0.059332	
bank_branch_count_8w	-0.019784	0.100776	
date_of_birth_distinct_emails_4w	0.015786	-0.012100	
credit_risk_score	-0.085697	-0.014962	
email_is_free	0.007625	-0.020873	
phone_home_valid	-0.043488	0.006145	
phone_mobile_valid	0.009163	0.042329	
bank_months_count	-0.027671	0.175411	
proposed_credit_limit	-0.064816	0.033850	
session_length_in_minutes	0.049374	0.018501	
keep_alive_session	0.000793	-0.006148	
device_distinct_emails_8w	0.011413	-0.024972	
month	-0.017966	-0.049242	

	zip_count_4w	velocity_6h	velocity_24h	…	\
income	-0.079997	-0.100350	-0.106405	…	

name_email_similarity	0.025874	0.027620	0.038162	…
prev_address_months_count	-0.027014	0.004293	0.017160	…
current_address_months_count	0.040709	0.021910	0.015533	…
customer_age	-0.016129	-0.028680	-0.006880	…
days_since_request	0.006127	0.033093	0.021932	…
intended_balcon_amount	0.007608	0.045158	0.070400	…
zip_count_4w	1.000010	0.133936	0.198823	…
velocity_6h	0.133936	1.000010	0.463804	…
velocity_24h	0.198823	0.463804	1.000010	…
velocity_4w	0.298165	0.400854	0.540729	…
bank_branch_count_8w	0.003664	0.020905	0.041774	…
date_of_birth_distinct_emails_4w	0.121708	0.122253	0.149496	…
credit_risk_score	-0.101330	-0.151244	-0.153062	…
email_is_free	0.024346	0.019873	0.026393	…
phone_home_valid	-0.038401	-0.028620	-0.046243	…
phone_mobile_valid	0.007479	-0.015739	-0.032518	…
bank_months_count	0.049366	0.017426	0.021521	…
proposed_credit_limit	-0.013902	-0.035302	0.003048	…
session_length_in_minutes	0.043676	0.053628	0.064216	…
keep_alive_session	0.017006	0.013564	0.007827	…
device_distinct_emails_8w	0.026515	0.031013	0.036795	…
month	-0.281729	-0.410802	-0.550383	…

	credit_risk_score	email_is_free	\
income	0.195343	-0.013721	
name_email_similarity	0.033618	-0.075405	
prev_address_months_count	-0.032034	-0.023588	
current_address_months_count	0.115030	-0.073249	
customer_age	0.192823	0.014226	
days_since_request	-0.085697	0.007625	
intended_balcon_amount	-0.014962	-0.020873	
zip_count_4w	-0.101330	0.024346	
velocity_6h	-0.151244	0.019873	
velocity_24h	-0.153062	0.026393	
velocity_4w	-0.169070	0.046455	
bank_branch_count_8w	-0.018674	-0.006948	
date_of_birth_distinct_emails_4w	-0.156319	0.034843	
credit_risk_score	1.000010	-0.016162	
email_is_free	-0.016162	1.000010	
phone_home_valid	-0.013571	-0.019829	
phone_mobile_valid	-0.002204	0.030560	
bank_months_count	-0.060329	0.006312	
proposed_credit_limit	0.622599	0.000535	
session_length_in_minutes	-0.029040	0.041217	
keep_alive_session	-0.055943	-0.036244	
device_distinct_emails_8w	-0.028274	0.004740	
month	0.176883	-0.058587	

	phone_home_valid	phone_mobile_valid	\
income	0.012676	0.010555	
name_email_similarity	0.003814	0.015459	
prev_address_months_count	-0.045306	0.010982	
current_address_months_count	0.115907	-0.097486	
customer_age	0.188598	-0.150076	
days_since_request	-0.043488	0.009163	
intended_balcon_amount	0.006145	0.042329	
zip_count_4w	-0.038401	0.007479	
velocity_6h	-0.028620	-0.015739	
velocity_24h	-0.046243	-0.032518	
velocity_4w	-0.047811	-0.031044	
bank_branch_count_8w	0.064747	-0.015609	
date_of_birth_distinct_emails_4w	-0.147730	0.102194	
credit_risk_score	-0.013571	-0.002204	
email_is_free	-0.019829	0.030560	
phone_home_valid	1.000010	-0.275398	
phone_mobile_valid	-0.275398	1.000010	
bank_months_count	0.057916	0.005845	
proposed_credit_limit	-0.014093	-0.025538	
session_length_in_minutes	-0.044417	-0.004365	
keep_alive_session	0.054068	0.023697	
device_distinct_emails_8w	-0.004077	-0.046356	
month	0.048598	0.035254	

	bank_months_count	proposed_credit_limit	\
income	-0.001819	0.135009	
name_email_similarity	-0.011884	0.053971	
prev_address_months_count	-0.048139	-0.042681	
current_address_months_count	0.066281	0.149883	
customer_age	0.012682	0.175815	
days_since_request	-0.027671	-0.064816	
intended_balcon_amount	0.175411	0.033850	
zip_count_4w	0.049366	-0.013902	
velocity_6h	0.017426	-0.035302	
velocity_24h	0.021521	0.003048	
velocity_4w	0.035290	0.038256	
bank_branch_count_8w	0.144942	0.003451	
date_of_birth_distinct_emails_4w	-0.005195	-0.072189	
credit_risk_score	-0.060329	0.622599	
email_is_free	0.006312	0.000535	
phone_home_valid	0.057916	-0.014093	
phone_mobile_valid	0.005845	-0.025538	
bank_months_count	1.000010	-0.026512	
proposed_credit_limit	-0.026512	1.000010	
session_length_in_minutes	0.013237	0.003169	

keep_alive_session	0.006517	-0.053385
device_distinct_emails_8w	0.004568	-0.002564
month	-0.027446	-0.038215

	session_length_in_minutes	keep_alive_session \
income	-0.051490	-0.058315
name_email_similarity	0.000112	0.031243
prev_address_months_count	0.046938	0.041924
current_address_months_count	-0.027386	-0.059696
customer_age	0.029887	-0.051321
days_since_request	0.049374	0.000793
intended_balcon_amount	0.018501	-0.006148
zip_count_4w	0.043676	0.017006
velocity_6h	0.053628	0.013564
velocity_24h	0.064216	0.007827
velocity_4w	0.076175	0.027273
bank_branch_count_8w	0.005549	0.011107
date_of_birth_distinct_emails_4w	-0.033820	0.061008
credit_risk_score	-0.029040	-0.055943
email_is_free	0.041217	-0.036244
phone_home_valid	-0.044417	0.054068
phone_mobile_valid	-0.004365	0.023697
bank_months_count	0.013237	0.006517
proposed_credit_limit	0.003169	-0.053385
session_length_in_minutes	1.000010	-0.056252
keep_alive_session	-0.056252	1.000010
device_distinct_emails_8w	0.078448	-0.053139
month	-0.076507	-0.027120

	device_distinct_emails_8w	month
income	-0.002465	0.119413
name_email_similarity	-0.029268	-0.048978
prev_address_months_count	0.004498	-0.005397
current_address_months_count	0.004002	-0.019638
customer_age	0.042828	0.007649
days_since_request	0.011413	-0.017966
intended_balcon_amount	-0.024972	-0.049242
zip_count_4w	0.026515	-0.281729
velocity_6h	0.031013	-0.410802
velocity_24h	0.036795	-0.550383
velocity_4w	0.045437	-0.847785
bank_branch_count_8w	0.003071	-0.036585
date_of_birth_distinct_emails_4w	-0.021827	-0.238472
credit_risk_score	-0.028274	0.176883
email_is_free	0.004740	-0.058587
phone_home_valid	-0.004077	0.048598
phone_mobile_valid	-0.046356	0.035254

```
bank_months_count                                0.004568 -0.027446
proposed_credit_limit                           -0.002564 -0.038215
session_length_in_minutes                        0.078448 -0.076507
keep_alive_session                              -0.053139 -0.027120
device_distinct_emails_8w                        1.000010 -0.047966
month                                           -0.047966  1.000010

[23 rows x 23 columns]
```

Terdapat 23 kolom numerik yang sudah dilakukan scaling menggunakan StandardScaler

[30]:
```python
# sebelum di scaling
credit_num.cov().round(2)
```

[30]:
```
                                  income  name_email_similarity  \
income                              0.08                  -0.00
name_email_similarity              -0.00                   0.08
prev_address_months_count           0.11                  -0.12
current_address_months_count       -0.45                   0.98
customer_age                        0.49                  -0.24
days_since_request                 -0.02                  -0.01
intended_balcon_amount              0.26                   0.21
zip_count_4w                      -23.36                   7.57
velocity_6h                       -87.80                  24.20
velocity_24h                      -45.70                  16.41
velocity_4w                       -29.18                  11.60
bank_branch_count_8w                0.67                  -0.52
date_of_birth_distinct_emails_4w   -0.11                   0.05
credit_risk_score                   4.11                   0.71
email_is_free                      -0.00                  -0.01
phone_home_valid                    0.00                   0.00
phone_mobile_valid                  0.00                   0.00
bank_months_count                  -0.01                  -0.04
proposed_credit_limit              20.15                   8.07
session_length_in_minutes          -0.12                   0.00
keep_alive_session                 -0.01                   0.00
device_distinct_emails_8w          -0.00                  -0.00
month                               0.08                  -0.03

                                  prev_address_months_count  \
income                                                 0.11
name_email_similarity                                 -0.12
prev_address_months_count                           1852.74
current_address_months_count                       -1028.18
customer_age                                         -41.71
days_since_request                                    18.69
intended_balcon_amount                                 2.01
zip_count_4w                                       -1166.86
```

velocity_6h	555.58
velocity_24h	1090.26
velocity_4w	387.23
bank_branch_count_8w	-662.61
date_of_birth_distinct_emails_4w	14.09
credit_risk_score	-99.80
email_is_free	-0.51
phone_home_valid	-0.95
phone_mobile_valid	0.15
bank_months_count	-25.25
proposed_credit_limit	-942.52
session_length_in_minutes	16.75
keep_alive_session	0.90
device_distinct_emails_8w	0.04
month	-0.52

	current_address_months_count	customer_age	\
income	-0.45	0.49	
name_email_similarity	0.98	-0.24	
prev_address_months_count	-1028.18	-41.71	
current_address_months_count	7792.98	173.60	
customer_age	173.60	151.56	
days_since_request	-29.12	-2.26	
intended_balcon_amount	73.87	-6.70	
zip_count_4w	3606.32	-199.26	
velocity_6h	5815.84	-1061.68	
velocity_24h	2024.10	-125.03	
velocity_4w	1863.30	-55.62	
bank_branch_count_8w	1874.55	230.92	
date_of_birth_distinct_emails_4w	-82.54	-27.18	
credit_risk_score	734.96	171.81	
email_is_free	-3.22	0.09	
phone_home_valid	5.01	1.14	
phone_mobile_valid	-2.74	-0.59	
bank_months_count	71.31	1.90	
proposed_credit_limit	6788.15	1110.43	
session_length_in_minutes	-20.04	3.05	
keep_alive_session	-2.62	-0.31	
device_distinct_emails_8w	0.07	0.11	
month	-3.85	0.21	

	days_since_request	intended_balcon_amount	\
income	-0.02	0.26	
name_email_similarity	-0.01	0.21	
prev_address_months_count	18.69	2.01	
current_address_months_count	-29.12	73.87	
customer_age	-2.26	-6.70	

days_since_request	29.11	2.37
intended_balcon_amount	2.37	392.10
zip_count_4w	33.17	151.17
velocity_6h	536.86	2688.80
velocity_24h	174.66	2057.74
velocity_4w	104.15	1088.53
bank_branch_count_8w	-48.59	908.49
date_of_birth_distinct_emails_4w	0.43	-1.21
credit_risk_score	-33.46	-21.44
email_is_free	0.02	-0.21
phone_home_valid	-0.11	0.06
phone_mobile_valid	0.02	0.27
bank_months_count	-1.82	42.33
proposed_credit_limit	-179.40	343.88
session_length_in_minutes	2.21	3.04
keep_alive_session	0.00	-0.06
device_distinct_emails_8w	0.01	-0.10
month	-0.22	-2.17

	zip_count_4w	velocity_6h	velocity_24h	\
income	-23.36	-87.80	-45.70	
name_email_similarity	7.57	24.20	16.41	
prev_address_months_count	-1166.86	555.58	1090.26	
current_address_months_count	3606.32	5815.84	2024.10	
customer_age	-199.26	-1061.68	-125.03	
days_since_request	33.17	536.86	174.66	
intended_balcon_amount	151.17	2688.80	2057.74	
zip_count_4w	1007050.97	404153.34	294516.86	
velocity_6h	404153.34	9041762.56	2058631.09	
velocity_24h	294516.86	2058631.09	2178930.66	
velocity_4w	277228.51	1116780.94	739532.00	
bank_branch_count_8w	1673.99	28618.34	28073.15	
date_of_birth_distinct_emails_4w	616.87	1856.68	1114.56	
credit_risk_score	-7359.78	-32915.99	-16352.80	
email_is_free	12.17	29.77	19.41	
phone_home_valid	-18.85	-42.10	-33.39	
phone_mobile_valid	2.39	-15.06	-15.27	
bank_months_count	603.76	638.60	387.17	
proposed_credit_limit	-7157.28	-54458.99	2308.14	
session_length_in_minutes	363.32	1336.75	785.77	
keep_alive_session	8.48	20.27	5.74	
device_distinct_emails_8w	5.32	18.66	10.87	
month	-628.23	-2744.88	-1805.31	

	…	credit_risk_score	email_is_free	\
income	…	4.11	-0.00	
name_email_similarity	…	0.71	-0.01	

prev_address_months_count	…	-99.80	-0.51
current_address_months_count	…	734.96	-3.22
customer_age	…	171.81	0.09
days_since_request	…	-33.46	0.02
intended_balcon_amount	…	-21.44	-0.21
zip_count_4w	…	-7359.78	12.17
velocity_6h	…	-32915.99	29.77
velocity_24h	…	-16352.80	19.41
velocity_4w	…	-11337.78	21.44
bank_branch_count_8w	…	-615.34	-1.58
date_of_birth_distinct_emails_4w	…	-57.14	0.09
credit_risk_score	…	5238.57	-0.58
email_is_free	…	-0.58	0.25
phone_home_valid	…	-0.48	-0.00
phone_mobile_valid	…	-0.05	0.00
bank_months_count	…	-53.22	0.04
proposed_credit_limit	…	23118.60	0.14
session_length_in_minutes	…	-17.42	0.17
keep_alive_session	…	-2.01	-0.01
device_distinct_emails_8w	…	-0.41	0.00
month	…	28.45	-0.06

	phone_home_valid	phone_mobile_valid	\
income	0.00	0.00	
name_email_similarity	0.00	0.00	
prev_address_months_count	-0.95	0.15	
current_address_months_count	5.01	-2.74	
customer_age	1.14	-0.59	
days_since_request	-0.11	0.02	
intended_balcon_amount	0.06	0.27	
zip_count_4w	-18.85	2.39	
velocity_6h	-42.10	-15.06	
velocity_24h	-33.39	-15.27	
velocity_4w	-21.67	-9.15	
bank_branch_count_8w	14.42	-2.26	
date_of_birth_distinct_emails_4w	-0.37	0.16	
credit_risk_score	-0.48	-0.05	
email_is_free	-0.00	0.00	
phone_home_valid	0.24	-0.04	
phone_mobile_valid	-0.04	0.10	
bank_months_count	0.35	0.02	
proposed_credit_limit	-3.54	-4.17	
session_length_in_minutes	-0.18	-0.01	
keep_alive_session	0.01	0.00	
device_distinct_emails_8w	-0.00	-0.00	
month	0.05	0.02	

	bank_months_count	proposed_credit_limit	\
income	-0.01	20.15	
name_email_similarity	-0.04	8.07	
prev_address_months_count	-25.25	-942.52	
current_address_months_count	71.31	6788.15	
customer_age	1.90	1110.43	
days_since_request	-1.82	-179.40	
intended_balcon_amount	42.33	343.88	
zip_count_4w	603.76	-7157.28	
velocity_6h	638.60	-54458.99	
velocity_24h	387.17	2308.14	
velocity_4w	398.50	18184.64	
bank_branch_count_8w	804.22	806.08	
date_of_birth_distinct_emails_4w	-0.32	-187.06	
credit_risk_score	-53.22	23118.60	
email_is_free	0.04	0.14	
phone_home_valid	0.35	-3.54	
phone_mobile_valid	0.02	-4.17	
bank_months_count	148.54	-165.77	
proposed_credit_limit	-165.77	263210.30	
session_length_in_minutes	1.34	13.48	
keep_alive_session	0.04	-13.61	
device_distinct_emails_8w	0.01	-0.26	
month	-0.74	-43.57	

	session_length_in_minutes	\
income	-0.12	
name_email_similarity	0.00	
prev_address_months_count	16.75	
current_address_months_count	-20.04	
customer_age	3.05	
days_since_request	2.21	
intended_balcon_amount	3.04	
zip_count_4w	363.32	
velocity_6h	1336.75	
velocity_24h	785.77	
velocity_4w	585.06	
bank_branch_count_8w	20.94	
date_of_birth_distinct_emails_4w	-1.42	
credit_risk_score	-17.42	
email_is_free	0.17	
phone_home_valid	-0.18	
phone_mobile_valid	-0.01	
bank_months_count	1.34	
proposed_credit_limit	13.48	
session_length_in_minutes	68.72	
keep_alive_session	-0.23	

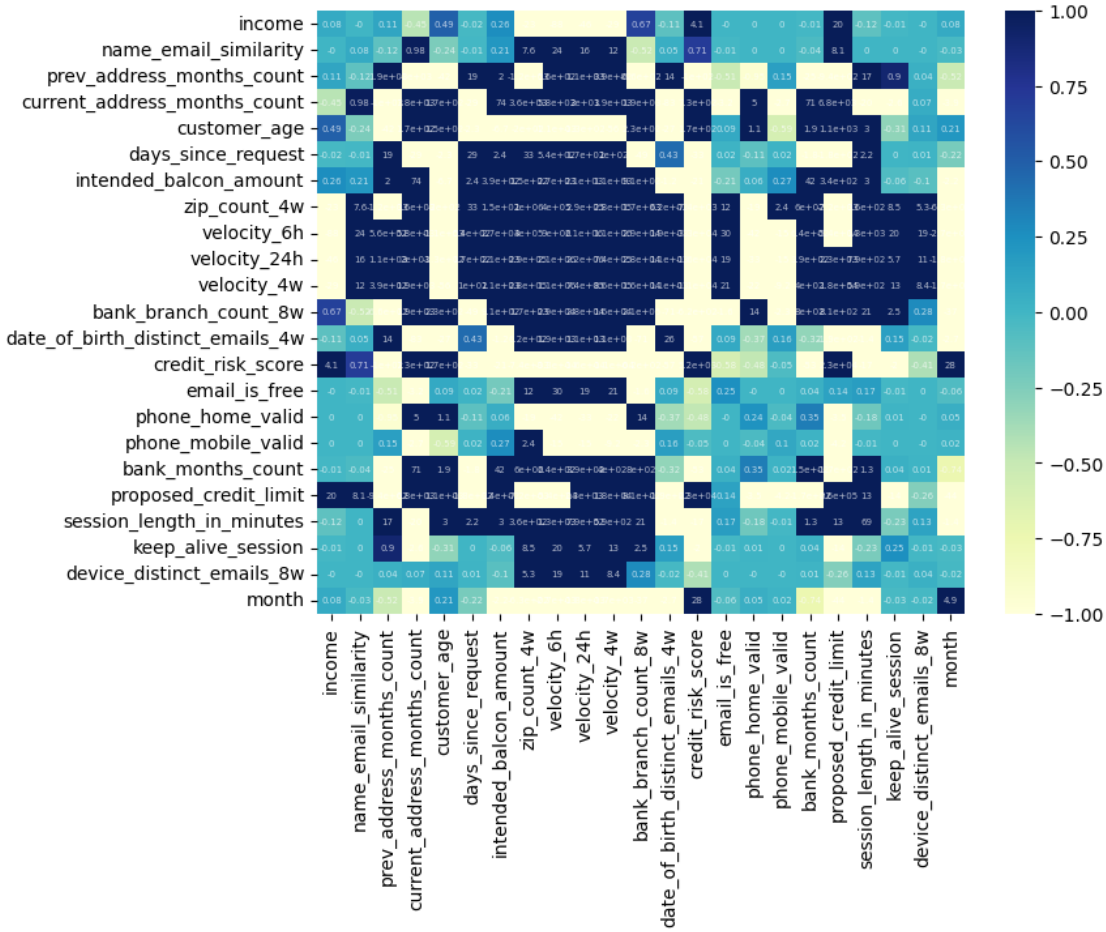
device_distinct_emails_8w	0.13
month	-1.41

	keep_alive_session \
income	-0.01
name_email_similarity	0.00
prev_address_months_count	0.90
current_address_months_count	-2.62
customer_age	-0.31
days_since_request	0.00
intended_balcon_amount	-0.06
zip_count_4w	8.48
velocity_6h	20.27
velocity_24h	5.74
velocity_4w	12.56
bank_branch_count_8w	2.51
date_of_birth_distinct_emails_4w	0.15
credit_risk_score	-2.01
email_is_free	-0.01
phone_home_valid	0.01
phone_mobile_valid	0.00
bank_months_count	0.04
proposed_credit_limit	-13.61
session_length_in_minutes	-0.23
keep_alive_session	0.25
device_distinct_emails_8w	-0.01
month	-0.03

	device_distinct_emails_8w	month
income	-0.00	0.08
name_email_similarity	-0.00	-0.03
prev_address_months_count	0.04	-0.52
current_address_months_count	0.07	-3.85
customer_age	0.11	0.21
days_since_request	0.01	-0.22
intended_balcon_amount	-0.10	-2.17
zip_count_4w	5.32	-628.23
velocity_6h	18.66	-2744.88
velocity_24h	10.87	-1805.31
velocity_4w	8.42	-1745.46
bank_branch_count_8w	0.28	-37.01
date_of_birth_distinct_emails_4w	-0.02	-2.68
credit_risk_score	-0.41	28.45
email_is_free	0.00	-0.06
phone_home_valid	-0.00	0.05
phone_mobile_valid	-0.00	0.02
bank_months_count	0.01	-0.74

```
proposed_credit_limit                                  -0.26   -43.57
session_length_in_minutes                               0.13    -1.41
keep_alive_session                                     -0.01    -0.03
device_distinct_emails_8w                               0.04    -0.02
month                                                  -0.02     4.94

[23 rows x 23 columns]
```

[31]:
```
# alternatif menggunakan seaborn heatmap, sebelum dilakukan scaled

plt.figure(figsize=(8, 6), dpi=100)
sns.heatmap(credit_num.cov().round(2), vmin=-1, vmax=1, annot=True,␣
↪cmap='YlGnBu',
            annot_kws={"size": 5, "color":'white', "alpha":0.7, "ha": 'center',␣
↪"va": 'center'});
```

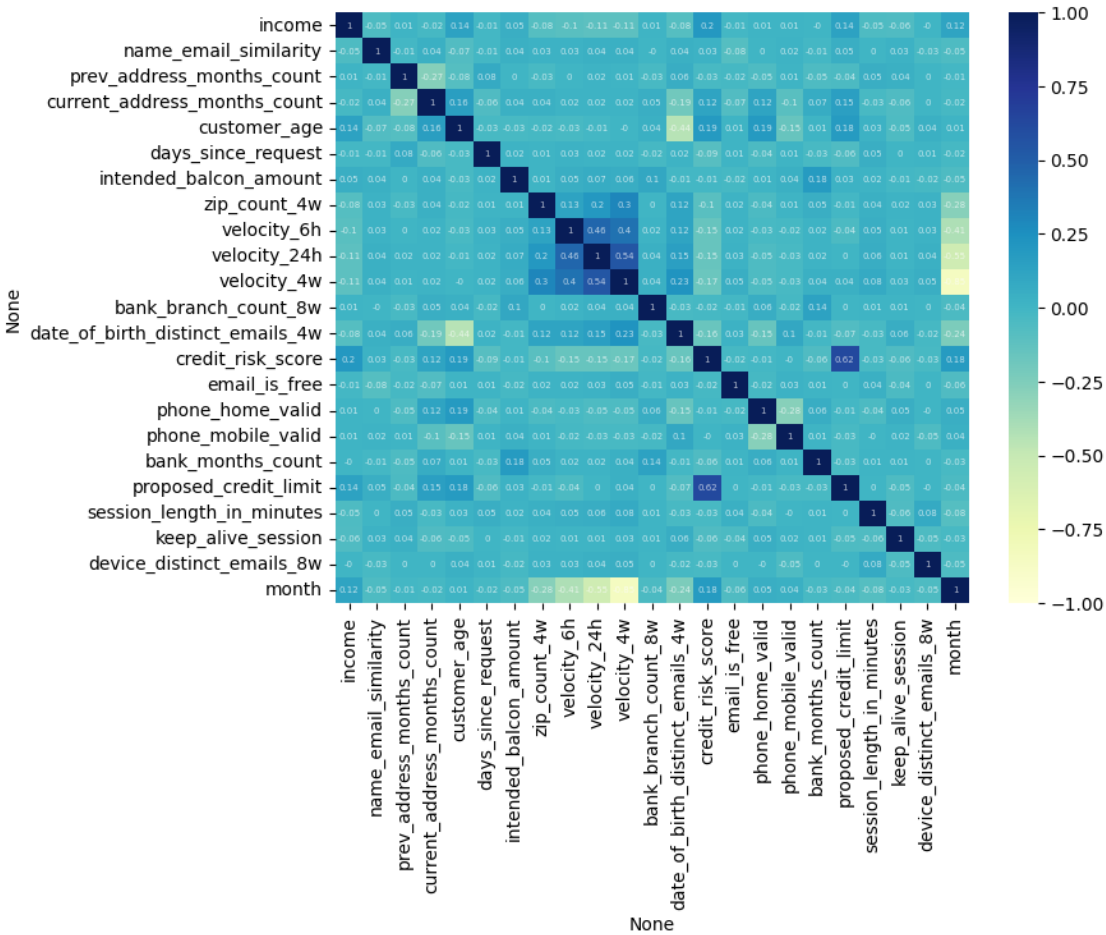


Di atas adalah distribusi nilai covariance dari data yang belum distandarisasi (scale). Variance dari masing-masing variabel berbeda jauh karena range/skala dari tiap variabel berbeda, begitupun

covariance. **Nilai variance dan covariance dipengaruhi oleh skala dari data**. Semakin tinggi skala, nilai variance atau covariance akan semakin tinggi.

Data dengan perbedaan skala antar variabel yang tinggi tidak baik untuk langsung dianalisis PCA karena dapat menimbulkan bias.

[32]:
```
# alternatif menggunakan seaborn heatmap, setelah dilakukan scaled
plt.figure(figsize=(8, 6), dpi=100)
sns.heatmap(credit_scaled.cov().round(2), vmin=-1, vmax=1, annot=True,␣
↪cmap='YlGnBu',
            annot_kws={"size": 5, "color":'white', "alpha":0.7, "ha": 'center',␣
↪"va": 'center'});
```



38.2.4 Principal Component Analysis menggunakan library sklearn

Setelah melakukan scaling, kita dapat menerapkan PCA pada data yang telah diubah skala, seperti yang ditunjukkan dalam kode berikut: menggunakan `credit_scaled`

[33]:
```
# membuat model PCA
pca = PCA(svd_solver='full') # implementasi full svd sehingga mendapatkan semua␣
↪PC yang terbentuk
pca.fit(credit_scaled)
```

[33]:
```
PCA(svd_solver='full')
```

[34]:
```
pca.n_components_
```

[34]:
```
23
```

`pca.components_` : berisi nilai *eigen vector* yang akan dijadikan formula untuk PC baru

[35]:
```
pd.DataFrame(pca.components_)
```

[35]:
```
           0         1         2         3         4         5         6   \
0  -0.151037  0.040877  0.030732 -0.037863 -0.115780  0.043330  0.050360
1   0.131211  0.027067 -0.200700  0.359607  0.431709 -0.111579  0.068807
2  -0.178849 -0.101274 -0.134318  0.157163  0.167072  0.021271  0.036078
3  -0.009587  0.213531 -0.262366  0.222816 -0.246917 -0.170547  0.438431
4   0.242364 -0.279994  0.235604 -0.217804  0.135823  0.228995  0.375115
5   0.188357  0.155747  0.531049 -0.309087  0.038336  0.119368  0.185062
6  -0.235646  0.516401  0.077805  0.167756 -0.062101  0.314409  0.132201
7   0.172621 -0.112981 -0.114172  0.128873  0.214158  0.456829  0.127624
8  -0.417507  0.173197  0.009419 -0.031988  0.112507  0.106199 -0.056853
9  -0.021937  0.202547 -0.046761  0.028459 -0.244928  0.266050  0.170098
10  0.121905 -0.113744 -0.121785  0.111820 -0.058301  0.598057 -0.014284
11 -0.206624 -0.062817 -0.166514  0.069482 -0.104654  0.332235 -0.175366
12 -0.199769 -0.423826  0.131820  0.065009 -0.081208  0.085576 -0.264096
13 -0.308344 -0.514776 -0.014358  0.156806 -0.173163 -0.032976  0.304332
14  0.547238 -0.026695 -0.390116  0.038245 -0.128592 -0.002828 -0.044823
15 -0.108939 -0.127094  0.013075  0.044177 -0.046784 -0.124393  0.580360
16  0.111762 -0.046885  0.215445  0.223301 -0.199189 -0.009768 -0.079542
17  0.128360  0.074671  0.020785  0.057311 -0.070542 -0.070303  0.020096
18  0.170970  0.010739  0.499058  0.697312 -0.107183 -0.046154 -0.096163
19 -0.088448  0.041605  0.061080  0.081162  0.543356 -0.009293  0.092915
20 -0.047239  0.024611  0.028759  0.094929  0.384675  0.012462  0.020061
21  0.037005 -0.000327  0.004650 -0.028080  0.006940 -0.021828 -0.033059
22 -0.007561  0.006024 -0.004642 -0.002325  0.000733 -0.003045 -0.007773

           7         8         9   …        13        14        15        16  \
0   0.239616  0.352018  0.414803  … -0.231746  0.043622 -0.072833  0.010492
1   0.069862  0.106972  0.156302  …  0.352513 -0.015874  0.202134 -0.217273
2  -0.007396  0.016889 -0.022723  … -0.462617 -0.047896  0.425957 -0.324666
3   0.041106 -0.051532 -0.049706  …  0.021995 -0.146909 -0.062562  0.254078
4  -0.055353  0.001611  0.019117  … -0.024789  0.225189 -0.162471  0.169431
5  -0.155093  0.035027  0.060782  …  0.103966 -0.252733  0.311583 -0.251187
6  -0.021946  0.028791 -0.000634  … -0.009434 -0.491929 -0.118447  0.005524
```

```
7  -0.103286  0.183392  0.118983  … -0.113106 -0.098916 -0.166988  0.262103
8   0.119617 -0.082872 -0.066426  …  0.085306  0.473197  0.042277  0.120429
9  -0.164871 -0.000459 -0.012297  …  0.028219  0.368136  0.228230 -0.389565
10  0.477876 -0.240225 -0.167987  …  0.059772  0.036078  0.067682 -0.094456
11 -0.476396  0.171030  0.080309  …  0.090041  0.111344 -0.059934  0.027938
12  0.244599 -0.113872 -0.049655  …  0.079829 -0.389076 -0.079060 -0.149737
13 -0.314724  0.141569  0.050763  …  0.107626 -0.116529  0.011363 -0.124214
14 -0.141151  0.000537 -0.025960  … -0.065052 -0.086060  0.105649 -0.075051
15  0.257813 -0.166999 -0.047259  … -0.036045  0.071306  0.029780 -0.047163
16  0.303979  0.580973  0.094537  …  0.082986  0.086032  0.364064  0.264860
17  0.170067  0.277484  0.086507  …  0.003160  0.130061 -0.606180 -0.572511
18 -0.180817 -0.249173 -0.076325  … -0.123527  0.179212 -0.104604  0.040148
19 -0.036077  0.332556 -0.534780  … -0.059841 -0.020769 -0.104467 -0.013200
20 -0.001482 -0.289882  0.653228  … -0.018125  0.003599 -0.023472  0.024701
21 -0.002987  0.001374 -0.007820  … -0.716723 -0.005595 -0.014360  0.013627
22 -0.016671  0.006550  0.010857  … -0.015082  0.010368 -0.001510 -0.003617

          17        18        19        20        21        22
0   0.037212 -0.092509  0.070897  0.039108  0.037528 -0.483264
1   0.066798  0.417700  0.032357 -0.100973  0.054958 -0.179102
2   0.211714 -0.476955  0.000565  0.062404  0.064336  0.065935
3   0.451115  0.046819 -0.207881  0.112532 -0.234753  0.050471
4   0.330485 -0.021074  0.360367 -0.250153  0.206724  0.020152
5   0.023646  0.110266 -0.144659  0.397021 -0.183692 -0.041902
6  -0.073182  0.004529  0.397022 -0.132294  0.218359  0.068536
7  -0.182277 -0.146097 -0.243146  0.002718 -0.503271 -0.011479
8   0.040502  0.095741  0.395842  0.409462 -0.364502  0.017052
9  -0.001579  0.071445 -0.025815 -0.523836 -0.284846  0.011334
10  0.150392  0.103756 -0.173721  0.250832  0.276410 -0.006428
11 -0.200212  0.109315 -0.179733  0.145609  0.274389  0.011601
12 -0.005072  0.090865  0.173012 -0.238445 -0.409997  0.000138
13  0.143333  0.150016  0.145484  0.221379  0.043604  0.034776
14 -0.190783 -0.072610  0.552374  0.246501 -0.119041 -0.027070
15 -0.683521 -0.020119 -0.034851  0.025647  0.075767  0.007789
16 -0.065003  0.020630  0.056199 -0.032660  0.040016  0.295578
17  0.054067 -0.057457 -0.038607  0.184381 -0.036715  0.192215
18 -0.019240 -0.086775  0.019397  0.071578  0.001405 -0.136164
19 -0.033328 -0.033491  0.010747 -0.013980 -0.008865 -0.012687
20 -0.005242 -0.047001  0.025411  0.011798  0.003196  0.251476
21 -0.016415  0.682026 -0.010833  0.004329 -0.015434  0.070253
22 -0.004895  0.012329 -0.000285 -0.000011  0.001875  0.711832

[23 rows x 23 columns]
```

Rasio Varians yang Dijelaskan

Rasio varians yang dijelaskan memberikan informasi proporsi tentang seberapa banyak informasi yang diatributkan pada setiap PC. Berdasarkan informasi di bawah ini, kita mendapatkan informasi

bahwa PC1 mengandung sekitar 13,18% informasi, PC2 sebanyak 0,08%, dan PC3 sebanyak 0,06%.

[36]:
```
pca.explained_variance_ratio_
```

[36]:
```
array([0.1318414 , 0.08761348, 0.06657119, 0.05687241, 0.05320949,
       0.0503121 , 0.04685173, 0.04350163, 0.04271713, 0.04131351,
       0.04033185, 0.03925077, 0.03842389, 0.03796283, 0.03597653,
       0.03405586, 0.03011896, 0.02978514, 0.02918552, 0.02171452,
       0.02086706, 0.01493503, 0.00658796])
```

Untuk lebih mudah dipahami, kita dapat menghitung proporsi kumulatif dari PC1 hingga PC20. Berikut adalah output yang menyatakan bahwa seluruh dimensi dalam PC mengandung sekitar 95.77% informasi dari dataset asli kita.

[37]:
```
np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100)
```

[37]:
```
array([ 13.18,  21.94,  28.6 ,  34.29,  39.61,  44.64,  49.33,  53.68,
        57.95,  62.08,  66.11,  70.04,  73.88,  77.68,  81.28,  84.69,
        87.7 ,  90.68,  93.6 ,  95.77,  97.86,  99.35, 100.01])
```

Observasi:

Dalam output yang diberikan diatas, nilai pada setiap posisi adalah kumulatif dari rasio varians yang dijelaskan oleh tiap komponen utama. Ketika kita melihat persentase kumulatif tersebut, kita mencari titik di mana persentase varians yang dijelaskan sudah cukup tinggi untuk kebutuhan analisis kita.

Dari hasil yang diberikan, tampaknya sekitar 95.77% dari varians dapat dijelaskan oleh 20 komponen utama pertama. Oleh karena itu, Anda mungkin memilih untuk mengambil 20 komponen utama ini, yang mencakup sebagian besar varians data Anda.

Namun, keputusan akhir bergantung pada kebutuhan spesifik analisis Anda. Beberapa praktik umum adalah memilih jumlah komponen utama yang menjelaskan sekitar 80-95% dari varians total, tetapi Anda dapat memilih nilai yang sesuai dengan tujuan dan konteks analisis Anda.

[38]:
```
# Nilai dimensi baru dari PCA Model
credit_pca = pd.DataFrame(pca.transform(credit_scaled),
                          columns=[f'PC{i+1}' for i in range(pca.
 ↪n_components_)])
credit_pca.head()
```

[38]:
```
        PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0  1.692755  3.323030 -0.889182  0.568785  1.539485  0.669139 -0.882377
1  0.853039  0.094970  1.362371 -1.279626 -1.114656 -1.041867 -1.349552
2  0.933672  2.583289 -0.083890 -1.519840  1.423196 -0.311211 -1.014184
3  1.463710 -2.074312  0.009756  0.335156 -0.323293 -0.674480 -1.214947
4 -2.179767  0.666897 -1.064235 -0.738059 -0.202709  0.682306 -1.596558

        PC8       PC9      PC10  …      PC14      PC15      PC16      PC17  \
0 -0.816128  0.464054  0.338666  … -2.648092  0.622580  0.200997  1.858074
```

```
1 -1.287512 -0.509089  1.731860  … -0.446959  0.185748  1.788388  1.661765
2 -2.772364 -0.670852 -2.078549  …  1.083076 -1.209996 -0.610770 -1.318451
3 -0.397285  0.673089 -0.550966  …  0.111886 -0.864106  0.344133  0.582058
4  0.048435  0.442683 -0.068521  …  0.599069  0.769384  0.144125  0.493707

       PC18      PC19      PC20      PC21      PC22      PC23
0 -0.260204 -1.150616 -0.780847 -0.111586 -0.012969 -0.163692
1  1.457426 -0.587829 -0.324188 -0.340631 -0.273005  0.074517
2 -1.986781 -0.225584 -1.288779  0.460784  0.320538  0.082087
3  0.913164 -0.487811 -0.710987  0.779752  0.278370 -0.591655
4 -0.715133 -0.387881 -1.316316 -0.288874 -0.550302  0.135780

[5 rows x 23 columns]
```

[39]:
```
credit_pca.shape
```

[39]:
```
(100000, 23)
```

[40]:
```
credit_pca20 = credit_pca.iloc[:,0:20]
credit_pca20.head()
```

[40]:
```
        PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0  1.692755  3.323030 -0.889182  0.568785  1.539485  0.669139 -0.882377
1  0.853039  0.094970  1.362371 -1.279626 -1.114656 -1.041867 -1.349552
2  0.933672  2.583289 -0.083890 -1.519840  1.423196 -0.311211 -1.014184
3  1.463710 -2.074312  0.009756  0.335156 -0.323293 -0.674480 -1.214947
4 -2.179767  0.666897 -1.064235 -0.738059 -0.202709  0.682306 -1.596558

        PC8       PC9      PC10      PC11      PC12      PC13      PC14  \
0 -0.816128  0.464054  0.338666 -0.942384  1.925547  1.243251 -2.648092
1 -1.287512 -0.509089  1.731860  2.152441 -1.882548  1.821081 -0.446959
2 -2.772364 -0.670852 -2.078549  2.107466  1.285194 -2.136757  1.083076
3 -0.397285  0.673089 -0.550966  1.103438 -0.722769  0.301028  0.111886
4  0.048435  0.442683 -0.068521  0.133925  0.679665 -0.644503  0.599069

       PC15      PC16      PC17      PC18      PC19      PC20
0  0.622580  0.200997  1.858074 -0.260204 -1.150616 -0.780847
1  0.185748  1.788388  1.661765  1.457426 -0.587829 -0.324188
2 -1.209996 -0.610770 -1.318451 -1.986781 -0.225584 -1.288779
3 -0.864106  0.344133  0.582058  0.913164 -0.487811 -0.710987
4  0.769384  0.144125  0.493707 -0.715133 -0.387881 -1.316316
```

[41]:
```
credit_scaled.head(3)
```

[41]:
```
     income name_email_similarity prev_address_months_count  \
0  1.125802              0.454534                 -0.387324
1  0.094766             -1.410073                 -0.387324
```

```
2 -0.248913            -1.395895                -0.387324

  current_address_months_count customer_age days_since_request  \
0                    -0.431527     0.459337          -0.188833
1                     0.406739    -1.165255          -0.189457
2                    -0.590118     1.271634          -0.189578

  intended_balcon_amount zip_count_4w velocity_6h velocity_24h  …  \
0              -0.442254     1.075569    1.790845     1.827162  …
1              -0.455380     4.005273    0.226647    -0.256489  …
2              -0.468534    -0.226854   -1.418498     1.297549  …

  credit_risk_score email_is_free phone_home_valid phone_mobile_valid  \
0          1.623733      0.916752         1.233515           0.359129
1         -0.324384      0.916752         1.233515          -2.784511
2          0.877646      0.916752         1.233515           0.359129

  bank_months_count proposed_credit_limit session_length_in_minutes  \
0         -0.144869              1.860207                  0.655308
1         -0.965382             -0.673720                 -0.118348
2          1.414106              1.860207                 -0.380592

  keep_alive_session device_distinct_emails_8w     month
0          -1.114677                 -0.124266 -1.038034
1          -1.114677                 -0.124266  0.312032
2           0.897121                  4.874361 -1.488056

[3 rows x 23 columns]
```

38.3 Visualisasi PCA

38.3.1 Visualisasi Biplot

Biplot adalah visualisasi data multivariat yang menggabungkan **individual factor map** dan **variables factor map**.

PCA tidak hanya berguna untuk dimensionality reduction namun baik untuk visualisasi high-dimensional data. Visualisasi dapat menggunakan **biplot** yang menampilkan:

1. **Individual factor map**, yaitu sebaran data secara keseluruhan menggunakan 2 PC. Tujuannya untuk:

- observasi yang serupa
- outlier dari keseluruhan data

2. **Variables factor map**, yaitu plot yang menunjukkan korelasi antar variable dan kontribusinya terhadap PC.

[42]:
```
sample_plot = credit_scaled.iloc[0:50, :] # ambil sample data
sample_plot.head(3)
```

[42]:

```
   income name_email_similarity prev_address_months_count  \
0  1.125802              0.454534                 -0.387324
1  0.094766             -1.410073                 -0.387324
2 -0.248913             -1.395895                 -0.387324

  current_address_months_count customer_age days_since_request  \
0                    -0.431527     0.459337          -0.188833
1                     0.406739    -1.165255          -0.189457
2                    -0.590118     1.271634          -0.189578

  intended_balcon_amount zip_count_4w velocity_6h velocity_24h  …  \
0              -0.442254     1.075569    1.790845     1.827162  …
1              -0.455380     4.005273    0.226647    -0.256489  …
2              -0.468534    -0.226854   -1.418498     1.297549  …

  credit_risk_score email_is_free phone_home_valid phone_mobile_valid  \
0          1.623733      0.916752         1.233515           0.359129
1         -0.324384      0.916752         1.233515          -2.784511
2          0.877646      0.916752         1.233515           0.359129

  bank_months_count proposed_credit_limit session_length_in_minutes  \
0         -0.144869              1.860207                  0.655308
1         -0.965382             -0.673720                 -0.118348
2          1.414106              1.860207                 -0.380592

  keep_alive_session device_distinct_emails_8w     month
0          -1.114677                 -0.124266 -1.038034
1          -1.114677                 -0.124266  0.312032
2           0.897121                  4.874361 -1.488056

[3 rows x 23 columns]
```

[43]:

```
sample_plot.shape
```

[43]: (50, 23)

[44]:

```
import plotly.express as px
# ambil nama tiap kolom balance_saled sesuai urutan di df
features = list(sample_plot.columns.values)
# hasil PCA. sama seperti objek transform_ di atas, tapi beda nama kolom
components = pca.fit_transform(sample_plot)
# dot product/perkalian matriks antara eigenvector dengan akar kuadrat eigen␣
↪value
loadings = pca.components_.T * np.sqrt(pca.explained_variance_)

# Pilih fitur dengan loadings terbesar pada PC1 dan PC2
```

```
important_features = [features[i][0] for i in np.abs(loadings[:, :5]).
 ↪argmax(axis=0)]

fig = px.scatter(components, x=0, y=1, hover_name=list(range(len(components))))

for i, feature in enumerate(features):
    # Hanya tambahkan garis panah dan anotasi untuk fitur yang penting
    if feature[0] in important_features:
        fig.add_shape(
            type='line',
            x0=0, y0=0,
            x1=loadings[i, 0],
            y1=loadings[i, 1],
        )

        fig.add_annotation(
            x=loadings[i, 0],
            y=loadings[i, 1],
            ax=0, ay=0,
            xanchor="center",
            yanchor="bottom",
            text=features[i][0],
        )

        fig.add_annotation(
            x=loadings[i, 0],
            y=loadings[i, 1],
            ax=0, ay=0,
            xref='x', yref='y',
            axref='x', ayref='y',
            text='',  # if you want only the arrow
            showarrow=True,
            arrowhead=4,
            arrowsize=1.5,
            arrowwidth=1,
            arrowcolor='black'
        )

# Plotly figure layout
fig.update_layout(title='Biplot PCA', title_x=0.5, width=1000, height=800)
fig.update_xaxes(title_text='PC1')
fig.update_yaxes(title_text='PC2')
fig.show()
```

Individual Factor Map

Pada **individual factor map**, observasi diplot berdasarkan skor mereka pada PC1 dan PC2. Observasi yang serupa akan berada dekat satu sama lain, sedangkan observasi yang berbeda akan

berada jauh satu sama lain.

Pada biplot yang Anda kirimkan, observasi 1, 2, dan 3 berada dekat satu sama lain, menunjukkan bahwa mereka memiliki karakteristik yang mirip. Observasi 4, 5, dan 6 juga berada dekat satu sama lain, menunjukkan bahwa mereka juga memiliki karakteristik yang mirip. Observasi 7, 8, dan 9 berada jauh dari observasi lainnya, menunjukkan bahwa mereka memiliki karakteristik yang berbeda.

Outlier

Outlier adalah observasi yang berada jauh dari distribusi data lainnya. Pada biplot, outlier biasanya terletak di tepi plot. Pada biplot diatas, observasi 42 adalah outlier. Observasi ini berada jauh di tepi plot, menunjukkan bahwa observasi ini memiliki karakteristik yang sangat berbeda dari observasi lainnya.

Variables Factor Map

Pada **variables factor map**, variabel diplot berdasarkan skor mereka pada PC1 dan PC2. Korelasi antar variabel dapat dilihat dari sudut antar dua panah

- Panah saling berdekatan (sudut antar panah < 90), maka korelasi positif
- Panah saling tegak lurus (sudut antar panah $= 90$), maka tidak berkorelasi
- Panah saling bertolak belakang (sudut antar panah mendekati 180), maka korelasi negatif

Contoh interpretasi plot diatas:

- Variabel `bank_branch_count_8w` dan `phone_mobile_valid` berkorelasi positif. Variabel ini berada dekat satu sama lain, menunjukkan bahwa variabel ini cenderung meningkat atau menurun bersama-sama.
- Variabel `bank_branch_count_8w` dan `month` tidak berkorelasi.
- Variabel `month` dan `name_email_similarity` berkorelasi negatif. Variabel ini berada jauh satu sama lain, menunjukkan bahwa variabel ini cenderung meningkat atau menurun secara berlawanan.

Kesimpulan Berdasarkan biplot diatas, dapat disimpulkan bahwa: …

38.3.2 [Optional] Visualisasi explained variance menggunakan library

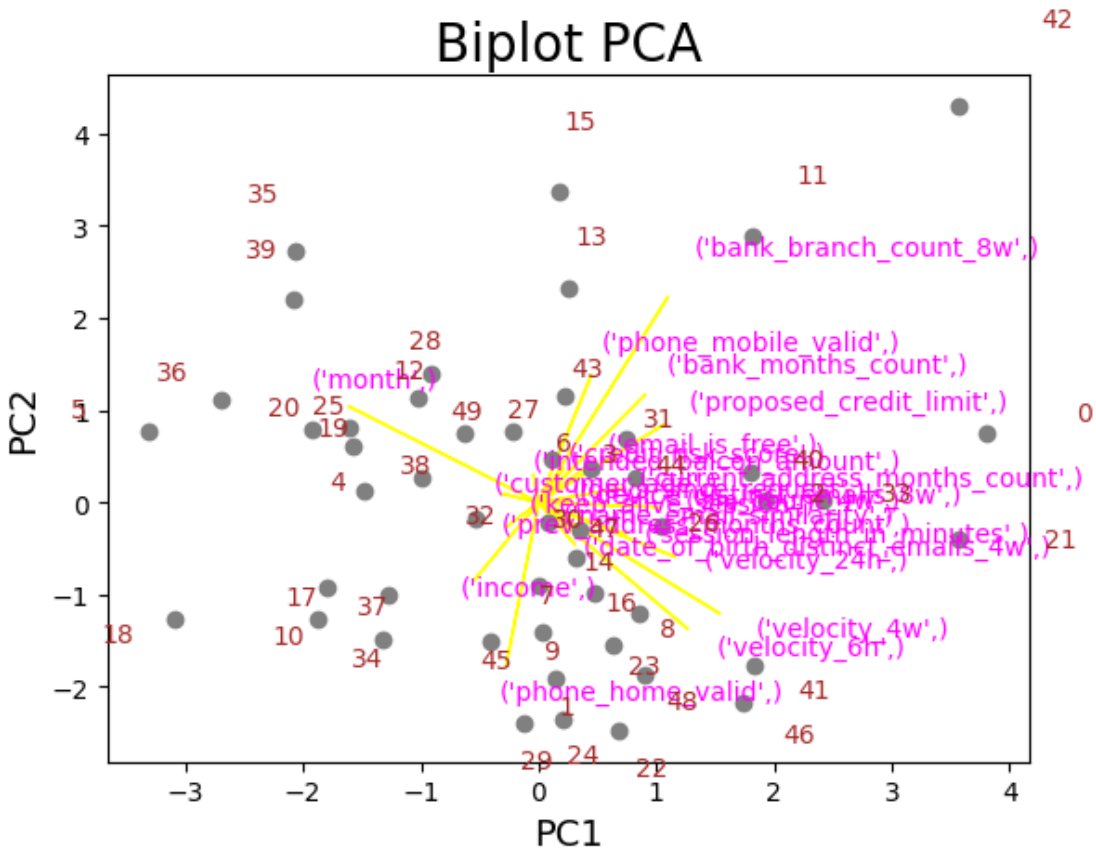
[45]:
```
exp_var_cumul = np.cumsum(np.round(pca.explained_variance_ratio_,␣
↪decimals=4)*100)

px.area(
    x=range(1, exp_var_cumul.shape[0] + 1),
    y=exp_var_cumul,
    labels={"x": "Jumlah Components", "y": "Explained Variance"}
)
```

Plot di atas adalah plot kumulatif varians yang dijelaskan oleh komponen utama dalam analisis PCA. Pada sumbu x, kita memiliki jumlah komponen utama, sedangkan sumbu y menunjukkan persentase varians yang dijelaskan. Kurva area menunjukkan akumulasi persentase varians seiring

penambahan komponen utama. Poin penting pada kurva adalah di mana Anda memutuskan jumlah komponen yang akan diambil untuk menjelaskan sejumlah tertentu dari varians total dalam data.

38.3.3 [Optional] Simple biplot : helper.py

[46]:
```
biplot_pca(sample_plot)
```



38.3.4 [Optional] Scree Plot Proporsi Eigenvalue

[47]:
```
import plotly.graph_objects as go

# Dapatkan nilai eigenvalue dari komponen utama
eigenvalues = pca.explained_variance_

# Hitung proporsi variasi yang dijelaskan oleh setiap komponen utama
explained_var_ratio = pca.explained_variance_ratio_

# Buat scree plot menggunakan plotly
fig = go.Figure()
```

```
# Plot eigenvalue
fig.add_trace(go.Scatter(x=list(range(1, len(eigenvalues) + 1)),
                         y=eigenvalues, mode='lines+markers',␣
 ↪name='Eigenvalue'))

# Plot proporsi variasi yang dijelaskan
fig.add_trace(go.Scatter(x=list(range(1, len(explained_var_ratio) + 1)),
                         y=explained_var_ratio, mode='lines+markers',
                         name='Explained Variance Ratio'))

# Atur layout dan tampilkan
fig.update_layout(title='Scree Plot',
                  xaxis_title='Principal Component (PC)',
                  yaxis_title='Eigenvalue / Explained Variance Ratio',
                  showlegend=True,
                  width=1200, height=800)

fig.show()
```

Dalam contoh ini, scree plot menunjukkan eigenvalue dan proporsi variasi yang dijelaskan oleh setiap komponen utama. Anda dapat mengatur `ncp` sesuai dengan jumlah komponen utama yang ingin Anda tampilkan pada plot. Namun, dalam hal ini kita pakai semua PC nya yakni 23

[48]:
```
# Buat dataframe untuk hasil PCA
pca_df = credit_pca20.copy()#pd.DataFrame(data=x_pca, columns=[f'PC{i+1}' for i␣
 ↪in range(x_pca.shape[1])])

# Tambahkan kolom kelas (jika ada)
# Jika x memiliki kolom kelas, ganti 'class_column_name' dengan nama kolomnya
pca_df['fraud_bool'] = credit_clean['fraud_bool'].astype(str)

# Buat scatter plot PCA
fig = px.scatter(pca_df, x='PC1', y='PC2',
                 title='Plot PCA berdasarkan Label',
                 color='fraud_bool',
                #   color=outliers.astype(str),
                 color_discrete_map={'0': '#a6c4ff', '1': '#ffa07a'})

# Atur layout dan tampilkan plot
fig.update_layout(xaxis_title='PC 1', yaxis_title='PC 2',
                  width=1000, height=800)
fig.show()
```

Tujuan dari plot PCA berdasarkan label dengan PC1 dan PC2 adalah untuk memvisualisasikan distribusi data pada dimensi-dimensi utama (PC1 dan PC2) setelah proses reduksi dimensi menggunakan metode PCA. Plot tersebut juga memberikan pemahaman visual tentang bagaimana kelas atau label data tersebar dalam ruang yang direduksi dimensinya, membantu mengidentifikasi pola

atau cluster yang mungkin ada dalam data.

Important:

Jika pada data awal tidak ada kolom label (misalnya, kolom target seperti 'fraud_bool'), maka melakukan plot PCA berdasarkan label tidak akan memberikan informasi yang relevan atau bermakna. Plot tersebut berguna ketika kita ingin melihat pola atau cluster berdasarkan kelas atau label tertentu dalam data.

Jika tidak ada kolom label, lebih relevan untuk fokus pada visualisasi distribusi data berdasarkan komponen utama (PC1 dan PC2) tanpa mempertimbangkan kelas atau label. Plot tersebut dapat memberikan wawasan tentang struktur internal data tanpa melibatkan informasi label. Dengan demikian, fokusnya akan pada variasi dan pola intrinsik dalam data, bukan pada pemisahan berdasarkan kelas.

[49]:
```
# Dapatkan loadings dari PCA
loadings = pca.components_

# Buat dataframe untuk loadings
loadings_df = pd.DataFrame(data=loadings.T, columns=[f'PC{i+1}' for i in␣
 ↪range(loadings.shape[0])])

# Tambahkan kolom nama variabel
loadings_df['Variable'] = credit_scaled.columns

# Tampilkan loadings yang signifikan (misalnya, absolute loadings > 0.3)
significant_loadings = loadings_df[abs(loadings_df['PC1']) > 0.2]
significant_loadings
```

[49]:
```
         PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
7   0.272998 -0.011632  0.170400  0.219698  0.207326 -0.062567  0.481160
8   0.332235 -0.318315  0.022183  0.002883  0.061321  0.090515 -0.271476
9   0.305289 -0.137153 -0.099439 -0.094422 -0.047025 -0.088929  0.107691
10  0.404106 -0.279941 -0.028704 -0.125352 -0.010460  0.015621 -0.002203
11  0.288576  0.518364 -0.167788  0.239467 -0.186757 -0.261848 -0.153593
17  0.237831  0.272516  0.245491  0.111663  0.020680  0.173127 -0.058940
18  0.278637  0.196126 -0.415571 -0.105264  0.019541 -0.064495  0.230005
22 -0.423333  0.242621  0.002793  0.179051  0.087539  0.029009  0.008577

         PC8       PC9      PC10  …      PC15      PC16      PC17      PC18  \
7   0.035475 -0.124596  0.036945  … -0.207396 -0.047797  0.075471  0.091207
8  -0.418818 -0.129885 -0.283481  …  0.091688 -0.234790 -0.010717 -0.087979
9  -0.006446  0.037211 -0.243017  … -0.284732 -0.329304  0.400055  0.165205
10  0.192010 -0.040707  0.027677  …  0.306045  0.084556 -0.107509  0.006424
11  0.170006 -0.333268 -0.197671  …  0.306944  0.112792  0.027198  0.162583
17  0.166192  0.658737 -0.307781  …  0.005918  0.084181  0.082705 -0.056958
18 -0.136513  0.119629  0.321415  … -0.163517 -0.012256 -0.180826  0.044341
22 -0.069880 -0.147456 -0.135594  … -0.142119 -0.032587  0.104566  0.100315
```

```
        PC19      PC20      PC21      PC22      PC23                  Variable
7  -0.174159 -0.002844  0.363897 -0.026282  0.034646           (zip_count_4w,)
8   0.215133 -0.196382  0.281986  0.029008 -0.234222            (velocity_6h,)
9  -0.088738 -0.093960 -0.318765  0.096215  0.132749           (velocity_24h,)
10 -0.351867  0.037824 -0.155432 -0.188126 -0.480096            (velocity_4w,)
11  0.127145 -0.006676 -0.042809 -0.213896  0.018050   (bank_branch_count_8w,)
17 -0.226750  0.004153 -0.092958  0.074932 -0.035575      (bank_months_count,)
18  0.185553  0.320035 -0.131691  0.364386 -0.327143  (proposed_credit_limit,)
22 -0.272304 -0.224998  0.042353  0.126969 -0.649989                  (month,)

[8 rows x 24 columns]
```

Observasi:

Interpretasi loadings dari hasil PCA dapat memberikan pemahaman tentang kontribusi setiap variabel terhadap pembentukan setiap komponen utama (PC). Di sini, saya akan memberikan contoh interpretasi untuk beberapa variabel yang memiliki loadings signifikan pada PC1:

1. **zip_count_4w:** Variabel ini memiliki loading positif yang cukup besar pada PC1 (0.27). Ini menunjukkan bahwa nilai tinggi pada `zip_count_4w` berkontribusi positif terhadap nilai PC1. Dengan kata lain, observasi dengan nilai `zip_count_4w` yang tinggi cenderung memiliki nilai PC1 yang tinggi.

2. **velocity_6h:** Variabel ini memiliki loading positif yang kuat pada PC1 (0.33). Hal ini menandakan bahwa nilai tinggi pada `velocity_6h` berhubungan positif dengan nilai PC1. Observasi dengan `velocity_6h` yang tinggi cenderung memiliki nilai PC1 yang tinggi.

3. **velocity_24h:** Variabel ini memiliki loading negatif yang signifikan pada PC1 (-0.31). Ini menandakan bahwa nilai tinggi pada `velocity_24h` berhubungan negatif dengan nilai PC1. Dengan kata lain, observasi dengan nilai `velocity_24h` yang tinggi cenderung memiliki nilai PC1 yang rendah.

4. **velocity_4w:** Variabel ini memiliki loading positif yang signifikan pada PC1 (0.40). Ini menunjukkan bahwa nilai tinggi pada `velocity_4w` berhubungan positif dengan nilai PC1. Observasi dengan `velocity_4w` yang tinggi cenderung memiliki nilai PC1 yang tinggi.

Demikian pula, Anda dapat melanjutkan untuk menganalisis loadings pada PC lainnya. Loadings yang tinggi (baik positif maupun negatif) menunjukkan sejauh mana variabel tersebut berkontribusi terhadap pembentukan komponen utama tersebut. Variabel dengan loadings tinggi pada suatu PC dapat dianggap sebagai variabel yang memiliki pengaruh besar pada PC tersebut.

39 Studi Kasus Local Outlier Factor (LOF): Deteksi Anomali dengan Menggunakan Library PyOD

39.1 Overview

Anomaly detection adalah teknik yang digunakan untuk mengidentifikasi data yang tidak sesuai dengan pola normal data. Teknik ini dapat digunakan untuk berbagai tujuan, seperti deteksi fraud, deteksi gangguan pada sistem, dan deteksi penyimpangan data.Anomali dapat merugikan suatu perusahaan, terutama dalam kondisi di mana observasi tidak biasa dapat menjadi indikator

adanya masalah serius. Untuk mengatasi tantangan ini, deteksi anomali menjadi kritis, dan salah satu pendekatan yang dapat digunakan adalah algoritma Local Outlier Factor (LOF).

Penerapan deteksi anomali memiliki dampak besar di berbagai industri, termasuk e-commerce dan industri perbankan. Mencegah terjadinya anomali atau situasi tak lazim yang dapat berkembang menjadi masalah lebih besar akan membantu perusahaan mencegah kerugian keuangan.

Dalam kasus ini, kita akan membahas secara mendalam mengenai deteksi anomali menggunakan teknis unsupervised learning yang mana dataset diasumsikan tidak memimlikilabel dengan menggunakan algoritma Local Outlier Factor (LOF), konsep dasar di balik algoritma ini, dan bagaimana algoritma ini bekerja untuk mendeteksi observasi yang tidak biasa. Penggunaan library `PyOD` dalam implementasi LOF akan menjadi fokus utama, memperkuat kehandalan dan fleksibilitas deteksi anomali dalam konteks pemrograman Python.

Referensi: PyOD Documentation. https://pyod.readthedocs.io/en/latest/.

40 Local Outlier Factor

Dalam deteksi anomali , faktor outlier lokal (LOF) adalah algoritma yang diusulkan oleh Markus M. Breunig, Hans-Peter Kriegel , Raymond T. Ng dan Jörg Sander pada tahun 2000 untuk menemukan titik data anomali dengan mengukur deviasi lokal dari titik data tertentu sehubungan dengan tetangganya. [1]

LOF adalah algoritma unsupervied learning salah satu teknik anomaly detection yang populer. Teknik ini bekerja dengan cara menghitung seberapa jauh suatu data dari tetangga terdekatnya. Data yang memiliki nilai LOF yang tinggi dianggap sebagai outlier. Implementasi LOF dapat diaplikasikan pada berbagai kondisi, seperti mengidentifikasi titik-titik tak lazim dalam suatu dataset. Selain itu, algoritma ini dapat digunakan untuk mendeteksi kondisi anomali, yang penting untuk mencegah situasi yang tidak diinginkan berkembang lebih jauh.

Kesesuaian LOF untuk Deteksi Fraud:

LOF dapat menjadi pilihan yang baik untuk deteksi fraud karena memiliki beberapa kelebihan, yaitu:

- Efektif dalam menemukan outlier lokal: LOF dapat mengidentifikasi outlier yang tidak dapat ditemukan oleh metode global, seperti outlier yang berada di dalam cluster yang padat.
- Tidak sensitif terhadap distribusi data: LOF dapat bekerja dengan baik pada data dengan distribusi yang tidak normal.
- Mudah diimplementasikan: LOF dapat diimplementasikan dengan mudah menggunakan library Python seperti Pyod.

Namun, LOF juga memiliki beberapa kekurangan, yaitu:

- Dapat menjadi lambat untuk data yang besar: LOF memerlukan komputasi yang cukup berat untuk dataset yang besar.
- Memerlukan pemilihan parameter yang tepat: Parameter k (jumlah tetangga terdekat) yang digunakan dalam LOF dapat mempengaruhi hasil deteksi outlier.

Lof akan menghitung kepadatan lokal suatu titik dibandingkan dengan kepadatan lokal k tetangganya. Misalkan kita mempunyai contoh pada gambar berikut.

Pada gambar di atas terlihat bahwa titik o1 merupakan outlier karena jarak titik tersebut jauh lebih besar dibandingkan titik terdekat. Pertanyaannya adalah, bagaimana kita memberi label o2? Dengan algoritma faktor outlier lokal, kita dapat menghitung skor o2 dan melihat apakah itu outlier atau bukan. Faktor outlier lokal akan menghasilkan skor dengan interpretasi sebagai berikut:

- Jika LOF > 1, kemungkinan besar merupakan titik outlier
- Jika LOF < 1, kemungkinan besar titik tersebut bukan merupakan titik outlier

Jika titik tersebut kurang padat dibandingkan area dengan kepadatan tinggi di C2, kemungkinan besar titik tersebut merupakan outlier. Semakin terisolasi suatu titik dibandingkan dengan titik tetangganya yang memiliki kepadatan tinggi, semakin besar kemungkinan suatu observasi dikategorikan sebagai outlier.

Contoh lainnya adalah o1, dan o2 dapat dianggap sebagai outlier, dan istilah untuk outlier ini bersifat lokal. o3 merupakan outlier global, sedangkan o4 bukan merupakan outlier karena lingkungan sekitar titik tidak sepadat kepadatan di dekat o1, o2, atau o3.

PyOD LOF Documentation. https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.lof

Dataset yang digunakan masih sama, menggunakan dataset dari Bank Account Fraud dari kaggle yang sudah dilakukan PCA sebelumnya dengan mempertahankan 20PC

40.1 Membuat Model LOF

Disini kita menggunakan semua observasi dengan 20 PC

[50]:
```
credit_pca20.head()
```

[50]:
```
        PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0  1.692755  3.323030 -0.889182  0.568785  1.539485  0.669139 -0.882377
1  0.853039  0.094970  1.362371 -1.279626 -1.114656 -1.041867 -1.349552
2  0.933672  2.583289 -0.083890 -1.519840  1.423196 -0.311211 -1.014184
3  1.463710 -2.074312  0.009756  0.335156 -0.323293 -0.674480 -1.214947
4 -2.179767  0.666897 -1.064235 -0.738059 -0.202709  0.682306 -1.596558

        PC8       PC9      PC10      PC11      PC12      PC13      PC14  \
0 -0.816128  0.464054  0.338666 -0.942384  1.925547  1.243251 -2.648092
1 -1.287512 -0.509089  1.731860  2.152441 -1.882548  1.821081 -0.446959
2 -2.772364 -0.670852 -2.078549  2.107466  1.285194 -2.136757  1.083076
3 -0.397285  0.673089 -0.550966  1.103438 -0.722769  0.301028  0.111886
4  0.048435  0.442683 -0.068521  0.133925  0.679665 -0.644503  0.599069

       PC15      PC16      PC17      PC18      PC19      PC20
0  0.622580  0.200997  1.858074 -0.260204 -1.150616 -0.780847
1  0.185748  1.788388  1.661765  1.457426 -0.587829 -0.324188
2 -1.209996 -0.610770 -1.318451 -1.986781 -0.225584 -1.288779
3 -0.864106  0.344133  0.582058  0.913164 -0.487811 -0.710987
4  0.769384  0.144125  0.493707 -0.715133 -0.387881 -1.316316
```

[51]:
```
# Menggunakan LOF dari PyOD (Python Outlier Detection)
lof_model = LOF()
outliers = lof_model.fit_predict(credit_pca20)
```

Pada kode di atas, kita menggunakan model LOF (Local Outlier Factor) dari modul yang disebut sebagai `LOF` dari library Pyod, tetapi kode lengkapnya tidak diberikan). Model LOF adalah algoritma deteksi outlier yang bekerja dengan menilai tingkat kedekatan setiap titik data dengan tetangganya. Output dari `fit_predict(credit_pca20)` adalah label yang menunjukkan setiap titik data sebagai outlier (label -1) atau tidak (label 1).

Sejauh ini, Kita sudah benar memasukkan hasil PCA (`credit_pca20`) sebagai input untuk deteksi outlier menggunakan LOF. Namun, beberapa hal yang perlu diperhatikan:

1. **Jumlah Cluster:**
   - Model LOF memerlukan informasi tentang jumlah cluster atau tetangga yang digunakan untuk menghitung faktor lokal. Jika Kita tidak memberikan parameter ini, LOF akan menganggap bahwa seluruh dataset adalah satu cluster.
2. **Penyesuaian Parameter:**
   - Terkadang, performa LOF dapat dipengaruhi oleh parameter yang diatur, seperti `n_neighbors` (jumlah tetangga yang digunakan dalam perhitungan). Pastikan untuk menyesuaikan parameter-parameter ini sesuai dengan karakteristik dataset Kita.
3. **Interpretasi Hasil:**
   - Setelah Kita mendapatkan label outlier, Kita dapat menggunakan informasi ini untuk analisis lebih lanjut. Label 1 menunjukkan bahwa titik data tersebut dianggap sebagai outlier.

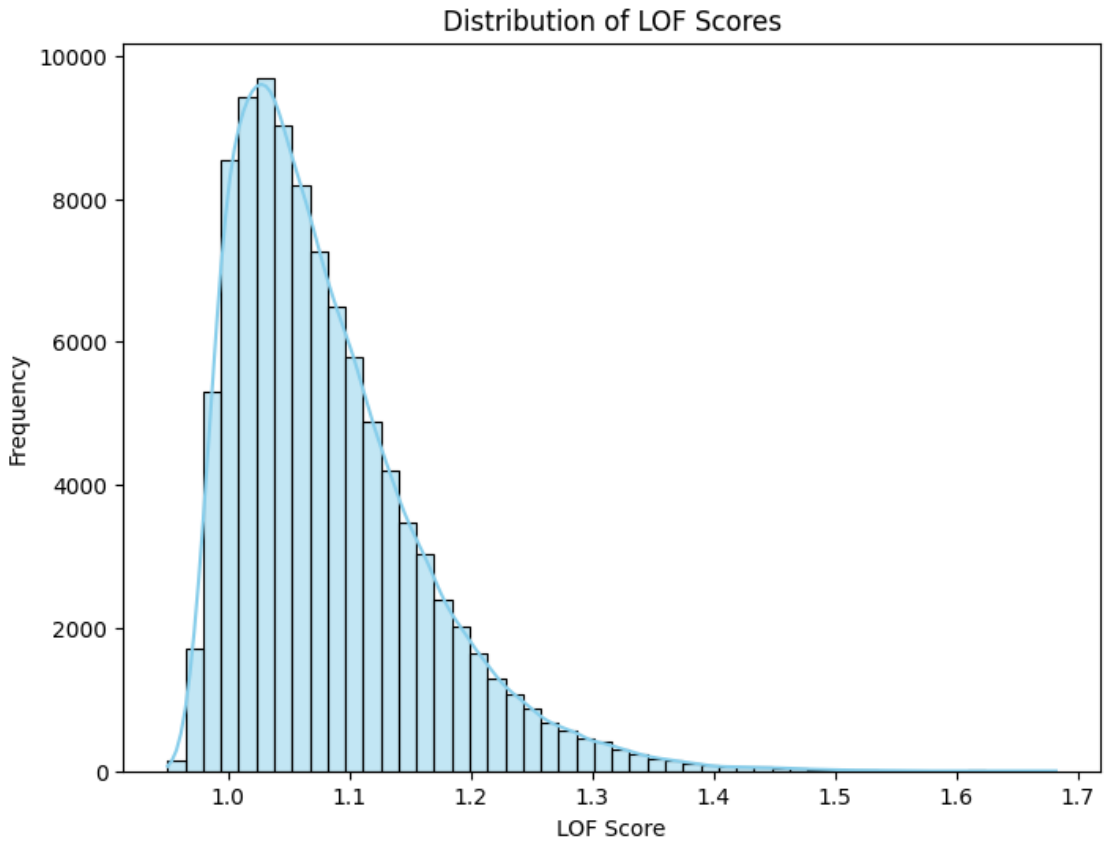
Cek jumlah antara LOF 0 (normal) dan LOF 1 (anomali)

[52]:
```
unique_labels, label_counts = np.unique(outliers, return_counts=True)
print("Label Counts:", dict(zip(unique_labels, label_counts)))
```

```
Label Counts: {0: 90000, 1: 10000}
```

[53]:
```
# Membuat model LOF
lof_model = LOF(contamination=0.05, n_neighbors=40)

# Fitting model pada data hasil PCA
lof_model.fit(credit_pca)

# Menghitung nilai LOF
lof_scores = lof_model.decision_function(credit_pca)

# Menampilkan distribusi nilai LOF
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
sns.histplot(lof_scores, bins=50, kde=True, color='skyblue')
plt.title('Distribution of LOF Scores')
plt.xlabel('LOF Score')
```

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



0 mewakili inliers (data normal), 1 mewakili outliers (data anomali).

Dalam hal ini, Anda dapat menganggap data dengan nilai LOF 1 sebagai anomali. Dengan memeriksa distribusi nilai LOF, Anda dapat melihat seberapa baik model dapat memisahkan inliers dan outliers berdasarkan skor LOF yang diberikan. Jika distribusinya lebih terfokus di sekitar nilai 1, itu mungkin menunjukkan bahwa model secara efektif memisahkan data menjadi dua kategori tersebut.

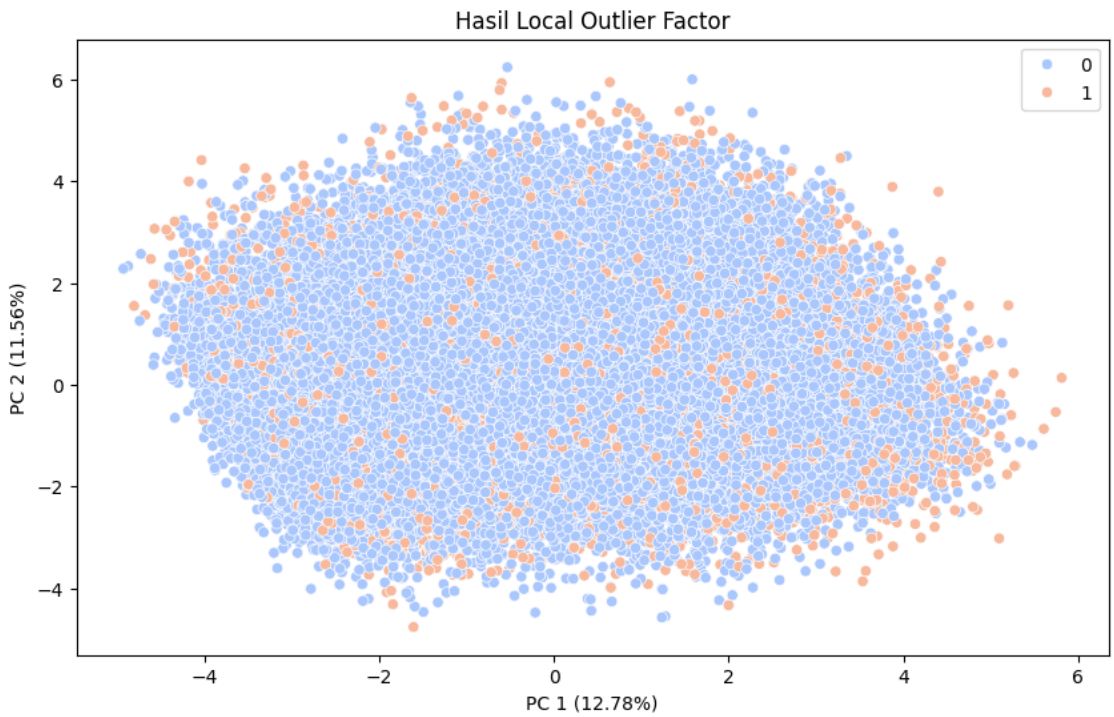
40.2 Cek Index Anomali

[54]:
```
# cek index yang dideteksi merupakan anomali
outlier_indices = np.where(outliers == 1)[0]
outlier_indices
```

[54]:
```
array([   44,    55,    56, …, 99992, 99995, 99996], dtype=int64)
```

40.3 Visualisasi Plot LOF

[55]:
```
# menampilkan plot anomali
plt.figure(figsize=(10, 6))
sns.scatterplot(x=credit_pca20['PC1'],
                y=credit_pca20['PC2'],
                hue=outliers,
                palette='coolwarm')
plt.title('Hasil Local Outlier Factor')
plt.xlabel(f'PC 1 ({pca.explained_variance_ratio_[0]*100:.2f}%)')
plt.ylabel(f'PC 2 ({pca.explained_variance_ratio_[1]*100:.2f}%)')
plt.show()
```



[56]:
```
# Plot hasil LOF menggunakan Plotly Express
fig = px.scatter(credit_pca20, x='PC1', y='PC2',
                 color=outliers.astype(str),
                 color_discrete_map={'0': '#a6c4ff', '1': '#ffa07a'},
                 title='LOF Results',
                 labels={'PC1': f'PC 1 ({pca.explained_variance_ratio_[0]*100:.
 ↪2f}%)',
                         'PC2': f'PC 2 ({pca.explained_variance_ratio_[1]*100:.
 ↪2f}%)'})

# Menampilkan plot
```

```
fig.update_layout(width=1000, height=800)
fig.show()
```

40.4 LOF Menggunakan Treshold

40.4.1 Buat model LOF menggunakan treshold

Kita coba pada sebagian dataset, yakni 500 observasi dan menggunakan 15 PC

[57]:
```
credit_lof = credit_pca20.iloc[0:500,:15]
credit_lof.shape
```

[57]: (500, 15)

[58]:
```
# Set seed (kunci)
np.random.seed(123)

# Set the outlier detection and thresholding methods
lof_model1 = LOF(contamination= 0.05,
                 n_neighbors = 40)
outliers1 = lof_model1.fit_predict(credit_lof)
```

Cek jumlah antara LOF 0 (normal) dan LOF 1 (anomali)

[59]:
```
unique_labels1, label_counts1 = np.unique(outliers1, return_counts=True)
print("Label Counts:", dict(zip(unique_labels1, label_counts1)))
```

```
Label Counts: {0: 475, 1: 25}
```

40.4.2 Visualisasi Plot LOF (treshold)

[60]:
```
# Plot hasil LOF menggunakan Plotly Express
fig = px.scatter(credit_lof, x='PC1', y='PC2',
                 color=outliers1.astype(str),
                 color_discrete_map={'0': '#a6c4ff', '1': '#ffa07a'},
                 title='LOF Results',
                 labels={'PC1': f'PC 1 ({pca.explained_variance_ratio_[0]*100:.
 ↪2f}%)',
                         'PC2': f'PC 2 ({pca.explained_variance_ratio_[1]*100:.
 ↪2f}%)'})

# Menampilkan plot
fig.update_layout(width=1000, height=800)
fig.show()
```

40.4.3 Cek Index Anomali

[61]:
```
anomaly_indices = np.where(outliers1 == 1)[0]
anomaly_data = credit_lof.iloc[anomaly_indices]
anomaly_data
```

[61]:
```
          PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
26   0.925788 -0.585639 -2.657096 -0.728718 -0.205464  1.473026  1.728699
56   3.204368  0.470792  0.738523 -0.564910  5.984894  2.183383  1.967510
89  -2.136645  0.645510 -0.718518 -1.946945  1.091322 -1.485979  0.814064
90   1.291360  0.735105 -1.765434 -1.853514 -0.373382 -0.099603 -0.876788
98   1.377495  0.966175 -0.609939  0.066280  4.838258 -1.458191  1.318682
106 -1.039148  0.727984 -1.256450 -2.378548  1.391495 -1.120396  0.754681
112  1.201808  1.000082  0.742217 -1.546939  2.458487 -2.290524 -0.470125
127  2.494227 -1.640534 -0.833048 -0.657298  0.256379 -2.131187  0.369281
128 -1.223339 -1.470065 -2.359066 -2.628642  1.023193  4.879602  0.213449
144  4.372939 -2.528413 -0.153119  0.360090  1.588415  3.316612  2.317017
164  2.737500  0.017354  0.640464 -1.766394  1.446785 -1.363379 -0.688976
165  0.389156  0.087253 -0.587367 -3.157713  4.365842  1.347478  4.362516
169 -0.888533  3.209603 -1.413784 -2.537523  2.480841 -3.167606  3.168509
180  0.267759 -1.773736 -0.262213 -1.943138  1.240789  1.044085  1.491958
230  0.033705 -2.354364 -0.399532 -0.377762  2.515687  4.602028  2.486514
275 -2.872665 -0.302327  3.131976 -1.762586  0.096531 -0.325052  0.809636
297 -0.607758 -0.866862  2.307437 -1.784542  1.812124  4.060226  1.140934
363 -2.224941  0.901970 -0.472185  3.057095  0.532174  0.884117 -1.793480
413  1.292793 -0.952288  0.125778 -1.740493  0.789963 -1.967448 -0.200018
438  0.548546  0.686341  1.337680 -0.227938 -2.329503  1.309343 -1.531209
460  1.305142 -0.766397 -0.624385  1.553603 -2.741367  0.345322 -2.243837
476  3.399779 -1.300193  0.317476 -2.254566  0.745243 -1.465471  1.834862
487 -0.530611  0.593323  2.747431 -3.972989  0.677513  1.504566  1.294437
491 -0.390247  0.830536  0.232818  0.233699 -1.159435  0.528379  0.042014
496 -1.518274 -0.582115 -1.428216 -2.237218  2.291936  0.098974  1.776954

          PC8       PC9      PC10      PC11      PC12      PC13      PC14  \
26   1.927637  0.229754  0.792727  3.285204  2.187178  0.815790  0.275277
56  -1.954748  3.755181 -1.544258 -3.057926  0.243010  1.795828 -0.661952
89  -1.295176 -3.419550 -2.066404  0.832588  1.393269 -1.411975  0.216122
90  -4.067350 -2.528821  0.239315  2.860650  1.983418 -1.511423  1.193488
98  -2.564892  2.438550  0.074616 -1.711888  0.295400  2.712190 -0.373146
106 -2.918113  0.232411 -2.342232  1.331730  1.572635 -1.630373  0.990650
112 -1.652067 -2.171192 -1.978085  1.709270 -0.037374 -1.930605 -0.582780
127 -3.137536 -2.787540 -0.185417  2.588982 -0.026080 -2.245416 -2.023214
128 -1.944008  1.623873  0.048183 -0.840907 -0.676905  0.722867  0.458059
144  0.288303  0.139930  0.010181 -0.745707 -0.443143 -0.101578  1.251905
164 -1.599671 -1.764261 -2.567007  1.668575  0.833009 -2.601999 -0.087475
165 -3.772920 -0.852562 -2.191494 -0.539066 -1.548064 -0.563336 -0.307221
169 -2.982817  0.563544 -1.565100  1.109453 -0.216495 -1.225239 -0.327231
180  2.666928  0.305286 -0.105567  2.389946  1.730006  1.062993  0.496760
```

```
230 -0.358744  1.303898 -1.127548 -0.263021  0.639413  2.290664 -1.089875
275 -4.068504 -0.435376 -0.318930  1.160053  2.885682 -1.600034  0.025239
297 -0.989366  0.879993  0.222881  0.680894 -1.304075  1.817346  2.064097
363  2.413132  3.191402  0.696436 -1.562514  0.847268  2.846223  0.027721
413 -2.382215 -1.357306 -2.408881  2.807734  0.080766 -1.828665 -0.739077
438  2.355259  0.817970  3.729084 -1.520350 -1.990273  1.093611 -1.819768
460  5.976468  3.919949  3.026939 -3.061516 -3.051402  3.866855 -1.718555
476 -1.171487 -0.501594 -0.978475  1.599640  3.146881 -2.371861  0.986433
487 -1.023461 -2.375233 -1.340301  1.633276  2.114501 -1.731802  1.083333
491  4.280527  0.957927  0.402216 -2.397744 -2.572900  1.235735 -2.798863
496 -2.865892 -1.624487 -0.715342  0.115193  0.681182 -2.653834 -1.016960

         PC15
26   0.401502
56   2.037928
89  -0.639252
90   1.210835
98   4.358628
106 -1.043118
112 -1.516049
127  0.050240
128 -2.900389
144 -1.709838
164 -0.162840
165 -0.999097
169  1.154403
180  0.992689
230 -3.313149
275 -0.153866
297 -2.237290
363  0.261570
413  0.251758
438  1.185235
460 -0.627523
476 -0.311971
487 -0.616784
491  0.393138
496 -1.474479
```

Kode diatas bertujuan untuk menemukan indeks indeks dari data yang dianggap sebagai anomali oleh model LOF. Setelah itu, data anomali tersebut diekstrak dari DataFrame `credit_lof` yang merupakan hasil PCA.

Interpretasi hasilnya dapat dilakukan sebagai berikut:

1. **`anomaly_indices`**: Array ini berisi indeks-indeks dari anomali dalam array `outliers1`. Indeks-indeks ini sesuai dengan baris-baris dalam DataFrame `credit_lof` yang merupakan data hasil dari PCA.

2. **anomaly_data**: DataFrame ini berisi data aktual (baris-baris) dari DataFrame `credit_lof` yang diidentifikasi sebagai anomali oleh model LOF. DataFrame ini sudah berada dalam ruang hasil transformasi PCA.

Interpretasi dari hasil ini melibatkan pemahaman terhadap karakteristik atau pola yang membuat titik-titik ini dianggap sebagai anomali. Anda mungkin ingin menganalisis nilai-nilai fitur dalam DataFrame `anomaly_data` dan membandingkannya dengan distribusi keseluruhan data. Fitur-fitur dengan nilai ekstrem atau pola yang sangat berbeda dari mayoritas data mungkin menjadi indikator anomali.

Berikut langkah-langkah yang bisa diambil:

- Periksa nilai-nilai fitur dalam `anomaly_data` dan bandingkan dengan data normal.
- Visualisasikan anomali dalam ruang hasil transformasi PCA untuk melihat apakah ada pola yang jelas.
- Periksa fitur-fitur asli dalam data anomali untuk memahami sifat anomali tersebut.

Misalnya, Anda dapat menggunakan teknik visualisasi seperti scatter plot atau visualisasi relevan lainnya untuk menjelajahi anomali lebih lanjut.

40.4.4 [Optional] Membandingkan kolom label dengan hasil lof

[62]:
```
# Buat DataFrame baru dengan kolom fraud_bool dari credit_clean500
df_combined_500 = pd.concat([credit_lof, credit_clean.loc[0:
 ↪499,['fraud_bool']]], axis=1)
df_combined_500
```

[62]:
```
          PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0    1.692755  3.323030 -0.889182  0.568785  1.539485  0.669139 -0.882377
1    0.853039  0.094970  1.362371 -1.279626 -1.114656 -1.041867 -1.349552
2    0.933672  2.583289 -0.083890 -1.519840  1.423196 -0.311211 -1.014184
3    1.463710 -2.074312  0.009756  0.335156 -0.323293 -0.674480 -1.214947
4   -2.179767  0.666897 -1.064235 -0.738059 -0.202709  0.682306 -1.596558
..        …       …       …       …       …       …       …
495 -1.828468  0.715119 -2.557044 -1.129321  0.404440  1.250127  0.114934
496 -1.518274 -0.582115 -1.428216 -2.237218  2.291936  0.098974  1.776954
497 -0.336184 -1.133876 -0.578677  0.354667  1.376334  0.193115  0.062013
498  2.117674 -2.183381 -0.813468 -1.752209  0.403727  1.375570  3.092221
499 -1.428220  1.084061 -1.964436 -0.236965 -0.578958 -0.199866  1.369869

          PC8       PC9      PC10      PC11      PC12      PC13      PC14  \
0   -0.816128  0.464054  0.338666 -0.942384  1.925547  1.243251 -2.648092
1   -1.287512 -0.509089  1.731860  2.152441 -1.882548  1.821081 -0.446959
2   -2.772364 -0.670852 -2.078549  2.107466  1.285194 -2.136757  1.083076
3   -0.397285  0.673089 -0.550966  1.103438 -0.722769  0.301028  0.111886
4    0.048435  0.442683 -0.068521  0.133925  0.679665 -0.644503  0.599069
..        …       …       …       …       …       …       …
495 -0.071610 -0.870993 -0.689261  0.388458 -1.431135  1.112148 -1.383164
496 -2.865892 -1.624487 -0.715342  0.115193  0.681182 -2.653834 -1.016960
```

```
497  0.813487 -1.966106  0.245186 -0.638806 -0.801928  0.177842  0.760963
498  0.224868  0.367147  0.568256 -0.692106 -0.380947  0.992528 -0.858177
499  0.339294 -0.581789  0.031773 -0.400552 -0.335871  0.059226 -1.186435

         PC15  fraud_bool
0    0.622580           0
1    0.185748           0
2   -1.209996           1
3   -0.864106           0
4    0.769384           0
..        …         …
495 -1.431717           0
496 -1.474479           0
497  0.695872           0
498 -1.551977           0
499  0.095109           0

[500 rows x 16 columns]
```

[63]:
```
df_combined_500['lof_score'] = outliers1
df_combined_500
```

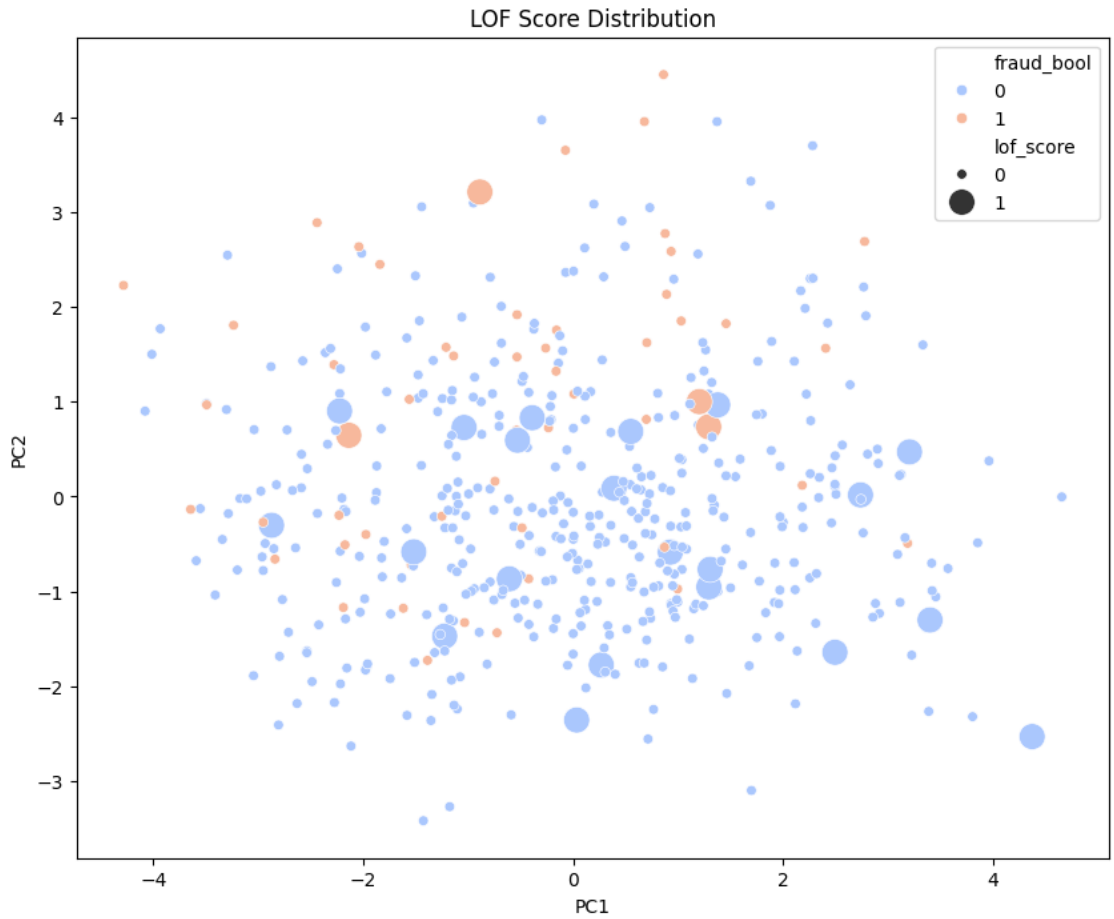
[63]:
```
           PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0    1.692755  3.323030 -0.889182  0.568785  1.539485  0.669139 -0.882377
1    0.853039  0.094970  1.362371 -1.279626 -1.114656 -1.041867 -1.349552
2    0.933672  2.583289 -0.083890 -1.519840  1.423196 -0.311211 -1.014184
3    1.463710 -2.074312  0.009756  0.335156 -0.323293 -0.674480 -1.214947
4   -2.179767  0.666897 -1.064235 -0.738059 -0.202709  0.682306 -1.596558
..        …       …       …       …       …       …       …
495 -1.828468  0.715119 -2.557044 -1.129321  0.404440  1.250127  0.114934
496 -1.518274 -0.582115 -1.428216 -2.237218  2.291936  0.098974  1.776954
497 -0.336184 -1.133876 -0.578677  0.354667  1.376334  0.193115  0.062013
498  2.117674 -2.183381 -0.813468 -1.752209  0.403727  1.375570  3.092221
499 -1.428220  1.084061 -1.964436 -0.236965 -0.578958 -0.199866  1.369869

           PC8       PC9      PC10      PC11      PC12      PC13      PC14  \
0   -0.816128  0.464054  0.338666 -0.942384  1.925547  1.243251 -2.648092
1   -1.287512 -0.509089  1.731860  2.152441 -1.882548  1.821081 -0.446959
2   -2.772364 -0.670852 -2.078549  2.107466  1.285194 -2.136757  1.083076
3   -0.397285  0.673089 -0.550966  1.103438 -0.722769  0.301028  0.111886
4    0.048435  0.442683 -0.068521  0.133925  0.679665 -0.644503  0.599069
..        …       …       …       …       …       …       …
495 -0.071610 -0.870993 -0.689261  0.388458 -1.431135  1.112148 -1.383164
496 -2.865892 -1.624487 -0.715342  0.115193  0.681182 -2.653834 -1.016960
497  0.813487 -1.966106  0.245186 -0.638806 -0.801928  0.177842  0.760963
498  0.224868  0.367147  0.568256 -0.692106 -0.380947  0.992528 -0.858177
499  0.339294 -0.581789  0.031773 -0.400552 -0.335871  0.059226 -1.186435
```

```
         PC15  fraud_bool  lof_score
0    0.622580           0          0
1    0.185748           0          0
2   -1.209996           1          0
3   -0.864106           0          0
4    0.769384           0          0
..        …          …        …
495 -1.431717           0          0
496 -1.474479           0          1
497  0.695872           0          0
498 -1.551977           0          0
499  0.095109           0          0

[500 rows x 17 columns]
```

[64]:
```
(df_combined_500['fraud_bool'].value_counts() / len(df_combined_500)) * 100
```

[64]:
```
0    89.2
1    10.8
Name: fraud_bool, dtype: float64
```

[71]:
```
# Buat scatter plot
# Buat scatter plot dengan ukuran titik berdasarkan kolom 'Outlier'
plt.figure(figsize=(10, 8))
sns.scatterplot(x='PC1', y='PC2', hue='fraud_bool', size='lof_score',
                sizes=(200, 30), data=df_combined_500, palette='coolwarm')
plt.title('LOF Score Distribution')
plt.show()
```



Dari visualisasi di atas kita dapat melihat bahwa transaksi asli dan penipuan mempunyai pola yang berbeda-beda. Semakin tinggi skor lof yang dimilikinya, maka titik tersebut semakin tebal dan menonjol.

Aturan praktis skor lof mengatakan bahwa jika skor LOF = 1, kemungkinan besar skor tersebut merupakan outlier. Entah bagaimana, ambang batas dapat disesuaikan dengan distribusi data. Mari kita lihat dulu statistik skor LOFnya.

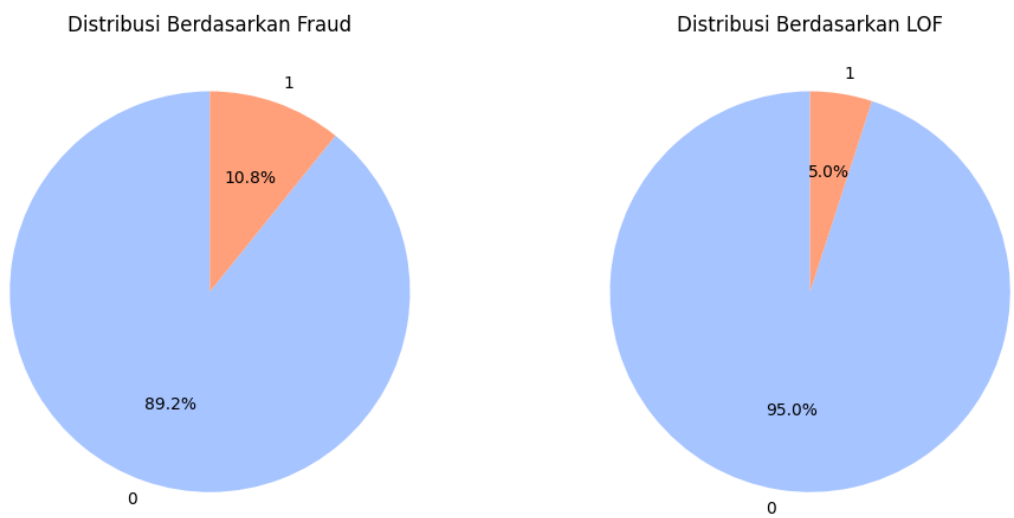
hasil dari model LOF adalah 0 dan 1, ini menunjukkan bahwa setiap sampel dianggap sebagai outlier (1) atau bukan outlier (0) berdasarkan skor LOF yang dihitung. Secara umum, model LOF pada PyOD akan memberikan label -1 untuk outlier dan 1 untuk sampel normal. Oleh karena itu, Anda mungkin perlu melakukan penyesuaian atau interpretasi tergantung pada bagaimana label outlier diatur pada dataset atau implementasi LOF yang digunakan.

[66]:
```
print(df_combined_500['fraud_bool'].value_counts())
print(df_combined_500['lof_score'].value_counts())
```

```
0    446
1     54
Name: fraud_bool, dtype: int64
```

```
0    475
1     25
Name: lof_score, dtype: int64
```

[67]:
```
# Hitung nilai untuk pie plot
fraud_counts = df_combined_500['fraud_bool'].value_counts()
lof_counts = df_combined_500['lof_score'].value_counts()

# Atur subplot
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# Pie plot untuk kolom 'fraud_bool'
axs[0].pie(fraud_counts, labels=fraud_counts.index, autopct='%1.1f%%',␣
 ↪startangle=90, colors=['#a6c4ff', '#ffa07a'])
axs[0].set_title('Distribusi Berdasarkan Fraud')

# Pie plot untuk kolom 'lof_score'
axs[1].pie(lof_counts, labels=lof_counts.index, autopct='%1.1f%%',␣
 ↪startangle=90, colors=['#a6c4ff', '#ffa07a'])
axs[1].set_title('Distribusi Berdasarkan LOF')

plt.show()
```



40.5 Catatan Penting tentang Local Outlier Factor

Banyak kasus yang terlibat dalam deteksi outlier atau anomali tidak dilengkapi dengan variabel yang dilabeli. Jika dataset dilengkapi dengan variabel target yang diberi label, ini dapat menilai kualitas deteksi. Local Outlier Factor akan menilai skor yang tidak biasa tanpa mengetahui kondisi sebenarnya dari pengamatan; untuk kasus ini, ia tidak tahu nilai sebenarnya dari suatu dataset

apakah fraud/outlier atau tidak. Realitas dari suatu dataset sering memberikan kita kondisi apakah suatu titik data adalah outlier atau tidak. Local outlier factor adalah pembelajaran mesin yang tidak supervisi yang seringkali tidak melibatkan variabel yang dilabeli. Skor yang dihasilkan oleh algoritma ini dapat menjadi sudut pandang tambahan dalam algoritma pembelajaran mesin yang supervisi.

Beberapa catatan penting tentang memprediksi kondisi outlier atau anomali adalah terlalu sedikitnya kasus dalam mendeteksi penyakit atau perilaku yang berubah dengan cepat dalam transaksi penipuan. Jika kita menghadapi situasi seperti ini, kita tidak dapat hanya menggunakan algoritma pembelajaran mesin tanpa supervisi untuk mendeteksi. Beberapa teknik yang kokoh perlu dinilai untuk mendapatkan hasil yang lebih baik.

Local outlier factor akan menghasilkan angka dalam interval rasio. Jika skornya lebih dari 1, itu berarti kerapatan tetangga lebih tinggi. Interpretasi dari suatu rasio dapat disesuaikan berdasarkan rekomendasi bisnis pengguna daripada hanya mengartikan lof > 1 sebagai outlier. Jika pengguna ingin lebih selektif dalam mendeteksi pengamatan outlier, pengguna dapat meningkatkan ambang batas skor lof berdasarkan domain bisnis.

41 Kesimpulan

Outlier adalah kondisi di mana suatu pengamatan dalam suatu dataset berbeda dari sisa data. Untuk mendeteksi apakah suatu pengamatan adalah outlier atau tidak, beberapa metode dapat digunakan.

Kita dapat menggunakan `local outlier factor` untuk menangani situasi di mana kita dapat menggunakan deteksi outlier dalam data multivariabel. Algoritma ini akan menggunakan perbandingan kepadatan pengamatan terhadap kepadatan pengamatan tetangga K-terdekat.

LOF membutuhkan K sebagai atribut untuk menghitung pengamatan tetangganya, dan dengan mensimulasikan K yang berbeda, algoritma ini akan menghasilkan skor lof yang berbeda. Semakin banyak nilai K yang digunakan, semakin stabil algoritma dapat mendeteksi outlier. Meskipun algoritma ini digunakan untuk mendeteksi pengamatan outlier, ia dapat digunakan untuk mendeteksi situasi anomali. Meskipun LOF dapat digunakan untuk menilai masalah pembelajaran mesin yang terawasi, perhatikan bahwa beberapa latar belakang kasus bisnis perlu diperhatikan, seperti proporsi dataset yang tidak seimbang.

Setelah data PCA, kita tidak bisa lagi mengetahui secara pasti kolom-kolom mana yang signifikan memberikan identitas bahwa observasi tersebut anomali. Namun, kita masih bisa memperkirakannya dengan menggunakan beberapa metode, antara lain:

Analisis komponen utama. Analisis komponen utama dapat digunakan untuk melihat bagaimana masing-masing komponen utama berkontribusi terhadap anomali. Komponen utama yang memiliki kontribusi yang signifikan terhadap anomali dapat dianggap sebagai kolom-kolom yang signifikan memberikan identitas anomali.

- Misalkan, kita memiliki data PCA dengan dua komponen utama, yaitu PC1 dan PC2. Jika komponen PC1 memiliki kontribusi yang signifikan terhadap anomali, maka kita dapat memperkirakan bahwa kolom-kolom yang memiliki korelasi yang tinggi dengan PC1 juga signifikan memberikan identitas anomali.

42 Glossary

- *Variance*: Ukuran seberapa jauh sebuah kumpulan bilangan tersebar. Nilai variance yang mendekati 0 menunjukkan data tidak banyak beragam.
- *Std (Standard Deviation)*: Akar kuadrat dari variance, dapat digunakan untuk menentukan keragaman data. Semakin kecil nilai *Std* maka data semakin tidak beragam dan sebaliknya.
- *Covariance*: Nilai yang menunjukkan hubungan antara variansi pada variable X dan variansi pada variable Y. Tidak dapat di interpretasikan, dikarenakan memiliki nilai negatif tak hingga sampai positif tak hingga.
- *Correlation*: Nilai yang menunjukkan hubungan antar variable X dengan variable Y. Memiliki rentang nilai -1 sampai 1. Nilai korelasi mendekati 1 menujukkan variable X dan Y memiliki hubungan positif kuat, sedangkan nilai korelasi mendekati -1 menunjukkan variable X dan Y memiliki hubungan negatif kuat.

43 Referensi

- In Depth: Principal Component Analysis (Jake VanderPlas)
- Intuisi PCA
- PCA (chapter 3)
- The Importance of Feature Scaling before PCA

Contact Us:
Algoritma Data Indonesia
(+62) 815-1752-5695