Rakamin Academy

Machine Learning Preparation

Data Pre-processing

Outline Pembelajaran

Rakamin Academy

Data Preprocessing

- Mengapa data perlu dipreproses?
- Handling Missing Data
- Handling Duplicated Data
- Handling Outliers
- Feature Transformation
- Feature Encoding
- Class Imbalance

Outline Pembelajaran

Rakamin Academy

Topic Name

- Lorem
- Lorem
- Lorem
- Lorem
- Lorem
- Lorem
- Lorem
- Lorem
- Lorem
- Lorem
- Lorem
- Lorem

Mengapa Data perlu Diprepproses?

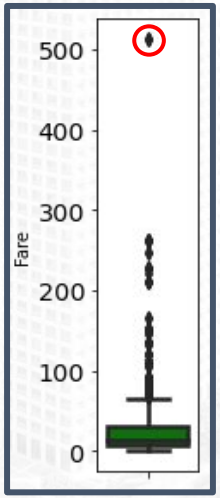
Kebersihan data adalah sebagian dari sukses

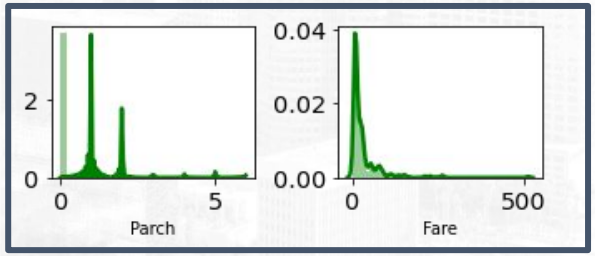
... sebelumnya di Rakamin...

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
male	NaN	0	0	A/S 2816	8.0500	NaN	S
male	32.0	0	0	1601	56.4958	NaN	S
male	26.0	1	2	C.A. 2315	20.5750	NaN	S
male	25.0	1	0	347076	7.7750	NaN	S
male	1.0	5	2	CA 2144	46.9000	NaN	S





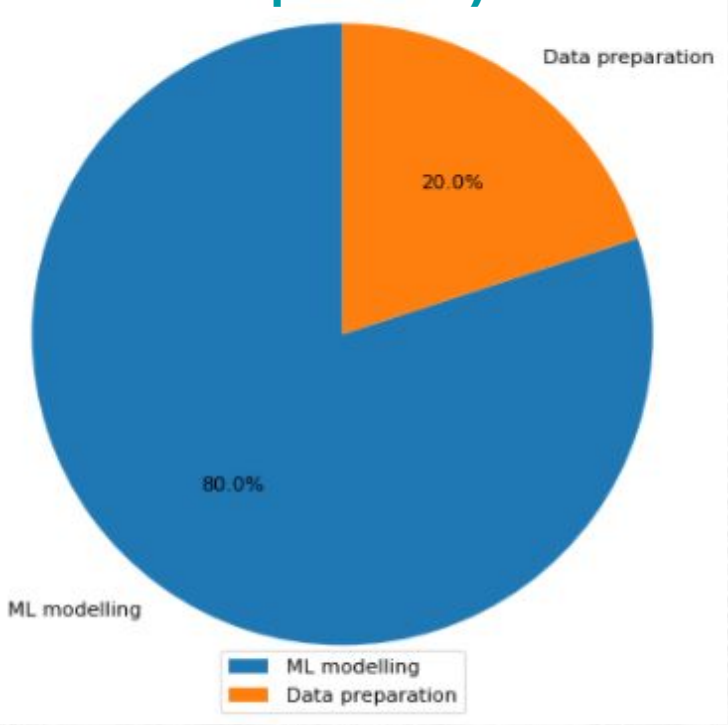
Rakamin Academy

Seberapa kotorkah data-data di dunia nyata? Sangat kotor!

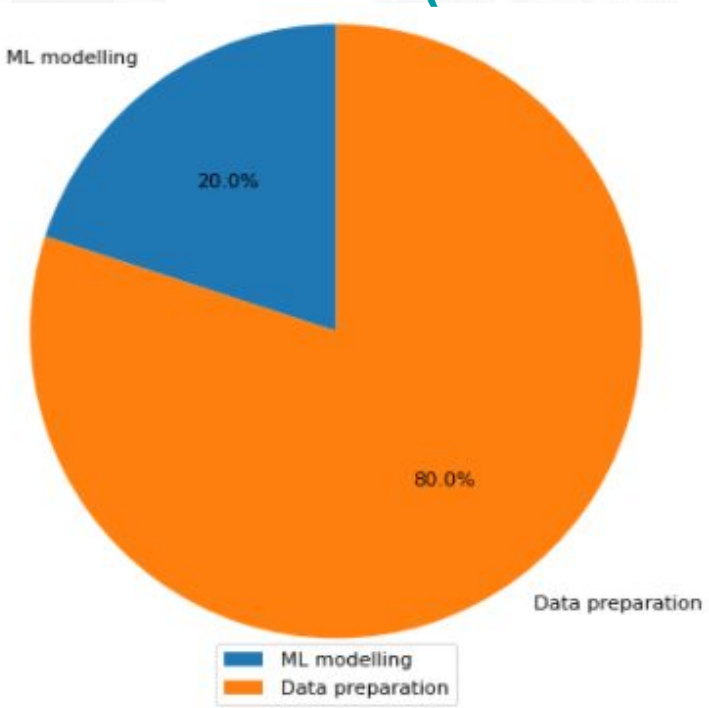
Contoh penyebab ketidakbersihan data:

- User asal mengisi data
- Jenis input yang tidak wajib
- Kesalahan implementasi *tracker* data/ *engineering* mistakes
- *Scammer*, *abuser*
- Keterbatasan responden/sumber data
- Salah logic join tabel
- Dll

Ekspektasi :)



Realita :(



Pemrosesan yang diperlukan seperti apa saja sih?

- Mengubah tipe/format data
- Membersihkan/menambal data-data yang kosong
- Menghilangkan data duplikat yang tidak diinginkan
- Menseleksi data/fitur yang redundan
- Mengubah skala/distribusi data untuk mempermudah *learning*
- Menambahkan data sintetis/duplikat

Ada berbagai macam cara untuk setiap proses di atas masing-masing dengan pertimbangannya sendiri!

Salah satu prinsip penting Machine Learning

Garbage

In

Garbage

Out

Rakamin Academy

Data Pre-processing

Dataset

botak.csv

- **Deskripsi**:

  Dataset sintetik. Memprediksi peluang botaknya seseorang dari beberapa atribut mengenai orang tersebut.

- **Data**:

  Setiap baris mewakili satu orang, setiap kolom berisi atribut orang tersebut.

**ilustrasi tidak ada hubungannya dengan data*

Photo of bald dirty mad man on gray background

Klik disini untuk mengakses folder Hands-On dan Dataset

Load data ke Dataframe

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from matplotlib import rcParams
6
7 print('Numpy version: ', np.__version__)
8 print('Pandas version: ',pd.__version__)
9 print('Seaborn version: ',sns.__version__)
```

```
Numpy version:  1.18.5
Pandas version:  1.0.5
Seaborn version:  0.10.1
```

```
1 rcParams['figure.figsize'] = (10,7)
2 rcParams['lines.linewidth'] = 2.5
3 rcParams['xtick.labelsize'] = 'x-large'
4 rcParams['ytick.labelsize'] = 'x-large'
```

```
[ ]  df = pd.read_csv('botak.csv')
```

Outline Pembelajaran

Rakamin Academy

Data Preprocessing

- [x] Mengapa data perlu dipreproses?
- [ ] Handling Missing Data
- [ ] Handling Duplicated Data
- [ ] Handling Outliers
- [ ] Feature Transformation
- [ ] Feature Encoding
- [ ] Class Imbalance

Missing Data

1: Data yang Hilang

Berapa banyak data yang hilang?

```
df.info()
```

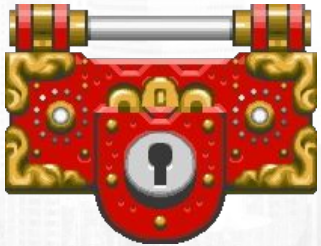
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7991 entries, 0 to 7990
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   umur            7991 non-null   float64
 1   jenis_kelamin   7982 non-null   object
 2   pekerjaan       7924 non-null   object
 3   provinsi        7991 non-null   object
 4   gaji            7968 non-null   float64
 5   is_menikah      7991 non-null   int64
 6   is_keturunan    7976 non-null   float64
 7   berat           7952 non-null   float64
 8   tinggi          7991 non-null   float64
 9   sampo           7934 non-null   object
 10  is_merokok      7991 non-null   int64
 11  pendidikan      7991 non-null   object
 12  botak_prob      7991 non-null   float64
dtypes: float64(6), int64(2), object(5)
memory usage: 811.7+ KB
```

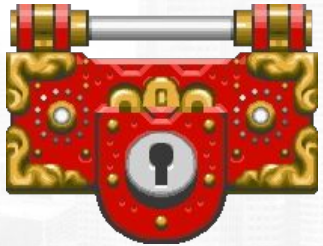
```
df.isna().sum()
```

```
umur             0
jenis_kelamin    9
pekerjaan       67
gaji            23
is_menikah       0
is_keturunan    15
berat           39
tinggi           0
sampo           57
is_merokok       0
pendidikan       0
botak_prob       0
dtype: int64
```

Ada 2 cara mengecek jumlah nilai yang hilang pada dataframe:

- `df.info()`
- `df.isna().sum()`





Kira-kira bagaimana menghadapi data yang kosong?

Teknik #1: Hapus (drop) baris-baris dengan data yang hilang

Rakamin Academy

Ketika kita punya cukup banyak data dan jumlah data yang hilang tidak signifikan, biasanya cukup hapus baris-baris dengan data yang hilang.

```
1 df = df.dropna()
2 df = df.dropna(subset=['umur', 'jenis_kelamin'])
3 df.dropna(inplace=True)
4 df.dropna(subset=['umur', 'jenis_kelamin'], inplace=True)
```

Penghapusan dapat dilakukan dengan fungsi `df.dropna()`. Kode di atas menunjukkan contoh 4 cara berbeda menggunakan `df.dropna()`.

Berikut penjelasan untuk 2 parameter yang dipakai di atas:

1. `subset`: hanya hapus baris dengan nilai kosong di kolom-kolom yang diberikan.
2. `inplace`: Nilai `True` atau `False`. Apabila `True`, tidak mengembalikan dataframe baru tapi langsung menghapus di dataframe awal.

Teknik #2: Isi data-data yang kosong/Imputation (Numeric)

Rakamin Academy

Ketika kita tidak mau menghapus satu pun baris data, kita bisa mengisi kekosongan data secara manual.

Kita isi dengan apa? Biasanya ada 2 hal yang dipertimbangkan:

- Konteks masalah: nilai apa yang paling masuk akal?
- Performa model ML: nilai apa yang menghasilkan performa model tertinggi?

```
df = df.fillna(df.mean())
df = df.fillna(df.min())
df = df.fillna(df.max())
df = df.fillna(0)
```

Pengisian dapat dilakukan dengan fungsi `df.fillna()`. Kode di atas menunjukkan contoh pengisian data dengan 4 jenis nilai:

1. Rata-rata nilai pada kolom
2. Minimal nilai pada kolom
3. Maksimal nilai pada kolom
4. Nilai konstan

Teknik #2: Isi data-data yang kosong/Imputation (Numeric)

```
1 df.fillna(df.mean(), inplace=True)
```

(1)

```
1 df['umur'].fillna(df['umur'].mean(), inplace=True)
2 df['umur'] = df['umur'].fillna(df['umur'].mean())
```

(2)

Kode di atas menunjukkan contoh-contoh alternatif penggunaan `df.fillna()`. Berikut penjelasan untuk 2 alternatif di atas:

1. Menggunakan parameter `inplace` untuk langsung mengisi di dataframe awal
2. Melakukan pengisian hanya di kolom tertentu, nomor 1 dengan parameter `inplace`, nomor 2 tanpa parameter `inplace`.

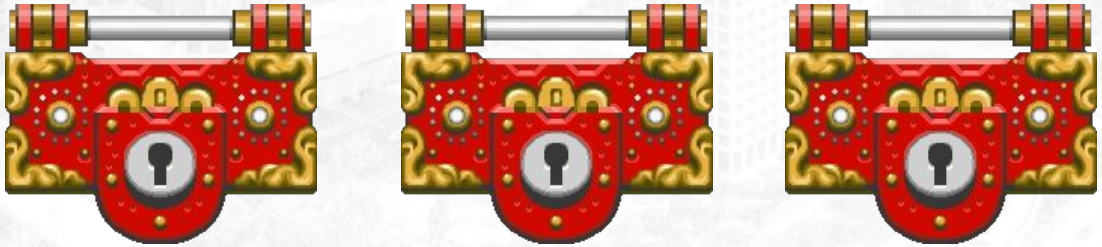
** Penting: pengisian untuk seluruh dataframe (seperti contoh di slide sebelumnya dan nomor 1 di slide ini) hanya akan mengisi kolom numerik.*

Quizz!!!

Misalkan kalian adalah DS yang mengembangkan Algoritma Search di e-commerce TokoPakEdi.

Diisi apakah missing data pada fitur-fitur berikut?

1. Presentase transaksi legit (BUKAN fraud) pada product_id
2. Rating product_id
3. Presentase penolakan proses transaksi oleh seller_id



Teknik #2: Isi data-data yang kosong/Imputation (Categorical)

```
1 df['pekerjaan'] = df['pekerjaan'].fillna('PNS')
2 df['pekerjaan'] = df['pekerjaan'].fillna(df['pekerjaan'].mode()[0])
```

Pengisian kolom categorical hanya bisa dilakukan manual per-kolom!

Kode di atas menunjukkan contoh-contoh penggunaan `df.fillna()` untuk kolom categorical dengan 2 jenis nilai:

1. Nilai konstan
2. Modus pada kolom

Outline Pembelajaran

Rakamin Academy

Data Preprocessing

- [x] Mengapa data perlu dipreproses?
- [x] Handling Missing Data
- [ ] Handling Duplicated Data
- [ ] Handling Outliers
- [ ] Feature Transformation
- [ ] Feature Encoding
- [ ] Class Imbalance

Duplicated Data

2: Data yang Sama

Berapa banyak data yang duplikat?

```
1 df.duplicated().sum()
```

```
0
```

Pengecekan jumlah nilai yang hilang pada dataframe dapat dilakukan dengan `df.duplicated().sum()`.

```
1 df.duplicated(subset=['umur', 'sampo', 'berat']).sum()
```

```
15
```

Parameter `subset` dapat digunakan untuk mengecek duplikat pada kumpulan kolom tertentu saja. Contoh:

umur	sampo	berat	is_merokok
22	Moonsilk	78	1
22	Moonsilk	78	0

- Tanpa subset (memperhitungkan semua kolom): 0 duplikat
- Dengan subset [‘umur’, ‘sampo’, ‘berat’]: 1 duplikat

Hapus (drop) baris-baris duplikat (1)

Ketika kita yakin kita tidak memerlukan baris-baris duplikat, biasanya cukup hapus baris-baris tersebut.

```
1 df = df.drop_duplicates()
2 df = df.drop_duplicates(subset=['umur', 'jenis_kelamin'])
3 df.drop_duplicates(inplace=True)
4 df.drop_duplicates(subset=['umur', 'jenis_kelamin'], inplace=True)
```

Penghapusan dapat dilakukan dengan fungsi `df.drop_duplicates()`. Kode di atas menunjukkan contoh 4 cara berbeda menggunakan `df.drop_duplicates()`.

Berikut penjelasan untuk 2 parameter yang dipakai di atas:

1. `subset`: hanya hapus baris duplikat berdasarkan kolom-kolom yang diberikan.
2. `inplace`: Nilai `True` atau `False`. Apabila `True`, tidak mengembalikan dataframe baru tapi langsung menghapus di dataframe awal.

Parameter `keep`

```
1 df = df.drop_duplicates(keep='first')
2 df = df.drop_duplicates(keep='last')
3 df = df.drop_duplicates(keep=False)
```

Selain 2 fungsi tadi, `df.drop_duplicates()` juga memiliki parameter penting lain bernama `keep`. Kode di atas menunjukkan 3 pilihan untuk parameter tersebut. Berikut penjelasannya:

1. `first`: Simpan baris duplikat yang paling atas, sisanya dihapus (default)
2. `last`: Simpan baris duplikat yang paling bawah, sisanya dihapus
3. `False`: Hapus semua baris duplikat

Parameter keep

Rakamin Academy

first

...	23	Laki-laki	...
~~...~~	~~23~~	~~Laki-laki~~	~~...~~
~~...~~	~~23~~	~~Laki-laki~~	~~...~~

last

~~...~~	~~23~~	~~Laki-laki~~	~~...~~
~~...~~	~~23~~	~~Laki-laki~~	~~...~~
...	23	Laki-laki	...

False

~~...~~	~~23~~	~~Laki-laki~~	~~...~~
~~...~~	~~23~~	~~Laki-laki~~	~~...~~
~~...~~	~~23~~	~~Laki-laki~~	~~...~~

Outline Pembelajaran

Rakamin Academy

Data Preprocessing

- [x] Mengapa data perlu dipreproses?
- [x] Handling Missing Data
- [x] Handling Duplicated Data
- [ ] Handling Outliers
- [ ] Feature Transformation
- [ ] Feature Encoding
- [ ] Class Imbalance

Outliers

3: Data yang Berbeda (jauh)

Outlier itu apa?

Outlier adalah data point (baris) yang nilainya ekstrim/jauh berbeda dari data-data lain pada umumnya. Bisa muncul dari:

- Kesalahan pada pengambilan data
- Keberadaan individu-individu yang ‘spesial’

Kenapa outlier jadi masalah?

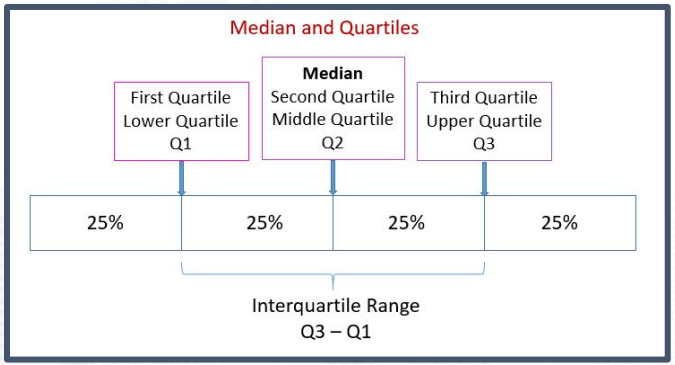
Rakamin Academy

Data Nilai Ujian Biologi Murid SMP Negeri 666 Paku

Jenis Kelamin	Presensi Kelas Bio	Uang Jajan/Hari	Nilai Ujian
Laki-laki	60%	100 Ribu	70
Laki-laki	80%	50 ribu	85
Perempuan	100%	80 ribu	92
Laki-laki	80%	0	90
Perempuan	40%	0	70
Perempuan	60%	50 ribu	75
Perempuan	100%	**100 juta**	90

Rumus nilai ujian ???

Menghapus outlier berdasarkan *IQR*



IQR: lebar Q3-Q1

Outlier: Lebih ekstrim dari 1.5 IQR dari Q1 atau Q3

```
1 Q1 = df['umur'].quantile(0.25)
2 Q3 = df['umur'].quantile(0.75)
3 IQR = Q3 - Q1
4 low_limit = Q1 - (1.5 * IQR)
5 high_limit = Q3 + (1.5 * IQR)
6 filtered_entries = ((df['umur'] >= low_limit) & (df['umur'] <= high_limit))
7 df = df[filtered_entries]
```

Menghapus outlier berdasarkan *IQR*

```
1 Q1 = df['umur'].quantile(0.25)
2 Q3 = df['umur'].quantile(0.75)
3 IQR = Q3 - Q1
4 low_limit = Q1 - (1.5 * IQR)
5 high_limit = Q3 + (1.5 * IQR)
6 filtered_entries = ((df['umur'] >= low_limit) & (df['umur'] <= high_limit))
7 df = df[filtered_entries]
```

Kode di atas menunjukkan cara menghapus baris berdasarkan outlier di kolom `umur` menggunakan IQR. Berikut penjelasannya:

1. Hitung Q1
2. Hitung Q3
3. Hitung IQR
4. Hitung batas bawah untuk outlier
5. Hitung batas atas untuk outlier
6. Buat filter `boolean` berdasarkan apakah nilai di bawah batas bawah atau di atas batas atas
7. Pakai filter untuk memilih baris yang BUKAN merupakan outlier

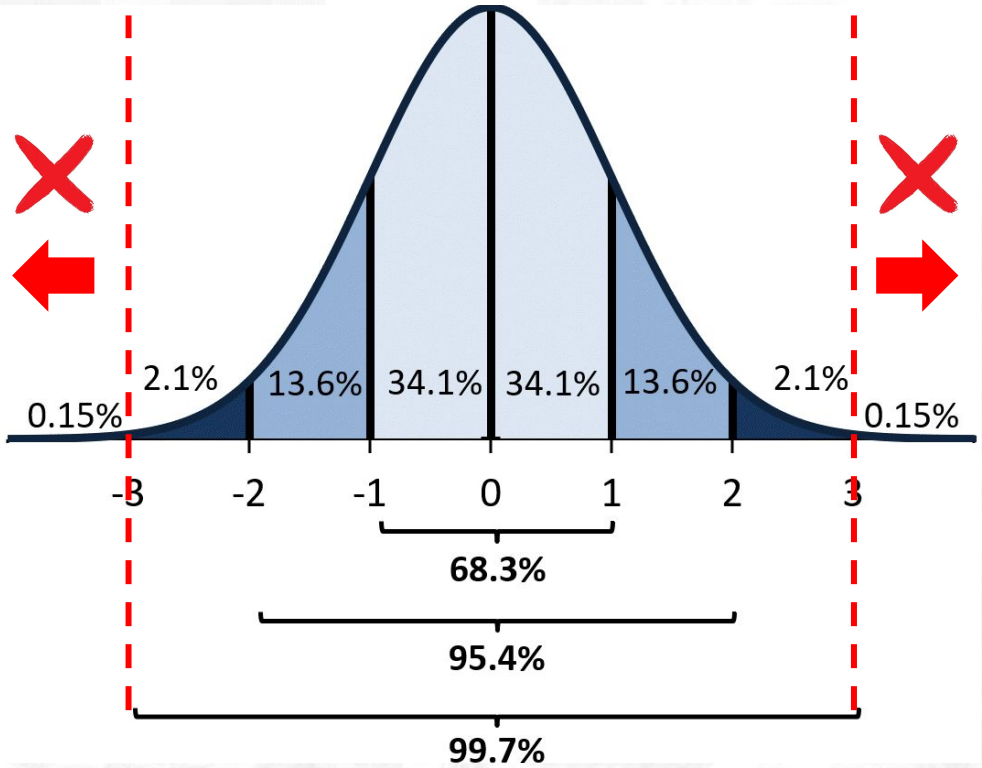
Menghapus outlier berdasarkan *Z-score*

Z-score: berapa kali *standard deviation* jarak sebuah nilai dari rata-rata kolom

Outlier: abs(Z-score) > 3

- Kita membuang ~0.3% data paling ekstrim (asumsi data berdistribusi normal)

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



Menghapus outlier berdasarkan *Z-score*

```
1 from scipy import stats
```

```
1 z_scores = np.abs(stats.zscore(df['umur']))
2 filtered_entries = (z_scores < 3)
3 df = df[filtered_entries]
```

Kode di atas menunjukkan cara menghapus baris berdasarkan outlier di kolom `umur` menggunakan Z-score. Berikut penjelasannya:

1. Hitung Z-score absolut untuk semua baris dari kolom `umur`
2. Buat filter `boolean` berdasarkan apakah Z-score kurang dari 3
3. Pakai filter untuk memilih baris data yang BUKAN outlier

umur	abs_zscore	filtered_entries
63.0	2.391707	True
14.0	2.595184	True
69.0	3.002347	False
64.0	2.493481	True

umur	abs_zscore	filtered_entries
63.0	2.391707	True
14.0	2.595184	True
[redacted]		
64.0	2.493481	True

umur	abs_zscore
63.0	2.391707
14.0	2.595184
64.0	2.493481

Short break!
Salah satu lebah di gambar ini adalah ‘outlier’ Ratu Lebah
Penemu tercepat akan mendapatkan ucapan selamat

Outline Pembelajaran

Rakamin Academy

Data Preprocessing

- [x] Mengapa data perlu dipreproses?
- [x] Handling Missing Data
- [x] Handling Duplicated Data
- [x] Handling Outliers
- [ ] Feature Transformation
- [ ] Feature Encoding
- [ ] Class Imbalance

Feature Transformation

Fitur-fitur dengan skala berbeda & terkait dengan normalitas data

Apa saja transformasi fitur yang umum digunakan?

1. *Normalization* adalah proses mengubah nilai-nilai suatu feature menjadi skala tertentu

2. *Standardization* adalah proses mengubah nilai-nilai feature sehingga `mean = 0` *dan* `standard deviation = 1`

3. *Log Transformation* adalah proses mengubah nilai-nilai suatu feature dengan menggunakan fungsi logaritma

Apa efek dari setiap transformasi?

Transformasi	Efek
Normalization	• Hard feature scaling: merubah range dari nilai fitur dengan batas range yang pasti/rigid • TIDAK merubah bentuk sebaran data
Standardization	• Soft feature scaling: merubah range dari nilai fitur dengan batas yang tidak saklek • MERUBAH bentuk sebaran data menjadi mendekati distribusi normal • **Hasil transformasi memiliki nilai rataan 0 & simpangan baku 1**
Log transformation	• Soft feature scaling: merubah range dari nilai fitur dengan batas yang tidak saklek (lebih *loose* dibanding standardization) • MERUBAH bentuk sebaran data menjadi mendekati normal • **Digunakan pada fitur yang sebaran aslinya right-skewed (long right tailed)**

Kenapa kita perlu *Normalization/Standardization*?

- Data dengan skala yang sama akan menjamin algoritma pembelajaran memperlakukan semua feature dengan adil
- Data dengan skala yang sama dan *centered* akan mempercepat algoritma pembelajaran (model training)
- Data dengan skala yang sama akan mempermudah interpretasi beberapa model ML

Rakamin Academy

Data Nilai Ujian Biologi Murid SMP Negeri 666 Paku

Jenis Kelamin	Presensi Kelas Bio	Uang Jajan/Hari	Nilai Ujian
Laki-laki	60%	100 Ribu	70
Laki-laki	80%	50 ribu	85
Perempuan	100%	80 ribu	92
Laki-laki	80%	0	90
Perempuan	40%	0	70
Perempuan	60%	50 ribu	75
Perempuan	100%	100 ribu	90

Rumus nilai ujian:

- 50 + (Presensi Kelas / 2) - (Uang Jajan / 10 ribu) ATAU
- 50 + (0.5 * presensi) - (0.0001 * jajan)

Apakah presensi 5000x lebih penting dalam menentukan nilai ujian?

Normalizing/Standardizing dengan `sklearn.preprocessing`

```
1 from sklearn.preprocessing import MinMaxScaler, StandardScaler
2 df['gaji_norm'] = MinMaxScaler().fit_transform(df['gaji'].values.reshape(len(df), 1))
3 df['gaji_std'] = StandardScaler().fit_transform(df['gaji'].values.reshape(len(df), 1))
```

Normalization **dan** ***standardization*** **dapat dilakukan dengan menggunakan** **`MinMaxScaler`** **dan** **`StandardScaler`** **pada library** **`sklearn.preprocessing`**.

Kode di atas menunjukkan cara melakukan ***normalization*** **dan** ***standardization*** **data dikolom** **`gaji`** **(hanya berbeda jenis scalernya). Berikut penjelasan tiap bagiannya:**

1. `df['gaji'].values` mengeluarkan nilai-nilai dari kolom `gaji`
2. `.reshape(len(df), 1)` mengubah bentuk array menjadi format yang diperlukan
3. `MinMaxScaler().fit_transform()` atau `StandardScaler().fit_transform()` melakukan *normalization/standardization*
4. Hasilnya disimpan di kolom `gaji_norm` dan `gaji_std`

Normalizing/Standardizing dengan `sklearn.preprocessing`

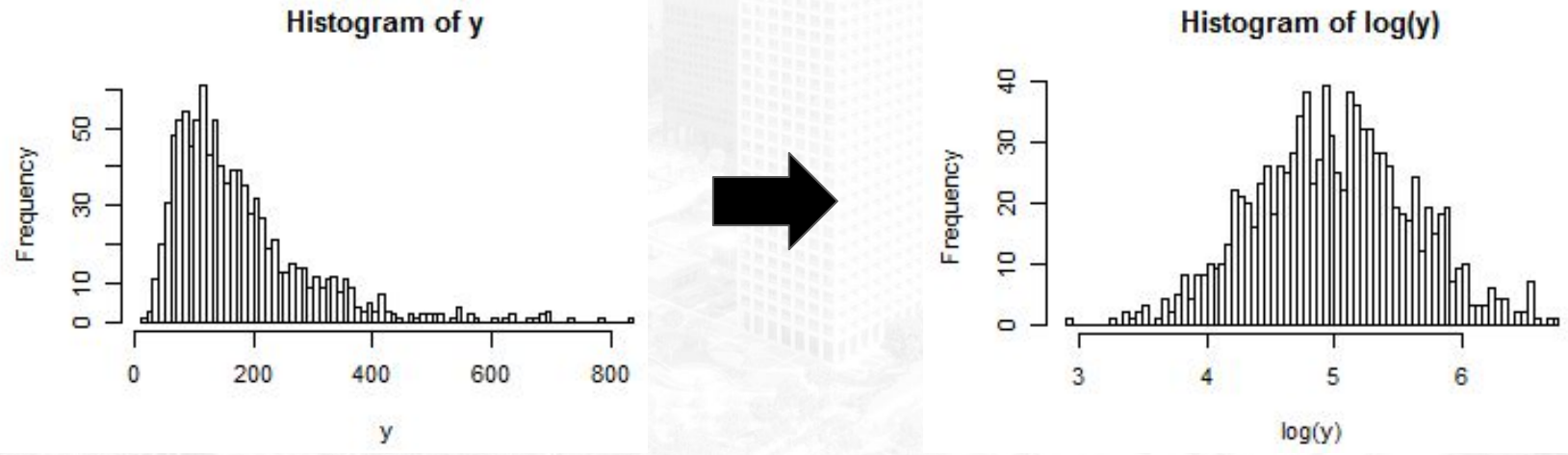
```
1 df[['gaji', 'gaji_norm', 'gaji_std']].describe()
```

	gaji	gaji_norm	gaji_std
count	7.759000e+03	7759.000000	7.759000e+03
mean	7.464218e+06	0.122974	8.453664e-17
std	3.728615e+06	0.076879	1.000064e+00
min	1.500000e+06	0.000000	-1.599683e+00
25%	4.893492e+06	0.069969	-6.895031e-01
50%	6.637149e+06	0.105921	-2.218311e-01
75%	9.139816e+06	0.157522	4.494177e-01
max	5.000000e+07	1.000000	1.140867e+01

Ketika kita tampilkan statistik kolom yang dihasilkan menggunakan `describe()`, dapat terlihat bahwa *normalization* dan *standardization* telah berhasil.

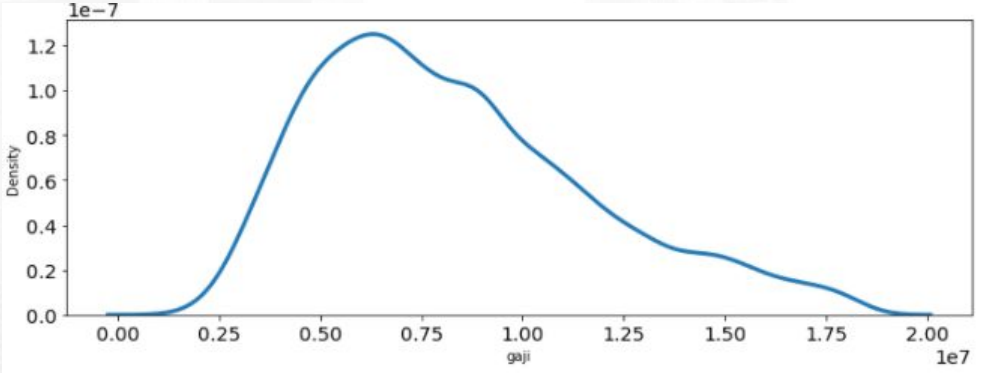
Log Transformation

- Log Transformation digunakan pada data yang right-skewed (i.e. punya 'buntut' yang panjang di kanan)
- Distribusi hasil transformasi akan mendekati distribusi normal, seperti gambar di bawah
  - Perhatikan perubahan skala pada sumbu x!

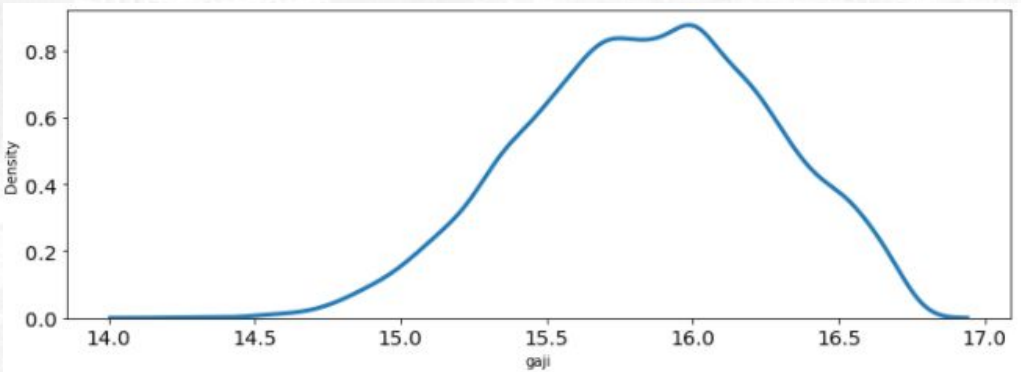


Log Transformation

```
# distribusi gaji (nilai asli)
sns.kdeplot(df['gaji'])
```




```
# distribusi gaji (setelah log transformation)
sns.kdeplot(np.log(df['gaji']))
# lebih mendekati distribusi normal!
```




```
# kita tranformasi
df['log_gaji'] = np.log(df['gaji'])

# drop kolom gaji (nilai asli)
df = df.drop(columns='gaji')
```

Outline Pembelajaran

Rakamin Academy

Data Preprocessing

- [x] Mengapa data perlu dipreproses?
- [x] Handling Missing Data
- [x] Handling Duplicated Data
- [x] Handling Outliers
- [x] Feature Transformation
- [ ] Feature Encoding
- [ ] Class Imbalance

Rakamin Academy

Feature Encoding

Mengakomodasi feature categorical

Feature Encoding itu apa?

Feature Encoding **adalah proses mengubah feature categorical menjadi feature numeric.**

Mengapa kita perlu *feature encoding*? Tak semua model/algoritma ML dapat menggunakan feature categorical.

Data Besaran Penghasilan dari Survei Abhal^2

Gender	Pendidikan	Pekerjaan	Penghasilan (juta)
Laki-laki	S1	SWASTA	7
Laki-laki	SMA	PNS	13
Perempuan	S1	PNS	15
Laki-laki	S2	FREELANCE	24
Perempuan	S3	PNS	17
Perempuan	S1	SWASTA	23
Perempuan	SMA	FREELANCE	12

Pertanyaan

Bagaimana cara menulis rumus dari Penghasilan secara matematis?

Teknik #1: Label Encoding

Label Encoding adalah perubahan feature categorical menjadi numeric dengan memberikan angka yang berbeda bagi masing-masing nilai unik

```
mapping_gender = {
    'Laki-laki': 0,
    'Perempuan': 1
}
df['gender'] = df['gender'].map(mapping_gender)

mapping_pendidikan = {
    'SMA': 0,
    'S1': 1,
    'S2': 2,
    'S3': 3
}
df['pendidikan'] = df['pendidikan'].map(mapping_pendidikan)
```

	gender	pendidikan	pekerjaan	penghasilan
0	0	1	SWASTA	7
1	0	0	PNS	13
2	1	1	PNS	15
3	0	2	FREELANCE	24
4	1	3	PNS	17
5	1	1	SWASTA	23
6	1	0	FREELANCE	12

Lalu bagaimana dengan kolom ‘pekerjaan’?

Teknik #2: One-hot Encoding

One-hot encoding adalah perubahan feature categorical menjadi numeric dengan menjadikan masing-masing nilai unik feature tersendiri

```
1 pd.get_dummies(df['pekerjaan'], prefix='kerja')
```

Kode di atas menunjukkan cara melakukan one-hot encoding pada kolom `pekerjaan` menggunakan `get_dummies()`. Berikut penjelasannya:

1. Parameter pertama adalah kolom yang ingin di one-hot encoding (`pekerjaan`)
2. Parameter `prefix` diisi dengan nama awalan dari kolom-kolom baru yang akan dihasilkan
3. Fungsi ini akan mengembalikan dataframe baru yang berisi feature-feature numerik

Teknik #2: One-hot Encoding (lanjutan)

	kerja_FREELANCE	kerja_PNS	kerja_SWASTA
0	0	0	1
1	0	1	0
2	0	1	0
3	1	0	0
4	0	1	0
5	0	0	1
6	1	0	0

Ketika kita tampilkan dataframe yang dihasilkan terlihat bahwa setiap nilai unik berubah menjadi kolom baru. Awalan nama kolom-kolom baru ini sesuai dengan isi parameter `prefix`.

Data Besaran Penghasilan dari Survei Abhal^2 (encoded)

gender	pendidikan	freelance	PNS	swasta	penghasilan
0	1	0	0	1	7
0	0	0	1	0	13
1	1	0	1	0	15
0	2	1	0	0	24
1	3	0	1	0	17
1	1	0	0	1	23
1	0	1	0	0	12

Rumus nilai penghasilan:

- A * gender + B * pendidikan + C * freelance + D * PNS + E * swasta
- Dimana A, B, C, D, E didapat dari melatih model machine learning (regresi)

Recap: Label Encoding atau OHE?

- **Gunakan Label Encoding pada:**
  - Kolom kategorikal dengan jumlah distinct values = 2. E.g. Gender, respon ya/tidak, etc
  - Kolom kategorikal dengan tipe ordinal (punya urutan). E.g. tingkat pendidikan, intensitas (rendah/medium/tinggi), socio-economic status (A/B/C/D), etc

- **Selainnya, gunakan One Hot Encoding (OHE)**

Bagaimana kalau nilai unik feature terlalu banyak dan berpotensi memanggil kutukan dimensi? (misal: Kota)

- **Kerucutkan feature menjadi feature baru dengan nilai unik lebih sedikit (misal: propinsi)**
- **Batasi sejumlah N nilai fitur dengan frekuensi tertinggi, lalu encode sisanya sebagai value 'others'**
- Buang feature
- Gunakan model/algoritma ML yang mengakomodasi feature categorical
- ... atau beli komputer yang lebih kuat

Model/algoritma Machine Learning dapat mengakomodasi ribuan bahkan jutaan feature dengan teknik-teknik tertentu dan komputer yang cukup kuat

Outline Pembelajaran

Rakamin Academy

Data Preprocessing

- Mengapa data perlu dipreproses?
- Handling Missing Data
- Handling Duplicated Data
- Handling Outliers
- Feature Transformation
- Feature Encoding
- Class Imbalance

Rakamin Academy

Class Imbalance

Kesenjangan antar kelas

Class imbalance itu apa?

Class imbalance **adalah sebuah kondisi dalam masalah klasifikasi dimana distribusi nilai unik pada target *sangat timpang*.**

Mengapa *class imbalance* dapat menjadi masalah?

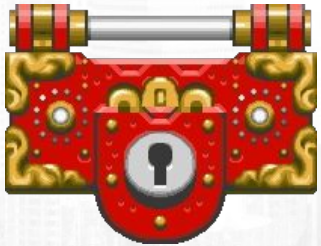
Data Rekap Kelulusan Murid SMP X ++

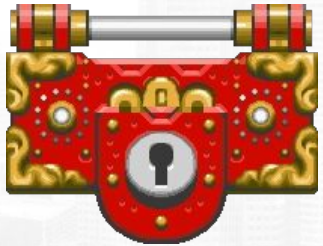
Presensi	Nilai Ujian	Donasi	# Skors	Prestasi	Beasiswa	...	...	Lulus
90%	60	1 Milyar	4 Hari	...	...	...	...	YA
70%	70	0	2 Hari	...	...	...	...	YA
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
20%	10	5 Milyar	0	...	...	...	...	YA
80%	80	1 Milyar	0	...	...	...	...	YA

Misalkan:

- **Ada 100 ribu baris data**
- **Ada sekitar 700 anak yang tidak lulus**

Bisakah kita mencari aturan yang memiliki akurasi >99%?





Kira-kira aturan sederhana apa yang menjamin akurasi > 99%?

Jawaban: luluskan saja semuanya.

Apakah aturan ini berguna di dunia nyata?

Algoritma pembelajaran pada Machine Learning biasanya bertujuan untuk memaksimalkan ukuran akurasi.

***Class imbalance* bermasalah karena kondisi ini membuat algoritma machine learning menjadi bodoh.**

Bagaimana cara mengatasi *class imbalance*?

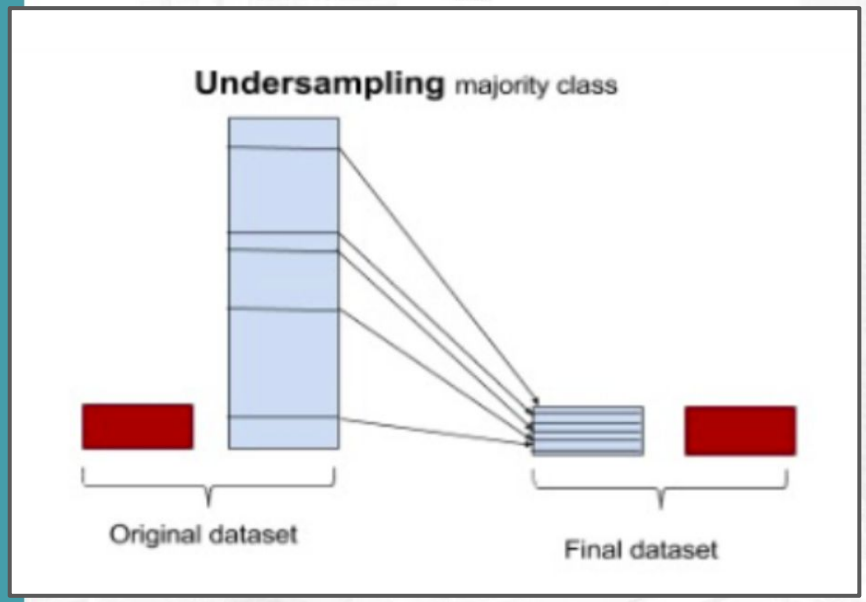
- Berikan ukuran akurasi yang lebih 'pintar' (sesi selanjutnya)
- Hilangkan *class imbalance* pada data dengan over/undersampling

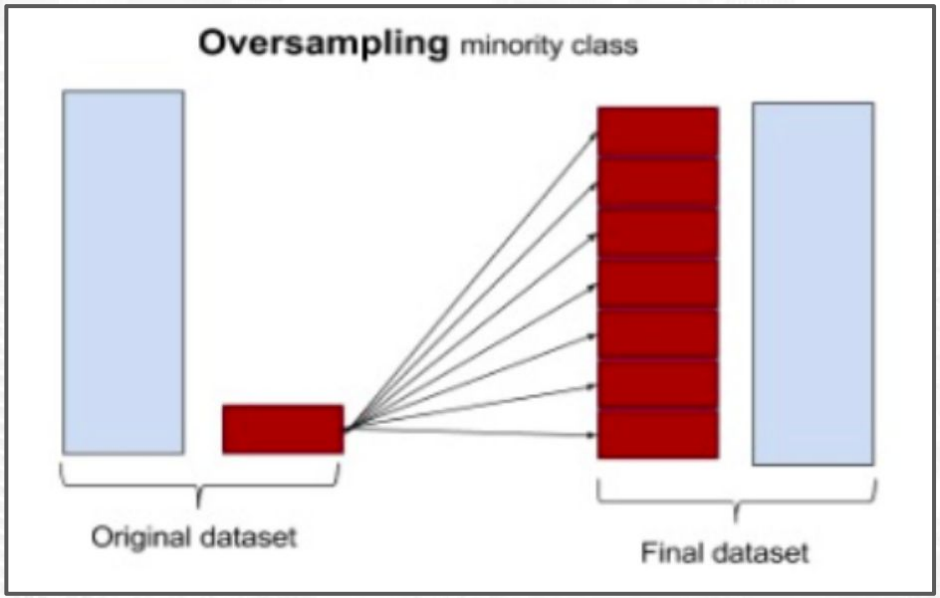
Derajat Ketimpangan Data

Degree of imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	<1% of the data set

Sumber: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data

Oversampling + Undersampling





Oversampling + Undersampling dengan `imblearn` (1)

- **Oversampling:** menduplikat data minoritas
- **Undersampling:** menghapus data mayoritas

```
1 df['botak_class'] = df['botak_prob'] > 0.8
2 print(df['botak_class'].value_counts())
```

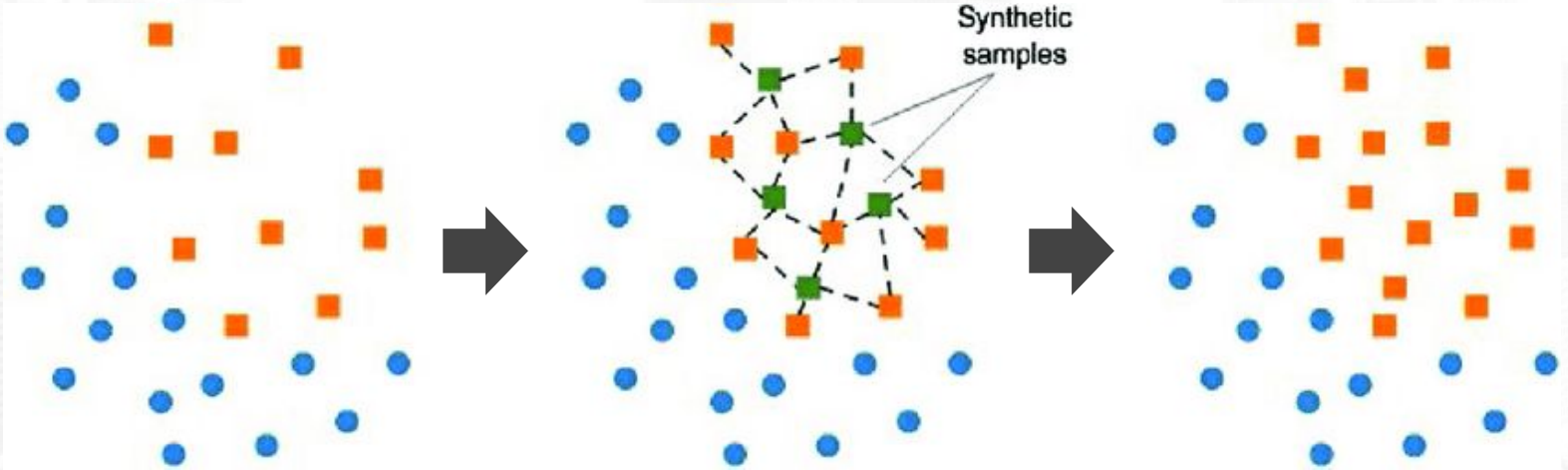
```
False    6810
True      111
Name: botak_class, dtype: int64
```

Sebelum melakukan over/under sampling, kita membuat target baru yang bertipe categorical (kiri) dan memisahkan feature dan target kedalam variabel `X` dan `y` (bawah)

```
1 X = df[[col for col in df.columns if col not in ['botak_class', 'botak_prob']]].values
2 y = df['botak_class'].values
3 print(X.shape)
4 print(y.shape)
```

```
(6921, 15)
(6921,)
```

SMOTE: Synthetic Minority Oversampling TEchnique



SMOTE merupakan salah satu algoritma oversampling yang umum digunakan. Data sintetis akan dibuat di antara kelas minoritas (interpolasi antar titik-titik data original)

Oversampling + Undersampling dengan `imblearn` (2)

```
1 from imblearn import under_sampling, over_sampling
2 X_under, y_under = under_sampling.RandomUnderSampler(0.5).fit_resample(X, y)
3 X_over, y_over = over_sampling.RandomOverSampler(0.5).fit_resample(X, y)
4 X_over_SMOTE, y_over_SMOTE = over_sampling.SMOTE().fit_resample(X, y)
```

***Oversampling* dan *undersampling* dapat dilakukan dengan menggunakan `RandomOverSample`, `RandomUnderSampler`, `SMOTE` pada library `imblearn`.**

Kode di atas menunjukkan cara melakukan *oversampling* dan *undersampling* data (hanya berbeda jenis samplernya). Berikut penjelasan tiap bagian kodenya:

1. `over_sampling.RandomOverSampler().fit_resample(X, y)` melakukan oversampling/undersampling /SMOTE

2. Hasilnya disimpan di pasangan variabel `X_over, y_over` (untuk *oversampling*), `X_over_SMOTE, y_over_SMOTE, dan X_under, y_under` (untuk *undersampling*)

Oversampling + Undersampling dengan `imblearn` (3)

```
1 print(pd.Series(y).value_counts())
```

```
False    6810
True      111
dtype: int64
```

```
1 print(pd.Series(y_over).value_counts())
```

```
True     6810
False    6810
dtype: int64
```

```
1 print(pd.Series(y_under).value_counts())
```

```
True     111
False    111
dtype: int64
```

3 potongan kode di samping memperlihatkan distribusi kelas target sebelum dan setelah sampling:

- *Class imbalance* pada data awal (atas)
- Kelas seimbang melalui duplikat (*oversampling*, tengah)
- Kelas seimbang melalui penghapusan (*undersampling*, bawah)

Outline Pembelajaran

Rakamin Academy

Data Preprocessing

- Mengapa data perlu dipreproses?
- Handling Missing Data
- Handling Duplicated Data
- Handling Outliers
- Feature Transformation
- Feature Encoding
- Class Imbalance

Hands-on: Prediksi Waktu Kedatangan Pesanan

Hands-On Required :

Hands - On :

Hands-On - Data Preprocessing.ipynb

Dataset :

1. Food_Delivery_Dataset.csv

Klik disini untuk mengakses folder Hands-On dan Dataset

Sudah.

Sesi tanya-jawab