Jane Doe and Max Power

zalza Book

To blah, blah, and blah.

Table of contents

Preface

This is a Quarto book.

Software conventions

```
1 + 1
```

```
2
```

To learn more about Quarto books visit https://quarto.org/docs/books.

Acknowledgments

Blah, blah, blah...

1

TUGAS KLASIFIKASI DATA PROYEK SAINS DATA - B

Nama : Zalzabila Rani
NIM : 210411100082
Kelas : B

1.1 — BUSSINESS UNDERSTANDING —

Tujuan = Untuk membangun model yang dapat memprediksi hasil dan perkembangan kanker payudara pada pasien dengan node positif. Dengan menganalisis kumpulan data kasus kanker payudara, model ini akan mempelajari pola dan korelasi untuk membuat prediksi yang akurat, sehingga memungkinkan tenaga kesehatan profesional untuk mengidentifikasi pasien berisiko tinggi dan membantu dokter memilih strategi perawatan yang lebih baik.

Kanker payudara adalah penyakit di mana sel-sel di dalam payudara berkembang tidak terkendali dan membentuk massa atau benjolan yang disebut tumor. Tumor ini dapat menjadi ganas (kanker) jika sel-selnya dapat menyebar ke bagian tubuh lain. Kanker payudara bisa terjadi pada wanita dan pria, tetapi lebih umum pada wanita.

Faktor risiko kanker payudara termasuk genetika, usia, jenis kelamin, dan faktor lingkungan. Pemeriksaan payudara sendiri, mamografi, dan kunjungan rutin ke dokter membantu dalam deteksi dini. Gejala kanker payudara mungkin meliputi benjolan atau perubahan bentuk payudara, perubahan pada kulit, atau keluarnya cairan dari puting.

Pengobatan kanker payudara dapat melibatkan pembedahan, radioterapi, kemoterapi, dan terapi hormon, tergantung pada jenis dan stadium kanker. Deteksi dini dan perawatan yang tepat dapat meningkatkan peluang kesembuhan. Penting untuk meningkatkan kesadaran tentang kanker payudara, mendorong pemeriksaan payudara rutin, dan mendukung mereka yang mengalami penyakit ini.

Kumpulan data berisi catatan pasien dari uji coba tahun 1984-1989 yang dilakukan oleh Kelompok German Breast Cancer Study Group (GBSG) terhadap 720 pasien dengan kanker payudara nodus positif.

Referensi data bisa diakses disini[1].

Dataset ini memiliki 686 data dengan rincian pelabelan sebagai berikut :

Label

Meaning

0

Hidup tanpa kambuh

1

Kambuh atau kematian(sudah meninggal)

1.1.1 Ciri - ciri yang digunakan untuk mengklasifikasi kanker payudara

Untuk mendeteksi kanker payudara masyarakat German, dapat dilihat dari ciri-ciri sebagai berikut :

1. `age (age, years)`
2. `meno (menopausal status)`
3. `size (tumor size, mm)`
4. `grade (tumor grade)`
5. `nodes (number of positive lymph nodes)`
6. `pgr (progesterone receptors, fmol/l)`
7. `er (estrogen receptors, fmol/l)`
8. `Hormon (hormonal therapy)`
9. `rfstime (recurrence free survival time; days to first of recurrence, death or last follow-up`

Deskripsi Fitur:

1. `age (age, years)`

   Ini mencatat usia pasien dalam tahun. Informasi ini membantu untuk memahami distribusi usia dalam sampel.

2. `meno (menopausal status)`

[1] https://www.kaggle.com/datasets/utkarshx27/breast-cancer-dataset-used-royston-and-altman

Ini mencatat status menopause pasien. Nilai 0 mewakili pre-menopause, sedangkan nilai 1 mewakili postmenopause.

3. `size (tumor size, mm)`

   Ukuran tumor dalam milimeter. Informasi ini membantu untuk memahami seberapa besar tumor pada pasien.

4. `grade (tumor grade)`

   Ini mencatat tingkat keparahan tumor. Menggambarkan seberapa "berat" atau "parah" tumor itu. Nilai yang lebih tinggi menunjukkan tumor yang lebih agresif atau lebih berbahaya.

5. `nodes (number of positive lymph nodes)`

   Ini mencatat jumlah nodus limfe positif pada pasien. ni memberikan informasi tentang seberapa banyak kanker menyebar ke kelenjar getah bening.

6. `pgr (progesterone receptors, fmol/l)`

   Menunjukkan kadar reseptor progesteron dalam darah pasien. Ini berkaitan dengan bagaimana hormon progesteron berinteraksi dengan sel-sel kanker.

   Progesteron adalah hormon seks yang diproduksi oleh ovarium pada wanita dan dalam jumlah kecil oleh testis pada pria. Hormon ini memainkan peran penting dalam siklus menstruasi dan kehamilan pada wanita.

7. `er (estrogen receptors, fmol/l)`

   Menunjukkan kadar reseptor estrogen dalam darah pasien. Ini berkaitan dengan bagaimana hormon estrogen berinteraksi dengan sel-sel kanker.

   Estrogen adalah kelompok hormon steroid yang berperan penting dalam perkembangan dan fungsi sistem reproduksi pada wanita. Terdapat beberapa jenis estrogen yang diproduksi di dalam tubuh manusia, tetapi yang paling umum adalah estradiol, estron, dan estriol.

   Hormon steroid adalah jenis hormon yang berasal dari lipid (lemak) dan memiliki struktur kimia yang mirip dengan steroid. Hormon steroid memainkan peran penting dalam berbagai fungsi tubuh, termasuk regulasi metabolisme, perkembangan seksual, dan respons terhadap stres.

8. `Hormon (hormonal therapy)`

Ini mencatat apakah pasien menjalani terapi hormonal atau tidak. Nilai 0 menunjukkan tidak, sementara nilai 1 menunjukkan ya.

9. `rfstime (recurrence free survival time; days to first of recurrence, death or last follow-up)`

   Ini mencatat waktu bertahan hidup bebas kekambuhan. Ini dapat diukur dalam jumlah hari dari saat rekurensi pertama, kematian, atau kunjungan tindak lanjut terakhir.

2

— DATA UNDERSTANDING —

Adapun hal - hal yang perlu dilakukan untuk memahami dataset ini, yakni 1. Mendeskripsikan setiap fitur pada data * tipe data * deskripsi data 2. Mengidentifikasi missing values setiap fitur atau kolom 3. Eksplorasi data (grafikan fitur) 4. Mengidentifikasi outlier 5. Mengidentifikasi jumlah data (proporsi data perkelas -untuk mengetahui balancing dataset atau keseimbangan data per kelas)

2.0.1 Load Dataset

```
import pandas as pd

data = pd.read_csv('dataset.csv')
data.head(10)
```

	age	meno	size	grade	nodes	pgr	er	hormon	rfstime	status
0	49	0	18	2	2	0	0	0	1838	0
1	55	1	20	3	16	0	0	0	403	1
2	56	1	40	3	3	0	0	0	1603	0
3	45	0	25	3	1	0	4	0	177	0
4	65	1	30	2	5	0	36	1	1855	0
5	48	0	52	2	11	0	0	0	842	1
6	48	0	21	3	8	0	0	0	293	1
7	37	0	20	2	9	0	0	1	42	0
8	67	1	20	2	1	0	0	1	564	1
9	45	0	30	2	1	0	0	0	1093	1

```
# Rincian dataset (banyak data dan kolom)

print("Banyaknya data : ", data.shape[0])
print("Banyaknya kolom : ", data.shape[1])
```

```
Banyaknya data :  686
```

```
Banyaknya kolom :  10
```

2.0.2 Mendeskripsikan setiap fitur

```
data.columns
```

```
Index(['age', 'meno', 'size', 'grade', 'nodes', 'pgr', 'er', 'hormon',
       'rfstime', 'status'],
      dtype='object')
```

2.0.3 Mengidentifikasi missing value

2.0.3.1 *Missing Value*

Missing value atau nilai yang hilang merujuk pada keadaan di mana suatu data atau observasi tidak memiliki nilai atau informasi untuk suatu atribut atau fitur tertentu. Nilai yang hilang sering kali direpresentasikan dengan simbol tertentu, seperti `NaN` (Not a Number) dalam beberapa lingkungan pemrograman.

Berikut penyebab adanya *missing value*: 1. Ketidakhadiran Data: - Observasi atau catatan tidak terdokumentasi atau tidak dicatat.

2. Kesalahan Pengukuran:
   - Kesalahan atau ketidakpastian dalam pengukuran atau pencatatan data.
3. Ketidaklengkapan Pengisian Formulir:
   - Formulir atau entri data tidak diisi secara lengkap oleh individu atau responden.
4. Proses Pengumpulan Data yang Kompleks:
   - Proses pengumpulan data yang melibatkan banyak tahap atau pihak dapat menyebabkan kehilangan data.

Dampak *missing value*: 1. Analisis Bias: - Missing value dapat menyebabkan bias dalam hasil analisis statistik.

2. Performa Model Machine Learning:
   - Model machine learning dapat memberikan hasil yang tidak akurat atau bias jika data training mengandung missing value.
3. Kesalahan Pengambilan Keputusan:
   - Keputusan yang diambil berdasarkan data yang tidak lengkap dapat menjadi tidak akurat atau tidak dapat diandalkan.

Penanganan *missing value*: 1. Menghapus Data: - Jika atribut tersebut memiliki banyak missing value, maka atribut tersebut perlu dihapus dari dataset.

Namun jika hanya terdapat beberapa data yang missing value bisa dilakukan drop dari baris yang memiliki missing value atau mengisinya dengan rata - rata nilai pada atribut yang bersangkutan.

2. Imputasi (Pengisian Nilai):
    - Menggantikan missing value dengan nilai yang dihitung atau ditentukan berdasarkan suatu metode, seperti rata-rata atau median.
3. Pemodelan Prediktif:
    - Menggunakan model prediktif untuk memprediksi nilai yang hilang berdasarkan pola data yang ada.
4. Penggunaan Default Value:
    - Menggantikan missing value dengan nilai default yang sudah ditentukan.

Contoh adanya missing value :

Nama

Usia

Pendapatan

John

28

50000

Jane

35

NaN

Bob

NaN

75000

NaN

45

NaN

Alice

30

60000

Dari tabel di atas menunjukkan dataset dengan missing value pada fitur ‘Usia’.

Karena fitur ini memiliki banyak nilai yang hilang (represented by `NaN`), kita perlu mengatasi keberadaan nilai yang hilang ini.

```
# Menghitung apakah ada nilai yang hilang dalam setiap kolom
missing_values = data.isna().any()

# Menampilkan hasil
print("Apakah ada nilai yang hilang dalam setiap kolom:")
print(missing_values)
```

```
Apakah ada nilai yang hilang dalam setiap kolom:
age        False
meno       False
size       False
grade      False
nodes      False
pgr        False
er         False
hormon     False
rfstime    False
status     False
dtype: bool
```

Noted : tidak ada *missing value* pada data

2.0.3.2 *Duplikat Data*

Duplikat data terjadi ketika satu baris dalam dataset memiliki nilai yang identik di semua kolomnya dengan baris lainnya. Keberadaan data yang redundan, atau berulang, dapat mengakibatkan gangguan dalam hasil analisis dan menghasilkan akurasi yang tidak akurat. Dalam konteks analisis data, penting untuk mengidentifikasi dan menangani duplikat agar hasil analisis mencerminkan keadaan sebenarnya dari dataset, dan untuk memastikan bahwa hasil yang dihasilkan adalah akurat dan dapat diandalkan.

Contoh adanya duplikat data :

Nama

Usia

Pendapatan

John

28

50000

Jane

35

60000

Bob

30

75000

Jane

35

60000

Alice

30

80000

Dalam contoh di atas, baris dengan nama ‘Jane’ muncul dua kali dan dianggap duplikat.

```
jumlah_duplikat = data.duplicated().sum()

# Menampilkan jumlah data yang duplikat
print("Jumlah data yang duplikat:", jumlah_duplikat)
```

```
Jumlah data yang duplikat: 0
```

Noted : tidak ada duplikat data

2.0.4 Mengidentifikasi Outlier

Outlier, dalam konteks data, dapat dijelaskan sebagai nilai yang jauh berbeda atau sangat ekstrem dibandingkan dengan sebagian besar nilai lain dalam dataset. Kita dapat membayangkan outlier sebagai data yang “aneh” atau “tidak biasa” yang muncul di tengah nilai-nilai yang lebih umum atau tipikal.

Pertimbangkan sebuah contoh sederhana dengan data pengukuran tinggi badan dalam suatu kelas. Mayoritas siswa memiliki tinggi badan yang mirip, tetapi ada satu siswa yang memiliki tinggi badan yang sangat tinggi atau rendah secara tidak wajar dibandingkan dengan siswa lainnya. Siswa ini dapat dianggap sebagai outlier karena tinggi badannya sangat tidak umum dalam konteks kelas tersebut.

Dengan kata lain, outlier adalah data yang berbeda secara signifikan dari mayoritas data lainnya, dan keberadaannya dapat mempengaruhi hasil analisis atau model yang dibangun dari dataset tersebut. Identifikasi dan penan-

ganan outlier penting untuk memastikan hasil analisis data lebih akurat dan representatif.

Contoh adanya outlier :

Nama

Pendapatan

John

5000

Jane

5200

Bob

5100

Alice

5500

Emily

5300

chris

20000

Pada tabel di atas, gaji Chris sangat jauh lebih tinggi dibandingkan dengan gaji karyawan lainnya. Gaji Chris dapat dianggap sebagai outlier karena nilai tersebut signifikan berbeda dari gaji mayoritas karyawan. Outlier seperti ini dapat mempengaruhi analisis statistik dan model prediksi, sehingga penting untuk mengidentifikasi dan memahami penyebabnya. Dalam konteks ini, gaji Chris mungkin merupakan kesalahan pengukuran atau mungkin ada faktor khusus yang menyebabkan gaji tersebut begitu tinggi.

```
from sklearn.neighbors import LocalOutlierFactor

# Menggunakan Local Outlier Factor
lof = LocalOutlierFactor(n_neighbors=5)
outlier_scores = lof.fit_predict(data)

outliers = data[outlier_scores == -1]
print("Banyaknya outlier : ", outliers.shape[0])

data_bersih = data[outlier_scores != -1]
print("Sisa data : ", data_bersih.shape[0])
```

```
Banyaknya outlier :  40
Sisa data :  646
```

Noted : data memiliki 40 outlier. Hanya terdapat 6% outlier sehingga tidak perlu ditangani.

2.0.5 Mengidentifikasi Jumlah Data

Dengan mengetahui proporsi data untuk masing - masing label, kita bisa mengetahui seberapa berbeda jumlah data di tiap - tiap label. Jika jumlah data antar label memiliki perbedaan yang sangat jauh maka akan mempengaruhi akurasi serta hasil klasifikasi sehingga nantinya perlu dilakukan penyeimbangan jumlah data di tiap labelnya.

```
dataa = data_bersih['status'].value_counts()

print("Jumlah data pada tiap target :")
print(dataa)
```

```
Jumlah data pada tiap target :
0    366
1    280
Name: status, dtype: int64
```

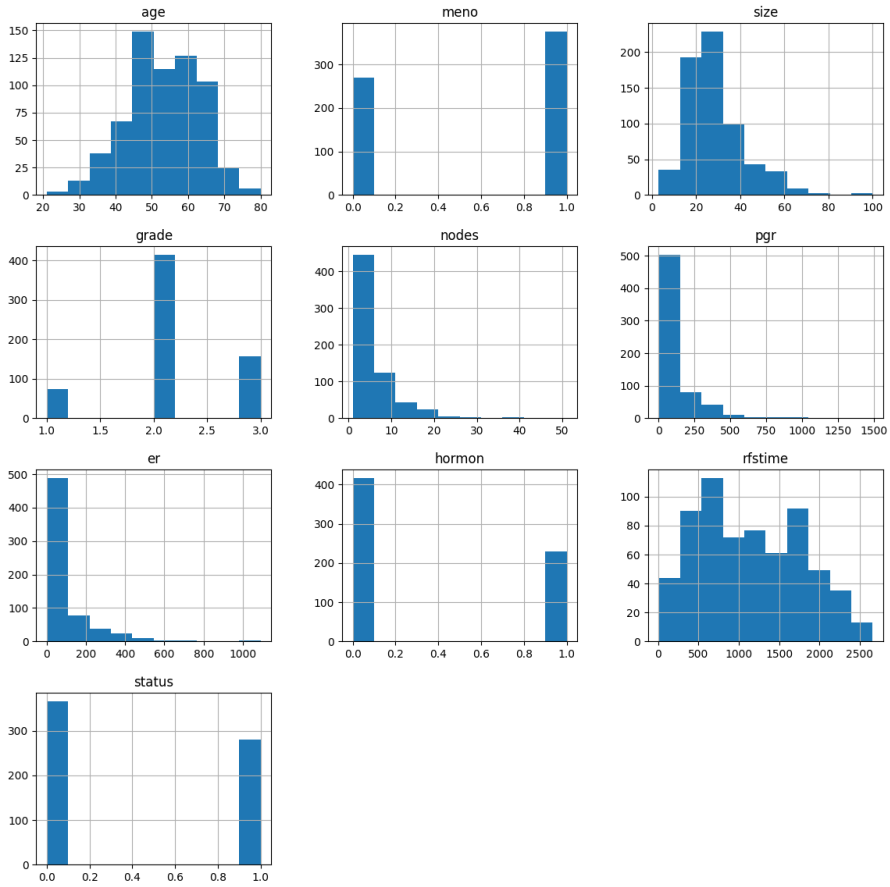
2.0.6 Eksplorasi Data

2.0.6.1 Visualisasi data beserta valuenya dan banyak datanya

Visualisasi data dilakukan untuk memudahkan kita memahami data. Melalui visualisasi data pula kita akan memperoleh informasi sebaran nilai dari dataset ini

```
import matplotlib.pyplot as plt

data_bersih.hist(figsize=(14,14))
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



2.0.6.2 Fitur beserta presentase kepentingannya

Dengan melakukan skoring fitur, kita akan mengetahui yang mana fitur yang penting dan yang tidak. Hal ini dikarenakan tidak semua fitur dapat dijadikan ciri untuk melakukan klasifikasi. Dengan menentukan beberapa ciri terbaik akan menghasilkan akurasi yang sama atau lebih baik dibandingkan dengan menggunakan semua ciri serta menghemat waktu komputasi.

2.0.6.2.1 Hasil skoring fitur dari tiap kolom / atribut

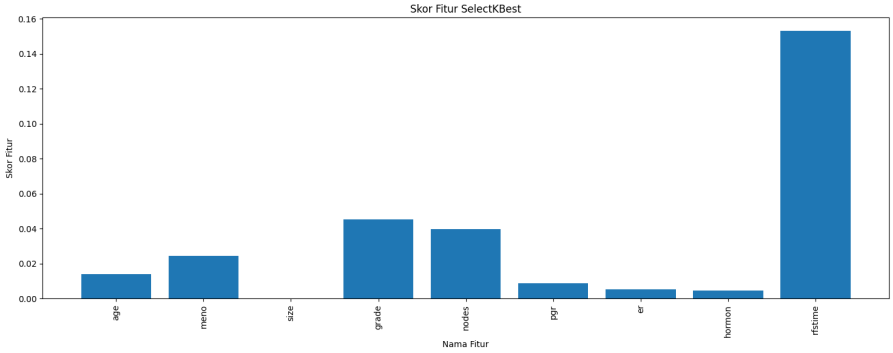
```
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.model_selection import train_test_split

# memisahkan kolom fitur dan target
```

```
fitur = data_bersih.drop(columns=['status'], axis =1)
target = data_bersih['status']

# Buat objek SelectKBest dengan mutual_info_classif sebagai fungsi skor
k_best = SelectKBest(score_func=mutual_info_classif, k='all')  # 'all' berarti akan mempertahan

# Hitung skor fitur
k_best.fit(fitur, target)
scores = k_best.scores_

# Dapatkan nama fitur dari kolom data Anda
fitur_names = fitur.columns

# Tampilkan skor fitur berserta namanya
for i, (score, fitur_name) in enumerate(zip(scores, fitur_names)):
    print(f"Fitur {i}: {fitur_name}, Skor: {score}")
```

```
Fitur 0: age, Skor: 0.013967268086709561
Fitur 1: meno, Skor: 0.02454763859169362
Fitur 2: size, Skor: 0.0
Fitur 3: grade, Skor: 0.045252411046041274
Fitur 4: nodes, Skor: 0.03969904661079671
Fitur 5: pgr, Skor: 0.00892882145382412
Fitur 6: er, Skor: 0.005163194801403703
Fitur 7: hormon, Skor: 0.004574344114665951
Fitur 8: rfstime, Skor: 0.15307373217244025
```

2.0.6.2.2 Grafik fitur dan nilai kepentingannya

```
import matplotlib.pyplot as plt

# Tampilkan skor fitur dalam grafik
plt.figure(figsize=(18, 6))
plt.bar(fitur_names, scores)
plt.xlabel("Nama Fitur")
plt.ylabel("Skor Fitur")
plt.title("Skor Fitur SelectKBest")
plt.xticks(rotation=90)
plt.show()
```



2.0.7 Hasil Analisa Pada Data Understanding :

1. Data tidak memiliki *missing values*
2. Tidak memiliki rerudan data
3. Data memiliki 40 outlier
4. Perbedaan jumlah atau proporsi data antar label sangat jauh
5. Hasil skoring fitur masih menggunakan data kotor sehingga perlu difilter kembali

3

— DATA PREPROCESSING —

Setelah memahami data, akan dilakukan tahap preprocessing untuk menangani masalah pada data yang sudah didefinisikan pada data understanding, yakni.

1. Menyeimbangkan proporsi data tiap target

Setelah data siap, akan dilakukan : 1. Skoring tiap fitur kembali 2. Normalisasi Data 3. Skenario Percobaan Menggunakan 2 Model yakni Naive Bayes, Support Vector Machine, dan Decision Tree

3.0.1 Load Dataset

```
import pandas as pd

data = pd.read_csv('dataset.csv')
data.head(5)
```

	age	meno	size	grade	nodes	pgr	er	hormon	rfstime	status
0	49	0	18	2	2	0	0	0	1838	0
1	55	1	20	3	16	0	0	0	403	1
2	56	1	40	3	3	0	0	0	1603	0
3	45	0	25	3	1	0	4	0	177	0
4	65	1	30	2	5	0	36	1	1855	0

```
# Rincian dataset (banyak data dan kolom)

print("Banyaknya data : ", data.shape[0])
print("Banyaknya kolom : ", data.shape[1])
```

```
Banyaknya data :  686
Banyaknya kolom :  10
```

3.0.2 Menyeimbangkan Data Tiap Target

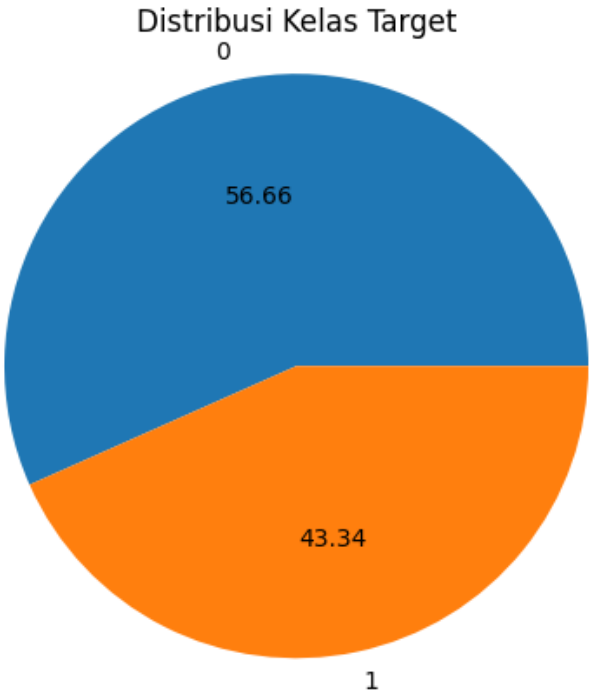
3.0.2.1 Banyaknya data di tiap label

```
fitur = data_bersih.drop(columns=['status'])
target = data_bersih['status']

target.value_counts()
```

```
0    366
1    280
Name: status, dtype: int64
```

3.0.2.1.1 Visualisasi banyaknya data di tiap label

```
import matplotlib.pyplot as plt

value_counts = target.value_counts()

plt.pie(value_counts, labels=value_counts.index, autopct='%.2f')
plt.title('Distribusi Kelas Target')
plt.axis('equal')
plt.show()
```



3.0.2.2 Penyeimbangan jumlah atau proporsi data

Perbandingan proporsi data tiap target sangat jauh sehingga untuk menghemat waktu komputasi output antar target akan diseimbangkan menggunakan metode UnderSampling. Undersampling adalah teknik untuk mengurangi jumlah data pada kelas mayoritas sehingga jumlahnya setara dengan kelas minoritas.

```
from imblearn.under_sampling import RandomUnderSampler

smote = RandomUnderSampler()
fitur_seimbang, target_seimbang = smote.fit_resample(fitur, target)

print("Jumlah sampel setelah diseimbangkan : ")
print(target_seimbang.value_counts())
```

```
Jumlah sampel setelah diseimbangkan :
0    280
1    280
Name: status, dtype: int64
```

simpan data yang telah siap diproses pada file csv baru

```
import pandas as pd

# Membuat DataFrame dari fitur dan target yang telah seimbang
data_seimbang = pd.concat([fitur_seimbang, target_seimbang], axis=1)

# Menyimpan DataFrame ke dalam file CSV
data_seimbang.to_csv('data_seimbang.csv', index=False)
```

```
fitur = data_seimbang.drop(columns=['status'])
target = data_seimbang['status']

print("Banyaknya data : ", fitur.shape[0])
print("Banyaknya fitur : ", fitur.shape[1])
```

```
Banyaknya data :  560
Banyaknya fitur :  9
```

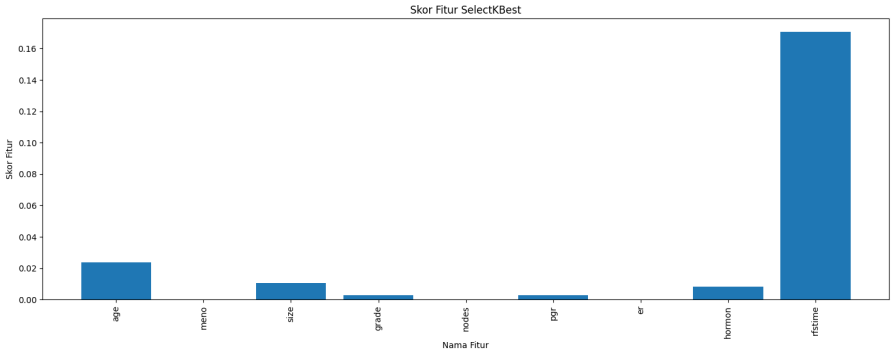
3.0.3 Eksplorasi Data (Skoring Fitur)

```
from sklearn.feature_selection import SelectKBest, mutual_info_classif

# Buat objek SelectKBest dengan mutual_info_classif sebagai fungsi skor
k_best = SelectKBest(score_func=mutual_info_classif, k='all')  # 'all' berarti akan mempertahan

# Hitung skor fitur
k_best.fit(fitur, target)
scores = k_best.scores_

# Dapatkan nama fitur dari kolom data Anda
fitur_names = fitur.columns

# Tampilkan skor fitur berserta namanya
for i, (score, fitur_name) in enumerate(zip(scores, fitur_names)):
    print(f"Fitur {i}: {fitur_name}, Skor: {score}")
```

```
Fitur 0: age, Skor: 0.023769722079653555
Fitur 1: meno, Skor: 0.0
Fitur 2: size, Skor: 0.010455660149520263
Fitur 3: grade, Skor: 0.002971926309707662
Fitur 4: nodes, Skor: 0.0
Fitur 5: pgr, Skor: 0.002634234084921916
Fitur 6: er, Skor: 0.0
Fitur 7: hormon, Skor: 0.008107100970603964
```

```
Fitur 8: rfstime, Skor: 0.17050292516468168
```

```python
import matplotlib.pyplot as plt

# Tampilkan skor fitur dalam grafik
plt.figure(figsize=(18, 6))
plt.bar(fitur_names, scores)
plt.xlabel("Nama Fitur")
plt.ylabel("Skor Fitur")
plt.title("Skor Fitur SelectKBest")
plt.xticks(rotation=90)
plt.show()
```



4

— MODELLING —

Note : Model Naive Bayes dipilih karena memiliki tingkat akurasi yang lebih baik dibandingkan model lainnya. Selengkapnya disini[1].

4.0.1 Konsep Naïve Bayes

Naive Bayes, atau kadang disebut Naïve Bayes Classifier, adalah algoritma machine learning probabilistik yang digunakan dalam berbagai macam tugas klasifikasi.

berikut rumus umum Teorema Bayes yang menjadi dasar dari Naive Bayes berikut:

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

`Keterangan :`

- P(A|B) adalah kondisional bahwa sampel termasuk dalam suatu kategori tertentu mengingat fitur-fiturnya.
- P(B|A) adalah probabilitas kondisional bahwa sampel memiliki fitur-fitur tertentu jika diketahui termasuk dalam kategori tertentu.
- P(A) adalah probabilitas prior bahwa sampel termasuk dalam suatu kategori tertentu tanpa mempertimbangkan fitur-fiturnya.
- P(B) adalah probabilitas bahwa sampel memiliki fitur-fitur tertentu, independen dari kategori.

Tujuan Utama Naive Bayes:

Tujuan utama Naive Bayes adalah melakukan klasifikasi, yaitu memprediksi kelas dari suatu instance atau data berdasarkan fitur-fiturnya. Model Naive Bayes digunakan untuk mengklasifikasikan data ke dalam salah satu kelas yang telah dipelajari selama fase pelatihan.

Kelebihan Naive Bayes:

- Cepat dan efisien, cocok untuk dataset besar.
- Bekerja baik dengan fitur-fitur yang banyak.

[1] https://colab.research.google.com/drive/15W2MCpFu_CnwQwqR4zcxFmwFZwoG4Kop?usp=sharing

Kelemahan Naive Bayes:

- Asumsi independensi dapat menjadi tidak realistis untuk beberapa dataset.
- Sensitif terhadap fitur yang tidak relevan.
- Tidak menangani baik fitur-fitur numerik kontinu.

4.0.2 Load Dataset

```
import pandas as pd

dataset = pd.read_csv('dataset_baru.csv')
dataset.head(5)
```

	age	size	grade	nodes	pgr	hormon	rfstime	status
0	49	18	2	2	0	0	1838	0
1	55	20	3	16	0	0	403	1
2	56	40	3	3	0	0	1603	0
3	45	25	3	1	0	0	177	0
4	65	30	2	5	0	1	1855	0

```
print("Banyaknya data : ", dataset.shape[0])
print("Banyaknya kolom : ", dataset.shape[1])
```

```
Banyaknya data :  686
Banyaknya kolom :  8
```

4.0.3 Split Dataset

```
from sklearn.model_selection import train_test_split

# memisahkan kolom fitur dan target
fitur = dataset.drop(columns=['status'], axis =1)
target = dataset['status']

# melakukan pembagian dataset, dataset dibagi menjadi 80% data training dan 20% data testing
fitur_train, fitur_test, target_train, target_test = train_test_split(fitur, target, test_size =
```

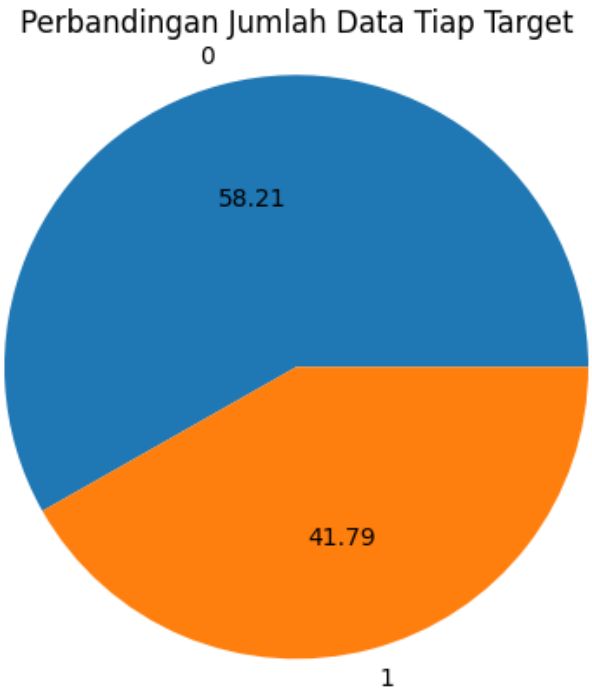
```
print("Banyaknya fitur atau ciri yang digunakan : ", fitur_train.shape[1])
print("Banyaknya data latih : ", fitur_train.shape[0])
print("Banyaknya data testing : ", fitur_test.shape[0])
```

```
Banyaknya fitur atau ciri yang digunakan :  7
Banyaknya data latih :  548
Banyaknya data testing :  138
```

```
target_train.value_counts()
```

```
0    319
1    229
Name: status, dtype: int64
```

```
import matplotlib.pyplot as plt

value_counts = target_train.value_counts()

plt.pie(value_counts, labels=value_counts.index, autopct='%.2f')
plt.title('Perbandingan Jumlah Data Tiap Target')
plt.axis('equal')
plt.show()
```



4.0.4 Normalisasi Data

Z-Score Scaler, atau disebut juga Z-Score Normalizer, adalah alat dalam pengolahan data yang digunakan untuk mengubah distribusi nilai-nilai dalam suatu dataset sehingga memiliki rata-rata nol (0) dan deviasi standar satu (1). Ini adalah teknik normalisasi yang berguna untuk menyamakan skala dari berbagai fitur atau variabel dalam dataset.

Cara Kerja Z-Score Scaler:

1. Hitung Rata-Rata( ):

   Hitung nilai rata-rata dari seluruh dataset atau dari setiap fitur yang akan di-normalisasi.

   Gunakan rumus:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

2. Hitung Deviasi Standar ( ):

   Hitung nilai deviasi standar dari seluruh dataset atau dari setiap fitur yang akan di-normalisasi.

   Gunakan rumus:

$$S = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}}$$

   Keterangan :

- Xi adalah setiap nilai dalam dataset.
- adalah rata-rata dari dataset.

- N adalah jumlah total nilai dalam dataset.

3. Hitung Z-Score untuk Setiap Nilai (Z):

   Gunakan rumus:

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

   Keterangan :

- Z adalah Z-score.

- X adalah nilai individual .
- adalah rata-rata (mean) dari fitur.
- adalah deviasi standar dari fitur.

untuk menghitung Z-Score untuk setiap nilai (X) dalam dataset atau setiap fitur.

4. Hasilnya:

   Setelah proses normalisasi, nilai-nilai dalam dataset atau setiap fitur akan memiliki distribusi dengan rata-rata ( ) yang sama (0) dan deviasi standar ( ) yang sama (1).

Contoh sederhana untuk penggunaan Z-score Scaler. Anggap kita memiliki dataset usia pasien sebagai berikut:

Usia = [25, 30, 35, 40, 45]

Langkah-langkah Z-core Scaling:

1. Hitung Rata-Rata( ):

$$\mu = \frac{25 + 30 + 35 + 40 + 45}{5} = \frac{175}{5} = 35$$

2. Hitung Deviasi Standar ( ):

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$$

$$S = \sqrt{\frac{(25-35)^2 + (30-35)^2 + (35-35)^2 + (40-35)^2 + (45-35)^2}{5}}$$

$$\mu = \sqrt{\frac{100 + 25 + 0 + 25 + 100}{5}} = \sqrt{\frac{250}{5}} = \sqrt{50} \approx 7.07$$

3. Hitung Z-Score untuk Setiap Nilai (Z):

$$Zi = \frac{Xi - \mu}{\sigma}$$

$$Z1 = \frac{24 - 35}{7.07} \approx -1.41$$

$$Z2 = \frac{30 - 35}{7.07} \approx -0.71$$

$$Z3 = \frac{35 - 35}{7.07} \approx 0$$

$$Z4 = \frac{40 - 35}{7.07} \approx 0.71$$

$$Z5 = \frac{45 - 35}{7.07} \approx 1.41$$

4. Hasilnya:

   Dataset yang telah di Z-score Scaling:

   Z= [-1.41, -0.71, 0, 0.71, 1.41]

```
import pickle
from sklearn.preprocessing import StandardScaler

# menentukan lokasi file pickle akan disimpan
path = 'zscore_scaler.pkl'

# membuat dan melatih objek StandardScaler
zscore_scaler = StandardScaler()
zscore_scaler.fit(fitur_train)

# menyimpan model ke dalam file pickle
with open(path, 'wb') as file:
    pickle.dump(zscore_scaler, file)

# memanggil kembali model normalisasi zscore dari file pickle
with open(path, 'rb') as file:
    zscore_scaler = pickle.load(file)

# menerapkan normalisasi zscore pada data training
zscore_training = zscore_scaler.transform(fitur_train)

# menerapkan normalisasi zscore pada data testing
zscore_testing = zscore_scaler.transform(fitur_test)
```

4.0.5 Membuat model

```
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import GaussianNB
```

```
# Inisialisasi model Naive Bayes
naive_bayes = GaussianNB()

# Buat objek Grid Search dengan model Naive Bayes (Gaussian) dan parameter grid
param_grid = {}  # Tidak ada parameter yang perlu di-tune untuk Gaussian Naive Bayes
grid_search = GridSearchCV(estimator=naive_bayes, param_grid=param_grid, cv=5, n_jobs=-1, verbo

# Lakukan grid search pada data latih
grid_search.fit(zscore_training, target_train)

# Melihat parameter terbaik yang ditemukan (Meskipun pada Naive Bayes tidak ada parameter yang
print("Parameter terbaik:", grid_search.best_params_)

# Evaluasi model terbaik pada data uji
best_model = grid_search.best_estimator_
accuracy = best_model.score(zscore_testing, target_test)
print("Akurasi model terbaik:", accuracy)
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
Parameter terbaik: {}
Akurasi model terbaik: 0.717391304347826
```

4.0.6 Simpan Model

```
import pickle

# Simpan model terbaik ke dalam file pickle
with open('model_nb.pkl', 'wb') as file:
    pickle.dump(best_model, file)
```

```
best_model
```

```
GaussianNB()
```

5

— *EVALUATION* —

Tahap evaluasi merupakan tahap penilaian atas kebenaran data. Melalui tahap ini kita dapat mengetahui seberapa jauh suatu data dapat dipercaya.

```
with open('model_nb.pkl', 'rb') as file:
    model_nb = pickle.load(file)

from sklearn.metrics import accuracy_score

model_nb.fit(zscore_training, target_train)
y_pred_nb = model_nb.predict(zscore_testing)
```

5.0.1 Akurasi

```
akurasi_nb = accuracy_score(target_test, y_pred_nb)
print('Akurasi Menggunakan Naive Bayes : ', akurasi_nb)
```

```
Akurasi Menggunakan Naive Bayes :  0.717391304347826
```

5.0.2 Presisi dan Recall

- **Presisi** mengukur sejauh mana hasil positif yang diprediksi oleh model adalah benar
- **Recall** mengukur sejauh mana model dapat mengidentifikasi dengan benar semua instance positif dalam data.

```
from sklearn.metrics import precision_score, recall_score

# Hitung presisi
precision = precision_score(target_test, y_pred_nb, average='macro')

# Hitung recall
recall = recall_score(target_test, y_pred_nb, average='macro')
```

```
print("Presisi :", precision)
print("Recall :", recall)
```

```
Presisi : 0.725471085120208
Recall : 0.7186974789915966
```

5.0.3 F-1 Score

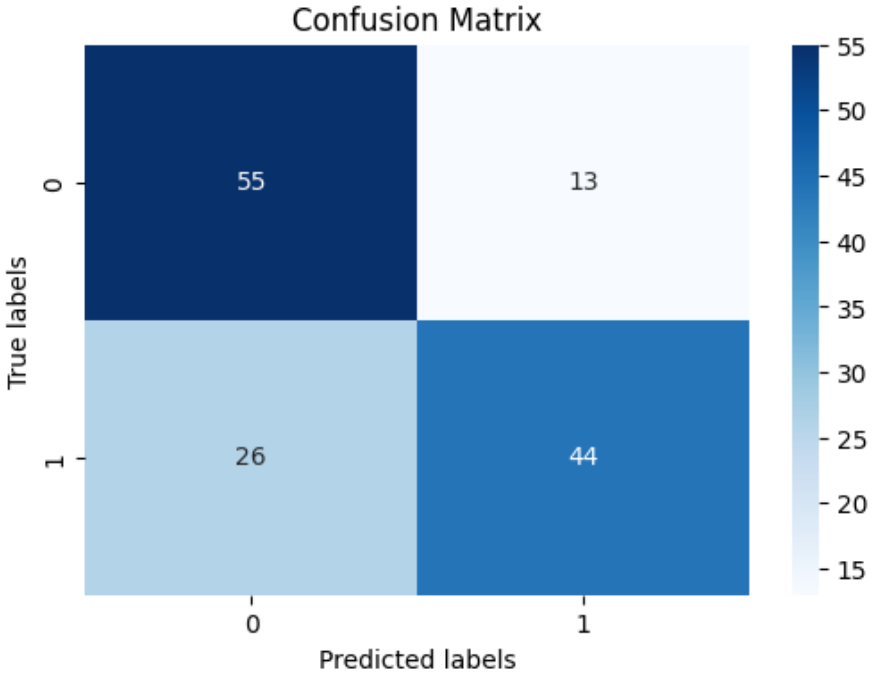
F1-Score adalah metrik gabungan yang mempertimbangkan presisi dan recall.

```
from sklearn.metrics import f1_score

# Hitung f1-score
f1_score = f1_score(target_test, y_pred_nb, average='macro')

print("F1 - Score :", precision)
```

```
F1 - Score : 0.725471085120208
```

5.0.4 Confussion Matrix

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns

conf_matrix = confusion_matrix(target_test, y_pred_nb)

plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```



Dari matrik di atas, diperoleh informasi sebagai berikut. * Orang yang hidup tanpa kambuh ada 55 data * Orang yang kambuh atau meninggal ada 44 data

6

— *DEPLOYMENT* —

code dilanjut pada file main.py untuk membangun sistem

7

Summary

In summary, this book has no content whatsoever.

References