Jane Doe and Max Power

Quarto CRC Book

To blah, blah, and blah.

Table of contents

References **39**

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 : Yulia Nanda Ivne Fara

NIM : 210411100158

Kelas : B

1.1 BUSSINESS UNDERSTANDING

1.1.1 Identifikasi kasus :

Glioma merupakan penyakit tumor yang berasal dari pertumbuhan abnormal pada sel-sel glial, yaitu sel-sel yang mendukung dan melindungi sel-sel saraf di dalam otak dan sumsum tulang belakang,dapat dibagi menjadi 2 kategori utama yaitu Lower-Grade Gliomas (LGG) adalah jenis glioma yang memiliki pertumbuhan lebih lambat dan umumnya dianggap sebagai tumor yang kurang agresif dibandingkan dengan Glioblastoma Multiforme (GBM) merupakan glioma yang sangat agresif dan tumbuh dengan cepat.

1.1.2 Tujuan Proyek :

Tujuan yaitu melakukan klasifikasi pasien menjadi dua kelompok berdasarkan ciri klinis dan molekuler/mutasi tertentu, yaitu apakah pasien tersebut memiliki Lower-Grade Glioma (LGG) atau Glioblastoma Multiforme (GBM). Ini merupakan tugas klasifikasi ingin mengidentifikasi subset gen mutasi dan gambaran klinis yang optimal untuk membantu memprediksi apakah seorang pasien memiliki LGG atau GBM

1.2 DATA UNDERSTANDING

Tahap Data Understanding merupakan tahap dimana kita perlu memahami data yang akan diolah. Adapun hal - hal yang perlu dilakukan nantinya untuk memahami dataset ini, yakni 1. Tentang data, mencakup : * Pengumpulan dataset * Pengenalan singkat mengenai data yang akan diolah 2. Mendeskripsikan data, mencakup : * analisa tipe data * deskripsi fitur 3. Eksplorasi data, mencakup : * Visualisasi data * Skoring fitur 4. Identifikasi kualitas data : * Identifikasi missing valye setiap fitur atau kolom * Identifikasi data duplikat * Identifikasi outlier (data aneh) * Identifikasi jumlah data (proporsi data perkelas -untuk mengetahui balancing dataset atau keseimbangan data per kelas)

1.2.1 About Data

1.2.1.1 Pengumpulan Dataset

Pada tanggal 13 Desember 2022, data fitur klinis dan mutasi glioma dievaluasi, yang diperoleh melalui platform Kaggle, menjadi sumber informasi yang relevan

```
import pandas as pd

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

	Grade	Gender	Age_at_diagnosis	Race	IDH1	TP53	ATRX	PTEN	EGFR	CIC	...	FUBP
0	0	0	51.30	0	1	0	0	0	0	0	...	1
1	0	0	38.72	0	1	0	0	0	0	1	...	0
2	0	0	35.17	0	1	1	1	0	0	0	...	0
3	0	1	32.78	0	1	1	1	0	0	0	...	0
4	0	0	31.51	0	1	1	1	0	0	0	...	0

1.2.1.2 Seputar dataset

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

```
Banyaknya data :  839
```

1.2.1.3 Mendeskripsikan Fitur

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

```
Banyaknya kolom :  24
```

Penjelasan fitur

1. Grade (Tingkat Keparahan): Menunjukkan sejauh mana tumor otak (glioma) tumbuh dan seberapa ganas tumor tersebut, berdasarkan pada sistem penilaian tertentu. Skala gradasi berkisar dari I (tumor jinak) hingga IV (tumor ganas).
2. Gender (Jenis Kelamin): Informasi tentang jenis kelamin pasien yang menderita glioma. Berguna untuk analisis epidemiologi dan pemahaman apakah terdapat perbedaan insidensi glioma antara laki-laki dan perempuan.
3. (Usia pada Saat Diagnosa): Usia pasien saat glioma pertama kali didiagnosis. Penting karena dapat memengaruhi pilihan pengobatan dan prognosis glioma, mengingat glioma dapat muncul pada berbagai kelompok usia.
4. Race (Ras): Informasi tentang ras pasien, membantu analisis untuk memahami apakah ada perbedaan kejadian glioma di antara kelompok etnis.
5. IDH1 (Isocitrate Dehydrogenase 1): Menunjukkan status mutasi gen IDH1, yang sering terlibat dalam perkembangan glioma dan dapat memengaruhi karakteristik dan prognosis glioma.
6. TP53 (Tumor Protein p53): Informasi tentang status gen TP53 dalam glioma, yang berperan sebagai gen penekan tumor dan mutasinya dapat memengaruhi pertumbuhan dan perkembangan glioma.
7. ATRX (Alpha Thalassemia/Mental Retardation Syndrome X-Linked): Menyajikan status gen ATRX, terkait dengan glioma, dan mutasinya dapat memengaruhi sifat molekuler glioma.
8. PTEN (Phosphatase and Tensin Homolog): Menunjukkan status gen PTEN yang berperan dalam perkembangan glioma dan mutasinya dapat memengaruhi pertumbuhan tumor otak.
9. EGFR (Epidermal Growth Factor Receptor): Informasi tentang gen EGFR dalam konteks glioma. Amplifikasi atau mutasi gen EGFR dapat ditemukan dalam beberapa jenis glioma dan memengaruhi respons terhadap pengobatan.
10. CIC (Capicua Transcriptional Repressor): Gen yang terlibat dalam regulasi pertumbuhan sel dan perkembangan glioma.
11. MUC16 (Mucin 16): Merujuk kepada gen atau molekul tertentu terkait dengan glioma, memerlukan informasi lebih lanjut untuk penjelasan yang lebih spesifik.

12. PIK3CA (Phosphatidylinositol-4,5-Bisphosphate 3-Kinase Catalytic Subunit Alpha): Menunjukkan informasi tentang gen PIK3CA yang terkait dengan glioma.
13. NF1 (Neurofibromatosis Type 1): Informasi tentang gen NF1 dalam konteks glioma.
14. PIK3R1 (Phosphoinositide-3-Kinase Regulatory Subunit 1): Menunjukkan informasi tentang gen PIK3R1.
15. FUBP1 (Far Upstream Element Binding Protein 1): Gen yang terkait dengan regulasi transkripsi gen dalam konteks glioma.
16. RB1 (Retinoblastoma 1): Menyajikan informasi tentang gen RB1, gen penekan tumor yang memengaruhi siklus sel dalam konteks glioma.
17. NOTCH1 (Notch Receptor 1): Informasi tentang gen NOTCH1 dalam glioma, yang terlibat dalam regulasi pertumbuhan sel dan pembentukan jaringan.
18. BCOR (BCL6 Corepressor): Merujuk kepada gen BCOR terkait dengan beberapa jenis kanker.
19. CSMD3 (CUB and Sushi Multiple Domains 3): Merujuk kepada gen atau molekul terkait dengan glioma.
20. SMARCA4 (SWI/SNF Related, Matrix Associated, Actin Dependent Regulator Of Chromatin, Subfamily A, Member 4): Menunjukkan informasi tentang gen SMARCA4 dalam keluarga gen regulasi kromatin.
21. GRIN2A (Glutamate Ionotropic Receptor NMDA Type Subunit 2A): Merujuk kepada gen GRIN2A terkait dengan glioma.
22. IDH2 (Isocitrate Dehydrogenase 2): Menunjukkan informasi tentang gen IDH2 yang terlibat dalam glioma dan sering dihubungkan dengan IDH1.
23. FAT4 (FAT Atypical Cadherin 4): Merujuk kepada gen FAT4 terkait dengan beberapa jenis kanker.
24. PDGFRA (Platelet-Derived Growth Factor Receptor Alpha): Menyajikan informasi tentang gen PDGFRA dalam konteks glioma, terlibat dalam jalur pertumbuhan sel dan pembentukan pembuluh darah.

1.2.2 Mendeskripsikan Data

```
data.columns
```

```
Index(['Grade', 'Gender', 'Age_at_diagnosis', 'Race', 'IDH1', 'TP53', 'ATRX',
       'PTEN', 'EGFR', 'CIC', 'MUC16', 'PIK3CA', 'NF1', 'PIK3R1', 'FUBP1',
       'RB1', 'NOTCH1', 'BCOR', 'CSMD3', 'SMARCA4', 'GRIN2A', 'IDH2', 'FAT4',
       'PDGFRA'],
      dtype='object')
```

1.2.2.1 Analisa Tipe data

Dalam analisa data, terdapat beberapa tipe data yang sering ditemukan. Pemahaman tipe data ini penting karena berbagai jenis analisis statistik dan pemrosesan data dapat memerlukan pendekatan yang berbeda tergantung pada jenis data yang digunakan.

Berikut adalah Macam - Macam Data yang ada pada data ini.

1. Tipe nominal
   - memiliki value 1 yang melambangkan ya dan 0 yang melambangkan tidak. > Pada data ini mencakup fitur : *'Grade', 'Gender', 'Race', 'IDH1', 'TP53', 'ATRX', 'PTEN', 'EGFR', 'CIC', 'MUC16', 'PIK3CA', 'NF1', 'PIK3R1', 'FUBP1', 'RB1', 'NOTCH1', 'BCOR', 'CSMD3', 'SMARCA4', 'GRIN2A', 'IDH2', 'FAT4', 'PDGFRA*
   - mencakup tipe data numeric. > yakni Age_at_diagnosis

1.2.3 Deskripsi Fitur

1. Grade (Kelas Tingkat Keparahan Tumor) Deskripsi: Menunjukkan tingkat keparahan tumor. Contoh Nilai: > 1 : Tumor dengan pertumbuhan lambat (Tingkat I)

---

2 : Tumor dengan pertumbuhan yang lebih cepat (Tingkat II)

---

3 : Tumor yang lebih agresif (Tingkat III)

---

4 : Tumor ganas dengan pertumbuhan paling cepat (Tingkat IV)

---

2. Gender (Jenis Kelamin) Deskripsi: Menunjukkan jenis kelamin pasien. Contoh Nilai: > 0 : perempuan

---

1 : laki - laki

---

3. Age_at_diagnosis (Usia pada Saat Diagnosis Tumor) Deskripsi: Menunjukkan usia pasien pada saat diagnosis tumor. Contoh Nilai: > 25: Pasien didiagnosis pada usia 25 tahun

---

60: Pasien didiagnosis pada usia 60 tahun

---

4. Race (Ras) Deskripsi: Menunjukkan ras pasien. Contoh Nilai: > 0 : Tidak memiliki ras

---

1 : memiliki ras

---

5. IDH1 Deskripsi: Menunjukkan status mutasi pada gen IDH1, yang dapat menjadi petunjuk untuk jenis tumor tertentu, seperti glioma. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

6. TP53 Deskripsi: Menunjukkan status mutasi pada gen TP53, yang terlibat dalam pengawasan pertumbuhan sel dan dapat terlibat

dalam perkembangan tumor. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

7. ATRX Deskripsi: Menunjukkan status mutasi pada gen ATRX, yang terlibat dalam regulasi panjang telomer dan stabilitas kromosom. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

8. PTEN Deskripsi: Menunjukkan status mutasi pada gen PTEN, yang berperan dalam pengawasan pertumbuhan sel dan dapat terlibat dalam perkembangan kanker. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

9. EGFR Deskripsi: Menunjukkan status mutasi pada gen EGFR, yang dapat terlibat dalam pertumbuhan sel dan sering diidentifikasi pada beberapa jenis kanker. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

10. CIC Deskripsi: Menunjukkan status mutasi pada gen CIC, yang

terlibat dalam regulasi siklus sel dan dapat terlibat dalam perkembangan tumor. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

11. MUC16 Deskripsi: Menunjukkan status mutasi pada gen MUC16, yang dapat terlibat dalam perkembangan beberapa jenis kanker. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

12. PIK3CA Deskripsi: Menunjukkan status mutasi pada gen PIK3CA, yang terlibat dalam jalur sinyal pertumbuhan sel. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

13. NF1 Deskripsi: Menunjukkan status mutasi pada gen NF1, yang terlibat dalam regulasi pertumbuhan sel dan dapat terlibat dalam perkembangan tumor. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasii

---

14. PIK3R1 Deskripsi: Menunjukkan status mutasi pada gen PIK3R1,

yang terlibat dalam regulasi jalur PI3K-Akt. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

15. FUBP1 Deskripsi: Menunjukkan status mutasi pada gen FUBP1, yang dapat terlibat dalam regulasi ekspresi gen. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

16. RB1 Deskripsi: Menunjukkan status mutasi pada gen RB1, yang terlibat dalam regulasi siklus sel. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

17. NOTCH1 Deskripsi: Menunjukkan status mutasi pada gen NOTCH1, yang dapat terlibat dalam regulasi diferensiasi sel dan pertumbuhan. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

18. BCOR Deskripsi: Menunjukkan status mutasi pada gen BCOR,

yang terlibat dalam regulasi ekspresi gen dan diferensiasi sel. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

19. CSMD3 Deskripsi: Menunjukkan status mutasi pada gen CSMD3, yang dapat terlibat dalam regulasi pertumbuhan sel. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

20. SMARCA4 Deskripsi: Menunjukkan status mutasi pada gen SMARCA4, yang terlibat dalam regulasi struktur kromosom dan pertumbuhan sel. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

21. GRIN2A Deskripsi: Menunjukkan status mutasi pada gen GRIN2A, yang terlibat dalam transmisi sinyal sel saraf. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

22. IDH2 Deskripsi: Menunjukkan status mutasi pada gen IDH2, yang

dapat terkait dengan jenis tumor tertentu. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

23. FAT4 Deskripsi: Menunjukkan status mutasi pada gen FAT4, yang terlibat dalam regulasi pertumbuhan sel. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

---

24. PDGFRA Deskripsi: Menunjukkan status mutasi pada gen PDGFRA, yang terlibat dalam pertumbuhan dan pemeliharaan sel. Contoh Nilai: > 0 : Tidak ada mutasi

---

1: Ada mutasi

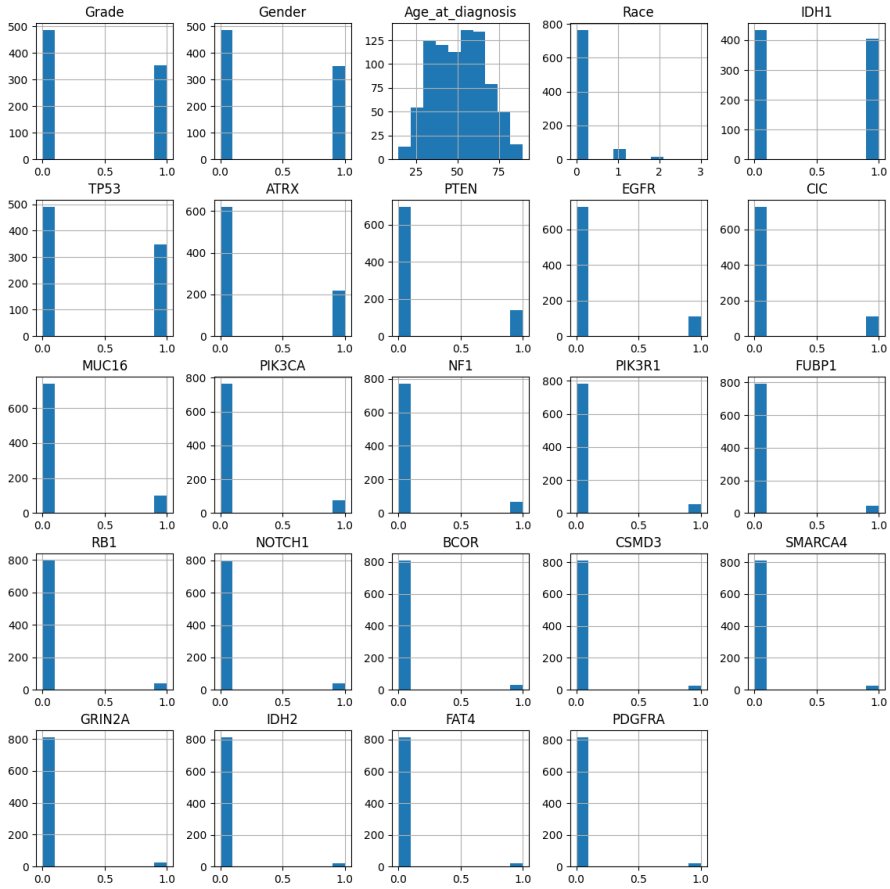
1.2.4 Eksplorasi Data

1.2.4.1 Visualisasi Data

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

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



1.2.4.2 Skoring Fitur

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.

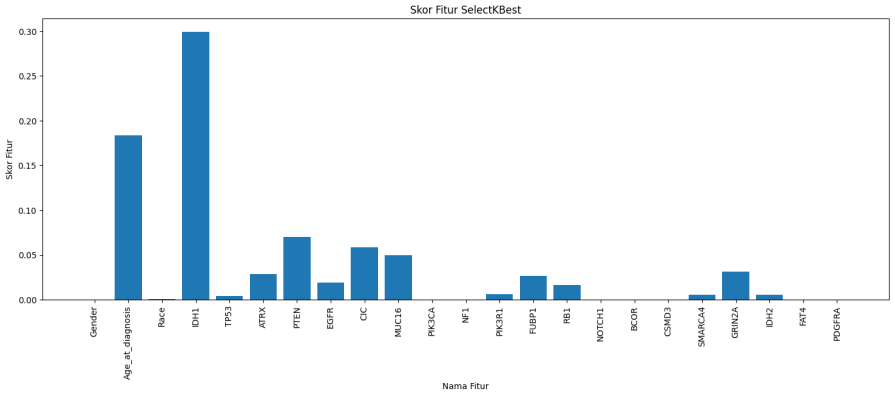
```
from sklearn.feature_selection import SelectKBest, mutual_info_classif

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

# 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: Gender, Skor: 0.0
Fitur 1: Age_at_diagnosis, Skor: 0.1836302604370481
Fitur 2: Race, Skor: 0.001083226817359284
Fitur 3: IDH1, Skor: 0.29916352355341314
Fitur 4: TP53, Skor: 0.004332139019691539
Fitur 5: ATRX, Skor: 0.028339027312104026
Fitur 6: PTEN, Skor: 0.07035477153576664
Fitur 7: EGFR, Skor: 0.019006818303878292
Fitur 8: CIC, Skor: 0.05872220363351821
Fitur 9: MUC16, Skor: 0.049386808646663116
Fitur 10: PIK3CA, Skor: 0.0
Fitur 11: NF1, Skor: 0.0
Fitur 12: PIK3R1, Skor: 0.006321600145554607
Fitur 13: FUBP1, Skor: 0.02634246570829135
Fitur 14: RB1, Skor: 0.01660825390759091
Fitur 15: NOTCH1, Skor: 0.0
Fitur 16: BCOR, Skor: 0.0
Fitur 17: CSMD3, Skor: 0.0
Fitur 18: SMARCA4, Skor: 0.005637498408886277
Fitur 19: GRIN2A, Skor: 0.031420500403958496
Fitur 20: IDH2, Skor: 0.0058230458756221015
Fitur 21: FAT4, Skor: 0.0
Fitur 22: PDGFRA, Skor: 0.0
```

```
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()
```



1.2.5 Mengidentifikasi Kualitas Data

1.2.5.1 Identifikasi missing value

Identifikasi nilai yang hilang (missing value) adalah proses untuk menemukan dan mengenali keberadaan nilai yang tidak ada atau kosong dalam suatu dataset. Langkah-langkah umum untuk mengidentifikasi missing value melibatkan: 1. Pemeriksaan Visual: Tinjau dataset secara visual, terutama jika ukurannya tidak terlalu besar. Identifikasi apakah ada nilai yang kosong atau tidak biasa. 2. Statistik Deskriptif: Gunakan metode statistik deskriptif seperti fungsi mean, median, dan std deviation untuk melihat apakah terdapat nilai yang tidak valid atau tidak sesuai dengan harapan. 3. Pemetaan Nilai yang Hilang: Representasikan missing value dengan visualisasi, seperti heatmap, untuk memahami pola distribusi nilai yang hilang dalam dataset. 4. Fungsi atau Metode Khusus: Gunakan fungsi atau metode khusus dalam bahasa pemrograman atau perangkat lunak analisis data untuk mengidentifikasi secara otomatis nilai yang hilang, seperti fungsi isna() atau isnull() dalam Python. 5. Analisis Korelasi: Evaluasi korelasi antara variabel yang memiliki missing value dengan variabel lain dalam dataset untuk memahami apakah ada pola atau hubungan yang dapat membantu dalam pengelolaan missing value.

Penanganan Missing Value : 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.

```
# mengecek 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:
Grade               False
Gender              False
Age_at_diagnosis    False
Race                False
IDH1                False
TP53                False
ATRX                False
PTEN                False
EGFR                False
CIC                 False
MUC16               False
PIK3CA              False
NF1                 False
PIK3R1              False
FUBP1               False
RB1                 False
NOTCH1              False
BCOR                False
CSMD3               False
SMARCA4             False
GRIN2A              False
IDH2                False
FAT4                False
PDGFRA              False
dtype: bool
```

Noted : tidak ada *missing value* pada data

1.2.5.2 Identifikasi Duplikat Data

Duplikat data merupakan suatu kondisi dimana suatu baris memiliki nilai yang sama persis di semua kolom pada baris lainnya. Adanya data yang redundan (berulang) dapat mengganggu hasil analisis dan menghasilkan akurasi yang tidak akurat. Untuk mengecek adanya duplikat data, maka digunakan fungsi *dupicated()*

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

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

```
Jumlah data yang duplikat: 1
```

Noted : terdapat 1 data duplikat

1.2.5.3 Identifikasi Outlier

Outlier, pada dasarnya, adalah suatu nilai yang secara signifikan menonjol dan berbeda jauh dari nilai-nilai lain dalam suatu set data. Kadang-kadang disebut sebagai nilai ekstrem, outlier tidak hanya mencolok tetapi juga memiliki sifat yang tidak representatif terhadap fenomena sekitarnya. Deteksi outlier menjadi penting dalam analisis data, karena keberadaannya dapat mengganggu hasil analisis keseluruhan dan memengaruhi kesimpulan yang dapat diambil dari dataset tersebut

```
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 :  29
Sisa data :  810
```

1.2.5.4 Identifikasi 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.

```
# Menghitung jumlah target pada data tanpa outlier
target_no_outliers = data['Grade'].value_counts()
```

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

```
Jumlah data pada tiap target :
Grade
0    487
1    352
Name: count, dtype: int64
```

1.2.6 Hasil Analisa Pada Data Understanding :

1. Data tidak memiliki *missing values*
2. Data memiliki 1 data redundan
3. Data memiliki 29 outlier

1.3 DATA PREPROCESSING

Setelah memahami data, akan dilakukan tahap preprocessing untuk menangani masalah pada data yang sudah didefinisikan pada data understanding, yakni. 1. Menghapus Data Duplikat 2. Menghapus Outlier 3. Menyeimbangkan proporsi data tiap target

Setelah data siap, akan dilakukan skoring fitur kembali.

```
import pandas as pd

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

	Grade	Gender	Age_at_diagnosis	Race	IDH1	TP53	ATRX	PTEN	EGFR	CIC	...	FUBP
0	0	0	51.30	0	1	0	0	0	0	0	...	1
1	0	0	38.72	0	1	0	0	0	0	1	...	0
2	0	0	35.17	0	1	1	1	0	0	0	...	0
3	0	1	32.78	0	1	1	1	0	0	0	...	0
4	0	0	31.51	0	1	1	1	0	0	0	...	0

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

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

```
Banyaknya data :  839
Banyaknya kolom :  24
```

1.3.1 Menghapus Data Duplikat

Duplikat data merupakan suatu kondisi dimana suatu baris memiliki nilai yang sama persis di semua kolom pada baris lainnya. Adanya data yang redundan (berulang) dapat mengganggu hasil analisis dan menghasilkan akurasi yang tidak akurat.

```
# Menghapus data yang duplikat
data_bersih = data.drop_duplicates()

print("Banyaknya sisa data : ", data_bersih.shape[0])
```

```
Banyaknya sisa data :  838
```

```
from sklearn.neighbors import LocalOutlierFactor

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

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

```
Sisa data :  810
```

1.3.2 Menyeimbangkan Data Tiap Target

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.

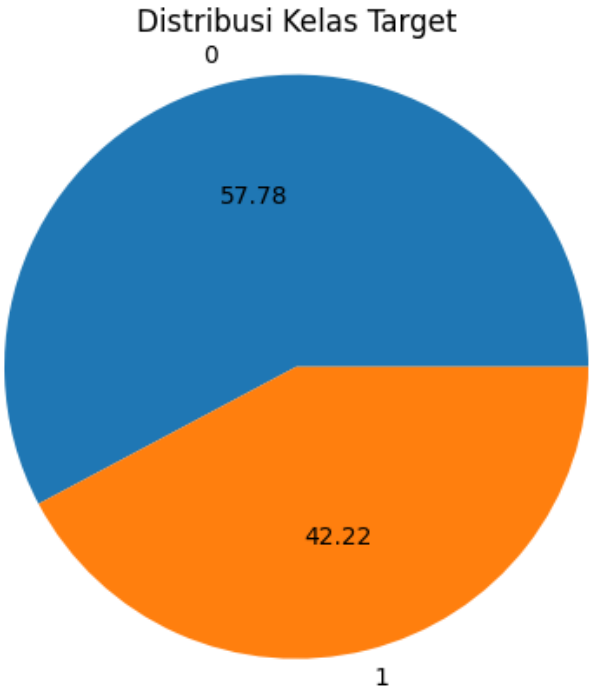
1.3.2.1 Proporsi Jumlah Data Di Tiap Label

```
fitur = data_oke.drop(columns=['Grade'])
target = data_oke['Grade']

target.value_counts()
```

```
Grade
0    468
1    342
Name: count, dtype: int64
```

1.3.2.1.1 ***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()
```



1.3.2.2 Penyeimbangan jumlah atau proporsi data

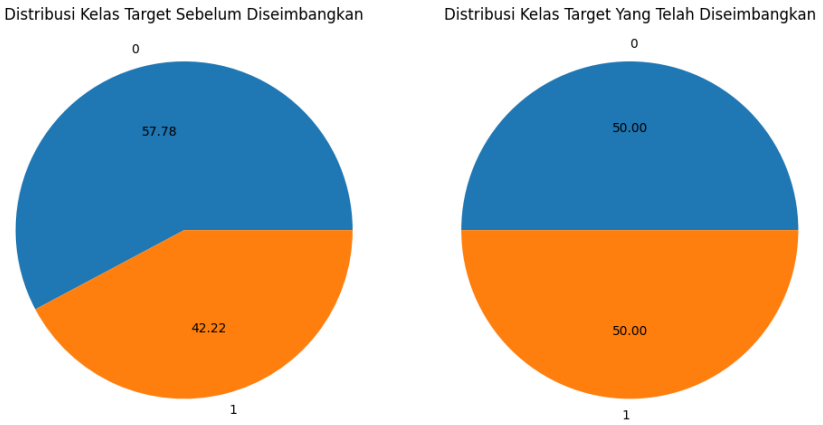
Perbandingan proporsi data tiap target sangat jauh sehingga untuk menghemat waktu komputasi output antar target akan diseimbangkan menggunakan metode UnderSampling. Undersampling adalah teknik untuk mengurangi jumlah data pada kelas mayoritas sehingga jumlahnya setara dengan kelas minoritas.

```
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 :
Grade
0    342
1    342
Name: count, dtype: int64
```

1.3.2.2.1 *Visualisasi banyaknya data di tiap label*

```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(1, 2, figsize=(10, 5))

# Plot distribusi kelas target sebelum penyeimbangan
target_tidak_seimbang = target.value_counts()
axs[0].pie(target_tidak_seimbang, labels=target_tidak_seimbang.index, autopct='%.2f')
axs[0].set_title('Distribusi Kelas Target Sebelum Diseimbangkan')
axs[0].axis('equal')

# Plot distribusi kelas target setelah penyeimbangan
target_sudah_seimbang = target_seimbang.value_counts()
axs[1].pie(target_sudah_seimbang, labels=target_sudah_seimbang.index, autopct='%.2f')
axs[1].set_title('Distribusi Kelas Target Yang Telah Diseimbangkan')
axs[1].axis('equal')

plt.tight_layout()
plt.show()
```



1.3.3 Hasil Preprocessing Data

```
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_seimbang1.csv', index=False)
```

1.3.4 Feature Selection

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.

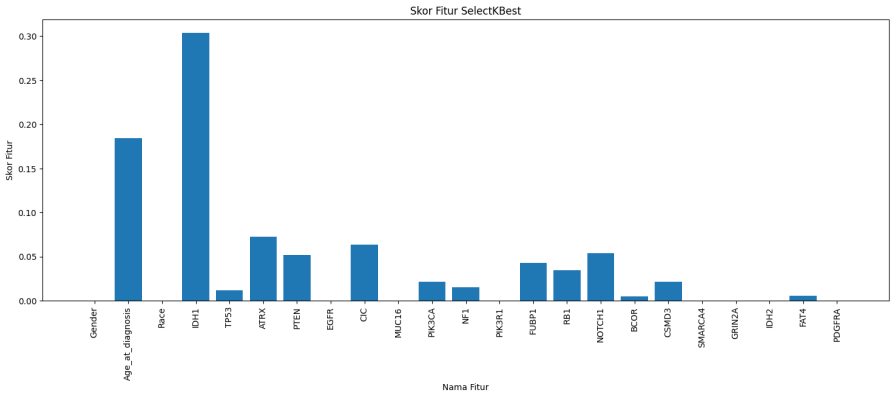
Skor informasi menggunakan *mutual_info_classif* berguna untuk mengevaluasi seberapa informatif suatu fitur terhadap variabel target. Skoring fitur menggunakan *mutual_info_classif* dapat mengukur seberapa banyak informasi dari variabel independen (fitur) yang terdapat pada variabel dependen (target) dalam dataset. Secara spesifik, skor ini menunjukkan seberapa banyak informasi dari suatu fitur yang dapat membantu dalam memprediksi target. **Semakin tinggi skornya, semakin informatif fitur tersebut terhadap variabel target.**

```
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_seimbang, target_seimbang)
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: Gender, Skor: 0.0
Fitur 1: Age_at_diagnosis, Skor: 0.1844928943766866
Fitur 2: Race, Skor: 0.0
Fitur 3: IDH1, Skor: 0.3037068691204612
Fitur 4: TP53, Skor: 0.0114767531002542
Fitur 5: ATRX, Skor: 0.07239657961576484
Fitur 6: PTEN, Skor: 0.05205249513954713
```

```
Fitur 7: EGFR, Skor: 0.0
Fitur 8: CIC, Skor: 0.0634755402890419
Fitur 9: MUC16, Skor: 0.0
Fitur 10: PIK3CA, Skor: 0.021167499071111262
Fitur 11: NF1, Skor: 0.01522729428307068
Fitur 12: PIK3R1, Skor: 0.0
Fitur 13: FUBP1, Skor: 0.043069867448098 04
Fitur 14: RB1, Skor: 0.034603512687797267
Fitur 15: NOTCH1, Skor: 0.05390188520546868
Fitur 16: BCOR, Skor: 0.004819449013254706
Fitur 17: CSMD3, Skor: 0.021559414485805117
Fitur 18: SMARCA4, Skor: 0.0
Fitur 19: GRIN2A, Skor: 0.0
Fitur 20: IDH2, Skor: 0.0
Fitur 21: FAT4, Skor: 0.005786907567380206
Fitur 22: PDGFRA, Skor: 0.0
```

```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()
```



1.4 MODELLING

1.4.1 Load Dataset

```
import pandas as pd

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

	Grade	Gender	Age_at_diagnosis	Race	IDH1	TP53	ATRX	PTEN	EGFR	CIC	...	FUBP
0	0	0	51.30	0	1	0	0	0	0	0	...	1
1	0	0	38.72	0	1	0	0	0	0	1	...	0
2	0	0	35.17	0	1	1	1	0	0	0	...	0
3	0	1	32.78	0	1	1	1	0	0	0	...	0
4	0	0	31.51	0	1	1	1	0	0	0	...	0

```
#Banyak data dan kolom

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

```
Banyaknya data :  839
Banyaknya kolom :  24
```

1.4.2 Split Dataset

```
from sklearn.model_selection import train_test_split

# 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 =
```

1.4.3 Grafik dan tingkat kepentingannya

```
import matplotlib.pyplot as plt

# Tampilkan skor fitur dalam grafik
plt.figure(figsize=(18, 6))
plt.bar(fitur_names, scores)
```

```
# Membuat label pada sumbu x dan y, serta judul pada grafiknya
plt.xlabel("Nama Fitur")
plt.ylabel("Skor Fitur")
plt.title("Skor Fitur SelectKBest")
# Menambahkan rotasi pada sumbu x
plt.xticks(rotation=90)
plt.show()
```

1.5 Data Preprocessing

Setelah memahami data, akan dilakukan tahap preprocessing untuk menangani masalah pada data yang sudah didefinisikan pada data understanding, yakni. 1. Menghapus Data Duplikat 2. Menghapus Outlier 3. Menyeimbangkan proporsi data tiap target

Setelah data siap, akan dilakukan : 1. Skoring tiap fitur kembali 2. Normalisasi Data 3. Eksplorasi Model

```
import pandas as pd

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

	Grade	Gender	Age_at_diagnosis	Race	IDH1	TP53	ATRX	PTEN	EGFR	CIC	...	FUBP
0	0	0	51.30	0	1	0	0	0	0	0	...	1
1	0	0	38.72	0	1	0	0	0	0	1	...	0
2	0	0	35.17	0	1	1	1	0	0	0	...	0
3	0	1	32.78	0	1	1	1	0	0	0	...	0
4	0	0	31.51	0	1	1	1	0	0	0	...	0

```
#Banyak data dan kolom

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

```
Banyaknya data :  839
Banyaknya kolom :  24
```

1.5.1 Menghapus Data Duplikat

```
# Menghapus data yang duplikat
data_bersih = data.drop_duplicates()

print("Banyaknya sisa data : ", data_bersih.shape[0])
```

1.5.2 Menghapus Outlier

```
from sklearn.neighbors import LocalOutlierFactor

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

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

1.5.3 Menyeimbangkan Data Tiap Target

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

target.value_counts()
```

1.5.4 Menyeimbangkan data target sesuai jumlahnya menggunakan jumlah pada target yang paling sedikit

```
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())
```

1.5.5 Simpan data yang telah seimbang di dalam database

```
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=['Grade'])
target = data_seimbang['Grade']
```

1.5.6 Eksplorasi Data (Skoring Fitur)

Melakukan skoring fitur dengan presentase kepentingannya

```
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: Gender, Skor: 0.0
Fitur 1: Age_at_diagnosis, Skor: 0.18672449077849174
Fitur 2: Race, Skor: 0.0010600399767952684
Fitur 3: IDH1, Skor: 0.2919775839551344
Fitur 4: TP53, Skor: 0.03145111467220385
Fitur 5: ATRX, Skor: 0.049446107986429455
Fitur 6: PTEN, Skor: 0.0814082771389042
Fitur 7: EGFR, Skor: 0.054840022727111526
Fitur 8: CIC, Skor: 0.07971535922808393
Fitur 9: MUC16, Skor: 0.0
```

```
Fitur 10: PIK3CA, Skor: 0.0
Fitur 11: NF1, Skor: 0.0
Fitur 12: PIK3R1, Skor: 0.0
Fitur 13: FUBP1, Skor: 0.009983749997428015
Fitur 14: RB1, Skor: 0.02414717554759105
Fitur 15: NOTCH1, Skor: 0.0
Fitur 16: BCOR, Skor: 0.003217834219356064
Fitur 17: CSMD3, Skor: 0.016862161428498057
Fitur 18: SMARCA4, Skor: 0.0
Fitur 19: GRIN2A, Skor: 0.0
Fitur 20: IDH2, Skor: 0.0
Fitur 21: FAT4, Skor: 0.0
Fitur 22: PDGFRA, Skor: 0.011853691523513898
```

```
import matplotlib.pyplot as plt

# Tampilkan skor fitur dalam grafik
plt.figure(figsize=(18, 6))
plt.bar(fitur_names, scores)
# Membuat label pada sumbu x dan y, serta judul pada grafiknya
plt.xlabel("Nama Fitur")
plt.ylabel("Skor Fitur")
plt.title("Skor Fitur SelectKBest")
# Menambahkan rotasi pada sumbu x
plt.xticks(rotation=90)
plt.show()
```

1.5.7 Split Data

```
from sklearn.model_selection import train_test_split

# 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 =
```

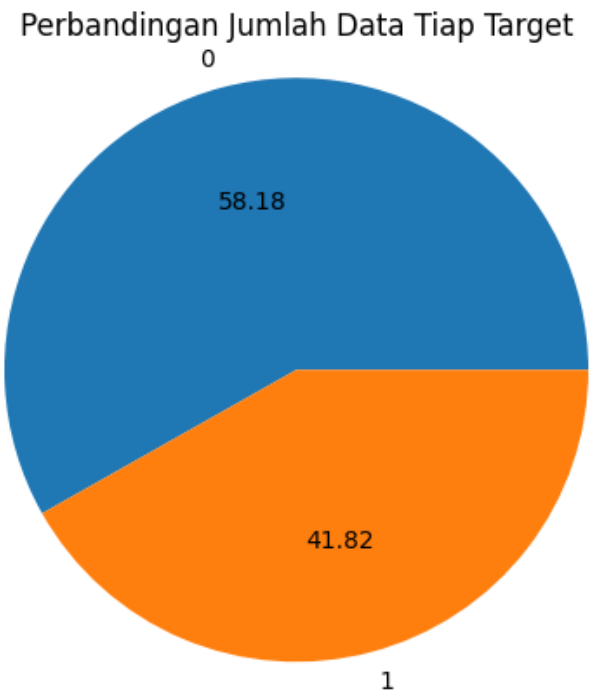
```
print("Banyaknya fitur  : ", fitur_train.shape[1])
print("Banyaknya data latih : ", fitur_train.shape[0])
print("Banyaknya data testing : ", fitur_test.shape[0])
```

```
Banyaknya fitur  :  23
Banyaknya data latih :  648
Banyaknya data testing :  162
```

```
target_train.value_counts()
```

```
Grade
0    377
1    271
Name: count, 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()
```



1.5.8 Normalisasi Data

1.5.9 Menggunakan Standarscaler (zscore)

Rumus untuk mencari Zscore: $$Z = \frac{X - \mu}{\sigma}$$

Penjelasan rumusnya: - Z = nilai standar - X = nilai teramati - = mean sampel - = simpangan baku sampel

Rumus untuk standar deviasi sendiri sebagai berikut.

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

```
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)
```

```
zscore_scaler
```

```
StandardScaler()
```

1.5.10 Membuat model

1.5.10.1 Konsep Naive Bayes

Naïve Bayes Classifier merupakan sebuah metoda klasifikasi yang berakar pada teorema Bayes . Metode pengklasifikasian dengan menggunakan metode probabilitas dan statistik.

Rumus Naive Bayes:

$$P(C|X) = \frac{P(C) - P(X|C)}{P(X)}$$

penjelasan: - P(C|X) = nilai probabilitas posterior dari kelas C given fitur X - P(C) = Nilai probabilitas prior dari kelas C - P(X|C) = likelihood dari fitur X jika kelasnya adalah C - P(X) = nilai probabilitas evidensi atau probabilitas dari fitur X

$$P(A) = \frac{n(A)}{n(s)}$$

Penjelasan: - P(A) = peluang - n(A) = jumlah peluang yang mungkin terjadi - n(S) = jumlah sampel dari sebuah kejadian

1.5.11 Menggunakan Minmaxscaler

Rumus untuk mencari Minmaxscaler:

$$X' = \frac{X - min}{max - min}$$

Penjelasan rumusnya: - X = nilai yang sudah di normalisasi - min = nilai data minimal - max = nilai data maksimal

Menggunakan Metode Naive Bayes

```
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

best_accuracy_nb_zscore = 0
best_k_zscore = 0
best_accuracy_nb_minmax = 0
best_k_minmax = 0

for k in range(1, fitur_train.shape[1] + 1):
    # Buat objek SelectKBest dengan mutual_info_classif sebagai fungsi skor
    k_best = SelectKBest(score_func=mutual_info_classif, k=k)
```

```
    # Fiturkan objek SelectKBest ke data training dan testing untuk kedua normalisasi (zscore d
    zscore_training_terbaik = k_best.fit_transform(zscore_training, target_train)
    zscore_testing_terbaik = k_best.transform(zscore_testing)
    minmaxtraining_terbaik = k_best.fit_transform(minmaxtraining, target_train)
    minmaxtesting_terbaik = k_best.transform(minmaxtesting)

    # Buat dan latih model dengan normalisasi zscore
    model_zscore = GaussianNB()
    model_zscore.fit(zscore_training_terbaik, target_train)

    # Lakukan prediksi pada data uji dengan normalisasi zscore
    y_pred_nb_zscore = model_zscore.predict(zscore_testing_terbaik)

    # Hitung akurasi dengan normalisasi zscore
    accuracy_nb_zscore = accuracy_score(target_test, y_pred_nb_zscore)

    # Buat dan latih model dengan normalisasi minmax
    model_minmax = GaussianNB()
    model_minmax.fit(minmaxtraining_terbaik, target_train)

    # Lakukan prediksi pada data uji dengan normalisasi minmax
    y_pred_nb_minmax = model_minmax.predict(minmaxtesting_terbaik)

    # Hitung akurasi dengan normalisasi minmax
    accuracy_nb_minmax = accuracy_score(target_test, y_pred_nb_minmax)

    # Memeriksa apakah akurasi dengan normalisasi zscore lebih baik dari yang sebelumnya
    if accuracy_nb_zscore > best_accuracy_nb_zscore:
        best_accuracy_nb_zscore = accuracy_nb_zscore
        best_k_zscore = k

    # Memeriksa apakah akurasi dengan normalisasi minmax lebih baik dari yang sebelumnya
    if accuracy_nb_minmax > best_accuracy_nb_minmax:
        best_accuracy_nb_minmax = accuracy_nb_minmax
        best_k_minmax = k

print("Dengan Normalisasi Zscore:")
print("Fitur terbaik yang bisa digunakan", best_k_zscore, "dengan akurasi : ", best_accuracy_nb

print("Dengan Normalisasi Minmax:")
print("Fitur terbaik yang bisa digunakan", best_k_minmax, "dengan akurasi : ", best_accuracy_nb

# Ambil indeks fitur terbaik dari SelectKBest
best_feature_indices_zscore = SelectKBest(score_func=mutual_info_classif, k=best_k_zscore).fit(
```

```
best_feature_indices_minmax = SelectKBest(score_func=mutual_info_classif, k=best_k_minmax).fit(

# Dapatkan nama fitur terbaik dari indeksnya
best_features_zscore = [fitur.columns[i] for i in best_feature_indices_zscore]
best_features_minmax = [fitur.columns[i] for i in best_feature_indices_minmax]


print("Fitur terbaik yang digunakan (dengan normalisasi Zscore):")
print(best_features_zscore)

print("Fitur terbaik yang digunakan (dengan normalisasi Minmax):")
print(best_features_minmax)
```

```
Dengan Normalisasi Zscore:
Fitur terbaik yang bisa digunakan 5 dengan akurasi :  0.8395061728395061
Dengan Normalisasi Minmax:
Fitur terbaik yang bisa digunakan 6 dengan akurasi :  0.8333333333333334
Fitur terbaik yang digunakan (dengan normalisasi Zscore):
['Age_at_diagnosis', 'IDH1', 'ATRX', 'PTEN', 'EGFR']
Fitur terbaik yang digunakan (dengan normalisasi Minmax):
['Age_at_diagnosis', 'IDH1', 'ATRX', 'PTEN', 'CIC', 'NOTCH1']
```

```
import pandas as pd

# kolom-kolom yang ingin Anda pertahankan
kolom_pilihan = ['IDH1', 'Age_at_diagnosis', 'CIC', 'NOTCH1', 'Grade']

# Buat dataset baru hanya dengan kolom yang dipilih
dataset_baru = data[kolom_pilihan]

# Simpan dataset baru dalam file CSV
dataset_baru.to_csv('dataset_baru_nb.csv', index=False)  # Simpan ke file CSV tanpa menyertakan
```

```
dataset_baru.head(5)
```

	IDH1	Age_at_diagnosis	CIC	NOTCH1	Grade
0	1	51.30	0	0	0
1	1	38.72	1	0	0
2	1	35.17	0	0	0
3	1	32.78	0	0	0
4	1	31.51	0	0	0

1.5.12 Split Dataset

```
from sklearn.model_selection import train_test_split

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

# 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 =
```

1.5.13 Normalisasi Data

```
import pickle
from sklearn.preprocessing import StandardScaler

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

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

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

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

# menerapkan normalisasi zscore pada data training
zscoretraining = zscorescaler.transform(fitur_train)

# menerapkan normalisasi zscore pada data testing
zscoretesting = zscorescaler.transform(fitur_test)
```

Membuat Model

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

```
# Definisi model Gaussian Naive Bayes
nb = GaussianNB()

# Definisi grid parameter yang akan diuji
param_grid = {
    'priors': [None],  # Anda dapat menentukan probabilitas prior jika memiliki pengetahuan seb
    'var_smoothing': [19, 20, 16, 30]  # Hyperparameter yang mengontrol sebagian varians terbes
}


# Inisialisasi GridSearchCV
grid_search = GridSearchCV(estimator=nb, param_grid=param_grid, cv=5, n_jobs=-1)

# Latih model dengan data latih dan mencari parameter terbaik
grid_search.fit(zscoretraining, target_train)

# Hasil parameter terbaik
best_params = grid_search.best_params_
print("Parameter terbaik:", best_params)

# Model dengan parameter terbaik
best_nb = grid_search.best_estimator_

# Evaluasi model terbaik pada data uji
accuracy = best_nb.score(zscoretesting, target_test)
print("Akurasi model terbaik:", accuracy)
```

```
Parameter terbaik: {'priors': None, 'var_smoothing': 19}
Akurasi model terbaik: 0.5297619047619048
```

1.5.14 Simpan Model

```
import pickle

# Model dengan parameter terbaik
best_nb = grid_search.best_estimator_

# Simpan model Decision Tree terbaik ke dalam file pickle
with open('best_nb_model.pkl', 'wb') as file:
    pickle.dump(best_nb, file)

# Evaluasi model terbaik pada data uji
```

```
accuracy = best_nb.score(zscoretesting, target_test)
print("Akurasi model terbaik:", accuracy)
```

```
Akurasi model terbaik: 0.5297619047619048
```

1.6 EVALUATION

- Evalution ini terdapat model Naive Bayes Classifier dengan menggunakan zscore
- Nilai akurasinya yaitu sebesar 0.52

```
from sklearn.metrics import accuracy_score

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

model_nb.fit(zscoretraining, target_train)
y_pred_nb = model_nb.predict(zscoretesting)

# akurasi
akurasi_nb = accuracy_score(target_test, y_pred_nb)
print('Akurasi Naive Bayes Classifier : ', akurasi_nb)
```

```
Akurasi Naive Bayes Classifier :  0.5297619047619048
```

2

Summary

In summary, this book has no content whatsoever.

References