

DOKUMEN PROYEK

12S3205 - PENAMBANGAN DATA

PREDICTION DEPRESSION OF EXPLORING MENTAL HEALTH DATA USING LOGISTIC REGRESSION



Disusun Oleh :

12S22030	Bryan Evans Simamora
12S22049	Agnes Monica Sanjani Harefa
12S22050	Yohana Christine Sitanggang

**PROGRAM STUDI SARJANA SISTEM INFORMASI
FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO
(FITE)
INSTITUT TEKNOLOGI DEL**

DAFTAR ISI

DAFTAR ISI

BAB 1.....	3
BUSINESS UNDERSTANDING	3
1.1 Determine Business Objective	3
1.2 Determine Project Goal	3
1.3 Produce Project Plan	3
BAB 2.....	4
DATA UNDERSTANDING	4
2.1 Pengumpulan Data.....	4
2.2 Describe Data	4
2.3 Validation Data.....	5
BAB 3.....	23
DATA PREPARATION.....	23
3.1 Data Selection	23
3.2 Data Cleaning	25
3.3 Data Construct	44
3.4 Labeling Data	46
3.5 Data Integration	47
BAB 4.....	48
MODELLING	48
4.1 Membangun Skenario Pengujian	48
4.2 Membangun Model	50
BAB 5.....	52
MODEL EVALUATION	52
5.1 Mengevaluasi Hasil Pemodelan.....	52
5.2 Melakukan Review Proses Pemodelan	55
BAB 6.....	56
DEPLOYMENT	56
6.1 Membuat Rencana Deployment Model	56
6.2 Melakukan Deployment Model.....	57
6.3 Melakukan Rencana Pemeliharaan.....	57
6.4 Melakukan Pemeliharaan.....	57
DAFTAR PUSTAKA	59

BAB 1

BUSINESS UNDERSTANDING

1.1 Determine Business Objective

Kesehatan mental merupakan salah satu aspek penting dalam kesejahteraan manusia, namun sering kali terabaikan, terutama dalam lingkungan kerja yang penuh tekanan. Berdasarkan laporan dari WHO, kesehatan mental di tempat kerja sering kali diabaikan meskipun dampak negatifnya terhadap karyawan sangat signifikan [1]. Gangguan mental seperti depresi menjadi masalah utama yang perlu segera ditangani, karena dapat berdampak pada produktivitas dan kesejahteraan individu [2]. Oleh karena itu, tujuan utama dari proyek ini adalah untuk menganalisis data survei kesehatan mental guna mengidentifikasi faktor-faktor risiko yang berkontribusi terhadap depresi, serta menyediakan dasar untuk pencegahan yang lebih baik di tempat kerja.

1.2 Determine Project Goal

Tujuan bisnis dari proyek ini adalah untuk mengidentifikasi faktor-faktor yang mempengaruhi risiko gangguan mental di kalangan pekerja. Dengan demikian, penelitian ini berfokus pada upaya untuk mengurangi risiko kesehatan mental di tempat kerja melalui pendekatan berbasis data. Sebagai contoh, studi menunjukkan bahwa stres di tempat kerja dapat meningkatkan risiko gangguan mental, yang dapat berakibat pada biaya kesehatan yang tinggi dan penurunan produktivitas [3]. Oleh karena itu, model klasifikasi yang dikembangkan bertujuan untuk memprediksi siapa yang berisiko mengalami depresi, yang pada akhirnya dapat membantu dalam merancang kebijakan dan program dukungan kesehatan mental yang lebih efektif di perusahaan.

1.3 Produce Project Plan

Dalam aspek teknis, kesuksesan proyek ini diukur berdasarkan kemampuan model klasifikasi untuk mencapai akurasi, precision, dan recall lebih dari 60%. Model ini juga harus dapat digunakan secara online dan mampu menerima input pengguna untuk prediksi. Beberapa tantangan yang dihadapi adalah penggunaan data sintesis yang mungkin tidak sepenuhnya mewakili kondisi lokal, seperti di Indonesia, serta adanya kemungkinan nilai kosong dalam dataset [4].

Meskipun demikian, penelitian ini akan memberikan wawasan yang berguna bagi perusahaan dalam merancang intervensi berbasis data untuk mendukung kesehatan mental di tempat kerja, yang telah dibuktikan efektif oleh beberapa penelitian sebelumnya [5].

BAB 2

DATA UNDERSTANDING

2.1 Pengumpulan Data

Dataset yang digunakan dalam proyek ini bersumber dari [Kaggle Playground Series S4E11](#) dan bersifat sintetis, dirancang khusus untuk keperluan pembelajaran dan eksperimen dalam pengembangan model prediksi kesehatan mental. Dataset ini berformat CSV (*Comma Separated Values*), yang umum digunakan untuk menyimpan data dalam bentuk tabel. Terdapat dua bagian dalam dataset, yaitu *train set* dan *test set*.

Persiapan dan Pemrosesan Awal

```
[ ] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
```

Untuk melakukan eksplorasi dan visualisasi data, digunakan beberapa library Python berikut: numpy untuk manipulasi angka dan array, matplotlib.pyplot untuk visualisasi dasar seperti histogram dan scatter plot, seaborn untuk visualisasi lanjutan dengan tampilan yang lebih menarik.

```
[ ] train = pd.read_csv("/content/train.csv")
test = pd.read_csv("/content/test.csv")
```

Dataset dimuat menggunakan fungsi pandas dimana :

- Di sini kami menggunakan metode `pd.read_csv()` untuk membaca file `train.csv` dan menyimpannya ke dalam variabel `train` dalam bentuk `DataFrame`. File ini biasanya berisi data latih (training data) yang digunakan untuk membangun dan melatih model machine learning.
- Di sini kami menggunakan `pd.read_csv()` juga untuk membaca file `test.csv` dan menyimpannya dalam variabel `test`. File ini umumnya berisi data uji (testing data) yang tidak memiliki label target, dan digunakan untuk memprediksi hasil akhir menggunakan model yang sudah dilatih.

2.2 Describe Data

Tahap ini bertujuan untuk memahami struktur dan karakteristik awal dari dataset yang akan digunakan dalam pelatihan dan pengujian model.

```
▶ train.head()
```

	id	Name	Gender	Age	City	Working Professional or Student	Profession	Academic Pressure	Work Pressure	CGPA	Study Satisfaction	Job Satisfaction	Sleep Duration	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Depression
0	0	Aaradhyia	Female	49.0	Ludhiana	Working Professional	Chef	NaN	5.0	NaN	NaN	2.0	More than 8 hours	Healthy	BHM	No	1.0	2.0	No	0
1	1	Vivan	Male	26.0	Varanasi	Working Professional	Teacher	NaN	4.0	NaN	NaN	3.0	Less than 5 hours	Unhealthy	LLB	Yes	7.0	3.0	No	1
2	2	Yuvraj	Male	33.0	Visakhapatnam	Student	NaN	5.0	NaN	8.97	2.0	NaN	5-6 hours	Healthy	B.Pharm	Yes	3.0	1.0	No	1
3	3	Yuvraj	Male	22.0	Mumbai	Working Professional	Teacher	NaN	5.0	NaN	NaN	1.0	Less than 5 hours	Moderate	BBA	Yes	10.0	1.0	Yes	1
4	4	Rhea	Female	30.0	Kanpur	Working Professional	Business Analyst	NaN	1.0	NaN	NaN	1.0	5-6 hours	Unhealthy	BBA	Yes	9.0	4.0	Yes	0

Pemanggilan fungsi `train.head()` dari library pandas bertujuan untuk menampilkan lima baris pertama dari DataFrame `train`.

```
[ ] test.head()
```

	id	Name	Gender	Age	City	Working Professional or Student	Profession	Academic Pressure	Work Pressure	CGPA	Study Satisfaction	Job Satisfaction	Sleep Duration	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness
0	140700	Shivam	Male	53.0	Visakhapatnam	Working Professional	Judge	NaN	2.0	NaN	NaN	5.0	Less than 5 hours	Moderate	LLB	No	9.0	3.0	Yes
1	140701	Sanya	Female	58.0	Kolkata	Working Professional	Educational Consultant	NaN	2.0	NaN	NaN	4.0	Less than 5 hours	Moderate	B.Ed	No	6.0	4.0	No
2	140702	Yash	Male	53.0	Jaipur	Working Professional	Teacher	NaN	4.0	NaN	NaN	1.0	7-8 hours	Moderate	B.Arch	Yes	12.0	4.0	No
3	140703	Nalini	Female	23.0	Rajkot	Student	NaN	5.0	NaN	6.84	1.0	NaN	More than 8 hours	Moderate	B.Sc	Yes	10.0	4.0	No
4	140704	Shaurya	Male	47.0	Kalyan	Working Professional	Teacher	NaN	5.0	NaN	NaN	5.0	7-8 hours	Moderate	BCA	Yes	3.0	4.0	No

Pemanggilan fungsi `test.head()` dari library pandas bertujuan untuk menampilkan lima baris pertama dari DataFrame `test`

```
[ ] print(train.shape)
print(test.shape)
```

```
(140700, 20)
(93800, 19)
```

Dataset `train` memiliki 140.700 baris dan 20 kolom, sedangkan dataset `test` memiliki 93.800 baris dan 19 kolom. Baris merepresentasikan jumlah sampel data, dan kolom merepresentasikan jumlah fitur atau atribut dalam dataset tersebut. Perbedaan jumlah kolom antara `train` dan `test` mengindikasikan bahwa ada satu kolom yang tidak disertakan dalam data pengujian. Umumnya, kolom yang hilang tersebut adalah kolom target atau label yang ingin diprediksi oleh model.

2.3 Validation Data

Validasi data dilakukan untuk memastikan kualitas dan konsistensi dataset sebelum masuk ke tahap pemodelan. Beberapa aspek utama yang divalidasi meliputi nilai kosong (*missing values*), keberadaan outlier atau anomali, duplikasi data, serta distribusi variabel target.

1. Check Missing Values

```
▶ train.info()
```

```
↗ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 140700 entries, 0 to 140699
Data columns (total 20 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   id                                         140700 non-null  int64
1   Name                                       140700 non-null  object
2   Gender                                    140700 non-null  object
3   Age                                        140700 non-null  float64
4   City                                       140700 non-null  object
5   Working Professional or Student          140700 non-null  object
6   Profession                                104070 non-null  object
7   Academic Pressure                        27897 non-null   float64
8   Work Pressure                            112782 non-null  float64
9   CGPA                                      27898 non-null   float64
10  Study Satisfaction                       27897 non-null   float64
11  Job Satisfaction                         112790 non-null  float64
12  Sleep Duration                           140700 non-null  object
13  Dietary Habits                           140696 non-null  object
14  Degree                                   140698 non-null  object
15  Have you ever had suicidal thoughts ?    140700 non-null  object
16  Work/Study Hours                         140700 non-null  float64
17  Financial Stress                         140696 non-null  float64
18  Family History of Mental Illness         140700 non-null  object
19  Depression                               140700 non-null  int64
dtypes: float64(8), int64(2), object(10)
memory usage: 21.5+ MB
```

Dataset ini berisi 140.700 entri dan 20 kolom, dengan kombinasi tipe data `int64`, `float64`, dan `object`. Dataset ini tampaknya digunakan untuk menganalisis faktor-faktor yang berkaitan dengan depresi, ditunjukkan oleh kolom target Depression. Berikut adalah penjelasan untuk masing-masing kolom

```
[ ] train.columns
```

```
Index(['id', 'Name', 'Gender', 'Age', 'City',
       'Working Professional or Student', 'Profession', 'Academic Pressure',
       'Work Pressure', 'CGPA', 'Study Satisfaction', 'Job Satisfaction',
       'Sleep Duration', 'Dietary Habits', 'Degree',
       'Have you ever had suicidal thoughts ?', 'Work/Study Hours',
       'Financial Stress', 'Family History of Mental Illness', 'Depression'],
      dtype='object')
```


menampilkan seluruh nama kolom (fitur) yang terdapat dalam dataset pelatihan train.

```
▶ train.isnull().sum()
```

	0
id	0
Name	0
Gender	0
Age	0
City	0
Working Professional or Student	0
Profession	36630
Academic Pressure	112803
Work Pressure	27918
CGPA	112802
Study Satisfaction	112803
Job Satisfaction	27910
Sleep Duration	0
Dietary Habits	4
Degree	2
Have you ever had suicidal thoughts ?	0
Work/Study Hours	0
Financial Stress	4
Family History of Mental Illness	0
Depression	0

dtype: int64

Fungsi `train.isnull().sum()` digunakan untuk memeriksa jumlah nilai yang hilang (missing values) di setiap kolom dalam dataset train.

 `test.isnull().sum()`


	0
id	0
Name	0
Gender	0
Age	0
City	0
Working Professional or Student	0
Profession	24632
Academic Pressure	75033
Work Pressure	18778
CGPA	75034
Study Satisfaction	75033
Job Satisfaction	18774
Sleep Duration	0
Dietary Habits	5
Degree	2
Have you ever had suicidal thoughts ?	0
Work/Study Hours	0
Financial Stress	0
Family History of Mental Illness	0

dtype: int64

Fungsi `test.isnull().sum()` digunakan untuk memeriksa jumlah nilai yang hilang (missing values) di setiap kolom dalam dataset test.

2. Data Duplikat

Pemeriksaan terhadap data duplikasi dilakukan pada dataset train dan test

```
 all_duplicates = train[train.duplicated(keep=False)]
print(all_duplicates)
```

Empty DataFrame

Columns: [id, Name, Gender, Age, City, Working Professional or Student, Profession, Academic Pressure, Work Pressure, CGPA, Study Satisfaction, Job Satisfaction, Sleep Duration, Dietary Habits, Degree, Have you ever had suicidal thoughts ?, Work/Study Hours, Financial Stress, Family History of Mental Illness, Depression]

Index: []

mendeteksi baris-baris duplikat dalam dataset `train`, dengan parameter `keep=False` yang berarti semua baris yang terindikasi sebagai duplikat akan ditampilkan (baik baris pertama maupun yang duplikatnya).

```
# Count duplicates
duplicate_count = train.duplicated().sum()
print(f"Number of duplicate rows: {duplicate_count}")
```

Number of duplicate rows: 0

Tidak ada baris duplikat dalam dataframe train.

```
[ ] # Count duplicates
duplicate_count = test.duplicated().sum()
print(f"Number of duplicate rows: {duplicate_count}")
```

Number of duplicate rows: 0

Tidak ada baris duplikat dalam dataframe test.

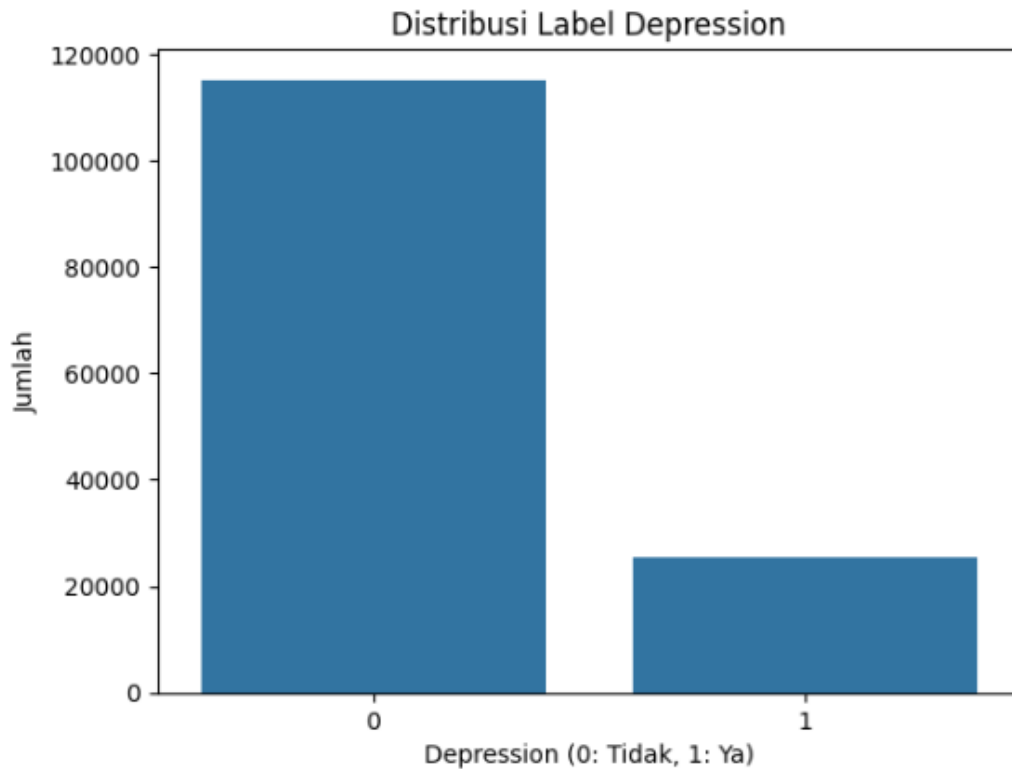
3. Distribusi Variabel Target

Visualisasi ini bertujuan untuk menunjukkan distribusi frekuensi dua kategori dalam label "Depression". Label tersebut memiliki dua nilai:

- 0 (Tidak): Mengindikasikan bahwa seseorang tidak mengalami depresi.
- 1 (Ya): Mengindikasikan bahwa seseorang mengalami depresi.

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='Depression', data=train)
plt.title('Distribusi Label Depression')
plt.xlabel('Depression (0: Tidak, 1: Ya)')
plt.ylabel('Jumlah')
plt.show()
```



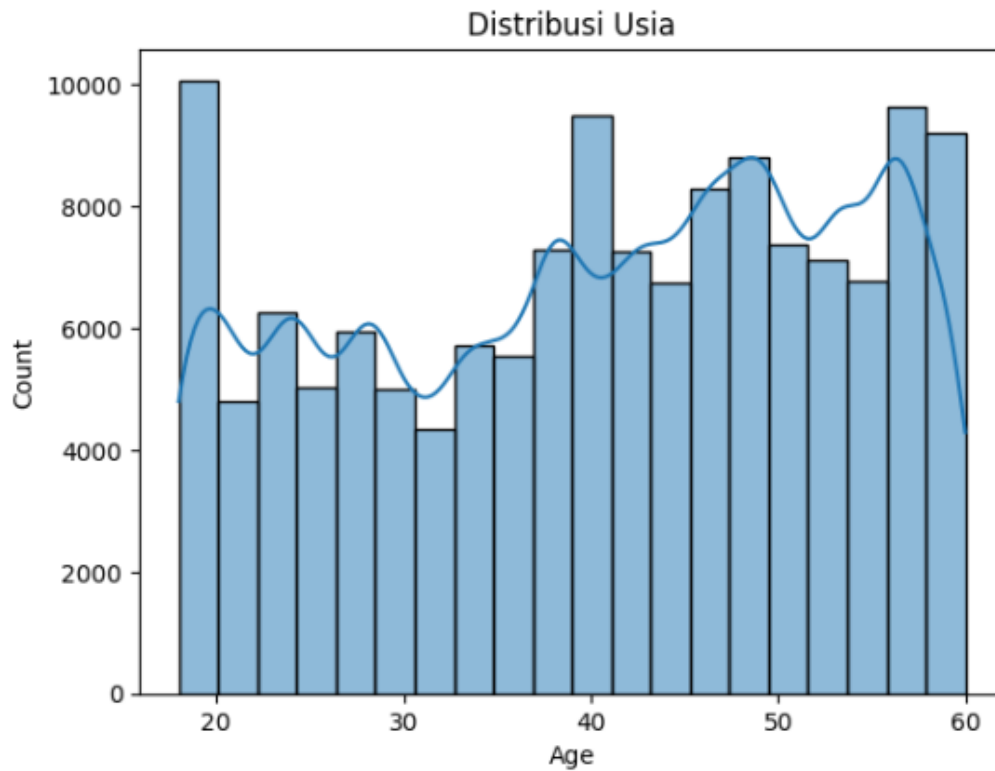
Distribusi data ini sangat tidak seimbang, dengan label "Tidak Depresi" (0) jauh lebih banyak dibandingkan dengan label "Ya Depresi" (1). Meskipun distribusi data yang tidak seimbang ini menunjukkan bahwa sebagian besar individu dalam dataset tidak depresi (label 0), ini tidak secara langsung menunjukkan bahwa kondisi tersebut bagus.

Ketidakseimbangan tersebut bisa menjadi masalah :

- Bias model : model machine learning cenderung lebih fokus pada kategori yang lebih dominan (dalam hal ini, label 0 - Tidak Depresi) karena lebih banyak data yang tersedia untuk kategori tersebut. Hal ini bisa membuat model kurang efektif dalam mengidentifikasi individu yang benar-benar mengalami depresi (label 1 - Ya Depresi).
- Evaluasi Tidak Akurat : Jika model memprediksi sebagian besar label sebagai "Tidak Depresi" (0), akurasi mungkin sangat tinggi (karena sebagian besar data memang tidak depresi), tetapi ini bukan indikasi bahwa model efektif dalam mendeteksi depresi. Dalam situasi ini, akurasi saja tidak cukup perlu menggunakan metrik lain seperti precision, recall, dan F1-score untuk menilai seberapa baik model dalam mengenali kelas minoritas (individu yang mengalami depresi).

4. Visualisasi Distribusi Numerikal dan Kategorikal

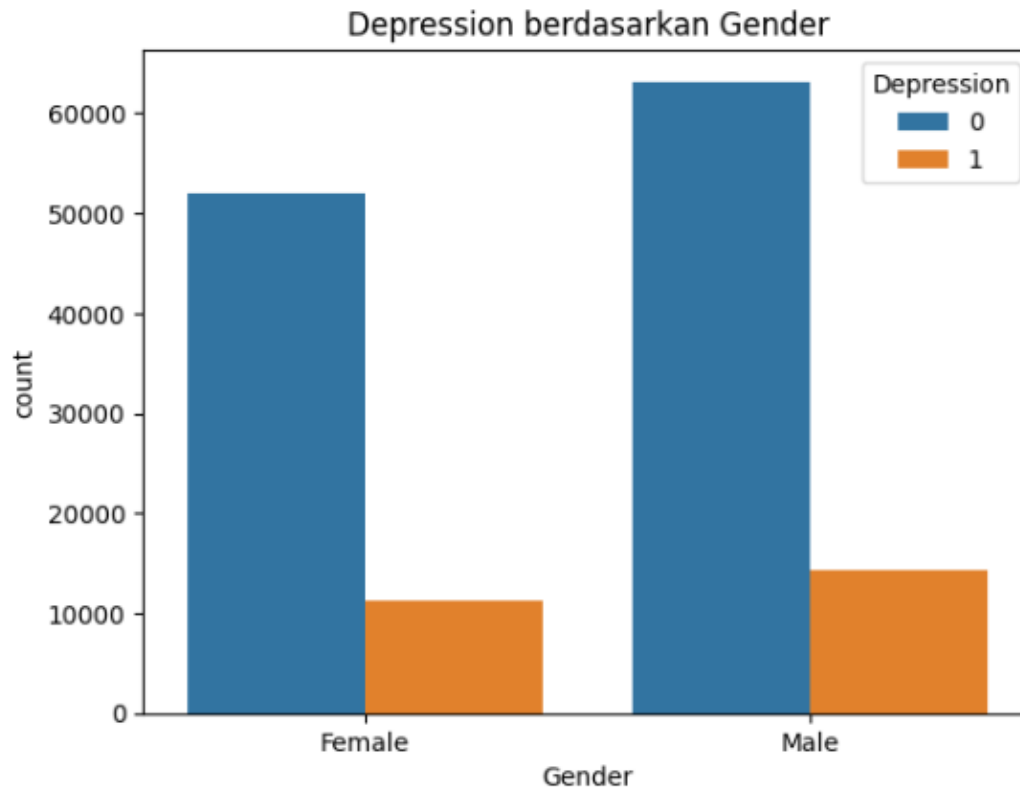
```
[ ] # Distribusi umur
sns.histplot(train['Age'], bins=20, kde=True)
plt.title('Distribusi Usia')
plt.xlabel('Age')
plt.show()
```



Hasil visualisasi menunjukkan bahwa data usia tersebar cukup merata pada rentang usia 18 hingga 60 tahun, dengan puncak jumlah data berada di usia sekitar 18-20 dan 55-60 tahun. Kurva KDE memberikan gambaran halus tentang tren distribusi, memperlihatkan bahwa terdapat peningkatan jumlah data pada usia menengah hingga lanjut.

- Visualisasi distribusi depresi berdasarkan gender

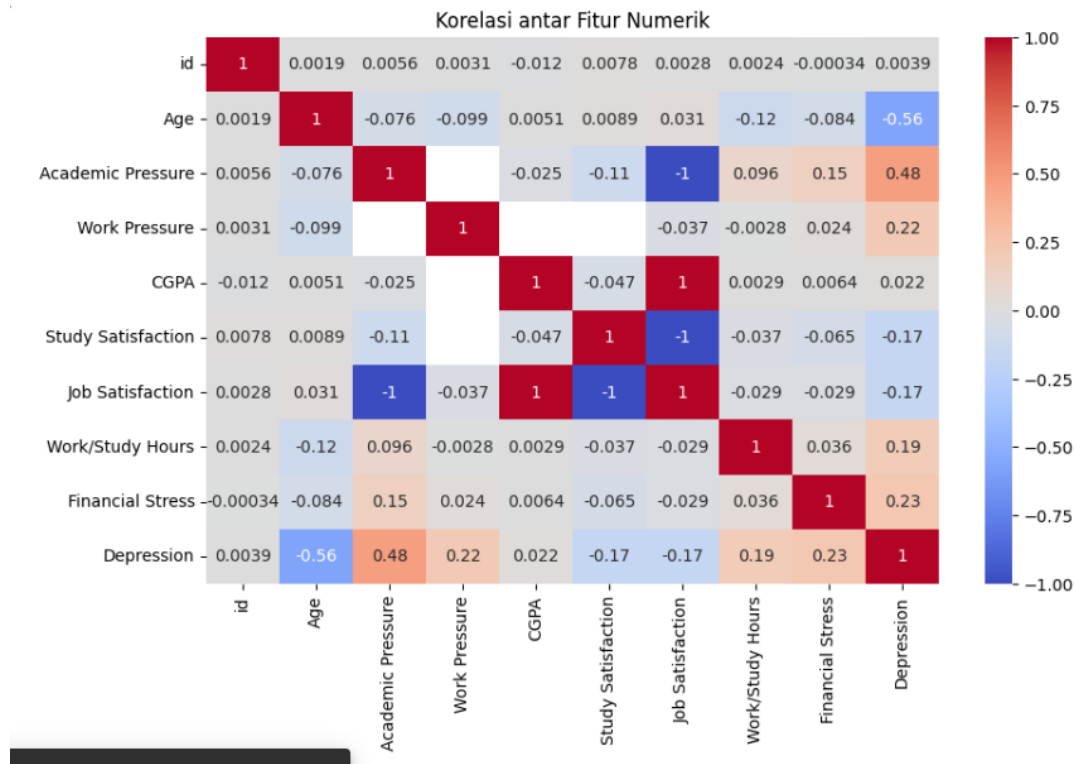
```
[ ] # Distribusi Gender vs Depression
sns.countplot(x='Gender', hue='Depression', data=train)
plt.title('Depression berdasarkan Gender')
plt.show()
```



- a. Distribusi Gender: Grafik ini menunjukkan bahwa jumlah individu yang tidak mengalami depresi lebih banyak pada kedua gender (perempuan dan laki-laki).
- b. Perbandingan Depresi Berdasarkan Gender: Meskipun lebih sedikit perempuan yang mengalami depresi dibandingkan laki-laki, laki-laki yang mengalami depresi relatif lebih banyak. Ini bisa memberi wawasan bahwa depresi mungkin lebih jarang terjadi pada perempuan secara keseluruhan, namun proporsinya lebih tinggi pada laki-laki dalam konteks ini.

5. Korelasi antar Fitur Numerik

```
corr = train.corr(numeric_only=True)
plt.figure(figsize=(10, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Korelasi antar Fitur Numerik')
plt.show()
```



Visualisasi heatmap pada fungsi ini digunakan untuk menunjukkan hubungan korelasi antar fitur numerik dalam dataset `train`. Korelasi dihitung menggunakan metode Pearson melalui fungsi `train.corr(numeric_only=True)`, yang menghasilkan matriks korelasi antar semua fitur numerik. Heatmap kemudian divisualisasikan dengan bantuan `seaborn.heatmap()`, di mana parameter `annot=True` digunakan untuk menampilkan nilai korelasi dalam masing-masing sel, dan `cmap='coolwarm'` memberikan gradasi warna dari biru (negatif) ke merah (positif).

Dari hasil heatmap yang ditampilkan dapat dilihat bahwa :

- Fitur Depression memiliki korelasi negatif cukup kuat dengan Age (-0.56), yang menunjukkan bahwa semakin muda usia responden, kecenderungan mengalami depresi lebih tinggi.
- Korelasi positif paling tinggi terhadap Depression berasal dari Academic Pressure (0.48), diikuti oleh Work Pressure (0.22) dan Financial Stress (0.23). Hal ini mengindikasikan bahwa tekanan akademik, pekerjaan, dan keuangan memiliki kontribusi terhadap tingkat depresi.
- Terdapat korelasi negatif antara Job Satisfaction dan beberapa variabel seperti Academic Pressure dan Depression, yang menyiratkan bahwa tingkat kepuasan kerja yang rendah dapat berkaitan dengan tekanan dan kondisi psikologis yang buruk.

```

▶ from scipy.stats import chi2_contingency

def cramers_v(x, y):
    confusion_matrix = pd.crosstab(x, y)
    chi2 = chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    phi2 = chi2 / n
    r, k = confusion_matrix.shape
    phi2corr = max(0, phi2 - ((k - 1)*(r - 1))/(n - 1))
    rcorr = r - ((r - 1)**2)/(n - 1)
    kcorr = k - ((k - 1)**2)/(n - 1)
    return np.sqrt(phi2corr / min((kcorr - 1), (rcorr - 1)))

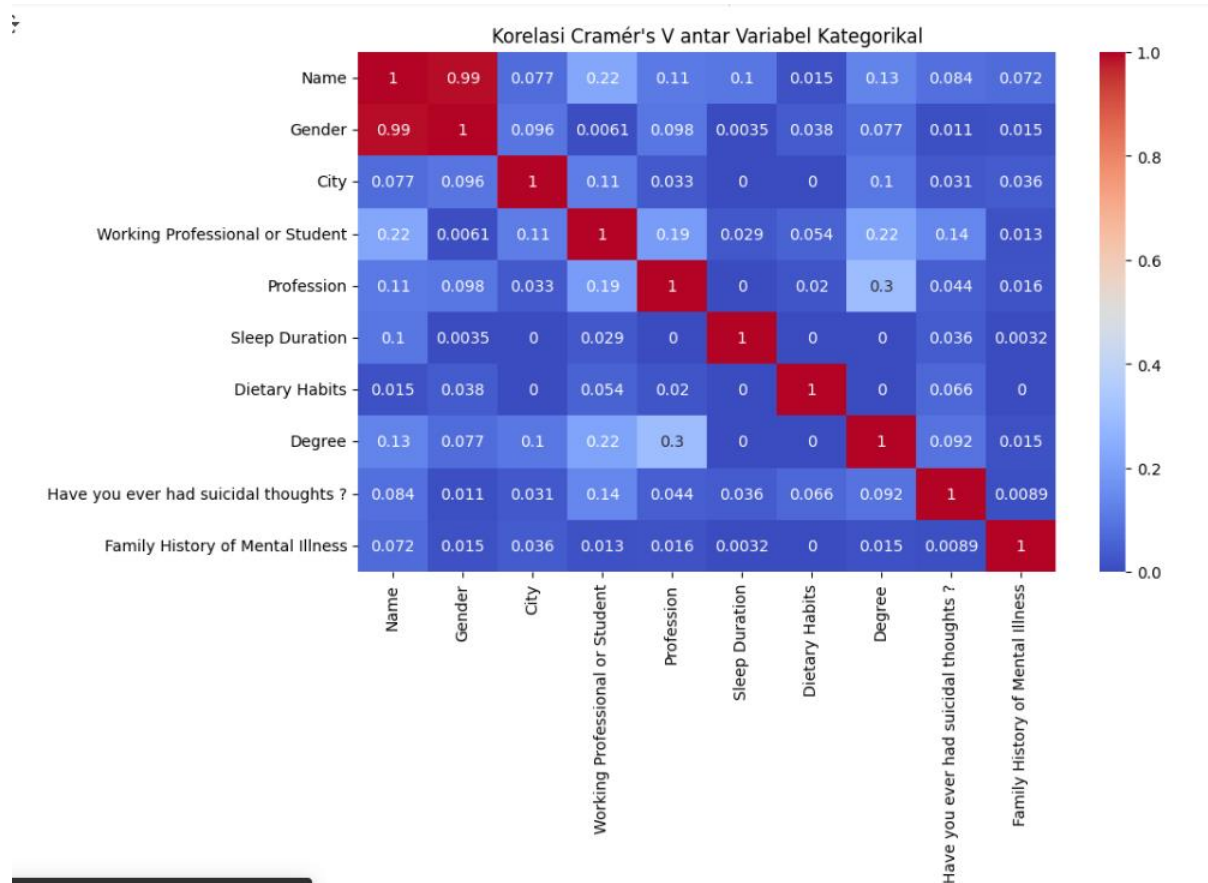
# Ambil kolom kategorikal
cat_cols = train.select_dtypes(include=['object', 'category']).columns
cramers_matrix = pd.DataFrame(index=cat_cols, columns=cat_cols)

# Hitung korelasi Cramér's V untuk semua pasangan
for col1 in cat_cols:
    for col2 in cat_cols:
        crammers_matrix.loc[col1, col2] = cramers_v(train[col1], train[col2])

# Konversi ke float
cramers_matrix = crammers_matrix.astype(float)

# Tampilkan heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(cramers_matrix, annot=True, cmap='coolwarm', vmin=0, vmax=1)
plt.title("Korelasi Cramér's V antar Variabel Kategorikal")
plt.show()

```



Penjelasan :

- Fungsi Cramér's V:

 - Fungsi `cramers_v(x, y)` menghitung asosiasi antara dua variabel kategorikal `x` dan `y`.
 - Fungsi ini pertama-tama membuat tabel kontingensi antara `x` dan `y` menggunakan `pd.crosstab()`.
 - Kemudian, uji Chi-kuadrat (`chi2_contingency()`) diterapkan untuk menghitung statistik Chi-kuadrat, yang selanjutnya dinormalisasi untuk mendapatkan nilai Cramér's V.
- Iterasi melalui kolom-kolom kategorikal:

 - Variabel `cat_cols` memilih semua kolom kategorikal dalam dataset pelatihan.
 - Kode ini kemudian melakukan iterasi pada semua pasangan kolom kategorikal, menghitung nilai Cramér's V untuk setiap pasangan dan menyimpan hasilnya dalam matriks.
- Visualisasi:

 - Hasilnya divisualisasikan menggunakan heatmap yang dibuat dengan `seaborn.heatmap()`. Skala warna menunjukkan kekuatan asosiasi antara variabel, dengan warna merah menunjukkan korelasi yang lebih kuat dan warna biru menunjukkan korelasi yang lebih lemah.
 - Korelasi tinggi (dekat dengan 1) menunjukkan bahwa dua variabel tersebut sangat berkaitan. Sebagai contoh, variabel seperti Name dan Gender memiliki korelasi yang sangat tinggi (Cramér's V ~ 0.99), yang menunjukkan bahwa nama kemungkinan terkait dengan gender dalam dataset.
 - Korelasi rendah (dekat dengan 0) menunjukkan sedikit atau tidak ada hubungan antara kedua variabel. Misalnya, Sleep Duration dan Dietary Habits mungkin memiliki sedikit atau tidak ada asosiasi berdasarkan nilai korelasi tersebut.

6. Missing Values pada setiap kolom dalam dataset train

```
def nullpercent(df):  
    value=(df.isnull().sum()/df.shape[0])*100  
    print(value)
```

```
▶ nullpercent(train)
```

```
id                0.000000  
Name              0.000000  
Gender            0.000000  
Age               0.000000  
City              0.000000  
Working Professional or Student  0.000000  
Profession        26.034115  
Academic Pressure  80.172708  
Work Pressure      19.842217  
CGPA               80.171997  
Study Satisfaction  80.172708  
Job Satisfaction   19.836532  
Sleep Duration     0.000000  
Dietary Habits     0.002843  
Degree            0.001421  
Have you ever had suicidal thoughts ?  0.000000  
Work/Study Hours   0.000000  
Financial Stress   0.002843  
Family History of Mental Illness  0.000000  
Depression         0.000000  
dtype: float64
```

Fungsi ini menggunakan `df.isnull().sum()` untuk menghitung jumlah nilai yang hilang di setiap kolom dalam dataset train.

Beberapa kolom, seperti Profession, Job Satisfaction, dan beberapa lainnya, memiliki missing values. Kolom dengan persentase tinggi nilai hilang seperti Study Satisfaction, Academic Pressure perlu diperhatikan lebih lanjut karena bisa mempengaruhi analisis data dan model yang akan dibangun.

- Academic Pressure


```
[ ] # Include NaN values, zeros, and negatives in value counts
train['Academic Pressure'].value_counts(dropna=False)
```



	count
Academic Pressure	
NaN	112803
3.0	7463
5.0	6296
4.0	5158
1.0	4801
2.0	4179

dtype: int64

- CGPA

```
train['CGPA'].value_counts(dropna=False)
```

	count
CGPA	
NaN	112802
8.0400	822
9.9600	425
5.7400	410
8.9500	371
...	...
6.6400	1
7.0625	1
6.9800	1
6.4400	1
6.0900	1

332 rows × 1 columns

dtype: int64

- Study Satisfaction

```
train['Study Satisfaction'].value_counts(dropna=False)
```



	count
Study Satisfaction	
NaN	112803
4.0	6360
2.0	5840
3.0	5823
1.0	5451
5.0	4423

dtype: int64

- Job Satisfaction



```
train['Job Satisfaction'].value_counts(dropna=False)
```

	count
Job Satisfaction	
NaN	27910
2.0	24783
5.0	22812
1.0	22324
3.0	21951
4.0	20920

dtype: int64

- Profession



```
train['Profession'].value_counts(dropna=False)
```



count	
Profession	
NaN	36630
Teacher	24906
Content Writer	7814
Architect	4370
Consultant	4229
...	...
Moderate	1
Analyst	1
Pranav	1
Visakhapatnam	1
Yuvraj	1

65 rows x 1 columns

dtype: int64

- Working Professional or Student

```
[ ] train['Working Professional or Student'].value_counts(dropna=False)
```



count	
Working Professional or Student	
Working Professional	112799
Student	27901

dtype: int64

- Sleep Duration

```
[ ] train['Sleep Duration'].value_counts()
```



count

Sleep Duration

Less than 5 hours	38784
7-8 hours	36969
More than 8 hours	32726
5-6 hours	32142
3-4 hours	12
6-7 hours	8
4-5 hours	7
4-6 hours	5
2-3 hours	5
6-8 hours	4
No	4
1-6 hours	4
10-11 hours	2
9-11 hours	2
8-9 hours	2
Sleep_Duration	2
Unhealthy	2

45	2
40-45 hours	1
1-2 hours	1
1-3 hours	1
9-6 hours	1
55-66 hours	1
Moderate	1
35-36 hours	1
8 hours	1
10-6 hours	1
Indore	1
than 5 hours	1
49 hours	1
Work_Study_Hours	1
3-6 hours	1
45-48 hours	1
9-5	1
Pune	1
9-5 hours	1

dtype: int64

Tujuan dari analisis ini adalah untuk memahami distribusi durasi tidur responden serta mengidentifikasi kemungkinan adanya entri data yang tidak konsisten atau tidak relevan.

Hasil dari perintah tersebut menunjukkan bahwa mayoritas responden mencatatkan durasi tidur yang tergolong dalam tiga kelompok utama:

- Less than 5 hours sebanyak 38.784 data,
- 7–8 hours sebanyak 36.969 data,
- More than 8 hours sebanyak 32.726 data.

Terdapat juga kategori lain seperti 5–6 hours (32.142 data), yang masih cukup umum, namun setelah itu mulai muncul berbagai nilai yang tidak konsisten, seperti 3–4 hours, 10–11 hours bahkan nilai tidak bermakna seperti Sleep_Duration, Unhealthy, Pune, Work_Study_Hours dan Moderate. Nilai-nilai ini kemungkinan besar merupakan hasil kesalahan input atau kolom lain yang tercampur ke dalam kolom ini.

Dengan melihat hasil ini, kita dapat menyimpulkan bahwa data pada kolom `Sleep Duration` perlu dilakukan pembersihan (data cleaning), seperti menyatukan kategori yang sejenis, menghapus nilai yang tidak relevan, serta mungkin mengubah semua entri menjadi format standar (durasi dalam satuan angka).

BAB 3

DATA PREPARATION

Pada data preparation ini, data mentah yang sudah diekplorasi atau dilakukan nya EDA (Exploratory Data Analysis) diubah menjadi format siap pakai untuk model.

3.1 Data Selection

Pada data selection ini, memilih kolom-kolom yang relevan untuk model dari dataset pelatihan dan pengujian seperti 'Gender', 'Age', 'City', 'Working Professional or Student', 'Profession', 'Academic Pressure', 'Work Pressure', 'Study Satisfaction', 'Job Satisfaction', 'Numeric Sleep Duration', 'Dietary Habits', 'Degree', 'Have you ever had suicidal thoughts ?', 'Work/Study Hours', 'Financial Stress', 'Family History of Mental Illness'

Kolom target Depression telah dipisahkan dari data pelatihan (y_train), sementara fitur lainnya digunakan untuk pelatihan (X_train).

1. Data Selection

```
[195] # Step 1: Data Selection
      # List of all relevant columns (without including target 'Depression')
      relevant_columns = ['Gender', 'Age', 'City', 'Working Professional or Student',
                          'Profession', 'Academic Pressure', 'Work Pressure',
                          'Study Satisfaction', 'Job Satisfaction', 'Numeric Sleep Duration',
                          'Dietary Habits', 'Degree', 'Have you ever had suicidal thoughts ?',
                          'Work/Study Hours', 'Financial Stress', 'Family History of Mental Illness']

      # Selecting relevant columns from both train and test data
      train_selected = train[relevant_columns + ['Depression']] # Including 'Depression' in training data
      test_selected = test[relevant_columns] # Only selecting features (no 'Depression')

      # Verify that the train and test data have the same columns
      print("Train columns:", train_selected.columns)
      print("Test columns:", test_selected.columns)

      train_selected.head(), test_selected.head()
```

```

Train columns: Index(['Gender', 'Age', 'City', 'Working Professional or Student',
'Profession', 'Academic Pressure', 'Work Pressure',
'Study Satisfaction', 'Job Satisfaction', 'Numeric Sleep Duration',
'Dietary Habits', 'Degree', 'Have you ever had suicidal thoughts ?',
'Work/Study Hours', 'Financial Stress',
'Family History of Mental Illness', 'Depression'],
dtype='object')

```

```

Test columns: Index(['Gender', 'Age', 'City', 'Working Professional or Student',
'Profession', 'Academic Pressure', 'Work Pressure',
'Study Satisfaction', 'Job Satisfaction', 'Numeric Sleep Duration',
'Dietary Habits', 'Degree', 'Have you ever had suicidal thoughts ?',
'Work/Study Hours', 'Financial Stress',
'Family History of Mental Illness'],
dtype='object')

```

```

(  Gender  Age      City Working Professional or Student \
0      0  49.0      Ludhiana      Working Professional
1      1  26.0      Varanasi      Working Professional
2      1  33.0  Visakhapatnam      Student
3      1  22.0      Mumbai      Working Professional
4      0  30.0      Kanpur      Working Professional

```

```

      Profession  Academic Pressure  Work Pressure  Study Satisfaction \
0      Chef      0.0      5.0      0.0
1      Teacher      0.0      4.0      0.0
2      other      5.0      0.0      2.0
3      Teacher      0.0      5.0      0.0
4  Business Analyst      0.0      1.0      0.0

```

```

      Job Satisfaction  Numeric Sleep Duration  Dietary Habits  Degree \
0      2.0      8.0      Healthy      BHM
1      3.0      5.0      Unhealthy      LLB
2      0.0      5.5      Healthy      B.Pharm
3      1.0      5.0      Moderate      BBA
4      1.0      5.5      Unhealthy      BBA

```

```

      Have you ever had suicidal thoughts ?  Work/Study Hours  Financial Stress \
0      0      1.0      2.0
1      1      7.0      3.0
2      1      3.0      1.0
3      1      10.0      1.0
4      1      9.0      4.0

```

```

      Family History of Mental Illness  Depression
0      0      0
1      0      1
2      0      1

```

```

      Family History of Mental Illness  Depression
0      0      0
1      0      1
2      0      1
3      1      1
4      1      0 ,

Gender  Age      City Working Professional or Student \
0      1  53.0  Visakhapatnam      Working Professional
1      0  58.0      Kolkata      Working Professional
2      1  53.0      Jaipur      Working Professional
3      0  23.0      Rajkot      Student
4      1  47.0      Kalyan      Working Professional

```

```

      Profession  Academic Pressure  Work Pressure \
0      Judge      0.0      2.0
1  Educational Consultant      0.0      2.0
2      Teacher      0.0      4.0
3      other      5.0      0.0
4      Teacher      0.0      5.0

```

```

      Study Satisfaction  Job Satisfaction  Numeric Sleep Duration \
0      0.0      5.0      5.0
1      0.0      4.0      5.0
2      0.0      1.0      7.5
3      1.0      0.0      8.0
4      0.0      5.0      7.5

```

```

      Dietary Habits  Degree  Have you ever had suicidal thoughts ? \
0      Moderate      LLB      0
1      Moderate      B.Ed      0
2      Moderate      B.Arch      1
3      Moderate      BSc      1
4      Moderate      BCA      1

```

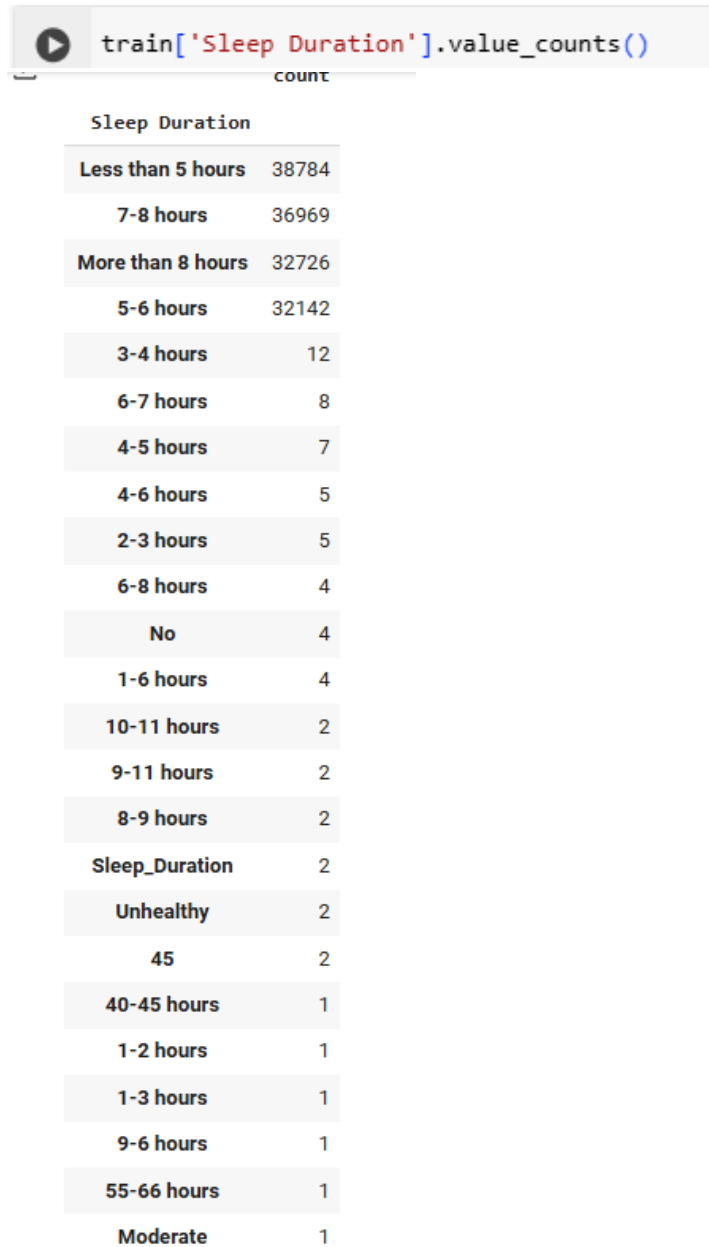
```

      Work/Study Hours  Financial Stress  Family History of Mental Illness
0      9.0      3.0      1
1      6.0      4.0      0
2      12.0      4.0      0
3      10.0      4.0      0
4      3.0      4.0      0 )

```


3.2 Data Cleaning

- Pada Sleep Duration



```
train['Sleep Duration'].value_counts()
```

Sleep Duration	count
Less than 5 hours	38784
7-8 hours	36969
More than 8 hours	32726
5-6 hours	32142
3-4 hours	12
6-7 hours	8
4-5 hours	7
4-6 hours	5
2-3 hours	5
6-8 hours	4
No	4
1-6 hours	4
10-11 hours	2
9-11 hours	2
8-9 hours	2
Sleep_Duration	2
Unhealthy	2
45	2
40-45 hours	1
1-2 hours	1
1-3 hours	1
9-6 hours	1
55-66 hours	1
Moderate	1

Penjelasan :

Fungsi `train['Sleep Duration'].value_counts()` digunakan untuk menghitung jumlah kemunculan masing-masing nilai dalam kolom 'Sleep Duration'. Tujuan dari analisis ini adalah untuk memahami distribusi durasi tidur responden serta mengidentifikasi kemungkinan adanya entri data yang tidak konsisten atau tidak relevan. Dengan melihat hasil ini, kita dapat menyimpulkan bahwa data pada kolom 'Sleep Duration' perlu dilakukan pembersihan (data cleaning), seperti menyatukan kategori yang sejenis, menghapus nilai yang tidak relevan, serta mungkin mengubah semua entri menjadi format standar (misalnya, durasi dalam satuan jam berbentuk angka).

```
[29] import re

def extract_numeric_duration(value):
    if isinstance(value, str):
        # Extract numeric ranges or single values
        match = re.findall(r'\d+\.\d*', value) # Find all numbers
        if len(match) == 1:
            return float(match[0]) # Single numeric value
        elif len(match) == 2:
            # Average for a range (e.g., 7-8 hours → (7+8)/2)
            return (float(match[0]) + float(match[1])) / 2
        return np.nan # Set non-numeric entries to NaN
```

Penjelasan :

Fungsi `extract_numeric_duration(value)` bertujuan untuk membersihkan dan mengonversi data dari kolom 'Sleep Duration' yang berisi nilai dalam bentuk string seperti "7-8 hours" atau "Less than 5 hours" menjadi format numerik agar lebih mudah dianalisis. Fungsi ini menggunakan pustaka `re` (regular expression) untuk mengekstraksi angka dari string tersebut. Pertama, fungsi akan memeriksa apakah input merupakan string. Jika ya, maka akan dilakukan pencarian semua angka dalam string tersebut menggunakan pola `r'\d+\.\d*'` — yang menangkap angka bulat maupun desimal.

Jika hanya ditemukan satu angka (misalnya "5 hours"), maka nilai tersebut dikembalikan dalam bentuk `float`. Namun, jika ditemukan dua angka (misalnya "7-8 hours"), maka fungsi akan mengembalikan rata-rata dari kedua angka tersebut, yaitu $(7+8)/2 = 7.5$. Hal ini berguna untuk merepresentasikan rentang waktu sebagai satu nilai estimasi tunggal.

Untuk entri yang tidak mengandung angka sama sekali atau berisi kata-kata yang tidak relevan seperti "Unhealthy" atau "Sleep_Duration", maka fungsi akan mengembalikan `np.nan` sebagai penanda nilai kosong.

```
[30] # Apply function to column
train['Numeric Sleep Duration'] = train['Sleep Duration'].apply(extract_numeric_duration)
```

Penjelasan :

Fungsi `train['Numeric Sleep Duration'] = train['Sleep Duration'].apply(extract_numeric_duration)` digunakan untuk menerapkan fungsi `extract_numeric_duration` pada setiap baris di kolom 'Sleep Duration' pada DataFrame `train`. Fungsi ini bertujuan untuk mengekstrak nilai numerik dari data yang awalnya berbentuk teks seperti "7-8 hours" atau "Less than 5 hours". Hasil dari proses ini adalah kolom baru bernama 'Numeric Sleep Duration' yang berisi nilai durasi tidur dalam format numerik (float).

train.head(30)

	id	Name	Gender	Age	City	Working Professional or Student	Profession	Academic Pressure	Work Pressure	CPA	...	Satisfaction	Job Duration	Sleep Duration	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	History of Mental Illness	Depression	Numeric Sleep Duration
0	0	Aaradhya	Female	49.0	Ludhiana	Working Professional	Chef	NaN	5.0	NaN	...	2.0	More than 8 hours	Healthy	BHM	No	1.0	2.0	No	0	8.0	
1	1	Vivan	Male	26.0	Varanasi	Working Professional	Teacher	NaN	4.0	NaN	...	3.0	Less than 5 hours	Unhealthy	LLB	Yes	7.0	3.0	No	1	5.0	
2	2	Yuvraj	Male	33.0	Visakhapatnam	Student	NaN	5.0	NaN	8.97	...	NaN	5-6 hours	Healthy	B.Pharm	Yes	3.0	1.0	No	1	5.5	
3	3	Yuvraj	Male	22.0	Mumbai	Working Professional	Teacher	NaN	5.0	NaN	...	1.0	Less than 5 hours	Moderate	BBA	Yes	10.0	1.0	Yes	1	5.0	
4	4	Rhea	Female	30.0	Kanpur	Working Professional	Business	NaN	1.0	NaN	...	1.0	5-6 hours	Unhealthy	BBA	Yes	9.0	4.0	Yes	0	5.5	

Penjelasan :

Menampilkan data set pada `train` yang sudah bertambah kolom `Numeric Sleep Duration` yang telah diterjemahkan menjadi angka.

```
train['Numeric Sleep Duration'].value_counts(dropna=False)
```

	count
Numeric Sleep Duration	
5.0	38790
7.5	36970
8.0	32728
5.5	32142
3.5	16
NaN	12
6.5	8
4.5	8
7.0	6
2.5	5
8.5	2
10.0	2
10.5	2
45.0	2
1.5	1
42.5	1
60.5	1
2.0	1
35.5	1
49.0	1
46.5	1

dtype: int64

Penjelasan :

Fungsi `train['Numeric Sleep Duration'].value_counts(dropna=False)` digunakan untuk menghitung jumlah kemunculan setiap nilai unik pada kolom `'Numeric Sleep Duration'`, termasuk nilai yang hilang (`NaN`). Kolom ini merupakan hasil transformasi dari kolom `'Sleep Duration'` yang sebelumnya berisi data dalam format teks, dan telah diubah menjadi format numerik menggunakan fungsi `'extract_numeric_duration'`.

```
[33] # Exclude values greater than 10 hours and less than 4.5 hours for mean calculation
      valid_values = train[(train['Numeric Sleep Duration'] <= 10) & (train['Numeric Sleep Duration'] >= 4.5)]

      # Calculate mean of valid values
      mean_value = valid_values['Numeric Sleep Duration'].mean()

      # Replace values where Numeric Sleep Duration > 10 or < 4.5 with the calculated mean
      train['Numeric Sleep Duration'] = train['Numeric Sleep Duration'].apply(
          lambda x: mean_value if x > 10 or x < 4.5 else x
      )
```


Penjelasan :

Fungsi di atas digunakan untuk membersihkan nilai outlier pada kolom `'Numeric Sleep Duration'`. Nilai-nilai yang dianggap tidak valid, yaitu durasi tidur lebih dari 10 jam atau kurang dari 4.5 jam,

dikecualikan dari perhitungan nilai rata-rata. Hal ini dilakukan dengan membuat subset `valid_values` yang hanya mencakup durasi tidur antara 4.5 hingga 10 jam. Kemudian, nilai rata-rata dari subset ini dihitung dan disimpan ke dalam variabel `mean_value`.

Langkah berikutnya adalah mengganti nilai-nilai yang berada di luar rentang valid (lebih dari 10 jam atau kurang dari 4.5 jam) dengan nilai rata-rata yang telah dihitung. Ini dilakukan menggunakan fungsi `.apply()` dengan `lambda`, di mana setiap nilai `x` akan digantikan oleh `mean_value` jika tidak berada dalam rentang yang diizinkan. Hasil dari proses ini adalah kolom `Numeric Sleep Duration` yang telah diperbaiki dari kemungkinan kesalahan input atau outlier ekstrem, sehingga lebih representatif untuk analisis selanjutnya.

```
[34] train['Numeric Sleep Duration'].value_counts(dropna=False)
```



	count
Numeric Sleep Duration	
5.000000	38790
7.500000	36970
8.000000	32728
5.500000	32142
6.469664	32
NaN	12
6.500000	8
4.500000	8
7.000000	6
8.500000	2
10.000000	2

dtype: int64

Penjelasan :

Fungsi `train['Numeric Sleep Duration'].value_counts(dropna=False)` digunakan untuk melihat distribusi nilai pada kolom `Numeric Sleep Duration` setelah proses pembersihan dilakukan. Parameter `dropna=False` memastikan bahwa nilai `NaN` (kosong) juga ditampilkan dalam hasil perhitungan.

Hasil pada gambar menunjukkan bahwa sebagian besar peserta memiliki durasi tidur yang berkisar antara 5 hingga 8 jam, dengan nilai yang paling umum adalah 5.0 jam (38.790 data), 7.5 jam (36.970 data), dan 8.0 jam (32.728 data). Terlihat juga adanya nilai baru, yaitu 6.469664, yang merupakan hasil dari penggantian outlier dengan nilai rata-rata durasi tidur valid yang telah dihitung sebelumnya. Nilai ini muncul sebanyak 32 kali, menunjukkan jumlah data yang sebelumnya memiliki nilai ekstrem (kurang dari 4.5 jam atau lebih dari 10 jam). Adanya nilai `NaN` (sebanyak 12) juga mengindikasikan bahwa beberapa data tidak dapat dikonversi ke format numerik selama proses ekstraksi dari kolom `Sleep Duration`.

```
[35] # Calculate the mean of the column, ignoring NaN values
mean_value = train['Numeric Sleep Duration'].mean()


# Replace NaN values with the calculated mean
train['Numeric Sleep Duration'] = train['Numeric Sleep Duration'].fillna(mean_value)
```

Penjelasan :

Disini kami menggunakannya untuk menangani nilai hilang (`NaN`) pada kolom `Numeric Sleep Duration`. Langkah pertama adalah menghitung nilai rata-rata dari kolom tersebut dengan fungsi `.mean()`, yang secara default mengabaikan nilai `NaN`. Nilai rata-rata ini disimpan dalam variabel

`mean_value`. Langkah berikutnya adalah mengganti seluruh nilai `NaN` pada kolom `Numeric Sleep Duration` dengan nilai rata-rata yang telah dihitung sebelumnya. Proses ini dilakukan menggunakan metode `.fillna(mean_value)`. Tujuan utama dari proses ini adalah untuk menjaga konsistensi data dan memastikan bahwa tidak ada nilai hilang yang dapat mengganggu proses analisis atau pemodelan.

```
[36] train['Numeric Sleep Duration'].value_counts(dropna=False)
```



	count
Numeric Sleep Duration	
5.000000	38790
7.500000	36970
8.000000	32728
5.500000	32142
6.469664	32
6.469664	12
6.500000	8
4.500000	8
7.000000	6
8.500000	2
10.000000	2

dtype: int64

Penjelasan :

Fungsi `train['Numeric Sleep Duration'].value_counts(dropna=False)` digunakan untuk melihat distribusi frekuensi dari nilai-nilai pada kolom `Numeric Sleep Duration` setelah dilakukan proses pembersihan dan imputasi data. Dengan menyertakan parameter `dropna=False`, kita memastikan bahwa nilai `NaN` juga akan ditampilkan jika masih ada. Hasil pada gambar menunjukkan bahwa nilai `NaN` sudah tidak muncul lagi, yang berarti proses penggantian nilai `NaN` dengan rata-rata telah berhasil. Selain itu, terlihat bahwa sebagian besar data memiliki durasi tidur sebesar 5.0, 7.5, 8.0, dan 5.5 jam. Munculnya nilai seperti `6.469664` sebanyak 32 kali menunjukkan nilai rata-rata yang digunakan untuk menggantikan data yang sebelumnya tidak valid (kurang dari 4.5 atau lebih dari 10 jam), yang juga sudah berhasil diterapkan.

```
[ ] # Replace values where Numeric Sleep Duration > 10 or < 4.5 with the calculated mean
test['Numeric Sleep Duration'] = test['Numeric Sleep Duration'].apply(
    lambda x: mean_value if x > 10 or x < 4.5 else x
)
```

Penjelasan :

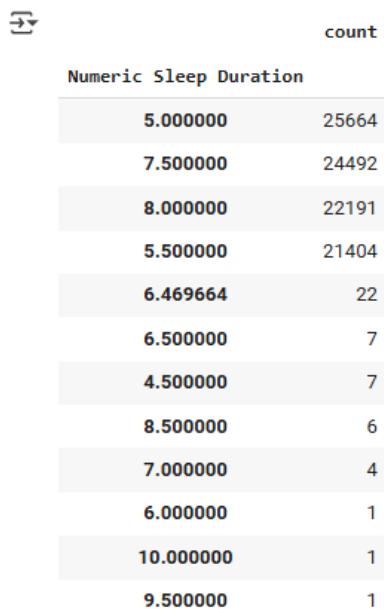
Fungsi yang digunakan `test['Numeric Sleep Duration'] = test['Numeric Sleep Duration'].apply(lambda x: mean_value if x > 10 or x < 4.5 else x)` bertujuan untuk membersihkan data dengan cara mengganti nilai durasi tidur yang dianggap tidak masuk akal (outlier) dengan nilai rata-rata yang telah dihitung sebelumnya (`mean_value`). Dalam konteks ini, nilai durasi tidur yang lebih dari 10 jam atau kurang dari 4.5 jam dianggap sebagai outlier. Nilai-nilai tersebut dapat terjadi karena kesalahan entri data atau kondisi yang tidak representatif secara umum. Dengan menerapkan fungsi `lambda`, setiap elemen pada kolom `Numeric Sleep Duration` diperiksa satu per satu. Jika nilainya di luar batas wajar tersebut, maka akan digantikan dengan `mean_value`. Jika masih dalam rentang normal (antara 4.5 hingga 10 jam), maka nilainya tetap dipertahankan. Hasil dari proses ini adalah pembersihan nilai-nilai ekstrem dalam data uji sehingga model yang akan dibangun nantinya tidak terpengaruh oleh data yang tidak wajar.

```
[ ] # Replace NaN values with the calculated mean
test['Numeric Sleep Duration'] = test['Numeric Sleep Duration'].fillna(mean_value)
```

Penjelasan :

Hasil dari proses ini dapat dilihat pada gambar, di mana sebelumnya terdapat beberapa nilai `NaN`, tetapi setelah penerapan kode ini, nilai `NaN` tersebut telah digantikan dengan nilai rata-rata (dalam contoh gambar terlihat sebagai 6.469664), sehingga nilai tersebut muncul beberapa kali (yaitu sebanyak jumlah `NaN` sebelumnya). Hal ini menunjukkan bahwa pengisian nilai yang hilang telah berhasil dilakukan.

```
test['Numeric Sleep Duration'].value_counts(dropna=False)
```



	count
5.000000	25664
7.500000	24492
8.000000	22191
5.500000	21404
6.469664	22
6.500000	7
4.500000	7
8.500000	6
7.000000	4
6.000000	1
10.000000	1
9.500000	1

dtype: int64

Penjelasan :

Dari hasil pada gambar, dapat dilihat bahwa sebagian besar nilai berada pada angka-angka umum seperti 5.0, 7.5, 8.0, dan 5.5, yang masing-masing muncul dalam puluhan ribu. Ini menunjukkan bahwa data tidur cenderung terkonsentrasi pada rentang waktu tidur yang wajar. Terlihat juga nilai `6.469664` muncul sebanyak 22 kali, yang mengindikasikan nilai ini adalah rata-rata yang digunakan untuk menggantikan data yang tidak valid (misalnya lebih dari 10 jam atau kurang dari 4.5 jam) atau data yang sebelumnya bernilai `NaN`. Fakta bahwa nilai `NaN` sudah tidak muncul lagi menunjukkan bahwa proses penggantian nilai hilang telah dilakukan dengan sukses.


```
[ ] # Drop the 'Sleep Duration' column from the DataFrame
train = train.drop(columns=['Sleep Duration'])
test = test.drop(columns=['Sleep Duration'])
```

Penjelasan :

Fungsi `.drop(columns=['Sleep Duration'])` berarti kita ingin menghapus kolom bernama 'Sleep Duration' dari DataFrame tersebut. Setelah kolom dihapus, hasilnya disimpan kembali ke variabel `train` dan `test`.

- Profession

```
[ ] train['Profession'].value_counts(dropna=False)
```



	count
Profession	
NaN	36630
Teacher	24906
Content Writer	7814
Architect	4370
Consultant	4229
...	...
Moderate	1
Analyst	1
Pranav	1
Visakhapatnam	1
Yuvraj	1

65 rows x 1 columns

dtype: int64

Penjelasan :

Ada nilai yang tidak masuk akal di kolom Profession. Nama orang: "Yuvraj", "Simran", "Manvi", "Pranav. Ini menunjukkan bahwa kolom Profession terkontaminasi oleh nilai-nilai dari kolom lain atau data yang salah format.


```
[ ] data = [
    "Student", "Academic", "Unemployed", "Profession", "Yogesh",
    "BCA", "MBA", "LLM", "PhD", np.nan, "Analyst", "Pranav",
    "Visakhapatnam", "M.Ed", "Moderate", "Nagpur", "B.Ed",
    "Unveil", "BBA", "MBBS", "Working Professional", "Medical Doctor",
    "City Manager", "FamilyVirar", "Dev", "BE", "B.Com",
    "Family Consultant", "Yuvraj", "Patna", "Unhealthy", "Surat",
    "MD", "City Consultant", "No", "MCA", "Surgeon", "M.Tech",
    "Simran", "B.Pharm", "Name", "Samar", "Manvi", "24th",
    "ME", "3M", "M.Pharm"
]
```

Penjelasan :

Tujuannya adalah untuk membersihkan data kategori sebelum digunakan dalam analisis atau pelatihan model machine learning.

Daftar ini mencakup:

- Status pekerjaan: "Student", "Unemployed", "Academic", "Working Professional"
- Gelar pendidikan: "MBA", "PhD", "LLM", "MCA", "B.Com", "BCA", dll.
- Nama orang: "Yogesh", "Pranav", "Yuvraj", "Samar", "Manvi", "Simran"
- Nama kota/tempat: "Nagpur", "Visakhapatnam", "Patna", "Surat"
- Nilai tidak masuk akal: "3M", "No", "24th", "Moderate", "Name", "Unveil", "Unhealthy"
- Kategori aneh lainnya: "FamilyVirar", "Family Consultant", "City Manager", "City Consultant"



```
train['Profession'].value_counts(dropna=False)
```

	count
Profession	
other	36680
Teacher	24906
Content Writer	7814
Architect	4370
Consultant	4229
HR Manager	4022
Pharmacist	3893
Doctor	3255
Business Analyst	3161
Entrepreneur	2968
Chemist	2967
Chef	2862
Educational Consultant	2852
Data Scientist	2390
Researcher	2328
Lawyer	2212
Customer Support	2055
Marketing Manager	1976
Pilot	1913
Travel Consultant	1860

Penjelasan :

Semua nilai sudah dimasukkan kedalam kolom other sesuai dengan daftar yang sudah diberikan dan juga nilai yang NaN pada data train.

```
# Replace using apply with lambda
test["Profession"] = test["Profession"].apply(
    lambda x: 'other' if x in data or pd.isna(x) else x
)
```

Penjelasan :

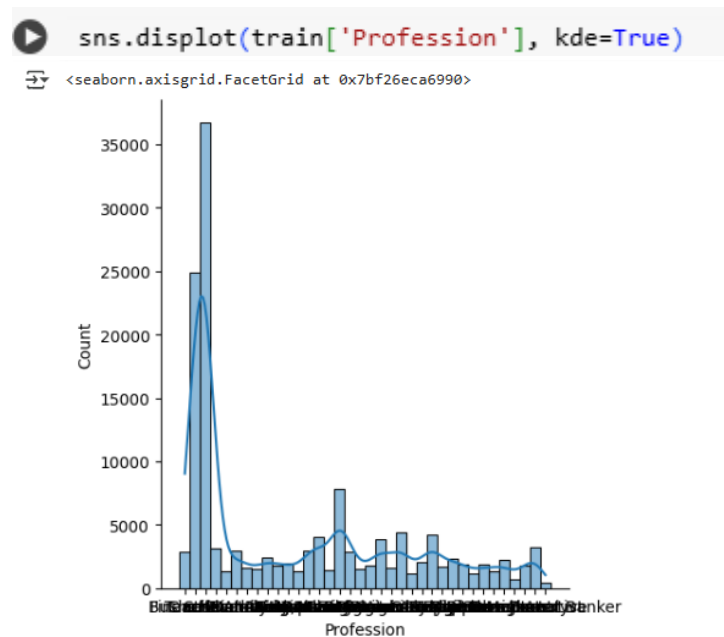
Telah digunakan untuk membersihkan kolom Profession di dataset test, dengan logika yang sama seperti dataset train.


```
test['Profession'].value_counts(dropna=False)
```

Profession	count
other	24676
Teacher	16385
Content Writer	5187
Architect	2982
Consultant	2920
Pharmacist	2656
HR Manager	2601
Doctor	2198
Business Analyst	2186
Chemist	1967
Entrepreneur	1935
Chef	1844
Educational Consultant	1827
Data Scientist	1582
Lawyer	1497
Researcher	1496
Pilot	1448
Customer Support	1422
Marketing Manager	1284

Penjelasan :

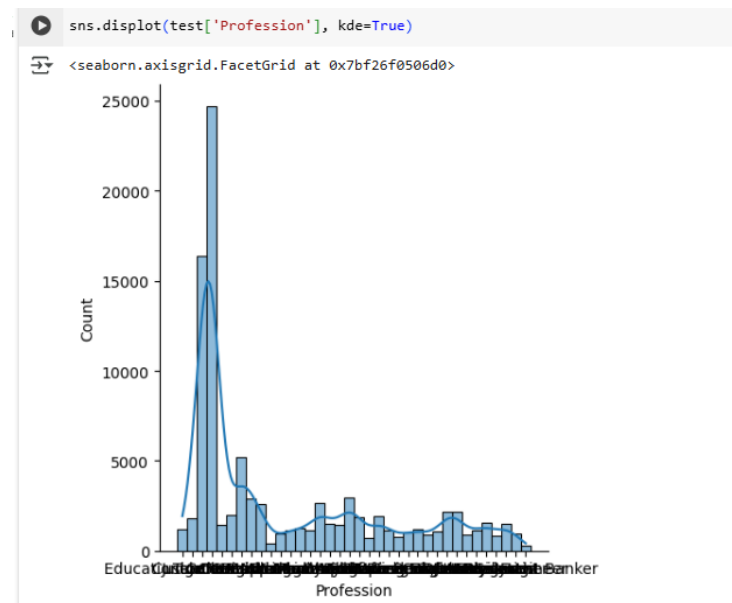
Semua nilai sudah dimasukkan kedalam kolom other sesuai dengan daftar yang sudah diberikan dan juga nilai yang NaN pada data test.



Penjelasan :

Puncak yang Sangat Tinggi: Ada satu kategori profesi yang memiliki jumlah frekuensi yang sangat tinggi (lebih dari 35.000), yang kemungkinan besar merupakan nilai yang paling sering muncul atau kategori yang paling dominan dalam dataset ini.

Distribusi Lebih Rata: Selain profesi dengan frekuensi tinggi, terlihat bahwa ada banyak kategori profesi lainnya yang memiliki frekuensi yang lebih rendah. Ini menunjukkan bahwa dataset mencakup beragam profesi, namun Sebagian besar data terkonsentrasi pada beberapa profesi tertentu.



Penjelasan :

Puncak yang Sangat Tinggi: Ada satu profesi yang sangat mendominasi jumlah data, terlihat pada puncak yang sangat tinggi. Ini menunjukkan bahwa satu profesi mungkin memiliki jumlah entri yang sangat besar dalam dataset test. Sebagai contoh, mungkin profesi yang sangat sering muncul di dataset ini seperti "Teacher" atau "Software Engineer". **Distribusi Lain Lebih Rata:** Profesi lainnya tersebar lebih merata, dengan sebagian besar profesi lainnya memiliki frekuensi yang lebih rendah. Ini menunjukkan bahwa meskipun ada banyak variasi profesi, hanya sedikit profesi yang benar-benar mendominasi dataset. **Ketidakseimbangan Data:** Data menunjukkan ketidakseimbangan yang jelas, di mana satu atau dua kategori profesi memiliki jumlah yang sangat tinggi dibandingkan dengan kategori lainnya. Ini dapat menyebabkan masalah dalam analisis dan pemodelan, terutama jika model terlalu fokus pada kategori profesi yang dominan.

- **Pada Academic Pressure**

```
# Include NaN values, zeros, and negatives in value counts
train['Academic Pressure'].value_counts(dropna=False)
```

	count
Academic Pressure	
NaN	112803
3.0	7463
5.0	6296
4.0	5158
1.0	4801
2.0	4179

dtype: int64

Penjelasan :

Fungsi `train['Academic Pressure'].value_counts(dropna=False)` digunakan untuk menghitung frekuensi kemunculan setiap nilai unik dalam kolom `'Academic Pressure'`, termasuk nilai `'NaN'` (hilang), dengan `'dropna=False'`. Hal ini penting untuk memahami distribusi nilai serta seberapa banyak data yang hilang dalam fitur tersebut sebelum melakukan imputasi atau pembersihan data. Hasil dari perintah tersebut menunjukkan bahwa terdapat 112.803 entri yang bernilai `'NaN'`, yang berarti bagian besar dari kolom ini memiliki nilai yang hilang. Selain itu, nilai-nilai lainnya berkisar antara 1 hingga 5, yang kemungkinan merepresentasikan tingkat tekanan akademik dalam bentuk skala ordinal (misalnya dari rendah ke tinggi). Nilai yang paling sering muncul setelah `'NaN'` adalah:

- Nilai 3 sebanyak 7.463 kali
- Nilai 5 sebanyak 6.296 kali
- Nilai 4 sebanyak 5.158 kali

Distribusi ini menunjukkan bahwa sebagian besar responden mengalami tekanan akademik pada level menengah hingga tinggi.

```
[160] train[train['Academic Pressure'].isna()].head()
```

	id	Name	Gender	Age	City	Working Professional or Student	Profession	Academic Pressure	Work Pressure	CGPA	Study Satisfaction	Job Satisfaction	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Depression	Numeric Sleep Duration
0	0	Aaradhy	Female	49.0	Ludhiana	Working Professional	Chef	NaN	5.0	NaN	NaN	2.0	Healthy	BHM	No	1.0	2.0	No	0	8.0
1	1	Vivan	Male	26.0	Varanasi	Working Professional	Teacher	NaN	4.0	NaN	NaN	3.0	Unhealthy	LLB	Yes	7.0	3.0	No	1	5.0
3	3	Yuvraj	Male	22.0	Mumbai	Working Professional	Teacher	NaN	5.0	NaN	NaN	1.0	Moderate	BBA	Yes	10.0	1.0	Yes	1	5.0
4	4	Rhea	Female	30.0	Kanpur	Working Professional	Business Analyst	NaN	1.0	NaN	NaN	1.0	Unhealthy	BBA	Yes	9.0	4.0	Yes	0	5.5
5	5	Vani	Female	59.0	Ahmedabad	Working Professional	Financial Analyst	NaN	2.0	NaN	NaN	5.0	Healthy	MCA	No	7.0	5.0	No	0	5.5

Penjelasan :

- Perintah `.isna()`:

Fungsi `.isna()` digunakan untuk memeriksa apakah sebuah nilai di kolom `'Academic Pressure'` adalah `NaN` (Not a Number), yang berarti nilai tersebut hilang atau tidak tersedia.

Hasilnya adalah `True` untuk baris-baris yang memiliki nilai `NaN` di kolom `'Academic Pressure'`.

- Hasil Output:

Dataset train menunjukkan baris-baris di mana `'Academic Pressure'` memiliki nilai `NaN`. Kolom lain seperti `'Work Pressure'`, `'CGPA'`, dan `'Study Satisfaction'` juga memiliki nilai `NaN`, yang berarti data untuk kolom tersebut tidak tercatat atau tidak tersedia.

Beberapa kolom lainnya seperti 'Gender', 'Age', 'City', dan 'Profession' tidak memiliki nilai NaN, yang berarti data untuk kolom tersebut lengkap.

```
[161] train['Academic Pressure']=train['Academic Pressure'].fillna(0)
      test['Academic Pressure']=test['Academic Pressure'].fillna(0)
```

Penjelasan :

train['Academic Pressure']: Mengakses kolom 'Academic Pressure' dalam dataset train.

.fillna(0): Fungsi .fillna(0) digunakan untuk menggantikan semua nilai NaN (missing values) di kolom 'Academic Pressure' dengan 0.

Hasilnya, kolom 'Academic Pressure' akan berisi 0 untuk setiap baris yang sebelumnya memiliki nilai NaN.

test['Academic Pressure']: Mengakses kolom 'Academic Pressure' dalam dataset test.

.fillna(0): Fungsi ini melakukan hal yang sama pada dataset test, mengganti NaN dengan 0 di kolom 'Academic Pressure'.

```
print(nullpercent(train))
print('*****')
print(nullpercent(test))
```

id	0.000000
Name	0.000000
Gender	0.000000
Age	0.000000
City	0.000000
Working Professional or Student	0.000000
Profession	0.000000
Academic Pressure	0.000000
Work Pressure	19.842217
CGPA	80.171997
Study Satisfaction	80.172708
Job Satisfaction	19.836532
Dietary Habits	0.002843
Degree	0.001421
Have you ever had suicidal thoughts ?	0.000000
Work/Study Hours	0.000000
Financial Stress	0.002843
Family History of Mental Illness	0.000000
Depression	0.000000
Numeric Sleep Duration	0.000000
dtype: float64	
None	

id	0.000000
Name	0.000000
Gender	0.000000
Age	0.000000
City	0.000000
Working Professional or Student	0.000000
Profession	0.000000
Academic Pressure	0.000000
Work Pressure	20.019190
CGPA	79.993603
Study Satisfaction	79.992537
Job Satisfaction	20.014925
Dietary Habits	0.005330
Degree	0.002132
Have you ever had suicidal thoughts ?	0.000000
Work/Study Hours	0.000000
Financial Stress	0.000000

Penjelasan :

Fungsi ini juga akan menghitung persentase nilai NaN di setiap kolom dataset train. Fungsi ini juga akan menghitung persentase nilai NaN di setiap kolom dataset test.

- Work Pressure

```
train['Work Pressure'].value_counts(dropna=False)
```

Work Pressure	count
NaN	27918
2.0	24373
4.0	22512
5.0	22436
3.0	21899
1.0	21562

dtype: int64

Penjelasan :
Menampilkan dataset pada train di Work Pressure

```
[164] train[train['Work Pressure'].isna()].head()
```

id	Name	Gender	Age	City	Professional or Student	Profession	Academic Pressure	Work Pressure	GPA	Study Satisfaction	Job Satisfaction	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Depression	Numeric Sleep Duration
2	Yunraj	Male	33.0	Visakhapatnam	Student	other	5.0	NaN	8.97	2.0	NaN	Healthy	B.Pharm	Yes	3.0	1.0	No	1	5.5
8	Aishwarya	Female	24.0	Bangalore	Student	other	2.0	NaN	5.90	5.0	NaN	Moderate	BSc	No	3.0	2.0	Yes	0	5.5
26	Aditya	Male	31.0	Srinagar	Student	other	3.0	NaN	7.03	5.0	NaN	Healthy	BA	No	9.0	1.0	Yes	0	5.0
30	Prisha	Female	28.0	Varanasi	Student	other	3.0	NaN	5.59	2.0	NaN	Moderate	BCA	Yes	4.0	5.0	Yes	1	7.5
32	Chhavi	Female	25.0	Jaipur	Student	other	4.0	NaN	8.13	3.0	NaN	Moderate	M.Tech	Yes	1.0	1.0	No	0	5.5

Penjelasan :
Hasil yang ditampilkan adalah beberapa entri di dataset train di mana kolom 'Work Pressure' memiliki nilai NaN (data yang hilang).

```
[165] train['Work Pressure']=train['Work Pressure'].fillna(0)
test['Work Pressure']=test['Work Pressure'].fillna(0)
```

Penjelasan :
Tujuan dari kode ini adalah untuk mengisi nilai NaN (missing values) dalam kolom 'Work Pressure' pada dataset train dan test dengan angka 0.

```
[166] train['Study Satisfaction'].value_counts(dropna=False)
```

Study Satisfaction	count
NaN	112803
4.0	6360
2.0	5840
3.0	5823
1.0	5451
5.0	4423

dtype: int64

Penjelasan :
Menampilkan data train pada Study Satisfaction beserta nilai yang ada

```
[167] train['Study Satisfaction']=train['Study Satisfaction'].fillna(0)
test['Study Satisfaction']=test['Study Satisfaction'].fillna(0)
```

Penjelasan :
Tujuan dari kode ini adalah untuk mengisi nilai NaN (missing values) dalam kolom 'Study Satisfaction' pada dataset train dan test dengan angka 0.

- Job Satisfaction

```
[168] train['Job Satisfaction']=train['Job Satisfaction'].fillna(0)
      test['Job Satisfaction']=test['Job Satisfaction'].fillna(0)
```

Penjelasan :

Tujuan dari kode ini adalah untuk mengisi nilai NaN (missing values) dalam kolom 'Job Satisfaction' pada dataset train dan test dengan angka 0.

- Dietary Habits

```
train['Dietary Habits'].value_counts(dropna=False)
```

	count
Dietary Habits	
Moderate	49705
Unhealthy	46227
Healthy	44741
NaN	4
More Healthy	2
No	2
Yes	2
Pratham	1
3	1
Gender	1
BSc	1
Mihir	1
1.0	1
Hormonal	1
Electrician	1
Less than Healthy	1
No Healthy	1
Less Healthy	1
M.Tech	1

Penjelasan :

Menampilkan data train pada Dietary Habits.

```
[170] values_list = [
    'NaN', 'Yes', 'No', 'More Healthy', 'Class 12', 'Indoor', 'Male', 'Vegas', 'M.Tech',
    'Less Healthy', 'No Healthy', 'Hormonal', 'Electrician', '1.0', 'Mihir', 'Less than Healthy',
    '3', 'Gender', 'BSc', 'Pratham', '2', 'Educational', 'Naina', 'Raghav', 'Vivaan',
    '5 Unhealthy', 'Soham', '5 Healthy', 'Academic', 'MCA', 'Resistant', 'Mealy', 'Prachi',
    'Kolkata'
]
```

Penjelasan :

List ini berisi berbagai nilai kategorikal yang kemungkinan besar digunakan untuk memproses data dalam analisis atau untuk mengisi nilai yang hilang (imputasi) dalam dataset.

```
train['Dietary Habits'].value_counts(dropna=False)
```

	count
Dietary Habits	
Moderate	49704
Unhealthy	46226
Healthy	44739
other	27

dtype: int64

Penjelasan :

```
test['Dietary Habits'].value_counts(dropna=False)
```

	count
Dietary Habits	
Moderate	33018
Unhealthy	30786
Healthy	29966
other	30

dtype: int64

- Degree

```
train['Degree'].value_counts(dropna=False)
```

	count
Degree	
Class 12	14729
B.Ed	11691
B.Arch	8742
B.Com	8113
B.Pharm	5856
...	...
LCA	1
B B.Com	1
RCA	1
Mihir	1
Advait	1

116 rows x 1 columns

dtype: int64

```
[ ] data_Degree = [
    np.nan, "M.Arch", "UX/UI Designer", "B.Sc", "Kalyan", "M", "LLBA", "NaN", "BArch", "L.Ed", "BPharm",
    "P.Com", "Nalini", "BEd", "B", "Degree", "Jhanvi", "Bhopal", "MEd", "LL B.Ed", "LLTech", "M_Tech",
    "5.88", "Pihu", "HCA", "Marsh", "Lata", "S.Arch", "BB", "LHM", "8.56", "Entrepreneur", "Aarav",
    "B.Student", "E.Tech", "M.S", "Navya", "Mihir", "RCA", "B B.Com", "LCA", "N.Pharm", "Doctor",
    "CGPA", "LLEd", "LLS", "Esha", "Working Professional", "Mthanya", "B.3.79", "K.Ed", "Mahika",
    "24", "M. Business Analyst", "Brithika", "ACA", "Badhya", "HR Manager", "Unite", "P.Pharm",
    "MPharm", "Data Scientist", "LL.Com", "Business Analyst", "H_Pharm", "Class 11", "20", "S.Tech",
    "Veda", "BH", "MPA", "S.Pharm", "Vrinda", "Bhavesh", "Brit", "B.B.Arch", "7.06", "B BA",
    "5.56", "Ritik", "B.03", "5.61", "0", "Plumber", "BPA", "Vivaan", "MTech", "29", "LLCom", "Advait",
    "BTech", "3.0", "B.M.Com", "Eshita", "M.UI", "B.H", "Mechanical Engineer", "I.Ed", "Magan", "B B.Tech",
    "M.B.Ed", "B Financial Analyst", "GCA", "G.Ed", "Rupak", "B.CA", "PCA", "J.Ed", "8.95", "Aadhya",
    "Banchal", "M.", "B.BA", "Moham", "B. Gender", "A.Ed", "Vibha", "B BCA", "B.Press", "Gagan",
    "Travel Consultant", "5.65", "B.Com", "E.Ed", "B._Pharm", "Pune", "Bian", "B.Study_Hours",
    "Kavya", "M.M.Ed", "BHCA"
]
```

```
test['Degree'].value_counts(dropna=False)
```

Degree	count
Class 12	9812
B.Ed	7762
B.Arch	6037
B.Com	5439
B.Pharm	3987
BCA	3869
M.Ed	3707
MCA	3438
BBA	3387
BS	3314
LLM	3133
MSc	3096
M.Tech	3017
M.Pharm	2995
LLB	2938

- Financial Stress

```
train['Financial Stress'].value_counts(dropna=False)
```

Financial Stress	count
2.0	31451
5.0	28279
4.0	27765
1.0	27211
3.0	25990
NaN	4

dtype: int64

```
[180] train = train.dropna(subset=['Financial Stress'])
      test = test.dropna(subset=['Financial Stress'])
```



```
print(nullpercent(train))
print('*****')
print(nullpercent(test))
```

```
id 0.000000
Name 0.000000
Gender 0.000000
Age 0.000000
City 0.000000
Working Professional or Student 0.000000
Profession 0.000000
Academic Pressure 0.000000
Work Pressure 0.000000
CGPA 80.173566
Study Satisfaction 0.000000
Job Satisfaction 0.000000
Dietary Habits 0.000000
Degree 0.000000
Have you ever had suicidal thoughts ? 0.000000
Work/Study Hours 0.000000
Financial Stress 0.000000
Family History of Mental Illness 0.000000
Depression 0.000000
Numeric Sleep Duration 0.000000
dtype: float64
None
*****
id 0.000000
Name 0.000000
Gender 0.000000
Age 0.000000
City 0.000000
Working Professional or Student 0.000000
Profession 0.000000
Academic Pressure 0.000000
Work Pressure 0.000000
CGPA 79.993603
Study Satisfaction 0.000000
Job Satisfaction 0.000000
Dietary Habits 0.000000
Degree 0.000000
Have you ever had suicidal thoughts ? 0.000000
Work/Study Hours 0.000000
Financial Stress 0.000000
```

```
[182] train.drop(columns='CGPA', inplace=True)
test.drop(columns='CGPA', inplace=True)
```

[183] train.head()

	id	Name	Gender	Age	City	Working Professional or Student	Profession	Academic Pressure	Work Pressure	Study Satisfaction	Job Satisfaction	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Depression	Numeric Sleep Duration
0	0	Aaradhy	Female	49.0	Ludhiana	Working Professional	Chef	0.0	5.0	0.0	2.0	Healthy	BHM	No	1.0	2.0	No	0	8.0
1	1	Vivan	Male	26.0	Varanasi	Working Professional	Teacher	0.0	4.0	0.0	3.0	Unhealthy	LLB	Yes	7.0	3.0	No	1	5.0
2	2	Yuvraj	Male	33.0	Visakhapatnam	Student	other	5.0	0.0	2.0	0.0	Healthy	B.Pharm	Yes	3.0	1.0	No	1	5.5
3	3	Yuvraj	Male	22.0	Mumbai	Working Professional	Teacher	0.0	5.0	0.0	1.0	Moderate	BBA	Yes	10.0	1.0	Yes	1	5.0
4	4	Rhea	Female	30.0	Kanpur	Working Professional	Business Analyst	0.0	1.0	0.0	1.0	Unhealthy	BBA	Yes	9.0	4.0	Yes	0	5.5

- City

```
train['City'].value_counts(dropna=False)
```



City	
Kalyan	6591
Patna	5924
Vasai-Virar	5765
Kolkata	5688
Ahmedabad	5613
...	...
Pooja	1
Khushi	1
Khaziabad	1
Jhanvi	1
Unirar	1

98 rows × 1 columns

dtype: int64

```
test['City'].value_counts(dropna=False)
```



City	
Kalyan	4387
Vasai-Virar	3897
Patna	3888
Kolkata	3726
Ahmedabad	3677
...	...
Vidhi	1
Abhinav	1
Rolkata	1
Ghopal	1
No.12	1

68 rows × 1 columns

dtype: int64

```
[186] names = [
    "Mihir", "Nandini", "Mahi", "Vidya", "City", "Pratyush",
    "Harsha", "Saanvi", "Vidya", "Siddhesh", "Bhavna", "Vikram",
    "Keshav", "Nalini", "City", "Hrithik", "San Vasai-Virar",
    "Vaikot", "Leela", "Chemist", "Ghopal", "No", "More Delhi",
    "Saanvi", "Pratham", "Vidhi", "Abhinav", "Rolkata", "Parth",
    "Aditi", "Saurav", "Sara", "Less Delhi", "Golkata", "Is Kanpur",
    "Unaly", "Thani", "Lawyer", "Vaishnavi", "Ira", "Avni",
    "Mhopal", "Less than 5 hours", "Pratyush", "Malyan", "No.12",
    "Bhavna", "Molkata", "MCA", "M.Com", "Atharv", "Nalini",
    "Keshav", "Ayush", "M.Tech", "Researcher", "Vaishnavi",
    "Chhavi", "Parth", "Vidhi", "Tushar", "MSc", "No",
    "Rashi", "ME", "Ishanabad", "Armaan", "Kagan", "Kashish",
    "Ithal", "Nalyan", "Dhruv", "Galesabad", "Itheg", "Aaradhya",
    "Pooja", "Khushi", "Khaziabad", "Jhanvi", "Kibara", "Harsh",
    "Reyansh", "Morena", "Less Delhi", "Malyansh", "Aditya", "Plata",
    "Aishwarya", "3.0", "Less than 5 Kalyan", "Krishna", "Mira",
    "Moreadhyay", "Ishkarsh", "Raghavendra", "Kashk", "Gurgaon",
    "Tolkata", "Anvi", "Krinda", "Ayansh", "Shrey", "Ivaan",
    "Vaanya", "Gaurav", "Unirar"
]
```

```
[189] train.drop(columns=['id', 'Name'], inplace=True)
      test.drop(columns=['id', 'Name'], inplace=True)
```

[190] train.head()

	Gender	Age	City	Working Professional or Student	Profession	Academic Pressure	Work Pressure	Study Satisfaction	Job Satisfaction	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Depression	Numeric Sleep Duration
0	Female	49.0	Ludhiana	Working Professional	Chef	0.0	5.0	0.0	2.0	Healthy	BHM	No	1.0	2.0	No	0	8.0
1	Male	26.0	Varanasi	Working Professional	Teacher	0.0	4.0	0.0	3.0	Unhealthy	LLB	Yes	7.0	3.0	No	1	5.0
2	Male	33.0	Visakhapatnam	Student	other	5.0	0.0	2.0	0.0	Healthy	B.Pharm	Yes	3.0	1.0	No	1	5.5
3	Male	22.0	Mumbai	Working Professional	Teacher	0.0	5.0	0.0	1.0	Moderate	BBA	Yes	10.0	1.0	Yes	1	5.0
4	Female	30.0	Kanpur	Working Professional	Business Analyst	0.0	1.0	0.0	1.0	Unhealthy	BBA	Yes	9.0	4.0	Yes	0	5.5

TEST.head

test.head()

	Gender	Age	City	Working Professional or Student	Profession	Academic Pressure	Work Pressure	Study Satisfaction	Job Satisfaction	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Depression	Numeric Sleep Duration
0	1	53.0	Visakhapatnam	Working Professional	Judge	0.0	2.0	0.0	5.0	Moderate	LLB	0	9.0	3.0	0	1	5.0
1	0	58.0	Kolkata	Working Professional	Educational Consultant	0.0	2.0	0.0	4.0	Moderate	B.Ed	0	6.0	4.0	0	0	5.0
2	1	53.0	Jaipur	Working Professional	Teacher	0.0	4.0	0.0	1.0	Moderate	B.Arch	1	12.0	4.0	0	0	7.5
3	0	23.0	Rajkot	Student	other	5.0	0.0	1.0	0.0	Moderate	B.Sc	1	10.0	4.0	0	0	8.0
4	1	47.0	Kalyan	Working Professional	Teacher	0.0	5.0	0.0	5.0	Moderate	BCA	1	3.0	4.0	0	0	7.5

Proses pembersihan dilakukan untuk mengatasi nilai yang hilang dalam dataset. Menggunakan SimpleImputer dari sklearn untuk mengimputasi nilai yang hilang pada kolom numerik dengan mean dan pada kolom kategorikal dengan mode (nilai yang paling sering muncul).

```
[196] from sklearn.compose import ColumnTransformer
      from sklearn.preprocessing import StandardScaler, OneHotEncoder
      from sklearn.pipeline import Pipeline
      from sklearn.impute import SimpleImputer
```

```
[197] # Preprocessing pipelines
      # For numerical columns, apply StandardScaler (and impute if necessary)
      numerical_pipeline = Pipeline(steps=[
          ('imputer', SimpleImputer(strategy='mean')), # Impute missing values if any
          ('scaler', StandardScaler())
      ])

      # For categorical columns, apply OneHotEncoding
      categorical_pipeline = Pipeline(steps=[
          ('imputer', SimpleImputer(strategy='most_frequent')), # Impute missing values if any
          ('onehot', OneHotEncoder(handle_unknown='ignore', drop='first'))
      ])
```

Penjelasan :

numerical_pipeline: Ini adalah pipeline untuk menangani kolom numerik. Pertama, nilai yang hilang akan diimputasi dengan mean menggunakan SimpleImputer(strategy='mean'). Kemudian, StandardScaler digunakan untuk menstandarisasi data numerik. categorical_pipeline: Ini adalah pipeline untuk menangani kolom kategorikal. Nilai yang hilang diimputasi dengan most_frequent (nilai yang paling sering muncul) menggunakan SimpleImputer(strategy='most_frequent'). Kemudian, OneHotEncoder digunakan untuk mengubah data kategorikal menjadi format numerik dengan representasi one-hot encoding. Argumen handle_unknown='ignore' memastikan bahwa jika ada nilai yang tidak terlihat selama pelatihan, itu tidak menyebabkan error, dan drop='first' menghindari kolinearitas dengan menghapus satu kategori.

```
[198] # Combine the numerical and categorical pipelines using ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_pipeline, numerical_columns),
        ('cat', categorical_pipeline, categorical_columns)
    ])

```

Penjelasan :

fit_transform pada data pelatihan (train) digunakan untuk mempelajari parameter transformasi dari data pelatihan (seperti mean dan standar deviasi untuk scaling, atau modus untuk imputasi) dan langsung mengaplikasikan transformasi tersebut.

train.drop('Depression', axis=1) digunakan untuk memisahkan kolom target Depression dari fitur yang digunakan untuk pelatihan model. Kolom Depression hanya digunakan untuk melatih model, tidak termasuk dalam fitur. Transform pada data pengujian (test) digunakan untuk mengaplikasikan transformasi yang sudah dipelajari pada data pelatihan ke data pengujian (tanpa mengubah parameter transformasi).

```
[199] # Apply the transformations
X = preprocessor.fit_transform(train.drop('Depression', axis=1))
y = train['Depression']
test_dataframe = preprocessor.transform(test)

```

Penjelasan :

fit_transform pada data pelatihan (train) digunakan untuk mempelajari parameter transformasi dari data pelatihan (seperti mean dan standar deviasi untuk scaling, atau modus untuk imputasi) dan langsung mengaplikasikan transformasi tersebut. train.drop('Depression', axis=1) digunakan untuk memisahkan kolom target Depression dari fitur yang digunakan untuk pelatihan model. Kolom Depression hanya digunakan untuk melatih model, tidak termasuk dalam fitur. transform pada data pengujian (test) digunakan untuk mengaplikasikan transformasi yang sudah dipelajari pada data pelatihan ke data pengujian (tanpa mengubah parameter transformasi).

[200] train.head()

	Gender	Age	City	Working Professional or Student	Profession	Academic Pressure	Work Pressure	Study Satisfaction	Job Satisfaction	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Depression	Numeric Sleep Duration
0	0	49.0	Ludhiana	Working Professional	Chef	0.0	5.0	0.0	2.0	Healthy	BHM	0	1.0	2.0	0	0	8.0
1	1	26.0	Varanasi	Working Professional	Teacher	0.0	4.0	0.0	3.0	Unhealthy	LLB	1	7.0	3.0	0	1	5.0
2	1	33.0	Visakhapatnam	Student	other	5.0	0.0	2.0	0.0	Healthy	B.Pharm	1	3.0	1.0	0	1	5.5
3	1	22.0	Mumbai	Working Professional	Teacher	0.0	5.0	0.0	1.0	Moderate	BBA	1	10.0	1.0	1	1	5.0
4	0	30.0	Kanpur	Working Professional	Business Analyst	0.0	1.0	0.0	1.0	Unhealthy	BBA	1	9.0	4.0	1	0	5.5

test.head()

	Gender	Age	City	Working Professional or Student	Profession	Academic Pressure	Work Pressure	Study Satisfaction	Job Satisfaction	Dietary Habits	Degree	Have you ever had suicidal thoughts ?	Work/Study Hours	Financial Stress	Family History of Mental Illness	Depression	Numeric Sleep Duration
0	1	53.0	Visakhapatnam	Working Professional	Judge	0.0	2.0	0.0	5.0	Moderate	LLB	0	9.0	3.0	1	1	5.0
1	0	58.0	Kolkata	Working Professional	Educational Consultant	0.0	2.0	0.0	4.0	Moderate	B.Ed	0	6.0	4.0	0	0	5.0
2	1	53.0	Jaipur	Working Professional	Teacher	0.0	4.0	0.0	1.0	Moderate	B.Arch	1	12.0	4.0	0	0	7.5
3	0	23.0	Rajkot	Student	other	5.0	0.0	1.0	0.0	Moderate	BSc	1	10.0	4.0	0	0	8.0
4	1	47.0	Kalyan	Working Professional	Teacher	0.0	5.0	0.0	5.0	Moderate	BCA	1	3.0	4.0	0	0	7.5

3.3 Data Construct

- Profession

```
[ ] # Replace using apply with lambda
train["Profession"] = train["Profession"].apply(
    lambda x: 'other' if x in data or pd.isna(x) else x
)
```

Penjelasan :

Semua nilai dalam kolom Profession yang masuk ke daftar ini atau kosong (NaN) akan diubah menjadi 'other'. Nilai yang tidak termasuk dalam daftar ini dianggap sebagai profesi valid dan tetap digunakan.

- Dietary Habits

```
[171] # Replace using apply with lambda
train["Dietary Habits"] = train["Dietary Habits"].apply(
    lambda x: 'other' if x in values_list or pd.isna(x) else x
)
```

```
# Replace using apply with lambda
test["Dietary Habits"] = test["Dietary Habits"].apply(
    lambda x: 'other' if x in values_list or pd.isna(x) else x
)
```

- Degree

```
[176] # Replace using apply with lambda
train["Degree"] = train["Degree"].apply(
    lambda x: 'other' if x in data_Degree or pd.isna(x) else x
)
```

```
[177] # Replace using apply with lambda
test["Degree"] = test["Degree"].apply(
    lambda x: 'other' if x in data_Degree or pd.isna(x) else x
)
```

- City

```
[187] # Replace using apply with lambda
train["City"] = train["City"].apply(
    lambda x: 'other' if x in names or pd.isna(x) else x
)
```

```
[188] # Replace using apply with lambda
test["City"] = test["City"].apply(
    lambda x: 'other' if x in names or pd.isna(x) else x
)
```

- Numeric Sleep Duration

```
[ ] # Apply function to column
    train['Numeric Sleep Duration'] = train['Sleep Duration'].apply(extract_numeric_duration)
```

Fungsi `train['Numeric Sleep Duration'] = train['Sleep Duration'].apply(extract_numeric_duration)` digunakan untuk menerapkan fungsi `extract_numeric_duration` pada setiap baris di kolom `Sleep Duration` pada DataFrame `train`. Fungsi ini bertujuan untuk mengekstrak nilai numerik dari data yang awalnya berbentuk teks seperti `"7-8 hours"` atau `"Less than 5 hours"`. Hasil dari proses ini adalah kolom baru bernama `Numeric Sleep Duration` yang berisi nilai durasi tidur dalam format numerik (float), sehingga lebih mudah untuk dilakukan analisis statistik atau dimasukkan ke dalam model machine learning.

```
[ ] # Exclude values greater than 10 hours and less than 4.5 hours for mean calculation
    valid_values = train[(train['Numeric Sleep Duration'] <= 10) & (train['Numeric Sleep Duration'] >= 4.5)]

    # Calculate mean of valid values
    mean_value = valid_values['Numeric Sleep Duration'].mean()

    # Replace values where Numeric Sleep Duration > 10 or < 4.5 with the calculated mean
    train['Numeric Sleep Duration'] = train['Numeric Sleep Duration'].apply(
        lambda x: mean_value if x > 10 or x < 4.5 else x
    )
```

Fungsi di atas digunakan untuk membersihkan nilai outlier pada kolom `Numeric Sleep Duration`. Nilai-nilai yang dianggap tidak valid, yaitu durasi tidur lebih dari 10 jam atau kurang dari 4.5 jam, dikecualikan dari perhitungan nilai rata-rata. Hal ini dilakukan dengan membuat subset `valid_values` yang hanya mencakup durasi tidur antara 4.5 hingga 10 jam. Kemudian, nilai rata-rata dari subset ini dihitung dan disimpan ke dalam variabel `mean_value`.

Langkah berikutnya adalah mengganti nilai-nilai yang berada di luar rentang valid (lebih dari 10 jam atau kurang dari 4.5 jam) dengan nilai rata-rata yang telah dihitung. Ini dilakukan menggunakan fungsi `.apply()` dengan `lambda`, di mana setiap nilai `x` akan digantikan oleh `mean_value` jika tidak berada dalam rentang yang diinginkan. Hasil dari proses ini adalah kolom `Numeric Sleep Duration` yang telah diperbaiki dari kemungkinan kesalahan input atau outlier ekstrem, sehingga lebih representatif untuk analisis selanjutnya.

3.4 Labeling Data

```
. # Convert Gender column: Male -> 1, Female -> 0
    train['Gender'] = train['Gender'].replace({'Male': 1, 'Female': 0})

    # Convert 'Have you ever had suicidal thoughts?' column: Yes -> 1, No -> 0
    train['Have you ever had suicidal thoughts ?'] = train['Have you ever had suicidal thoughts ?'].replace({'Yes': 1, 'No': 0})

    # Convert 'Family History of Mental Illness' column: Yes -> 1, No -> 0
    train['Family History of Mental Illness'] = train['Family History of Mental Illness'].replace({'Yes': 1, 'No': 0})

    # Convert Gender column: Male -> 1, Female -> 0
    test['Gender'] = test['Gender'].replace({'Male': 1, 'Female': 0})

    # Convert 'Have you ever had suicidal thoughts?' column: Yes -> 1, No -> 0
    test['Have you ever had suicidal thoughts ?'] = test['Have you ever had suicidal thoughts ?'].replace({'Yes': 1, 'No': 0})

    # Convert 'Family History of Mental Illness' column: Yes -> 1, No -> 0
    test['Family History of Mental Illness'] = test['Family History of Mental Illness'].replace({'Yes': 1, 'No': 0})
```

Penjelasan :

Mengubah data kategorikal menjadi format yang dapat digunakan dalam model machine learning. Kode ini memastikan bahwa model dapat memproses kolom Gender, Have you ever had suicidal thoughts?, dan Family History of Mental Illness dengan cara yang benar.

3.5 Data Integration

Pada tahap ini, kita memastikan bahwa data pelatihan dan pengujian siap digunakan bersama model.

```
# Menggabungkan dataset train dan test
combined_data = pd.concat([train, test], axis=0, ignore_index=True)

# Menampilkan beberapa baris pertama untuk memastikan data telah digabungkan dengan benar
print(combined_data.head())
```

```
Gender  Age  City Working Professional or Student \
0      0  49.0  Ludhiana      Working Professional
1      1  26.0  Varanasi      Working Professional
2      1  33.0  Visakhapatnam      Student
3      1  22.0  Mumbai      Working Professional
4      0  30.0  Kanpur      Working Professional

      Profession Academic Pressure Work Pressure Study Satisfaction \
0      Chef      0.0      5.0      0.0
1  Teacher      0.0      4.0      0.0
2  other      5.0      0.0      2.0
3  Teacher      0.0      5.0      0.0
4 Business Analyst      0.0      1.0      0.0

      Job Satisfaction Dietary Habits Degree \
0      2.0      Healthy BHM
1      3.0      Unhealthy LLB
2      0.0      Healthy B.Pharm
3      1.0      Moderate BBA
4      1.0      Unhealthy BBA

      Have you ever had suicidal thoughts ? Work/Study Hours Financial Stress \
0      0      1.0      2.0
1      1      7.0      3.0
2      1      3.0      1.0
3      1      10.0      1.0
4      1      9.0      4.0

      Family History of Mental Illness Depression Numeric Sleep Duration
0      0      0      8.0
1      0      1      5.0
2      0      1      5.5
3      1      1      5.0
4      1      0      5.5
```

Penjelasan :

Proses Data Integration adalah tentang menggabungkan data dari beberapa sumber (seperti train dan test) menjadi satu dataset yang kohesif dan siap untuk analisis lebih lanjut atau model machine learning.

BAB 4

MODELLING

4.1 Membangun Skenario Pengujian

1. Skenario Pengujian 1: Pengujian Akurasi Model

Tujuan Pengujian:

Memastikan bahwa model regresi logistik yang dibangun dapat menghasilkan prediksi yang akurat berdasarkan data uji.

Langkah Pengujian:

- Persiapkan data uji (test set) yang telah dibersihkan dan diproses sesuai dengan model pelatihan.
- Terapkan model regresi logistik pada data uji.
- Periksa hasil prediksi yang diberikan oleh model untuk setiap entri pada data uji.
- Bandingkan hasil prediksi dengan label target pada data uji untuk menghitung akurasi, precision, recall, dan F1-score.
- Lakukan evaluasi menggunakan confusion matrix untuk mengevaluasi distribusi prediksi model.
- Tinjau hasil dan pastikan model dapat mengklasifikasikan dengan benar individu yang mengalami depresi dan yang tidak.

Kriteria Keberhasilan:

- Akurasi model lebih dari 60% sesuai dengan target yang ditetapkan.
- Nilai precision, recall, dan F1-score yang tinggi pada kategori "Depresi" (label 1).
- Model dapat memprediksi dengan baik meskipun terdapat ketidakseimbangan antara label "Tidak Depresi" dan "Depresi".

2. Skenario Pengujian 2: Pengujian Kinerja Model pada Data Tidak Seimbang

Tujuan Pengujian:

Memastikan model dapat menangani masalah ketidakseimbangan kelas, di mana kategori "Tidak Depresi" jauh lebih banyak daripada "Depresi".

Langkah Pengujian:

- Pastikan data uji memiliki distribusi label yang tidak seimbang (lebih banyak "Tidak Depresi" daripada "Depresi").
- Terapkan model regresi logistik pada data uji yang tidak seimbang.
- Evaluasi hasil menggunakan metrik precision, recall, dan F1-score, dengan penekanan pada kategori minoritas ("Depresi").
- Tinjau confusion matrix untuk mengevaluasi seberapa baik model mengidentifikasi individu yang mengalami depresi meskipun jumlahnya lebih sedikit.
- Jika perlu, sesuaikan teknik penyeimbangan data (misalnya, oversampling atau undersampling) untuk meningkatkan kinerja pada kategori minoritas.

Kriteria Keberhasilan:

- Model menunjukkan recall yang tinggi pada kategori "Depresi".
- F1-score untuk kategori "Depresi" lebih besar daripada sekadar akurasi yang tinggi pada "Tidak Depresi".
- Model dapat menangani ketidakseimbangan kelas dengan baik.

3. Skenario Pengujian 3: Pengujian Model dengan Input Pengguna

Tujuan Pengujian:

Memastikan bahwa model dapat menerima input pengguna secara langsung dan memberikan prediksi yang valid.

Langkah Pengujian:

- Buka aplikasi atau antarmuka pengguna yang memungkinkan input data pengguna (misalnya, usia, tekanan kerja, durasi tidur).
- Masukkan data pengguna ke dalam aplikasi, termasuk data numerik dan kategorikal (seperti "Age", "Work Pressure", "Sleep Duration", dll.).
- Terapkan model regresi logistik untuk memprediksi apakah pengguna berisiko mengalami depresi berdasarkan input tersebut.
- Periksa hasil prediksi yang diberikan oleh model.
- Verifikasi apakah model memberikan output yang masuk akal berdasarkan input pengguna (misalnya, jika seseorang memiliki skor tekanan kerja yang tinggi dan durasi tidur yang rendah, model harus memprediksi risiko depresi yang lebih tinggi).

Kriteria Keberhasilan:

- Aplikasi dapat memproses input pengguna dengan benar.
- Model memberikan hasil yang valid dan akurat berdasarkan input.
- Pengguna dapat dengan mudah mengakses prediksi risiko depresi mereka.

4. Skenario Pengujian 4: Pengujian Pengaruh Data yang Tidak Konsisten

Tujuan Pengujian:

Memastikan bahwa model dapat menangani data yang tidak konsisten atau tidak valid (misalnya, nilai yang hilang atau tidak relevan).

Langkah Pengujian:

- Siapkan data uji dengan nilai yang hilang atau tidak konsisten (misalnya, nilai pada kolom "Sleep Duration" yang tidak valid).
- Terapkan model regresi logistik pada data dengan nilai yang hilang dan evaluasi bagaimana model menangani data yang tidak lengkap.
- Imputasi nilai yang hilang dengan nilai default atau menggunakan teknik imputasi seperti mean atau mode.
- Uji hasil prediksi dan periksa apakah model masih memberikan hasil yang valid meskipun data tidak lengkap.

Kriteria Keberhasilan:

- Model dapat menangani data yang hilang atau tidak konsisten dengan baik, memberikan hasil prediksi yang masih akurat.
- Imputasi atau penghapusan nilai yang hilang tidak mempengaruhi hasil model secara signifikan.

4.2 Membangun Model

```
# 1. Pisahkan fitur dan target
X = train.drop(columns=['Depression'])
y = train['Depression']
```

Penjelasan :

Memisahkan antara fitur (variabel input) dan target (label output) sebelum melakukan proses pelatihan model machine learning. Kolom 'Depression' merupakan target yang ingin diprediksi, sehingga dipisahkan dari variabel fitur `X` yang berisi atribut-atribut lain yang digunakan untuk melakukan prediksi. Pemisahan ini penting agar model hanya belajar dari fitur dan tidak "mengintip" label selama pelatihan.

```
[205] X = pd.get_dummies(X, drop_first=True)
```

Penjelasan :

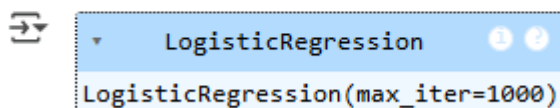
Mengubah variabel kategori menjadi representasi numerik agar dapat digunakan oleh algoritma machine learning. Fungsi `get_dummies` akan mengubah setiap kolom kategori menjadi beberapa kolom biner (0 atau 1). Argumen `drop_first=True` digunakan untuk menghindari dummy variable trap, yaitu situasi di mana variabel dummy saling berkorelasi sempurna sehingga menyebabkan multikolinearitas dalam model.

```
[206] X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

Penjelasan :

Membagi data menjadi dua bagian, yaitu data latih (train) dan data validasi (validation) dengan proporsi 80:20. Pembagian ini penting agar model dapat dilatih pada sebagian data dan diuji pada data yang belum pernah dilihat sebelumnya untuk mengevaluasi performanya. Parameter `random_state=42` digunakan untuk memastikan bahwa pembagian data bersifat konsisten dan dapat direproduksi pada setiap eksekusi kode.

```
[207] model = LogisticRegression(max_iter=1000)
      model.fit(X_train, y_train)
```



Penjelasan :

Membangun model klasifikasi untuk memprediksi apakah seseorang mengalami depresi atau tidak. Logistic Regression dipilih karena merupakan algoritma yang efisien untuk klasifikasi biner, seperti kasus ini. Parameter `max_iter=1000` digunakan untuk memastikan algoritma konvergen, yaitu memberikan cukup iterasi bagi algoritma agar menemukan solusi terbaik dalam pelatihan model.

Selanjutnya, kami melatih model dengan `model.fit(X_train, y_train)` agar model dapat belajar dari data latih.

BAB 5

MODEL EVALUATION

5.1 Mengevaluasi Hasil Pemodelan

```
[208] y_pred = model.predict(X_val)
```

```
print("Classification Report:\n", classification_report(y_val, y_pred))  
print("Accuracy:", accuracy_score(y_val, y_pred))
```

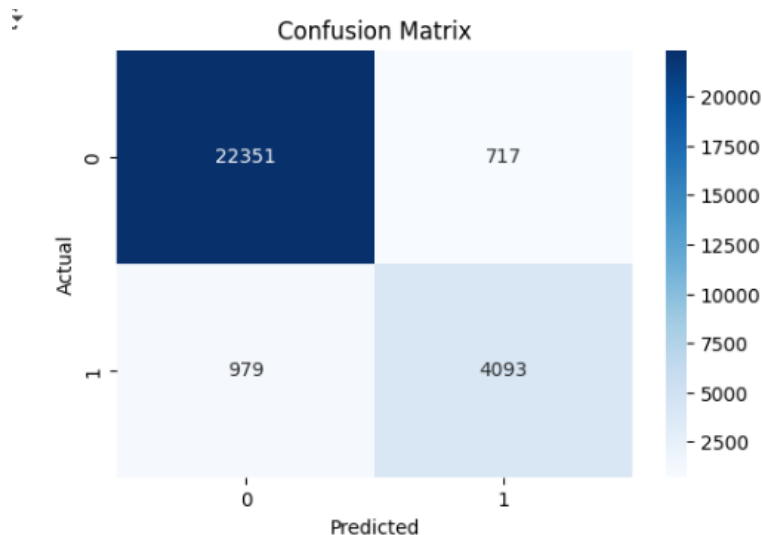
```
Classification Report:  
              precision    recall  f1-score   support  
  
     0           0.96       0.97      0.96       23068  
     1           0.85       0.81      0.83        5072  
  
    accuracy              0.94       28140  
   macro avg           0.90       0.89      0.90       28140  
  weighted avg           0.94       0.94      0.94       28140  
  
Accuracy: 0.939729921819474
```

Penjelasan :

Dari hasil evaluasi, terlihat bahwa model memiliki:

- Akurasi sebesar 0.94 atau 94%, yang menunjukkan bahwa 94% dari prediksi model sesuai dengan label sebenarnya.
- Untuk kelas 0 (tidak depresi), precision, recall, dan f1-score sangat tinggi (masing-masing 0.96, 0.97, dan 0.96), menandakan model sangat baik dalam mengenali individu yang tidak mengalami depresi.
- Untuk kelas 1 (depresi), precision, recall, dan f1-score masing-masing 0.85, 0.81, dan 0.83. Nilai ini masih cukup baik, meskipun lebih rendah dibanding kelas 0, yang menunjukkan bahwa model sedikit kurang optimal dalam mendeteksi individu yang mengalami depresi.
- Macro avg dan weighted avg menunjukkan bahwa secara keseluruhan, model memiliki performa seimbang, meskipun terdapat ketidakseimbangan jumlah data antara dua kelas (imbalance).

```
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.model_selection import cross_val_score  
  
# Confusion Matrix  
cm = confusion_matrix(y_val, y_pred)  
plt.figure(figsize=(6, 4))  
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  
plt.title('Confusion Matrix')  
plt.xlabel('Predicted')  
plt.ylabel('Actual')  
plt.show()
```



Penjelasan :

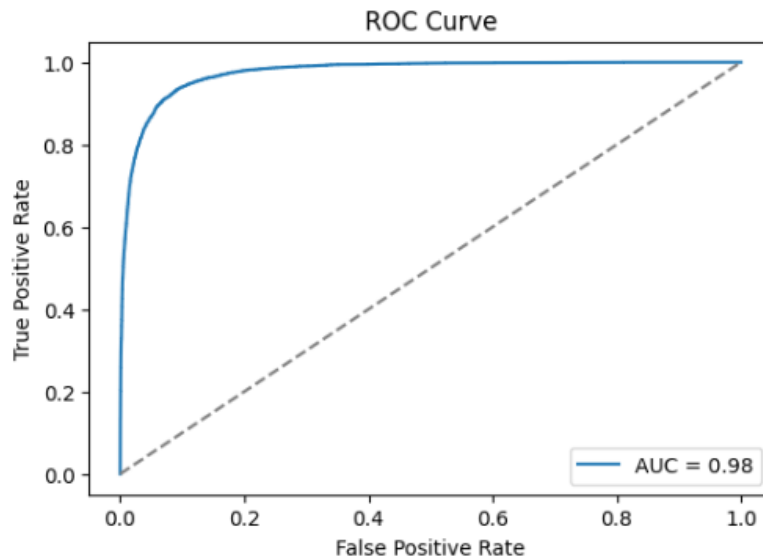
Visualisasi confusion matrix ini dibuat menggunakan `seaborn.heatmap` karena heatmap mempermudah interpretasi data secara visual dengan representasi warna dan anotasi nilai yang jelas. Confusion matrix di atas menunjukkan performa model klasifikasi biner. Label 0 dan 1 masing-masing merepresentasikan dua kelas yang berbeda. Berikut interpretasinya:

- True Negative (TN) = 22.351 : Model memprediksi 0 dan kenyataannya juga 0.
- False Positive (FP) = 717 : Model memprediksi 1 tetapi kenyataannya adalah 0.
- False Negative (FN) = 979 : Model memprediksi 0 tetapi kenyataannya adalah 1.
- True Positive (TP) = 4.093 : Model memprediksi 1 dan kenyataannya juga 1.

Dari confusion matrix ini, model terlihat memiliki performa cukup baik dengan jumlah prediksi benar yang jauh lebih besar daripada prediksi salah. Namun, false negative (979) sedikit lebih tinggi daripada false positive (717), yang bisa berdampak tergantung konteks aplikasinya (misalnya jika kesalahan melewati kelas 1 lebih berbahaya).

```
# ROC Curve & AUC
y_pred_proba = model.predict_proba(X_val)[: , 1]
fpr, tpr, thresholds = roc_curve(y_val, y_pred_proba)
auc_score = roc_auc_score(y_val, y_pred_proba)

plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, label=f"AUC = {auc_score:.2f}")
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```



Penjelasan :

ROC (Receiver Operating Characteristic) Curve di atas menunjukkan performa model klasifikasi dalam membedakan antara kelas positif dan negatif. Kurva ROC diplot berdasarkan nilai *True Positive Rate* (sensitivity) terhadap *False Positive Rate* di berbagai threshold klasifikasi.

- Garis diagonal abu-abu mewakili model acak (AUC = 0.5).
- Kurva biru menunjukkan kinerja model.
- AUC = 0.98 menandakan bahwa model memiliki performa sangat baik karena nilai AUC mendekati 1. Artinya, model mampu membedakan dengan sangat baik antara kelas positif dan negatif. Semakin dekat kurva ROC ke sudut kiri atas grafik, semakin baik performa model. Dalam hal ini, model menunjukkan performa klasifikasi yang sangat tinggi.

```
[211] # Cross-validation (optional)
      cv_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy')
      print("Cross-validation accuracy scores:", cv_scores)
      print("Mean CV accuracy:", cv_scores.mean())
```

Cross-validation accuracy scores: [0.9409204 0.93660877 0.9403847 0.93527609 0.93798587]
Mean CV accuracy: 0.9382351657376571

Penjelasan :

Cross-validation dilakukan sebanyak 5 kali (cv=5), menghasilkan skor akurasi sebagai berikut:

- Fold 1: 0.9409
- Fold 2: 0.9366
- Fold 3: 0.9404
- Fold 4: 0.9353
- Fold 5: 0.9379

Rata-rata akurasi cross-validation adalah 0.9382 . Ini menunjukkan bahwa model memiliki performa yang konsisten dan cukup tinggi di berbagai subset data, serta kemungkinan besar memiliki kemampuan generalisasi yang baik terhadap data baru.

5.2 Melakukan Review Proses Pemodelan

1. Tujuan Pemodelan

Jelaskan kembali secara singkat tujuan dari pemodelan, misalnya:

"Pemodelan dilakukan untuk memprediksi kemungkinan seseorang memiliki pemikiran bunuh diri berdasarkan data karakteristik individu dan faktor-faktor yang mempengaruhi kesehatan mental."

2. Teknik Pemodelan yang Digunakan

Sebutkan metode machine learning atau statistik yang digunakan, seperti:

"Dalam penelitian ini digunakan algoritma Logistic Regression, karena cocok untuk klasifikasi biner antara individu yang memiliki dan tidak memiliki pikiran bunuh diri."

3. Proses Pra-pemrosesan Data

Ulas kembali bagaimana data diproses sebelum pemodelan:

- Pembersihan data (missing values, duplikasi)
- Encoding data kategorik
- Normalisasi jika diperlukan
- Pembagian data (train/test split)

4. Hasil Pemodelan

Sampaikan ringkasan dari hasil pemodelan:

- Akurasi model
- Precision, Recall, F1-score jika digunakan
- Interpretasi koefisien (jika model seperti logistic regression digunakan)

5. Evaluasi dan Validasi

Refleksikan bagaimana validasi dilakukan:

- Cross-validation?
- Train-test split berapa persen?
- Apakah model overfitting atau underfitting?

6. Kelebihan dan Keterbatasan

Tunjukkan kekuatan dan kelemahan proses pemodelan yang dilakukan:

- Apakah dataset cukup representatif?
- Apakah ada bias dalam data?
- Apakah fitur yang digunakan sudah mencerminkan realita dengan baik?

7. Usulan Perbaikan

Berikan saran untuk pemodelan yang lebih baik ke depan:

- Coba model lain seperti Decision Tree, Random Forest, atau Neural Networks
- Tambahkan fitur lain yang relevan (misalnya data psikologis)
- Gunakan teknik feature selection

BAB 6

DEPLOYMENT

6.1 Membuat Rencana Deployment Model

Rencana deployment dilakukan untuk memastikan bahwa model machine learning yang telah dibangun dapat digunakan oleh pengguna akhir melalui antarmuka web. Dalam proyek ini, rencana deployment mencakup:

- Menyimpan model yang telah dilatih (`model_logreg.pkl`) dan scaler (`scaler.pkl`) dalam bentuk file menggunakan `joblib`.
- Membuat backend menggunakan framework Flask yang bertugas menerima input dari user, memproses data, dan menampilkan hasil prediksi.
- Membuat tampilan antarmuka sederhana menggunakan HTML (`index.html`) agar pengguna dapat mengisi formulir input.
- Menyiapkan server lokal (`localhost`) untuk pengujian awal, dengan kemungkinan untuk di-deploy ke server online seperti Heroku, PythonAnywhere, atau layanan VPS.
- Menentukan port dan host yang akan digunakan (`host='0.0.0.0'`, `port=5001`).

6.2 Melakukan Deployment Model

Deployment model dilakukan dengan langkah-langkah sebagai berikut:

1. Mempersiapkan Lingkungan

- Instalasi dependensi seperti Flask, numpy, dan joblib.
- Menyimpan file model_logreg.pkl dan scaler.pkl ke direktori yang sama dengan file aplikasi Flask (app.py).

2. Menjalankan Aplikasi

- Jalankan aplikasi menggunakan perintah:
"python app.py"
- Aplikasi akan berjalan di http://localhost:5001/.

3. Interaksi Pengguna

- Pengguna mengakses halaman index.html dan mengisi formulir.
- Data input dikirim ke endpoint /predict melalui metode POST.
- Data diskalakan menggunakan scaler.pkl, diprediksi menggunakan model_logreg.pkl, dan hasilnya ditampilkan di halaman.

4. Pengujian dan Validasi

- Dilakukan pengujian fungsi prediksi menggunakan berbagai kombinasi input untuk memastikan model bekerja sebagaimana mestinya.

6.3 Melakukan Rencana Pemeliharaan

Pemeliharaan sistem dilakukan untuk memastikan bahwa aplikasi tetap berjalan optimal dan dapat menyesuaikan dengan kebutuhan pengguna atau data baru. Rencana pemeliharaan mencakup:

• Pemantauan Kinerja Model:

- Mengevaluasi performa model secara berkala dengan data baru (real-time atau batch).
- Jika performa menurun, dilakukan retraining model.

• Perbaruan Antarmuka Pengguna:

- Menambahkan validasi form, desain UI responsif, atau fitur tambahan jika dibutuhkan pengguna.

• Pembaruan Dependensi dan Keamanan:

- Melakukan update paket/dependensi Python untuk keamanan dan kestabilan aplikasi.

• Backup dan Recovery:

- Menyimpan backup model dan data secara berkala.
- Menyusun prosedur pemulihan jika terjadi kegagalan sistem.

6.4 Melakukan Pemeliharaan

Implementasi rencana pemeliharaan dilakukan dengan tindakan-tindakan konkret sebagai berikut:

• Retraining Model:

- Apabila ditemukan data baru yang menunjukkan adanya perubahan tren, tim akan mengumpulkan data tersebut dan melakukan pelatihan ulang pada model.

• Monitoring Log Error:

- Flask menyediakan log error jika terjadi kegagalan dalam prediksi. Ini dianalisis untuk memperbaiki bug.

- **Perbaikan Bug pada Kode:**
 - Setiap kesalahan dalam pemrosesan input atau masalah kompatibilitas akan diperbaiki dengan memperbarui kode Python (app.py) maupun HTML (index.html).
- **Penambahan Fitur:**
 - Berdasarkan masukan pengguna, aplikasi akan diperbarui, misalnya dengan menambahkan grafik hasil prediksi atau penjelasan visual dari hasil.

Check Depression Prediction

Name: yohana sitanggang

Gender: Male | Age: 49

City (A=1, B=2, ...): 12 | Working Status: Working Professional

Profession (A=1, ... Z=26): 3 | Degree (numeric): 2

Academic Pressure: 0 | Work Pressure: 5

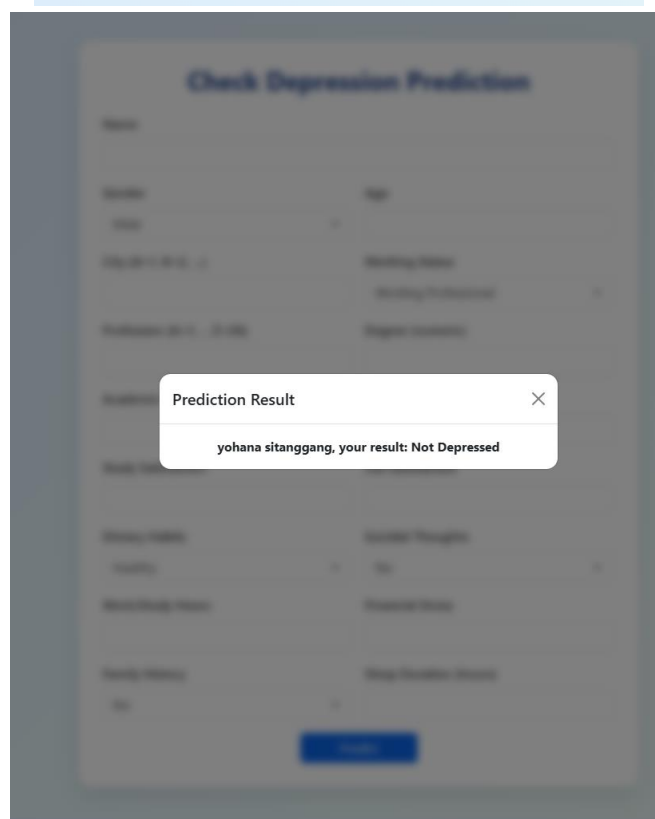
Study Satisfaction: 0 | Job Satisfaction: 2

Dietary Habits: Healthy | Suicidal Thoughts: No

Work/Study Hours: 1 | Financial Stress: 2

Family History: No | Sleep Duration (hours): 8

Predict



DAFTAR PUSTAKA

[1] World Health Organization, *Mental health action plan 2013-2020*, WHO, 2013. Available: <https://www.who.int>

[2] J. Goh, J. Pfeffer, and S. A. Zenios, "The relationship between workplace stressors and mortality and health costs in the United States," *Management Science*, vol. 62, no. 2, pp. 608–628, 2016, doi: 10.1287/mnsc.2015.2260.

[3] M. Sajjadian, R. W. Lam, R. Milev, S. Rotzinger, B. N. Frey, C. N. Soares, S. V. Parikh, J. A. Foster, G. Turecki, D. J. Müller, S. C. Strother, F. Farzan, S. H. Kennedy, and R. Uher, "Machine learning in the prediction of depression treatment outcomes: a systematic review

and meta-analysis," *Psychological Medicine*, vol. 51, no. 16, pp. 2742–2751, Dec. 2021, doi: 10.1017/S0033291721003871.

[4] S. Redzic, "Data distribution inside the OSMI database based on individual's location," *Using Machine Learning to Benefit Mental Health in the Tech Workplace*, Mar. 2023.

[5] R. Z. Goetzel, E. C. Roemer, C. Holingue, S. L. McGinty, S. M. Fikkan, J. A. Foster, and R. S. McIntyre, "Mental health in the workplace: A call to action proceedings from the mental health in the workplace—Public Health Summit," *Journal of Occupational and Environmental Medicine*, vol. 60, no. 4, pp. 322–330, Apr. 2018.