Logistic Regression, K-Nearest Neighbor, Naïve Bayes, Decision Tree

disusun untuk memenuhi tugas

mata kuliah Pembelajaran mesin

Oleh:

Kelompok 10

Anggota :

Muhammad Habil Aswad	(2208107010013)
Rafli Afriza Nugraha	(2208107010028)
Muhammad Khalid Al Ghifari	(2208107010044)
Muhammad Ridho	(2208107010064)
Muhammad Ilzam	(2208107010087)

JURUSAN INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SYIAH KUALA

DARUSSALAM, BANDA ACEH

2025

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam gaya hidup modern saat ini, banyak orang yang mengabaikan pentingnya tidur dan lupa akan manfaat besar yang diberikan tidur bagi tubuh manusia. Untuk menjawab tantangan ini, Smart-Yoga Pillow (SaYoPillow) dikembangkan sebagai solusi inovatif yang mengintegrasikan teknologi cerdas guna memahami hubungan antara stres dan kualitas tidur.

SaYoPillow mengusung konsep "Smart Sleeping" yang bertujuan tidak hanya meningkatkan kualitas tidur, tetapi juga memprediksi tingkat stres yang mungkin dialami seseorang pada keesokan harinya. Perangkat ini dilengkapi dengan edge processor, yaitu pemroses pintar yang mampu menganalisis perubahan fisiologis tubuh selama tidur, serta mencatat kebiasaan tidur pengguna.

Dataset ini memuat hubungan antara berbagai parameter tidur dan tingkat stres yang dikategorikan sebagai:

- **0** – Rendah/Normal
- **1** – Rendah-Menengah
- **2** – Menengah
- **3** – Menengah-Tinggi
- **4** – Tinggi

Parameter yang dianalisis meliputi:

- Rentang dengkuran
- Laju pernapasan
- Suhu tubuh
- Tingkat gerakan anggota tubuh
- Kadar oksigen dalam darah
- Pergerakan mata

- Jumlah jam tidur

- Detak jantung

1.2 Tujuan

- Menganalisis hubungan antara konsumsi bahan bakar dan emisi CO_2 menggunakan teknik eksplorasi data dan visualisasi.
- Membangun dan mengevaluasi model prediksi emisi CO_2 menggunakan regresi linier dan regresi polinomial untuk menentukan model yang paling akurat.

BAB II
PEMBAHASAN

2.1 Deskripsi Dataset

Dataset yang digunakan dalam analisis ini diperoleh dari kajian literatur yang relevan dan dikompilasi secara sistematis tanpa melibatkan subjek manusia secara langsung. Dataset ini diberi nama **"SayoPillow"** dan berisi data mengenai berbagai parameter fisiologis dan kebiasaan tidur yang berkaitan dengan tingkat stres seseorang. Dataset ini disusun untuk mendukung pengembangan model prediksi tingkat stres berdasarkan kualitas tidur dan kondisi fisik saat tidur.

Secara keseluruhan, dataset ini terdiri dari sejumlah entri data (jumlah baris tidak disebutkan secara eksplisit) dengan **9 fitur utama** yang mencerminkan indikator fisiologis pengguna selama tidur. Fitur-fitur tersebut antara lain:

- **Rentang dengkuran (Snoring Range)**
- **Laju pernapasan (Respiration Rate)**
- **Suhu tubuh (Body Temperature)**
- **Tingkat gerakan anggota tubuh (Limb Movement Rate)**
- **Kadar oksigen dalam darah (Blood Oxygen Levels)**
- **Pergerakan mata (Eye Movement)**
- **Jumlah jam tidur (Number of Hours of Sleep)**
- **Detak jantung (Heart Rate)**
- **Tingkat stres (Stress Level)** — sebagai variabel target dengan 5 kelas:
    - 0 = Rendah / Normal
    - 1 = Rendah-Menengah
    - 2 = Menengah
    - 3 = Menengah-Tinggi
    - 4 = Tinggi

2.2 Sampel Data

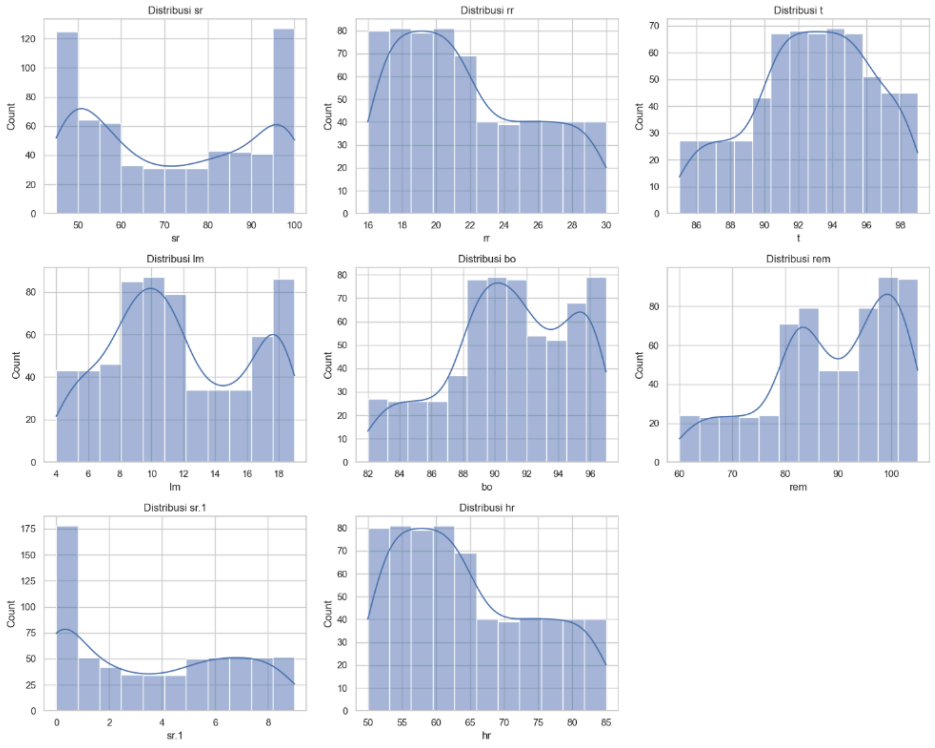
Beberapa sampel data dari dataset ini ditampilkan sebagai berikut:

```
5 Data Pertama:

Ukuran Dataset:
Jumlah Baris: 630, Jumlah Kolom: 9

Informasi Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 630 entries, 0 to 629
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   sr      630 non-null    float64
 1   rr      630 non-null    float64
 2   t       630 non-null    float64
 3   lm      630 non-null    float64
 4   bo      630 non-null    float64
 5   rem     630 non-null    float64
 6   sr.1    630 non-null    float64
 7   hr      630 non-null    float64
 8   sl      630 non-null    int64
dtypes: float64(8), int64(1)
memory usage: 44.4 KB
```

2.3 Data Loading

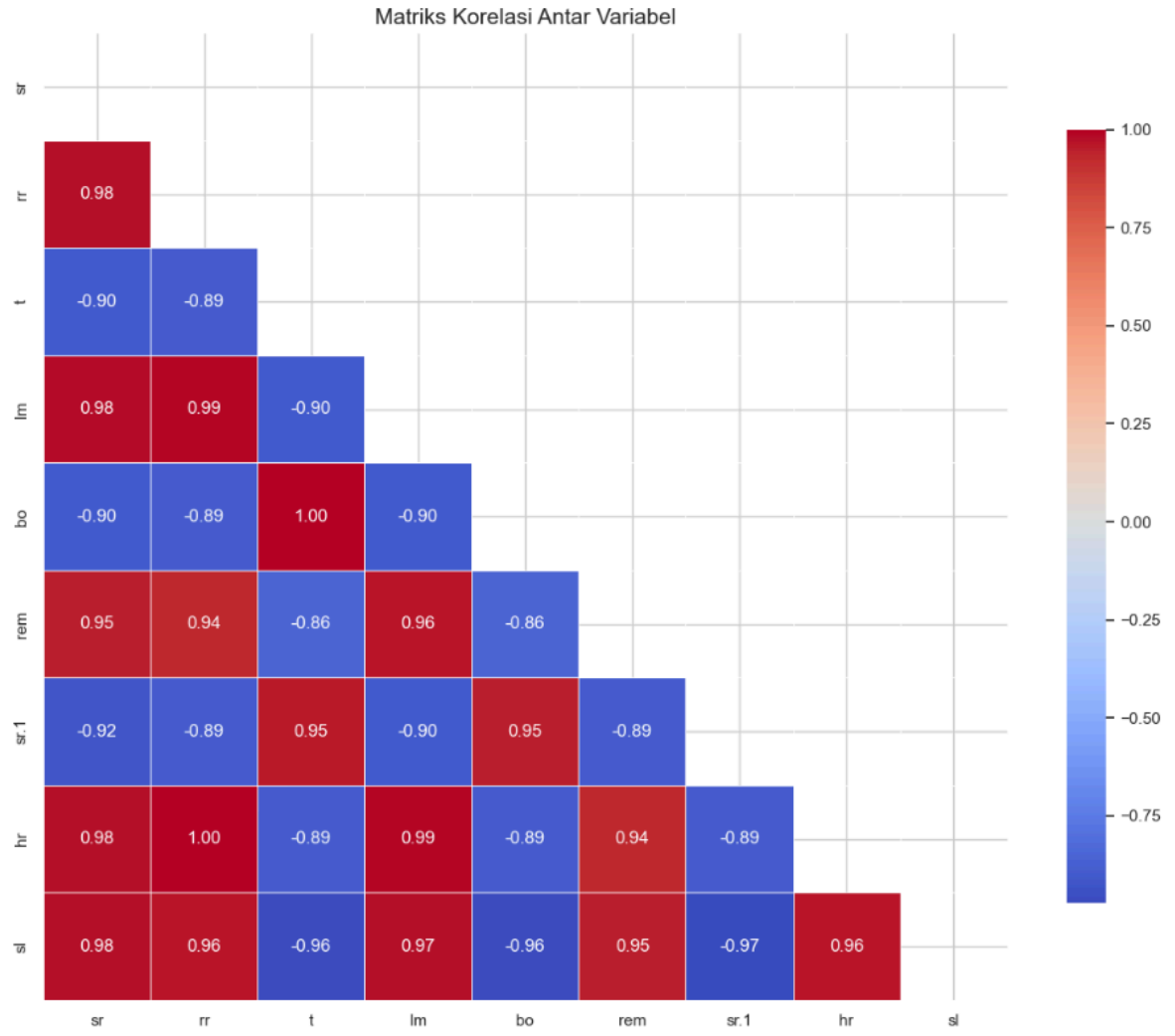
Proses pemuatan data dilakukan dengan membaca file CSV menggunakan library pandas. Data yang dimuat kemudian diperiksa untuk memastikan tidak ada nilai yang hilang dan bahwa semua atribut memiliki tipe data yang sesuai.

2.4 Distribusi Data



Gambar distribusi data menunjukkan variasi dari sembilan parameter fisiologis yang diamati selama tidur. Rentang dengkuran (sr) memiliki distribusi multimodal dengan dua puncak dominan di sekitar nilai 50 dan 100, mengindikasikan adanya dua kelompok pengguna: yang mendengkur ringan dan berat. Laju pernapasan (rr) terkonsentrasi antara 16–22, menunjukkan nilai yang umumnya normal. Suhu tubuh (t) cenderung berkumpul di sekitar 91–96°F, mencerminkan kondisi fisiologis yang stabil saat tidur. Gerakan anggota tubuh (lm) memperlihatkan dua puncak, yaitu pada nilai 8–10 dan 18, mengindikasikan sebagian pengguna tidur cukup gelisah. Kadar oksigen dalam darah (bo) memiliki distribusi normal dengan kecenderungan tinggi pada nilai 92–96. Pergerakan mata (rem) menunjukkan dua puncak signifikan di kisaran 80 dan 100, mencerminkan variasi tinggi dalam fase REM tidur. Parameter sr.1, yang mungkin merupakan indikator turunan dari mendengkur, didominasi oleh nilai rendah (sekitar 0–2), menunjukkan banyak pengguna mengalami mendengkur ringan atau tidak sama sekali. Terakhir, detak jantung (hr) mayoritas berkisar antara 55–65 bpm, menunjukkan banyak pengguna memiliki detak jantung yang tergolong tenang selama tidur. Seluruh distribusi ini memberikan gambaran menyeluruh tentang kondisi fisiologis pengguna selama tidur yang berpotensi mempengaruhi tingkat stres.

2.5 Korelasi Antar Variabel



Berdasarkan matriks korelasi antar variabel pada dataset SaYoPillow, terlihat bahwa sebagian besar variabel memiliki hubungan yang sangat kuat, baik positif maupun negatif. Variabel snoring rate (sr), respiration rate (rr), limb movement (lm), eye movement (rem), heart rate (hr), dan stress level (sl) menunjukkan korelasi positif yang sangat tinggi satu sama lain, dengan nilai korelasi mencapai 0.98 hingga 1.00. Hal ini mengindikasikan bahwa peningkatan aktivitas fisiologis seperti pernapasan, gerakan tubuh, dan detak jantung selama tidur cenderung berkaitan dengan meningkatnya tingkat stres. Sebaliknya, variabel suhu tubuh (t) menunjukkan korelasi negatif yang kuat terhadap hampir semua variabel lainnya, termasuk korelasi -0.96 terhadap stress level, yang menunjukkan bahwa suhu tubuh yang lebih tinggi selama tidur justru cenderung berkaitan dengan tingkat stres yang lebih rendah. Korelasi-korelasi ini menggambarkan adanya keterkaitan yang erat antara kondisi fisiologis saat tidur dengan prediksi tingkat stres yang dapat dianalisis secara efektif menggunakan pendekatan berbasis data seperti yang diimplementasikan dalam sistem SaYoPillow.

2.6 Missing Value

```
# Memeriksa missing values
print("Jumlah Missing Values per Kolom:")
print(df.isnull().sum())
```

```
Jumlah Missing Values per Kolom:
sr      0
rr      0
t       0
lm      0
bo      0
rem     0
sr.1    0
hr      0
sl      0
dtype: int64
```

Dari hasil eksplorasi awal terhadap dataset SaYoPillow, ditemukan bahwa tidak terdapat *missing value* pada atribut manapun. Hal ini menunjukkan bahwa kualitas data sangat baik dan siap digunakan untuk proses analisis lebih lanjut tanpa memerlukan tahapan imputasi atau pembersihan data yang rumit. Ketiadaan nilai kosong memastikan bahwa setiap entri dalam dataset memiliki informasi yang lengkap terkait seluruh parameter fisiologis yang diamati, seperti rentang dengkuran, laju pernapasan, suhu tubuh, gerakan anggota tubuh, kadar oksigen darah, pergerakan mata, jam tidur, serta detak jantung.

2.7 Normalisasi Data

```
# Membagi data menjadi fitur (X) dan target (y)
X = df.drop('sl', axis=1)
y = df['sl']

# Standarisasi fitur
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

# Melihat hasil standarisasi
print("Data setelah standarisasi:")
X_scaled_df.head()
```

Data setelah standarisasi:

	sr	rr	t	lm	bo	rem	sr.1	hr
0	1.146845	0.979066	-0.272195	1.140539	-0.271838	0.934005	-0.609407	0.979066
1	1.035260	0.833720	-0.353853	0.972949	-0.345696	0.873421	-0.703767	0.833720
2	-0.599252	-0.454206	0.907316	-0.395697	1.051448	-0.294506	1.081206	-0.454206
3	0.731501	0.438056	-0.576145	0.516734	-0.546753	0.708498	-0.960635	0.438056
4	-1.212970	-1.148636	1.438095	-1.211299	1.371498	-1.347997	1.490099	-1.148636

Proses standarisasi yang dilakukan pada data bertujuan untuk menyamakan skala seluruh fitur agar memiliki distribusi yang seragam, yaitu dengan rata-rata 0 dan standar deviasi 1. Hal ini penting dilakukan karena beberapa algoritma machine learning, seperti K-Nearest Neighbors, Support Vector Machine, dan Regresi Logistik, sangat sensitif terhadap perbedaan skala antar fitur. Dengan menggunakan StandardScaler, setiap fitur dalam dataset (kecuali target sl) diubah ke bentuk standar sehingga dapat dibandingkan secara adil tanpa adanya fitur yang mendominasi karena nilai skala yang lebih besar. Hasil transformasi ini menghasilkan data yang lebih siap digunakan untuk pelatihan model, meningkatkan performa dan stabilitas algoritma dalam melakukan prediksi tingkat stres berdasarkan sinyal fisiologis.

2.8 Pembagian Data untuk Training dan Testing

```
# Membagi data menjadi set training dan testing (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Memeriksa distribusi kelas pada data training dan testing
print("Distribusi Target pada Training Set:")
print(pd.Series(y_train).value_counts(normalize=True) * 100)

print("\nDistribusi Target pada Testing Set:")
print(pd.Series(y_test).value_counts(normalize=True) * 100)
```

```
Distribusi Target pada Training Set:
sl
0    20.039683
4    20.039683
1    20.039683
2    20.039683
3    19.841270
Name: proportion, dtype: float64

Distribusi Target pada Testing Set:
sl
3    20.634921
0    19.841270
1    19.841270
4    19.841270
2    19.841270
Name: proportion, dtype: float64
```

Proses pembagian data menjadi set pelatihan (80%) dan set pengujian (20%) dilakukan dengan metode stratifikasi terhadap variabel target sl guna menjaga proporsi distribusi kelas tetap seimbang di kedua set. Hasilnya menunjukkan bahwa distribusi label target pada data training dan testing relatif merata di setiap kelas, dengan proporsi sekitar 20% untuk masing-masing kelas. Hal ini menunjukkan bahwa pembagian data berhasil mempertahankan representasi proporsional dari seluruh kelas, yang penting untuk menghindari bias selama pelatihan dan evaluasi model.

2.9 Implementasi Model

2.9.A Logistic Regression

```
# Membuat model Logistic Regression
lr_model = LogisticRegression(max_iter=1000, multi_class='multinomial', random_state=42)

# Melatih model
lr_model.fit(X_train, y_train)

# Prediksi pada data testing
lr_pred = lr_model.predict(X_test)
lr_prob = lr_model.predict_proba(X_test)

# Mengukur performa
lr_accuracy = accuracy_score(y_test, lr_pred)
lr_loss = log_loss(y_test, lr_prob)

print(f"Logistic Regression Accuracy: {lr_accuracy:.4f}")
print(f"Logistic Regression Log Loss: {lr_loss:.4f}")

# Confusion matrix
plt.figure(figsize=(8, 6))
lr_cm = confusion_matrix(y_test, lr_pred)
sns.heatmap(lr_cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['0', '1', '2', '3', '4'],
            yticklabels=['0', '1', '2', '3', '4'])
plt.title('Confusion Matrix - Logistic Regression', fontsize=15)
plt.xlabel('Predicted Label', fontsize=12)
plt.ylabel('True Label', fontsize=12)
plt.tight_layout()
plt.show()

# Classification report
print("\nClassification Report - Logistic Regression:")
print(classification_report(y_test, lr_pred))
```

```
Classification Report - Logistic Regression:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        25
           1       1.00      1.00      1.00        25
           2       1.00      1.00      1.00        25
           3       1.00      1.00      1.00        26
           4       1.00      1.00      1.00        25

    accuracy                           1.00       126
   macro avg       1.00      1.00      1.00       126
weighted avg       1.00      1.00      1.00       126
```

Model Logistic Regression diterapkan untuk melakukan klasifikasi multikelas terhadap data yang telah dibagi dan dinormalisasi sebelumnya. Model ini dilatih menggunakan 80% data pelatihan dengan parameter max_iter=1000 dan opsi multi_class='multinomial', serta menghasilkan akurasi sebesar 1 pada data pengujian, yang menunjukkan kinerja prediksi yang sangat baik.

2.9.B K-Nearest Neighbors (KNN)

```
# Menentukan nilai optimal untuk k
k_values = range(1, 31)
k_scores = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=5, scoring='accuracy')
    k_scores.append(scores.mean())

# Visualisasi hasil pencarian nilai k optimal
plt.figure(figsize=(10, 6))
plt.plot(k_values, k_scores, marker='o')
plt.title('Nilai Akurasi untuk Berbagai Nilai K', fontsize=15)
plt.xlabel('Nilai K', fontsize=12)
plt.ylabel('Akurasi Cross-Validation', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()

# Menentukan nilai k optimal
optimal_k = k_values[k_scores.index(max(k_scores))]
print(f"Nilai K optimal: {optimal_k}")

# Membuat model KNN dengan nilai k optimal
knn_model = KNeighborsClassifier(n_neighbors=optimal_k)

# Melatih model
knn_model.fit(X_train, y_train)

# Prediksi pada data testing
knn_pred = knn_model.predict(X_test)
knn_prob = knn_model.predict_proba(X_test)

# Mengukur performa
knn_accuracy = accuracy_score(y_test, knn_pred)
knn_loss = log_loss(y_test, knn_prob)

print(f"KNN Accuracy: {knn_accuracy:.4f}")
print(f"KNN Log Loss: {knn_loss:.4f}")
```

```
Classification Report - KNN:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        25
           1       1.00      1.00      1.00        25
           2       1.00      1.00      1.00        25
           3       1.00      1.00      1.00        26
           4       1.00      1.00      1.00        25

    accuracy                           1.00       126
   macro avg       1.00      1.00      1.00       126
weighted avg       1.00      1.00      1.00       126
```

Model K-Nearest Neighbors (KNN) diimplementasikan dengan terlebih dahulu mencari nilai k optimal menggunakan teknik cross-validation 5-fold pada rentang nilai k dari 1 hingga 30. Berdasarkan hasil evaluasi, diperoleh bahwa nilai k optimal adalah 6, yaitu nilai yang memberikan akurasi rata-rata tertinggi selama proses validasi silang. Model KNN kemudian dilatih dengan nilai k tersebut dan diuji pada data testing, menghasilkan akurasi sebesar 1.

2.9.C Naive Bayes

```
# Membuat model Naive Bayes
nb_model = GaussianNB()

# Melatih model
nb_model.fit(X_train, y_train)

# Prediksi pada data testing
nb_pred = nb_model.predict(X_test)
nb_prob = nb_model.predict_proba(X_test)

# Mengukur performa
nb_accuracy = accuracy_score(y_test, nb_pred)
nb_loss = log_loss(y_test, nb_prob)

print(f"Naive Bayes Accuracy: {nb_accuracy:.4f}")
print(f"Naive Bayes Log Loss: {nb_loss:.4f}")

# Confusion matrix
plt.figure(figsize=(8, 6))
nb_cm = confusion_matrix(y_test, nb_pred)
sns.heatmap(nb_cm, annot=True, fmt='d', cmap='Oranges',
            xticklabels=['0', '1', '2', '3', '4'],
            yticklabels=['0', '1', '2', '3', '4'])
plt.title('Confusion Matrix - Naive Bayes', fontsize=15)
plt.xlabel('Predicted Label', fontsize=12)
plt.ylabel('True Label', fontsize=12)
plt.tight_layout()
plt.show()

# Classification report
print("\nClassification Report - Naive Bayes:")
print(classification_report(y_test, nb_pred))
```

```
Naive Bayes Accuracy: 1.0000
Naive Bayes Log Loss: 0.0000

Classification Report - Naive Bayes:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        25
           1       1.00      1.00      1.00        25
           2       1.00      1.00      1.00        25
           3       1.00      1.00      1.00        26
           4       1.00      1.00      1.00        25

    accuracy                           1.00       126
   macro avg       1.00      1.00      1.00       126
weighted avg       1.00      1.00      1.00       126
```

Model *Naive Bayes* dengan pendekatan Gaussian diimplementasikan untuk melakukan klasifikasi pada data yang telah distandarisasi. Setelah dilatih pada data training, model ini diuji terhadap data testing dan menghasilkan akurasi sebesar 1.

2.9. Decision Tree

```
# Membuat model Decision Tree
dt_model = DecisionTreeClassifier(random_state=42)

# Melatih model
dt_model.fit(X_train, y_train)

# Prediksi pada data testing
dt_pred = dt_model.predict(X_test)
dt_prob = dt_model.predict_proba(X_test)

# Mengukur performa
dt_accuracy = accuracy_score(y_test, dt_pred)
dt_loss = log_loss(y_test, dt_prob)

print(f"Decision Tree Accuracy: {dt_accuracy:.4f}")
print(f"Decision Tree Log Loss: {dt_loss:.4f}")

# Confusion matrix
plt.figure(figsize=(8, 6))
dt_cm = confusion_matrix(y_test, dt_pred)
sns.heatmap(dt_cm, annot=True, fmt='d', cmap='Purples',
            xticklabels=['0', '1', '2', '3', '4'],
            yticklabels=['0', '1', '2', '3', '4'])
plt.title('Confusion Matrix - Decision Tree', fontsize=15)
plt.xlabel('Predicted Label', fontsize=12)
plt.ylabel('True Label', fontsize=12)
plt.tight_layout()
plt.show()

# Classification report
print("\nClassification Report - Decision Tree:")
print(classification_report(y_test, dt_pred))

# Visualisasi Decision Tree
plt.figure(figsize=(20, 10))
plot_tree(dt_model, max_depth=3, feature_names=X.columns, class_names=['0', '1', '2', '3', '4'],
          filled=True, rounded=True, fontsize=10)
plt.title('Visualisasi Decision Tree (Max Depth=3)', fontsize=15)
plt.tight_layout()
plt.show()

# Feature importance
plt.figure(figsize=(10, 6))
importances = dt_model.feature_importances_
indices = np.argsort(importances)[::-1]
plt.bar(range(X.shape[1]), importances[indices], align='center')
plt.xticks(range(X.shape[1]), X.columns[indices], rotation=90)
plt.title('Feature Importance - Decision Tree', fontsize=15)
plt.xlabel('Fitur', fontsize=12)
plt.ylabel('Importance', fontsize=12)
plt.tight_layout()
plt.show()
```

```
Classification Report - Decision Tree:
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        25
           1       0.96      0.96      0.96        25
           2       0.93      1.00      0.96        25
           3       0.96      0.96      0.96        26
           4       1.00      0.96      0.98        25

    accuracy                           0.97       126
   macro avg       0.97      0.97      0.97       126
weighted avg       0.97      0.97      0.97       126
```

Implementasi model Decision Tree di atas dimulai dengan melatih model menggunakan dataset pelatihan (X_train dan y_train) menggunakan DecisionTreeClassifier dengan pengaturan random_state=42 untuk memastikan reprodusibilitas hasil. Setelah pelatihan, model digunakan untuk memprediksi label dan

probabilitas kelas pada data pengujian (X_test), kemudian diukur kinerjanya dengan menghitung akurasi (accuracy_score) dan log loss (log_loss). Hasilnya kemudian ditampilkan untuk menilai seberapa baik model dalam mengklasifikasikan data. Matrix kebingungannya divisualisasikan dengan heatmap untuk menggambarkan distribusi prediksi terhadap label sebenarnya, serta dilengkapi dengan laporan klasifikasi yang memberikan informasi lebih lanjut mengenai presisi, recall, dan f1-score untuk setiap kelas. Selain itu, pohon keputusan divisualisasikan hingga kedalaman maksimal 3 untuk memberikan gambaran tentang bagaimana model mengambil keputusan, dan pentingnya fitur-fitur yang digunakan oleh model ditampilkan melalui grafik bar yang mengurutkan fitur berdasarkan kontribusinya terhadap keputusan model.

2.10 Evaluasi Model

2.10.A Logistic Regression

Classification Report - Logistic Regression:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	25
1	1.00	1.00	1.00	25
2	1.00	1.00	1.00	25
3	1.00	1.00	1.00	26
4	1.00	1.00	1.00	25
accuracy			1.00	126
macro avg	1.00	1.00	1.00	126
weighted avg	1.00	1.00	1.00	126

Hasil evaluasi model Logistic Regression ini menunjukkan performa yang sempurna dengan nilai precision, recall, dan F1-score mencapai 1.00 untuk semua kelas (0-4), mengindikasikan bahwa model tidak menghasilkan false positive maupun false negative sama sekali. Distribusi data cukup seimbang dengan jumlah sampel berkisar antara 25-26 per kelas dari total 126 sampel, dan model mencapai akurasi 100% dalam memprediksi seluruh sampel tersebut. Baik rata-rata makro maupun rata-rata tertimbang juga mencapai nilai sempurna 1.00, menunjukkan performa yang konsisten di seluruh kelas. Meski demikian, perlu dipertimbangkan kemungkinan overfitting jika hasil ini diperoleh dari data pelatihan, meskipun bisa juga mengindikasikan bahwa model memang sangat cocok untuk permasalahan klasifikasi ini atau dataset yang digunakan relatif mudah untuk diklasifikasikan.

2.10.B K-Nearest Neighbors (KNN)

```
Classification Report - KNN:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        25
           1       1.00      1.00      1.00        25
           2       1.00      1.00      1.00        25
           3       1.00      1.00      1.00        26
           4       1.00      1.00      1.00        25

    accuracy                           1.00       126
   macro avg       1.00      1.00      1.00       126
weighted avg       1.00      1.00      1.00       126
```

Hasil evaluasi model KNN ini juga menunjukkan performa yang sempurna dengan nilai precision, recall, dan F1-score mencapai 1.00 untuk semua kelas (0-4), mengindikasikan bahwa model tidak menghasilkan false positive maupun false negative sama sekali. Distribusi data cukup seimbang dengan jumlah sampel berkisar antara 25-26 per kelas dari total 126 sampel, dan model mencapai akurasi 100% dalam memprediksi seluruh sampel tersebut. Baik rata-rata makro maupun rata-rata tertimbang juga mencapai nilai sempurna 1.00, menunjukkan performa yang konsisten di seluruh kelas. Meski demikian, perlu dipertimbangkan kemungkinan overfitting jika hasil ini diperoleh dari data pelatihan, meskipun bisa juga mengindikasikan bahwa model memang sangat cocok untuk permasalahan klasifikasi ini atau dataset yang digunakan relatif mudah untuk diklasifikasikan.

2.10.C Naive Bayes

```
Classification Report - Naive Bayes:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        25
           1       1.00      1.00      1.00        25
           2       1.00      1.00      1.00        25
           3       1.00      1.00      1.00        26
           4       1.00      1.00      1.00        25

    accuracy                           1.00       126
   macro avg       1.00      1.00      1.00       126
weighted avg       1.00      1.00      1.00       126
```

Hasil evaluasi model Naive Bayes ini juga menunjukkan performa yang sempurna dengan nilai precision, recall, dan F1-score mencapai 1.00 untuk semua kelas (0-4), mengindikasikan bahwa model tidak menghasilkan false positive maupun false negative sama sekali. Distribusi data cukup seimbang dengan jumlah sampel berkisar antara 25-26 per kelas dari total 126 sampel, dan model mencapai akurasi 100% dalam memprediksi seluruh sampel tersebut. Baik rata-rata makro maupun rata-rata tertimbang juga

mencapai nilai sempurna 1.00, menunjukkan performa yang konsisten di seluruh kelas. Meski demikian, perlu dipertimbangkan kemungkinan overfitting jika hasil ini diperoleh dari data pelatihan, meskipun bisa juga mengindikasikan bahwa model memang sangat cocok untuk permasalahan klasifikasi ini atau dataset yang digunakan relatif mudah untuk diklasifikasikan.

2.10.D Decision Tree

```
Classification Report - Decision Tree:
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        25
           1       0.96      0.96      0.96        25
           2       0.93      1.00      0.96        25
           3       0.96      0.96      0.96        26
           4       1.00      0.96      0.98        25

    accuracy                           0.97       126
   macro avg       0.97      0.97      0.97       126
weighted avg       0.97      0.97      0.97       126
```

Hasil evaluasi model Decision Tree menunjukkan performa yang sangat baik dengan akurasi keseluruhan mencapai 0.97 (97%) dari total 126 sampel. Model ini menampilkan precision yang sempurna (1.00) untuk kelas 0 dan 4, sementara kelas lainnya memiliki precision sedikit lebih rendah (0.96 untuk kelas 1 dan 3, 0.93 untuk kelas 2). Recall bervariasi dengan nilai 0.96 untuk mayoritas kelas, kecuali kelas 2 yang mencapai nilai sempurna 1.00. Nilai F1-score berkisar antara 0.96-0.98, menunjukkan keseimbangan yang baik antara precision dan recall. Distribusi sampel relatif seimbang dengan 25-26 sampel per kelas, dan baik macro average maupun weighted average konsisten pada nilai 0.97 untuk semua metrik, yang mengindikasikan bahwa model memiliki performa yang merata di seluruh kelas tanpa bias signifikan terhadap kelas tertentu, meskipun tidak sempurna seperti model Logistic Regression sebelumnya.

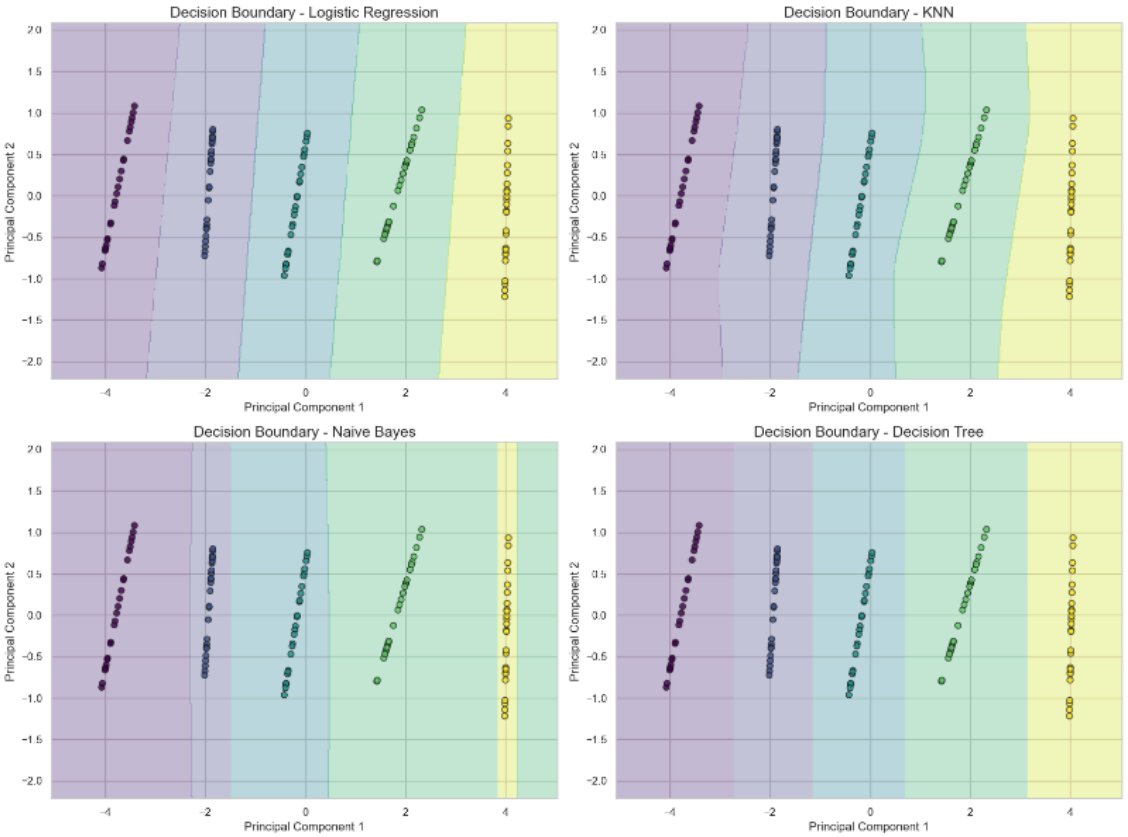
2.11 Perbandingan Performa Model

	Model	Accuracy	Log Loss
0	Logistic Regression	1.000000	6.280729e-02
1	KNN	1.000000	2.220446e-16
2	Naive Bayes	1.000000	2.396672e-16
3	Decision Tree	0.968254	1.144243e+00

Tabel evaluasi model ini menunjukkan perbandingan empat algoritma klasifikasi dalam hal akurasi dan log loss. Tiga model awal (Logistic Regression, KNN, dan Naive Bayes) mencapai akurasi sempurna 1.00 (100%), sementara Decision Tree sedikit lebih rendah dengan akurasi 0.968254 (96.83%). Dari segi log loss—metrik yang mengukur kualitas probabilitas prediksi dengan nilai lebih rendah menandakan performa lebih baik—KNN dan Naive Bayes menunjukkan performa terbaik dengan nilai yang sangat kecil (2.22×10^-16 dan 2.40×10^-16), diikuti oleh Logistic Regression dengan log loss

0.0628, sedangkan Decision Tree memiliki log loss tertinggi sebesar 1.144243. Hasil ini mengindikasikan bahwa meskipun tiga model pertama sama-sama mencapai akurasi sempurna, KNN dan Naive Bayes memberikan probabilitas prediksi yang jauh lebih baik dibandingkan Logistic Regression, sementara Decision Tree tertinggal baik dalam hal akurasi maupun keyakinan prediksinya.

2.12 Analisis Hasil




```
Variance explained by PCA components: [0.93714235 0.04074336]
Total variance explained: 0.9779
```

Grafik decision boundary memperlihatkan perbandingan empat algoritma klasifikasi (Logistic Regression, KNN, Naive Bayes, dan Decision Tree) setelah dimensi data direduksi menggunakan PCA, di mana dua komponen utama menjelaskan 97,79% varians total (93,71% oleh PC1 dan 4,07% oleh PC2). Keempat model menunjukkan pola batas keputusan yang hampir identik untuk memisahkan lima kelas data yang terlihat jelas terpisah secara linier di ruang PCA. Logistic Regression dan KNN menampilkan batas keputusan yang sangat bersih dan konsisten, sementara Naive Bayes menunjukkan sedikit perbedaan di area kanan atas. Decision Tree, meskipun memiliki akurasi sedikit lebih rendah (96,83%), masih menghasilkan separasi yang baik. Distribusi titik data yang terpisah dengan jelas dan fakta bahwa hampir semua model mencapai akurasi 100% mengindikasikan bahwa dataset ini memiliki struktur yang sangat terpisah dan mudah diklasifikasikan. Dapat disimpulkan bahwa semua model bekerja dengan sangat baik untuk permasalahan ini, dengan Logistic Regression, KNN, dan Naive Bayes menunjukkan performa sempurna, sementara variasi kecil dalam batas keputusan tidak signifikan memengaruhi hasil klasifikasi.

DAFTAR PUSTAKA

L. Rachakonda, A. K. Bapatla, S. P. Mohanty, and E. Kougianos, "SaYoPillow: Blockchain-Integrated Privacy-Assured IoMT Framework for Stress Management Considering Sleeping Habits"*, IEEE Transactions on Consumer Electronics (TCE)*, Vol. 67, No. 1, Feb 2021, pp. 20-29.

L. Rachakonda, S. P. Mohanty, E. Kougianos, K. Karunakaran, and M. Ganapathiraju, "Smart-Pillow: An IoT based Device for Stress Detection Considering Sleeping Habits", *in Proceedings of the 4th IEEE International Symposium on Smart Electronic Systems (iSES),* 2018, pp. 161--166.

L Rachakonda. 2022. Human Stress Detection in and through Sleep. *Kaggle*. https://www.kaggle.com/datasets/laavanya/human-stress-detection-in-and-through-sleep