Jennatul Macwe

Report Proyek Sain Data

To blah, blah, and blah.

Table of contents

Introduction **v**

Introduction **v**

Proyek Sains Data **vii**

Proyek Sains Data **vii**
langkah-langkah umum dalam Proyek Sains Data: vii

Tic-Tac-Toe EndGame (Memprediksi apakah permainan akan dimenangkan oleh pemain 'x' atau tidak) **ix**

Bussiness Understanding **xi**

Data Understanding **xiii**
0.1 **PENJELASAN DATASET :** xiii
0.2 **PENJELASAN FITUR** xv
0.3 **SUMBER DATASET:** xv
0.4 **Eksplorasi Data** xvi
0.4.1 **Jumlah per kelas** xvi
0.4.2 **Statistik ringkasan dari Dataset** xvii
0.4.3 **Missing Values** xviii
0.4.4 **Distribusi nilai fitur pada dataset** xix

Pre-Processing **xxi**

K-NN (k-Nearest Neighbors) **xxvii**

Naive Bayes **xxix**

Logistic Regression **xxxi**

Decision Tree **xxxiii**

Model dengan Akurasi Tertinggi (Decission Tree) **xxxv**

DEPLOYMENT **xxxvii**

Memprediksi Audio **xxxix**

DATASET **xli**
0.5 PENJELASAN DATASET xli
0.6 FITUR xli
0.7 SUMBER DATASET xlii

PRE-PROSESING **xlv**

Split Data **xlvii**
0.8 Normalisasi xlvii

Model K-Neirest Neighbors **xlix**
0.9 Penjelasan K-NN xlix
0.10 Langkah-Langkah K-NN: xlix
0.11 Rumus Perhitungan Jarak Euclidean l
0.12 REDUKSI DATA liv

Introduction

Proyek Sains Data

Proyek Sains Data adalah rangkaian kegiatan yang bertujuan untuk menerapkan metode sains data guna memahami, menganalisis, dan mengekstraksi pengetahuan atau informasi berharga dari data. Proyek-proyek ini seringkali melibatkan beberapa tahapan, mulai dari pengumpulan data, pembersihan data, analisis, hingga pembuatan model prediktif yang dapat digunakan untuk membuat keputusan atau mengidentifikasi pola.

langkah-langkah umum dalam Proyek Sains Data:

1. Pemahaman Bisnis (Business Understanding):

Meng-Identifikasi tujuan bisnis dan masalah yang ingin dipecahkan dan Memahami konteks bisnis, konstrain, dan kebutuhan pemangku kepentingan.

---

2. Pengumpulan Data:

Meng-Identifikasi dan kumpulkan data yang relevan dengan permasalahan. Data dapat berasal dari berbagai sumber, termasuk database internal, sumber eksternal, atau bahkan data yang dihasilkan oleh sensor atau perangkat IoT.

---

3. Pembersihan Data (Data Cleaning):

Meng-Identifikasi dan menangangani missing values (nilai yang hilang) dan outliers (data yang ekstrim atau tidak biasa).

---

4. Eksplorasi Data (Data Exploration):

Meng-Analisa statistik deskriptif

---

5. Pemodelan Data:

Memilih model atau algoritma yang sesuai kemudian melatih model dengan menggunakan data lati, lalu Evaluasi model menggunakan data uji.

---

6. Implementasi:

Meng-Implementasikan solusi berdasarkan temuan atau model yang dihasilkan. Me-Monitor performa solusi dan lakukan perbaikan jika diperlukan.

0

Tic-Tac-Toe EndGame (Memprediksi apakah permainan akan dimenangkan oleh pemain 'x' atau tidak)

```
#mneghubungkan ke drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/dr
```

```
cd /content/drive/MyDrive/PSD/tugas
```

```
/content/drive/MyDrive/PSD/tugas
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
```

0

Bussiness Understanding

TUJUAN ANALISIS DATA : Analisis data Tic-TacToe EndGame ini untuk memprediksi apakah permainan akan dimenangkan oleh pemain 'x' atau tidak. 'win for x,' yang berarti pemain 'x' harus berhasil menciptakan salah satu dari 8 kemungkinan cara untuk mencapai 'three-in-a-row,' yaitu menempatkan tiga tanda 'x' secara berurutan dalam satu baris, kolom, atau diagonal.

0

Data Understanding

0.1 PENJELASAN DATASET :

Dataset ini merupakan kumpulan lengkap konfigurasi papan di akhir permainan tic-tac-toe, di mana 'x' diasumsikan bermain terlebih dahulu. Konsep targetnya adalah 'kemenangan untuk x' (membuat 'tiga berjejer').

Dalam permainan Tic-Tac-Toe, ada dua pemain, biasanya disebut 'x' dan 'o,' yang bergiliran menempatkan tanda mereka di papan permainan 3x3. Tujuan permainan adalah mencapai 'win for x,' yang berarti pemain 'x' harus berhasil menciptakan salah satu dari 8 kemungkinan cara untuk mencapai 'three-in-a-row,' yaitu menempatkan tiga tanda 'x' secara berurutan dalam satu baris, kolom, atau diagonal.

Deskripsi tersebut mencatat bahwa Dataset ini mencakup semua konfigurasi papan pada akhir permainan Tic-Tac-Toe, dengan asumsi bahwa 'x' selalu bermain pertama.

Ini adalah tugas klasifikasi di mana model pembelajaran mesin mencoba memprediksi hasil permainan, yaitu apakah 'x' akan memenangkan permainan atau tidak.

1. **Tipe Data** > Tipe data dari Dataset ini adalah Categorical
2. **Subject Area** > Dataset ini digunakan dalam Game Tic-Tac-Toe
3. **jumlah data** : 958 bari > Dataset ini berisi 958 permainan yang mencakup semua konfigurasi papan pada akhir permainan Tic-Tac-Toe
4. **Missing Value** >Dataset ini TIDAK MEMILIKI Missing Values** yang berarti bahwa setiap bagian data pada dataset lengkap dengan informasi.

Dataset ini menggunakan Class sebagai Targetnya dimana Fitur ini menunjukan apakah akhir dari permainan dimenangkan oleh 'x' atau tidak.

DATASET

```
url = "https://raw.githubusercontent.com/jennamacwe/ProyekSainData/main/tic-tac-toe_Endgame2.csv
dataset = pd.read_csv(url, header=None)

# Menambahkan kolom
dataset.columns = ["top-left-square", "top-middle-square", "top-right-square", "middle-left-squa
# Menampilkan dataset dengan kolom tambahan
print(dataset)
```

```
   top-left-square top-middle-square top-right-square middle-left-square  \
0                x                 x                 x                   x
1                x                 x                 x                   x
2                x                 x                 x                   x
3                x                 x                 x                   x
4                x                 x                 x                   x
..             ...               ...               ...                 ...
953              o                 x                 x                   x
954              o                 x                 o                   x
955              o                 x                 o                   x
956              o                 x                 o                   o
957              o                 o                 x                   x

    middle-middle-square middle-right-square bottom-left-square  \
0                      o                   o                 x
1                      o                   o                 o
2                      o                   o                 o
3                      o                   o                 o
4                      o                   o                 b
..                   ...                 ...               ...
953                    o                   o                 o
954                    x                   o                 x
955                    o                   x                 x
956                    x                   x                 x
957                    x                   o                 o

    bottom-middle-square bottom-right-square     Class
0                      o                   o  positive
1                      x                   o  positive
2                      o                   x  positive
3                      b                   b  positive
4                      o                   b  positive
..                   ...                 ...       ...
953                    x                   x  negative
954                    o                   x  negative
955                    o                   x  negative
```

```
956                   o                   x  negative
957                   x                   x  negative

[958 rows x 10 columns]
```

0.2 PENJELASAN FITUR

```
# Menampilkan kolom pada dataset
print(dataset.columns)
```

```
Index(['top-left-square', 'top-middle-square', 'top-right-square',
     'middle-left-square', 'middle-middle-square', 'middle-right-square',
     'bottom-left-square', 'bottom-middle-square', 'bottom-right-square',
       'Class'],
      dtype='object')
```

Jumlah Fitur pada dataset Permainan Tic-Tac-toe ini sebanyak 9 Fitur dimana setiap fitur mempresentasikan setiap kotak pada permainan

- top-left-square : kotak pada bagian kiri baris teratas
- top-middle-square : kotak pada bagian tengah baris teratas
- top-right-square : kotak pada bagian kanan baris teratas
- middle-left-square : kotak pada bagian kiri baris tengah
- middle-middle-square : kotak pada bagian tengah baris tengah
- middle-right-square : kotak pada bagian kanan baris tengah
- bottom-left-square : kotak pada bagian kiri baris bawah
- bottom-middle-square : kotak pada bagian tengah baris bawah
- bottom-righ-square : kotak pada bagian kiri baris bawah

0.3 SUMBER DATASET:

Dataset didapatkan dari UCI Machine Learning Repository dan dapat diakses melalui link di bawah ini:

https://archive.ics.uci.edu/dataset/101/tic+tac+toe+endgam

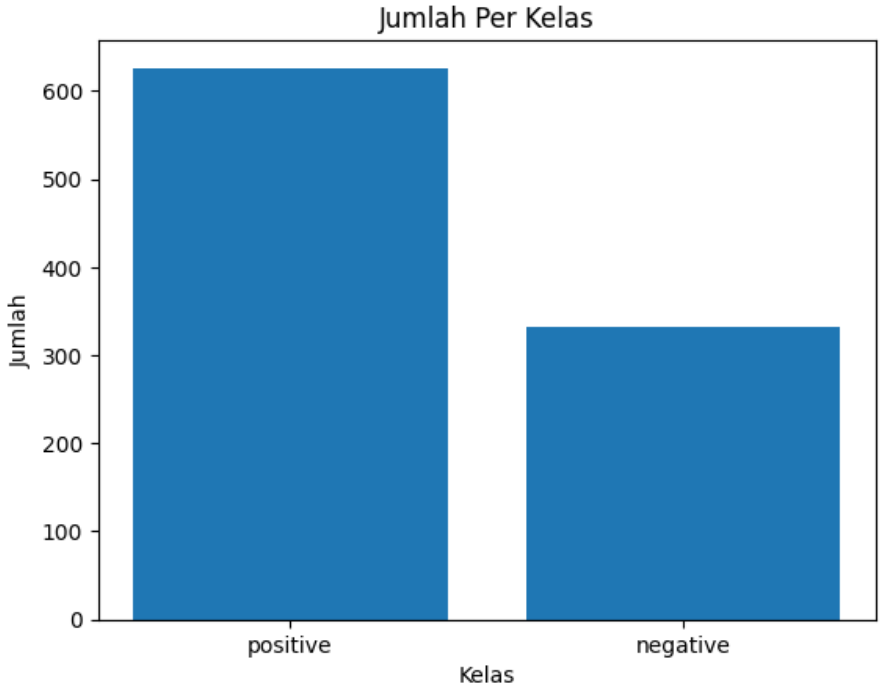
0.4 Eksplorasi Data

0.4.1 Jumlah per kelas

Menampilkan grafik jumlah per kelas (negatif dan positif) dari dataset dan Jumlah masing-masing Kelas untuk mengetahui kelas mana yang terbanyak. Jumlah data per Kelas beserta Grafik ditampilkan dibawah ini:

```
# Menghitung jumlah per kelas
class_counts = dataset['Class'].value_counts()

# Print jumlah per kelas
for class_label, count in class_counts.items():
    print(f'Kelas {class_label}: {count} data')

print("\n")

# Plot grafik jumlah per kelas
plt.bar(class_counts.index, class_counts.values)
plt.xlabel('Kelas')
plt.ylabel('Jumlah')
plt.title('Jumlah Per Kelas')
plt.show()

# Menghitung jumlah per kelas
# class_counts = dataset['Class'].value_counts()
```

```
Kelas positive: 626 data
Kelas negative: 332 data
```



0.4.2 Statistik ringkasan dari Dataset

Menampilkan gambaran umum tentang distribusi nilai dalam setiap kolom kategorikal pada dataset.

```
# Summary statistics
print("\nSummary statistics:")
print(dataset.describe(include='all'))
```

```
Summary statistics:
     top-left-square top-middle-square top-right-square middle-left-square  \
count             958                958              958                958
unique              3                  3                3                  3
top                 x                  x                x                  x
freq              418                378              418                378

       middle-middle-square middle-right-square bottom-left-square  \
count                   958                 958                958
unique                    3                   3                  3
top                       x                   x                  x
```

```
freq                    458                 378                 418

       bottom-middle-square bottom-right-square     Class
count                   958                 958       958
unique                    3                   3         2
top                       x                   x  positive
freq                    378                 418       626
```

Berikut adalah tafsir dari beberapa bagian dari output statistik ringkasan:

1. Count (Jumlah): Menunjukkan jumlah entri non-null (tidak kosong) dalam setiap kolom.
2. Unique (Unik): Menunjukkan jumlah nilai unik dalam setiap kolom.
3. Top (Teratas): Menunjukkan nilai yang paling sering muncul dalam setiap kolom.
4. Freq (Frekuensi): Menunjukkan frekuensi kemunculan nilai teratas dalam setiap kolom.

0.4.3 Missing Values

MEnampilkan apakah terdapat nilai yang hilang (missing values) dalam dataset.

Ini adalah langkah yang penting dalam eksplorasi data karena nilai yang hilang dapat mempengaruhi hasil analisis dan model yang dibangun.

```
# Checking for missing values
print("Missing values:")
print(dataset.isnull().sum())
```

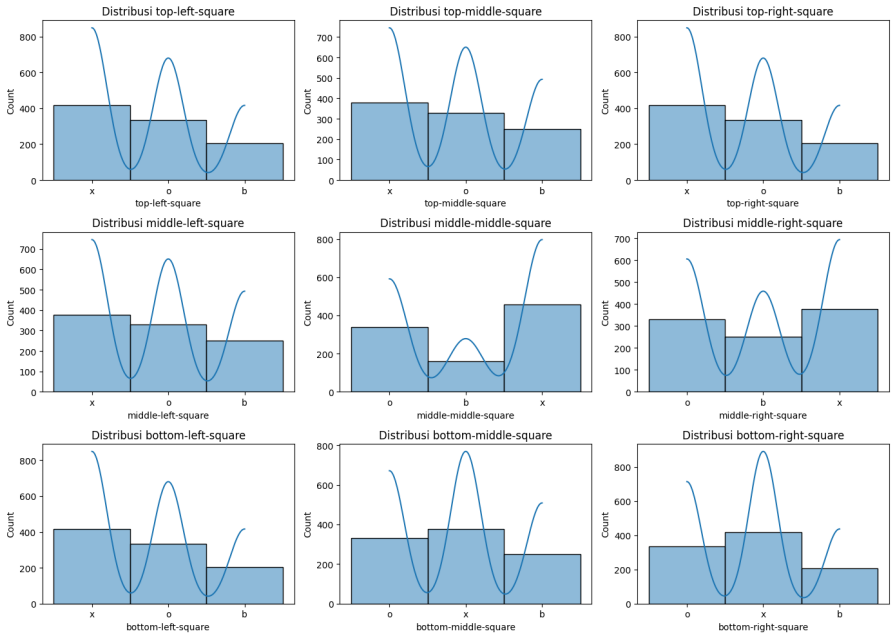
```
Missing values:
top-left-square         0
top-middle-square       0
top-right-square        0
middle-left-square      0
middle-middle-square    0
middle-right-square     0
bottom-left-square      0
bottom-middle-square    0
bottom-right-square     0
Class                   0
dtype: int64
```

Pada output yang di hasilkan terlihat bahwa tidak ada Missing Values dalam Dataset Permainan Tic-Tac-Toe

0.4.4 Distribusi nilai fitur pada dataset

Menampilkan serangkaian subplot yang masing-masing menunjukkan distribusi nilai untuk setiap fitur dalam dataset. Ini membantu memahami sebaran nilai dan pola distribusi untuk masing-masing kategori atau atribut dalam dataset permainan tic-tac-toe.

```
# Analisis distribusi nilai fitur
plt.figure(figsize=(14, 10))  # Membuat gambar dengan ukuran 14x10 inci untuk visualisasi

for i, feature in enumerate(dataset.columns[:-1], 1):  # Iterasi melalui setiap fitur kecuali k
    plt.subplot(3, 3, i)  # Membuat subplot dalam grid 3x3 untuk setiap fitur
    sns.histplot(dataset[feature], kde=True)  # Menggunakan seaborn untuk membuat histogram fitu
    plt.title(f"Distribusi {feature}")  # Menambahkan judul subplot dengan nama fitur untuk mem

plt.tight_layout()  # Mengoptimalkan tata letak subplot agar tidak tumpang tindih
plt.show()  # Menampilkan visualisasi distribusi nilai fitur
```



0

Pre-Processing

Karena Dataset bertipe Categorical maka akan diubah menjadi numerik menggunakan Label Encoding

Label encoding adalah suatu metode dalam pra-pemrosesan data yang melibatkan penggantian nilai-nilai kategori pada suatu fitur dengan nilai-nilai numerik yang unik.

Mengubah categorical menjadi numerik menggunakan Label Encoding dilakukan karena sebagian besar algoritma machine learning memerlukan input yang bersifat numerik. Beberapa algoritma, terutama yang berbasis pada perhitungan jarak (seperti k-Nearest Neighbors), dapat memberikan hasil yang lebih baik jika nilai kategorikal diubah menjadi bentuk numerik. Hal ini karena perhitungan jarak lebih mudah dilakukan pada data numerik.

catatan : Karena fitur-fitur ini berada dalam rentang yang sangat terbatas (hanya 1, 0, dan -1), normalisasi tidak diperlukan. Normalisasi biasanya digunakan pada data numerik yang memiliki rentang nilai yang berbeda agar fitur-fitur tersebut memiliki skala yang serupa.

```
# Memisahkan fitur (X) dan target (y)
X = dataset.drop('Class', axis=1)
y = dataset['Class']

# Membagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=670, test_size=288, random
```

```
# Mengimpor data CSV tanpa nama kolom (header)
data = pd.read_csv(url, header=None)

# Memberikan kolom pada dataset
column_names = ["top-left-square", "top-middle-square", "top-right-square", "middle-left-square
data.columns = column_names

# Mapping karakter ke angka (label encoding)
char_to_num = {'x': 1, 'o': 0, 'b': -1}
```

```
# Melakukan encoding untuk data train
encoded_data_train = []

# Mengonversi setiap baris dalam DataFrame X_train menjadi karakter dan menyimpannya dalam enco
for index, row in X_train.iterrows():
    encoded_row = [char_to_num[c] for c in row]
    encoded_data_train.append(encoded_row)

# Membuat DataFrame dari data train yang telah diencode
encoded_df_train = pd.DataFrame(encoded_data_train, columns=X_train.columns)

# Menampilkan DataFrame data train yang sudah diencode
print("Data Train yang sudah diencode:")
print(encoded_df_train)
```

```
Data Train yang sudah diencode:
    top-left-square  top-middle-square  top-right-square  middle-left-square  \
0                 1                 -1                 1                   0
1                 0                  1                 0                   1
2                 1                 -1                 1                  -1
3                 1                  0                -1                   0
4                 1                  1                 0                   1
..              ...                ...               ...                 ...
665               1                  1                -1                   0
666               1                 -1                 0                  -1
667               0                 -1                 1                   0
668               0                 -1                 1                   0
669               1                  1                -1                   1

     middle-middle-square  middle-right-square  bottom-left-square  \
0                       1                    0                   1
1                       1                    1                   0
2                       1                   -1                   0
3                       1                   -1                   0
4                       0                    1                   1
..                    ...                  ...                 ...
665                     1                    0                   0
666                     1                    0                   0
667                     1                   -1                   0
668                     1                    0                   1
669                     0                    0                   1

     bottom-middle-square  bottom-right-square
0                      -1                    0
```

```
1                      1                    0
2                      0                    0
3                      1                    1
4                      0                    0
..                   ...                  ...
665                   -1                    1
666                    1                    1
667                   -1                    1
668                   -1                    1
669                   -1                    0

[670 rows x 9 columns]
```

```
# Melakukan encoding untuk data test
encoded_data_test = []

# Mengonversi setiap baris dalam DataFrame X_train menjadi karakter dan menyimpannya dalam enco
for index, row in X_test.iterrows():
    encoded_row = [char_to_num[c] for c in row]
    encoded_data_test.append(encoded_row)

# Membuat DataFrame dari data test yang telah diencode
encoded_df_test = pd.DataFrame(encoded_data_test, columns=X_test.columns)

# Menampilkan DataFrame data test yang sudah diencode
print("Data Test yang sudah diencode:")
print(encoded_df_test)
```

```
Data Test yang sudah diencode:
    top-left-square  top-middle-square  top-right-square  middle-left-square  \
0                 0                  0                 0                  -1
1                 0                 -1                -1                   1
2                 0                  1                -1                   1
3                -1                  1                 0                   1
4                 0                 -1                 1                   1
..              ...                ...               ...                 ...
283              -1                  1                 0                   0
284               0                  1                -1                   0
285               1                  1                 0                   1
286               0                  0                 1                  -1
287               0                  1                 1                   1

     middle-middle-square  middle-right-square  bottom-left-square  \
0                       1                    1                   1
1                       1                    1                  -1
```

```
2                       1                    -1                    0
3                       1                     0                    1
4                       0                    -1                   -1
..                    ...                   ...                  ...
283                     1                     1                    0
284                     0                     1                    0
285                     0                    -1                    0
286                    -1                     1                    0
287                     0                     1                    0

     bottom-middle-square  bottom-right-square
0                       0                    1
1                      -1                    0
2                       1                    0
3                       0                    0
4                       1                    0
..                    ...                  ...
283                     1                   -1
284                     1                    1
285                     1                    0
286                     1                    1
287                    -1                    0

[288 rows x 9 columns]
```

menyimpan dataset yang telah di encoding ke csv dan pickle

```
import pickle

# Menyimpan DataFrame data train dan data test yang sudah diencode ke file CSV
encoded_df_train.to_csv("encoded_data_train.csv", index=False)
encoded_df_test.to_csv("encoded_data_test.csv", index=False)

# Menyimpan data train yang sudah diencode ke dalam file pickle
with open('encoded_data_train.pkl', 'wb') as file:
    pickle.dump(encoded_df_train, file)

# Menyimpan data test yang sudah diencode ke dalam file pickle
with open('encoded_data_test.pkl', 'wb') as file:
    pickle.dump(encoded_df_test, file)

# print("Data yang sudah diencoding telah disimpan ke CSV")
```

```
# Membaca data train yang sudah diencode dari file pickle
with open('encoded_data_train.pkl', 'rb') as file:
    loaded_encoded_df_train = pickle.load(file)

# Membaca data test yang sudah diencode dari file pickle
with open('encoded_data_test.pkl', 'rb') as file:
    loaded_encoded_df_test = pickle.load(file)
```

0

K-NN (k-Nearest Neighbors)

```
# Membangun model K-NN
# Hitung akurasi KNN dari k = 1 hingga 30
import numpy as np

k = 30
acc = np.zeros((k - 1))

for n in range(1, k, 2):
    knn = KNeighborsClassifier(n_neighbors=n, metric="euclidean").fit(loaded_encoded_df_train, 
    y_pred = knn.predict(loaded_encoded_df_test)
    acc[n - 1] = accuracy_score(y_test, y_pred)

best_accuracy = acc.max()
best_k = acc.argmax() + 1

print('Akurasi KNN terbaik adalah', best_accuracy, 'dengan nilai k =', best_k)

results_knn = pd.DataFrame({'Actual Label': y_test, 'Prediksi': y_pred})
results_knn
```

```
Akurasi KNN terbaik adalah 0.8472222222222222 dengan nilai k = 5
```

	Actual Label	Prediksi
836	negative	positive
477	positive	positive
350	positive	positive
891	negative	negative
855	negative	negative
...	...	...
501	positive	positive
796	negative	negative
634	negative	positive
405	positive	positive
741	negative	negative

0

Naive Bayes

```
# Inisialisasi model Naive Bayes Multinomial
nb = GaussianNB()
nb.fit(loaded_encoded_df_train, y_train)

# Lakukan prediksi pada data uji
y_pred = nb.predict(loaded_encoded_df_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print(f'Akurasi: {accuracy:.2f}')

results_nb = pd.DataFrame({'Actual Label': y_test, 'Prediksi': y_pred})
results_nb
```

```
Akurasi: 0.75
```

	Actual Label	Prediksi
836	negative	positive
477	positive	positive
350	positive	positive
891	negative	positive
855	negative	positive
...	...	...
501	positive	positive
796	negative	negative
634	negative	negative
405	positive	positive
741	negative	negative

0

Logistic Regression

```
# Inisialisasi model Logistic Regression
logreg_model = LogisticRegression()

# Melatih model menggunakan data train yang sudah diencode
logreg_model.fit(loaded_encoded_df_train, y_train)

# Membuat prediksi menggunakan data test yang sudah diencode
y_pred = logreg_model.predict(loaded_encoded_df_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
results_lr = pd.DataFrame({'Actual Label': y_test, 'Prediksi': y_pred})
results_lr
```

```
Accuracy: 0.6909722222222222
```

	Actual Label	Prediksi
836	negative	positive
477	positive	positive
350	positive	positive
891	negative	positive
855	negative	positive
...	...	...
501	positive	positive
796	negative	negative
634	negative	positive
405	positive	positive
741	negative	positive

0

Decision Tree

Decision tree adalah model prediktif dalam analisis data yang menggunakan struktur pohon untuk menentukan keputusan . Decision tree sering digunakan dalam klasifikasi dan regresi. Decision tree dengan teknik regresi yang paling terkenal yaitu CART (Classification and Regression Tree) yang diperkenalkan oleh Professor Breimann.

Ada beberapa tahapan yang harus dilakukan untuk membuat sebuah pohon keputusan, yaitu:

1. Menyiapkan data training yang sudah dikelompokkan ke dalam kelas-kelas tertentu.
2. Menentukan akar dari pohon keputusan. > Akar akan diambil dari atribut yang terpilih, dengan cara menghitung nilai gain dari masing-masing atribut, nilai gain yang paling tinggi yang akan menjadi akar pertama. Tapi, sebelum menghitung nilai gain kita harus menghitung nilai entropynya terlebih dahulu (note : atribut = kolom.) Untuk menghitungnya dapat digunakan rumus berikut:

$$Entropy(S) = \sum_{i=1}^{n} -pi.log_2 pi$$

Keterangan:

S = himpunan kasus

n = jumlah partisi S

pi = proporsi Si terhadap S

3. Kemudian, hitung nilai gain menggunakan rumus:

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|Si|}{|S|} * Entropy(Si)$$

Keterangan:

S = himpunan kasus

A = fitur

n = jumlah partisi atribut A

|Si| = proporsi Si terhadap S

|S| = jumlah kasus dalam S

```
from sklearn.tree import DecisionTreeClassifier

# Membangun model Decision Tree
dt = DecisionTreeClassifier()
dt.fit(loaded_encoded_df_train, y_train)

# Melakukan prediksi
y_pred = dt.predict(loaded_encoded_df_test)

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print(f'Akurasi: {accuracy:.2f}')


results_compare = pd.DataFrame({'Actual Label': y_test, 'Prediksi': y_pred})
results_compare
```

```
Akurasi: 0.92
```

	Actual Label	Prediksi
836	negative	negative
477	positive	positive
350	positive	positive
891	negative	negative
855	negative	negative
...	...	...
501	positive	positive
796	negative	negative
634	negative	negative
405	positive	positive
741	negative	negative

0

Model dengan Akurasi Tertinggi (Decission Tree)

```
# Membangun model Decision Tree
dt = DecisionTreeClassifier()
dt.fit(loaded_encoded_df_train, y_train)

# Input data untuk prediksi
user_inputs = {}
for feature in loaded_encoded_df_train.columns:
    user_input = input(f"Masukkan nilai untuk '{feature}': ")
    user_inputs[feature] = char_to_num[user_input]

# Membuat DataFrame dari input pengguna
new_data = pd.DataFrame([user_inputs])

# Melakukan prediksi CLASS untuk data baru menggunakan model yang dimuat
prediction = dt.predict(new_data)

# Menampilkan hasil prediksi
predicted_class = prediction[0]  # Menggunakan indeks 0 karena kita hanya memprediksi satu data

# Menampilkan hasil prediksi
if predicted_class == 'positive':
    print("Hasil Prediksi: Positif (Menang)")
else:
    print("Hasil Prediksi: Negatif (Kalah)")
```

```
Masukkan nilai untuk 'top-left-square': x
Masukkan nilai untuk 'top-middle-square': o
Masukkan nilai untuk 'top-right-square': b
Masukkan nilai untuk 'middle-left-square': b
Masukkan nilai untuk 'middle-middle-square': x
Masukkan nilai untuk 'middle-right-square': o
Masukkan nilai untuk 'bottom-left-square': x
Masukkan nilai untuk 'bottom-middle-square': o
```

```
Masukkan nilai untuk 'bottom-right-square': x
Hasil Prediksi: Positif (Menang)
```

0

DEPLOYMENT

Deploy Streamlit[1]

[1] https://https://tic-tac-toe-endgame.streamlit.app

0

Memprediksi Audio

```
pip install librosa
```

```
Requirement already satisfied: librosa in /usr/local/lib/python3.10/dist-packages (0.10.1)
Requirement already satisfied: audioread>=2.1.9 in /usr/local/lib/python3.10/dist-packages (from libr
Requirement already satisfied: numpy!=1.22.0,!=1.22.1,!=1.22.2,>=1.20.3 in /usr/local/lib/python3.10
Requirement already satisfied: scipy>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from librosa)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: joblib>=0.14 in /usr/local/lib/python3.10/dist-packages (from librosa)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from libr
Requirement already satisfied: numba>=0.51.0 in /usr/local/lib/python3.10/dist-packages (from librosa
Requirement already satisfied: soundfile>=0.12.1 in /usr/local/lib/python3.10/dist-packages (from lib
Requirement already satisfied: pooch>=1.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (
Requirement already satisfied: soxr>=0.3.2 in /usr/local/lib/python3.10/dist-packages (from librosa)
Requirement already satisfied: typing-extensions>=4.1.1 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: lazy-loader>=0.1 in /usr/local/lib/python3.10/dist-packages (from libr
Requirement already satisfied: msgpack>=1.0 in /usr/local/lib/python3.10/dist-packages (from librosa)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: platformdirs>=2.5.0 in /usr/local/lib/python3.10/dist-packages (from p
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from pooch
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from pooc
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.10/dist-packages (from soundfile>=
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0->
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from re
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from re
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
cd /content/drive/MyDrive/PSD/Emosi
```

```
/content/drive/MyDrive/PSD/Emosi
```

```
# Import Library
import os
import librosa
import numpy as np
import pandas as pd
from scipy.stats import skew, kurtosis, mode

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from pickle import dump

import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

0

DATASET

Sinyal audio emosi adalah data audio yang mengandung informasi tentang ekspresi emosi seseorang yang terdengar melalui suara, seperti intonasi suara, tempo, kecepatan bicara, pitch, dan berbagai karakteristik lainnya. Dalam konteks mata kuliah proyek sains data, analisis sinyal audio emosi adalah salah satu aplikasi penting dari pemrosesan sinyal digital dan pemelajaran mesin untuk memahami dan menggambarkan emosi dalam data suara.

0.5 PENJELASAN DATASET

Dataset audio ini terdapat 200 kata target yang diucapkan dalam frasa oleh dua aktris (berusia 26 dan 64 tahun). Rekaman dibuat dari set tersebut yang menggambarkan tujuh emosi (marah, jijik, takut, bahagia, kejutan, menyenangkan, kesedian dan netral).

Dataset ini terdiri dari 2746 data dengan 14 kelas dimana dua aktor merekan masing-masing 7 emosi.

Data set ini adalah data set sinyal audio yang telah dilakukan perhitungan statistika. Di mana data yang digunakan sebanyak 2810. Terdapat 10 fitur dalam perhitungan data ini, diantaranya yaitu ZCR Mean, ZCR Median, ZCR Standar Deviasi, ZCR Kurtosis, ZCR Skewness, RMSE, RMSE Median, RMSE Standar Deviasi, RMSE Kurtosis, dan RMSE Skewness.

0.6 FITUR

Pada dataset ini menjadikan perhitungan statistika sebagai fitur, dimana fitur-fitur tersebut terdiri sebanyak 10 fitur yaitu:

- zcr_mean
- zcr_median

- zcr_std_dev
- zcr_kurtosis
- zcr_skew
- rmse
- rmse_median
- rmse_std_dev
- rmse_kurtosis
- rmse_skew

0.7 SUMBER DATASET

Dataset Audio ini dapat dilihat di kaggle melalui link di bawah ini:

https://www.kaggle.com/datasets/ejlok1/toronto-emotional-speech-set-tess?resource=download

```
folders=['YAF_sad','YAF_pleasant_surprised','YAF_neutral',
         'YAF_happy','YAF_fear','YAF_disgust','YAF_angry',
         'OAF_Sad','OAF_Pleasant_surprise','OAF_neutral',
         'OAF_happy','OAF_Fear','OAF_disgust',
         'OAF_angry',
         ]
```

```
# Import library yang dibutuhkan
import os
import librosa
import numpy as np
import pandas as pd
from scipy.stats import skew, kurtosis, mode

# Fungsi untuk menghitung statistik audio
def calculate_statistics(audio_path):
    # Membaca file audio menggunakan librosa
    y, sr = librosa.load(audio_path)

    # UNTUK MENGHITUNG NILAI STATISTIKA
    mean = np.mean(y) # Rata-rata
    std_dev = np.std(y) # Standar deviasi
```

```
    max_value = np.max(y) # Nilai maksimum
    min_value = np.min(y) # Nilai minimum
    median = np.median(y) # Nilai tengah
    skewness = skew(y)  # Menghitung skewness
    kurt = kurtosis(y)  # Menghitung kurtosis
    q1 = np.percentile(y, 25)  # Kuartil pertama
    q3 = np.percentile(y, 75)  # Kuartil ketiga
    mode_value, _ = mode(y)  # Menghitung mode
    iqr = q3 - q1  # Rentang interkuartil

    # UNTUK MENGHITUNG NILAI ZCR (Zero Crossing Rate)
    zcr_mean = np.mean(librosa.feature.zero_crossing_rate(y=y))
    zcr_median = np.median(librosa.feature.zero_crossing_rate(y=y))
    zcr_std_dev = np.std(librosa.feature.zero_crossing_rate(y=y))
    zcr_kurtosis = kurtosis(librosa.feature.zero_crossing_rate(y=y)[0])
    zcr_skew = skew(librosa.feature.zero_crossing_rate(y=y)[0])

    # UNTUK MENGHITUNG NILAI RMSE (Root Mean Square Error)
    rmse = np.sum(y**2) / len(y)
    rmse_median = np.median(y**2)
    rmse_std_dev = np.std(y**2)
    rmse_kurtosis = kurtosis(y**2)
    rmse_skew = skew(y**2)

    return [zcr_mean, zcr_median, zcr_std_dev, zcr_kurtosis, zcr_skew, rmse, rmse_median, rmse_

# Inisialisasi list untuk menyimpan fitur-fitur
features = []

# Loop melalui folder dan file audio
for folder in folders:
    folder_path = f'{folder}'
    for filename in os.listdir(folder_path):
        if filename.endswith('.wav'):
            audio_path = os.path.join(folder_path, filename)
            # Menghitung statistik audio menggunakan fungsi yang telah dibuat
            statistics = calculate_statistics(audio_path)
            # Menambahkan label folder dan nama file ke fitur-fitur
            features.append([folder, filename] + statistics)

# Membuat DataFrame dari data fitur
columns =  ['Label', 'File'] + ['ZCR Mean', 'ZCR Median', 'ZCR Std Dev', 'ZCR Kurtosis', 'ZCR S
df = pd.DataFrame(features, columns=columns)
```

```
# Menampilkan DataFrame sebagai file CSV
df
```

	Label	File	ZCR Mean	ZCR Median	ZCR Std Dev	ZCR Kurtosi
0	YAF_sad	YAF_sour_sad.wav	0.167885	0.042480	0.239094	0.742978
1	YAF_sad	YAF_red_sad.wav	0.117450	0.035156	0.206319	4.820082
2	YAF_sad	YAF_seize_sad.wav	0.209233	0.041016	0.272344	-0.418738
3	YAF_sad	YAF_team_sad.wav	0.140697	0.040039	0.220488	2.793284
4	YAF_sad	YAF_moon_sad.wav	0.097233	0.035156	0.187809	5.664555
...	...	...	...	...	...	...
2527	OAF_disgust	OAF_jar_disgust.wav	0.114746	0.046387	0.163155	2.891574
2528	OAF_disgust	OAF_judge_disgust.wav	0.109677	0.045654	0.159212	4.190305
2529	OAF_disgust	OAF_keep_disgust.wav	0.118263	0.036865	0.178795	1.508613
2530	OAF_disgust	OAF_jug_disgust.wav	0.117042	0.041992	0.155269	1.017243
2531	OAF_disgust	OAF_hush_disgust.wav	0.137700	0.063232	0.160404	1.894541

```
# Menyimpannya ke CSV
df.to_csv('Audiofeatures.csv',index=False)
```

0

PRE-PROSESING

0

Split Data

Pembagian data (split data) dalam konteks sains data atau pembelajaran mesin penting untuk melatih, menguji, dan mengukur kinerja model. Data dibagi menjadi set pelatihan, dan pengujian. Set pelatihan digunakan untuk melatih model, sementara set pengujian memberikan estimasi kinerja model pada data yang belum pernah dilihat sebelumnya.

Pembagian ini membantu mencegah model hanya menghafal data pelatihan dan memastikan kemampuannya dalam menggeneralisasi pola pada data baru.

Pembagian data juga berguna untuk menguji hipotesis dan memastikan bahwa evaluasi kinerja model tidak terpengaruh oleh data yang telah dilihat sebelumnya. Hal ini sangat penting dalam menghasilkan model yang dapat diandalkan dan berguna untuk mengambil keputusan pada data baru. Dengan memisahkan data dengan hati-hati, kita dapat meningkatkan validitas dan keandalan model sains data atau pembelajaran mesin yang dikembangkan.

```
# Baca data dari file CSV
dataknn= pd.read_csv('Audiofeatures.csv')
# Pisahkan fitur (X) dan label (y)
X = dataknn.drop(['Label','File'], axis=1)  # Ganti 'target_column' dengan nama kolom target
y = dataknn['Label']

# split data into train and test sets
X_train,X_test,y_train, y_test= train_test_split(X, y, random_state=1, test_size=0.2)
```

0.8 Normalisasi

Normalisasi adalah proses mengubah nilai-nilai dalam suatu dataset sehingga dapat memiliki skala yang seragam. Tujuannya adalah untuk memastikan bahwa variabel-variabel dengan rentang nilai yang berbeda-beda memiliki dampak yang setara terhadap analisis atau pembelajaran mesin yang dilakukan. Normalisasi seringkali diterapkan pada data numerik sebelum digu-

nakan dalam model pembelajaran mesin atau analisis statistik seperti dibawah ini

```
# define scaler
scaler = StandardScaler()
# fit scaler on the training dataset
scaler.fit(X_train)
# save the scaler
dump(scaler, open('scaler.pkl', 'wb'))
# transform the training dataset
X_train_scaled = scaler.transform(X_train)

dataknn
```

	Label	File	ZCR Mean	ZCR Median	ZCR Std Dev	ZCR Kurtosi
0	YAF_sad	YAF_sour_sad.wav	0.167885	0.042480	0.239094	0.742978
1	YAF_sad	YAF_red_sad.wav	0.117450	0.035156	0.206319	4.820082
2	YAF_sad	YAF_seize_sad.wav	0.209233	0.041016	0.272344	-0.418738
3	YAF_sad	YAF_team_sad.wav	0.140697	0.040039	0.220488	2.793284
4	YAF_sad	YAF_moon_sad.wav	0.097233	0.035156	0.187809	5.664555
...	...	...	...	...	...	...
2527	OAF_disgust	OAF_jar_disgust.wav	0.114746	0.046387	0.163155	2.891574
2528	OAF_disgust	OAF_judge_disgust.wav	0.109677	0.045654	0.159212	4.190305
2529	OAF_disgust	OAF_keep_disgust.wav	0.118263	0.036865	0.178795	1.508613
2530	OAF_disgust	OAF_jug_disgust.wav	0.117042	0.041992	0.155269	1.017243
2531	OAF_disgust	OAF_hush_disgust.wav	0.137700	0.063232	0.160404	1.894541

```
# Membuka file pickle
with open('scaler.pkl', 'rb') as standarisasi:
    loadscal= pickle.load(standarisasi)

X_test_scaled=loadscal.transform(X_test) #normalisasi X testing dari hasil normalisasi X train
```

0

Model K-Neirest Neighbors

0.9 Penjelasan K-NN

K-NN, atau K-Nearest Neighbors, adalah algoritma pembelajaran mesin yang digunakan untuk klasifikasi dan regresi. Ide dasar di balik K-NN adalah mengklasifikasikan atau memprediksi suatu data berdasarkan mayoritas label (untuk klasifikasi) atau nilai rata-rata (untuk regresi) dari K tetangga terdekatnya di dalam ruang fitur. Artinya, sebuah data akan diatributkan label atau nilai yang paling umum atau rata-rata di antara tetangga-tetangganya.

0.10 Langkah-Langkah K-NN:

1. Pemilihan Jumlah Tetangga (K): > Pilih suatu nilai K, yang merupakan jumlah tetangga terdekat yang akan digunakan untuk membuat prediksi.
2. Menghitung Jarak: > Hitung jarak antara data yang akan diprediksi dengan semua data lain dalam dataset menggunakan suatu metrik jarak, seperti Euclidean distance atau Manhattan distance.
3. Identifikasi Tetangga Terdekat: > Pilih K tetangga terdekat berdasarkan nilai jarak yang dihitung.
4. Klasifikasi atau Regresi: > Untuk klasifikasi, atributkan label yang paling umum di antara tetangga-tetangga tersebut kepada data yang akan diprediksi. Untuk regresi, atributkan nilai rata-rata dari nilai target tetangga-tetangga tersebut kepada data yang akan diprediksi.

0.11 Rumus Perhitungan Jarak Euclidean

Dalam kasus klasifikasi dengan menggunakan jarak Euclidean, rumus perhitungan jarak antara dua titik $P(x_1, y_1)$ dan $Q(x_2, y_2)$ pada bidang dua dimensi adalah:

$$Euclidean\ Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_2)^2}$$

```
# Jumlah tetangga (neighbors) yang akan diuji
K = 10

# Inisialisasi array untuk menyimpan akurasi
acc = np.zeros((K-1))

# Melakukan iterasi untuk nilai n_neighbors dari 1 hingga K dengan langkah 2
for n in range(1, K, 2):
    # Membuat model KNN dengan jumlah tetangga n dan metrik Euclidean
    knn = KNeighborsClassifier(n_neighbors=n, metric="euclidean").fit(X_train_scaled, y_train)

    # Melakukan prediksi pada data uji
    y_pred = knn.predict(X_test_scaled)

    # Menghitung akurasi dan menyimpannya dalam array acc
    acc[n-1] = accuracy_score(y_test, y_pred)

# Menampilkan hasil akurasi terbaik dan nilai k yang sesuai
print('Akurasi terbaik adalah', acc.max(), 'dengan nilai k =', acc.argmax()+1)
```

```
Akurasi terbaik adalah 0.6844181459566075 dengan nilai k = 5
```

```
# Inisialisasi model KNeighborsClassifier dengan 13 tetangga terdekat (n_neighbors=13)
# dan menggunakan metrik jarak "euclidean" untuk mengukur jarak antar data.
knn = KNeighborsClassifier(n_neighbors=13, metric="euclidean")

# Simpan model KNeighborsClassifier ke dalam file 'modelknn.pkl' menggunakan fungsi dump dari j
dump(knn, open('modelknn.pkl', 'wb'))
```

```
# Membuka file 'modelknn.pkl' dalam mode pembacaan biner ('rb')
with open('modelknn.pkl', 'rb') as knn:
    # Melakukan deserialisasi objek K-Nearest Neighbors (KNN) dari file
    loadknn = pickle.load(knn)
```

```
# Melakukan pelatihan (fitting) kembali pada model KNN yang telah di-deserialize
# dengan menggunakan data yang telah di-scaling (X_train_scaled) dan labelnya (y_train)
loadknn.fit(X_train_scaled, y_train)
```

```
KNeighborsClassifier(metric='euclidean', n_neighbors=13)
```

```
y_pred = loadknn.predict(X_test_scaled)
y_pred
```

```
array(['YAF_disgust', 'OAF_Fear', 'OAF_disgust', 'OAF_Sad', 'OAF_Fear',
       'YAF_happy', 'YAF_happy', 'OAF_Pleasant_surprise', 'YAF_neutral',
       'YAF_neutral', 'YAF_happy', 'YAF_sad', 'OAF_Sad',
       'OAF_Pleasant_surprise', 'YAF_fear', 'YAF_neutral', 'YAF_angry',
       'YAF_pleasant_surprised', 'YAF_disgust', 'YAF_neutral',
       'YAF_pleasant_surprised', 'YAF_pleasant_surprised', 'YAF_fear',
       'YAF_happy', 'YAF_pleasant_surprised', 'OAF_Fear', 'OAF_happy',
       'YAF_sad', 'YAF_sad', 'YAF_happy', 'YAF_neutral', 'YAF_sad',
       'YAF_disgust', 'YAF_pleasant_surprised', 'OAF_neutral',
      'OAF_neutral', 'OAF_neutral', 'YAF_sad', 'OAF_happy', 'YAF_happy',
       'YAF_angry', 'YAF_happy', 'YAF_neutral', 'OAF_happy',
      'OAF_disgust', 'OAF_Fear', 'OAF_Fear', 'YAF_neutral', 'YAF_happy',
       'OAF_happy', 'OAF_Sad', 'YAF_disgust', 'OAF_Pleasant_surprise',
       'YAF_happy', 'YAF_angry', 'OAF_happy', 'OAF_Fear', 'YAF_fear',
       'YAF_disgust', 'OAF_Fear', 'OAF_happy', 'OAF_Sad', 'OAF_happy',
       'YAF_happy', 'OAF_Fear', 'YAF_pleasant_surprised',
       'YAF_pleasant_surprised', 'OAF_Pleasant_surprise', 'YAF_happy',
       'OAF_Fear', 'YAF_disgust', 'OAF_neutral', 'YAF_disgust',
      'OAF_disgust', 'YAF_neutral', 'OAF_Fear', 'OAF_Sad', 'YAF_happy',
      'YAF_happy', 'YAF_happy', 'YAF_angry', 'YAF_happy', 'OAF_disgust',
       'YAF_neutral', 'OAF_Pleasant_surprise', 'YAF_angry', 'YAF_sad',
      'OAF_Fear', 'YAF_pleasant_surprised', 'YAF_neutral', 'YAF_fear',
       'OAF_Pleasant_surprise', 'OAF_happy', 'OAF_happy', 'OAF_Sad',
       'OAF_neutral', 'YAF_disgust', 'YAF_neutral', 'YAF_fear',
      'OAF_happy', 'YAF_happy', 'YAF_angry', 'OAF_disgust', 'YAF_sad',
       'OAF_Pleasant_surprise', 'YAF_fear', 'YAF_happy', 'OAF_Sad',
       'YAF_pleasant_surprised', 'OAF_Fear', 'YAF_happy', 'OAF_Fear',
      'OAF_neutral', 'YAF_fear', 'YAF_pleasant_surprised', 'OAF_happy',
       'OAF_Pleasant_surprise', 'YAF_neutral', 'YAF_angry',
       'YAF_pleasant_surprised', 'YAF_sad', 'OAF_happy', 'YAF_angry',
       'OAF_disgust', 'OAF_Sad', 'OAF_Pleasant_surprise', 'YAF_angry',
       'YAF_angry', 'OAF_neutral', 'OAF_neutral', 'OAF_happy',
       'YAF_pleasant_surprised', 'OAF_happy', 'OAF_happy',
       'YAF_pleasant_surprised', 'OAF_neutral', 'OAF_Sad', 'OAF_Sad',
      'OAF_Pleasant_surprise', 'OAF_disgust', 'OAF_Pleasant_surprise',
```

```
  'YAF_disgust', 'YAF_angry', 'YAF_fear', 'OAF_Sad', 'YAF_happy',
  'OAF_neutral', 'YAF_neutral', 'YAF_disgust', 'YAF_disgust',
  'YAF_disgust', 'OAF_Fear', 'YAF_angry', 'YAF_happy',
'YAF_pleasant_surprised', 'OAF_disgust', 'YAF_pleasant_surprised',
  'YAF_pleasant_surprised', 'OAF_Pleasant_surprise', 'OAF_Sad',
  'OAF_Fear', 'YAF_sad', 'YAF_angry', 'YAF_happy', 'YAF_sad',
'YAF_angry', 'OAF_happy', 'YAF_disgust', 'YAF_neutral', 'OAF_Sad',
  'YAF_neutral', 'YAF_angry', 'OAF_Sad', 'OAF_Sad',
  'YAF_pleasant_surprised', 'OAF_neutral', 'YAF_disgust',
  'OAF_Pleasant_surprise', 'OAF_Pleasant_surprise',
  'OAF_Pleasant_surprise', 'OAF_neutral', 'OAF_happy',
  'OAF_Pleasant_surprise', 'OAF_disgust', 'YAF_angry',
  'YAF_pleasant_surprised', 'YAF_happy', 'YAF_angry', 'OAF_Fear',
 'OAF_neutral', 'YAF_sad', 'YAF_sad', 'OAF_neutral', 'OAF_happy',
  'YAF_happy', 'YAF_angry', 'YAF_disgust', 'OAF_neutral',
  'OAF_neutral', 'OAF_Fear', 'OAF_disgust', 'OAF_happy',
 'OAF_Pleasant_surprise', 'YAF_neutral', 'OAF_Pleasant_surprise',
  'OAF_Sad', 'YAF_pleasant_surprised', 'YAF_angry', 'YAF_angry',
  'OAF_happy', 'YAF_disgust', 'YAF_happy', 'YAF_happy',
  'YAF_disgust', 'OAF_neutral', 'OAF_Fear', 'OAF_Fear',
  'YAF_disgust', 'OAF_Pleasant_surprise', 'YAF_sad', 'YAF_fear',
'YAF_neutral', 'YAF_pleasant_surprised', 'YAF_pleasant_surprised',
  'OAF_Fear', 'YAF_angry', 'YAF_sad', 'YAF_fear', 'YAF_sad',
  'YAF_pleasant_surprised', 'OAF_Pleasant_surprise', 'OAF_happy',
  'YAF_disgust', 'YAF_happy', 'YAF_fear', 'YAF_neutral',
  'OAF_neutral', 'OAF_Fear', 'OAF_happy', 'YAF_neutral',
  'OAF_Pleasant_surprise', 'YAF_neutral', 'YAF_happy', 'OAF_Sad',
  'YAF_happy', 'OAF_Sad', 'YAF_disgust', 'OAF_Pleasant_surprise',
  'YAF_angry', 'YAF_angry', 'OAF_happy', 'YAF_angry', 'YAF_sad',
'OAF_happy', 'YAF_pleasant_surprised', 'OAF_disgust', 'OAF_happy',
'YAF_sad', 'YAF_angry', 'YAF_neutral', 'YAF_disgust', 'YAF_happy',
  'YAF_happy', 'OAF_happy', 'YAF_happy', 'YAF_happy',
  'OAF_Pleasant_surprise', 'YAF_neutral', 'OAF_Fear',
  'OAF_Pleasant_surprise', 'YAF_fear', 'YAF_fear', 'OAF_Fear',
  'OAF_neutral', 'OAF_disgust', 'YAF_disgust', 'OAF_neutral',
'YAF_angry', 'OAF_neutral', 'YAF_disgust', 'YAF_sad', 'YAF_angry',
  'OAF_happy', 'YAF_angry', 'OAF_Sad', 'OAF_disgust', 'YAF_sad',
 'YAF_pleasant_surprised', 'OAF_happy', 'YAF_neutral', 'OAF_Sad',
'YAF_pleasant_surprised', 'YAF_angry', 'YAF_neutral', 'YAF_angry',
  'OAF_happy', 'YAF_fear', 'YAF_neutral', 'YAF_angry',
  'YAF_pleasant_surprised', 'YAF_neutral', 'OAF_happy',
 'YAF_disgust', 'YAF_sad', 'YAF_disgust', 'YAF_happy', 'YAF_sad',
 'OAF_Fear', 'OAF_Pleasant_surprise', 'YAF_angry', 'YAF_neutral',
  'OAF_happy', 'OAF_happy', 'YAF_pleasant_surprised',
 'YAF_pleasant_surprised', 'OAF_happy', 'YAF_neutral', 'YAF_fear',
```

```
       'OAF_neutral', 'YAF_pleasant_surprised', 'OAF_Sad', 'OAF_Sad',
       'YAF_angry', 'YAF_angry', 'OAF_happy', 'YAF_happy', 'YAF_happy',
       'OAF_Sad', 'YAF_fear', 'YAF_angry', 'OAF_Sad', 'YAF_neutral',
       'OAF_neutral', 'YAF_disgust', 'YAF_neutral', 'YAF_neutral',
      'OAF_neutral', 'OAF_happy', 'YAF_pleasant_surprised', 'OAF_Fear',
       'YAF_disgust', 'YAF_happy', 'OAF_happy', 'YAF_disgust',
       'OAF_Pleasant_surprise', 'YAF_fear', 'YAF_happy', 'YAF_angry',
       'YAF_sad', 'OAF_Pleasant_surprise', 'YAF_fear', 'OAF_neutral',
       'OAF_neutral', 'OAF_happy', 'OAF_Sad', 'YAF_angry', 'OAF_Sad',
       'YAF_happy', 'YAF_happy', 'YAF_fear', 'OAF_neutral',
       'YAF_pleasant_surprised', 'YAF_neutral', 'OAF_happy',
     'OAF_neutral', 'OAF_happy', 'YAF_angry', 'OAF_Fear', 'OAF_neutral',
     'OAF_neutral', 'YAF_pleasant_surprised', 'OAF_happy', 'YAF_angry',
      'OAF_Sad', 'YAF_happy', 'OAF_Fear', 'OAF_neutral', 'OAF_neutral',
       'OAF_disgust', 'YAF_disgust', 'OAF_disgust', 'OAF_happy',
       'OAF_Sad', 'OAF_happy', 'YAF_fear', 'YAF_happy', 'OAF_happy',
      'OAF_happy', 'OAF_neutral', 'YAF_happy', 'OAF_happy', 'YAF_angry',
       'YAF_pleasant_surprised', 'OAF_happy', 'YAF_happy', 'YAF_fear',
     'OAF_Pleasant_surprise', 'YAF_neutral', 'OAF_happy', 'YAF_neutral',
      'YAF_happy', 'OAF_happy', 'YAF_pleasant_surprised', 'OAF_happy',
       'YAF_angry', 'OAF_Sad', 'OAF_Sad', 'YAF_pleasant_surprised',
       'OAF_happy', 'YAF_happy', 'YAF_angry', 'OAF_Sad', 'YAF_fear',
       'OAF_disgust', 'YAF_neutral', 'YAF_fear', 'OAF_neutral',
       'YAF_disgust', 'OAF_Pleasant_surprise', 'YAF_disgust',
      'OAF_neutral', 'YAF_angry', 'YAF_happy', 'YAF_happy', 'YAF_fear',
       'YAF_pleasant_surprised', 'OAF_happy', 'OAF_Fear', 'OAF_happy',
     'YAF_disgust', 'OAF_disgust', 'YAF_sad', 'YAF_angry', 'YAF_angry',
       'OAF_Fear', 'YAF_angry', 'OAF_happy', 'YAF_sad', 'OAF_happy',
     'YAF_pleasant_surprised', 'YAF_fear', 'YAF_neutral', 'OAF_neutral',
      'YAF_sad', 'OAF_Sad', 'YAF_angry', 'OAF_neutral', 'YAF_disgust',
      'YAF_angry', 'OAF_disgust', 'YAF_sad', 'YAF_angry', 'YAF_happy',
     'OAF_disgust', 'YAF_neutral', 'OAF_Sad', 'YAF_neutral', 'YAF_sad',
       'OAF_neutral', 'YAF_sad', 'OAF_Sad', 'YAF_pleasant_surprised',
       'OAF_disgust', 'YAF_disgust', 'YAF_angry', 'OAF_Fear',
      'YAF_disgust', 'OAF_Fear', 'OAF_happy', 'YAF_happy', 'YAF_angry',
       'OAF_Sad', 'OAF_neutral', 'YAF_pleasant_surprised', 'OAF_Sad',
       'OAF_neutral', 'YAF_pleasant_surprised', 'OAF_disgust',
      'OAF_happy', 'YAF_neutral', 'OAF_Fear', 'YAF_pleasant_surprised',
       'YAF_fear', 'OAF_Pleasant_surprise', 'OAF_happy', 'OAF_Sad',
       'YAF_angry', 'YAF_angry', 'OAF_Fear', 'OAF_happy', 'YAF_fear',
     'YAF_neutral', 'YAF_pleasant_surprised', 'YAF_pleasant_surprised',
       'YAF_disgust', 'YAF_neutral', 'YAF_disgust', 'OAF_Fear',
       'OAF_neutral', 'OAF_neutral', 'YAF_happy', 'OAF_neutral',
      'OAF_happy', 'YAF_happy', 'YAF_angry', 'OAF_Fear', 'YAF_happy'],
      dtype=object)
```

```
accuracy = accuracy_score(y_test,y_pred)
print("Akurasi:",accuracy)
```

```
Akurasi: 0.6824457593688363
```

0.12 REDUKSI DATA

reduksi ada 2 jenis bisa menggunakan seleksi data atau transformation data(contoh nya PCA / Principal component anlysis). seleksi data kita dapat emmilih fitur berdasarkan yang fitur/ kolom yang paling berpengaruh. sedangkan transformasi data kita perlu membuat kooordinat baru dari dari fitur yang ada sejumlah dengan jumlah fitur nya. mengapa reduksi data atau data reduction di perlukan? karna terlalu banyak kolom/fitur/ciri yang harus dikenali tidak baik untuk pemprosesan data dan memakan waktu komputasi yang lama, maka dari itu reduksi data di lakukan untuk mendapatkan data yang terbaik. ketika ingin mencari koordinat baru:

buat matriks covarian buat persamaan

note : konstanta yang paling besar mengartikan di koordinat tersebut merupakan ciri paling banyak atau penting

```
# Mengimpor modul PCA (Principal Component Analysis) dari pustaka scikit-learn
from sklearn.decomposition import PCA as sklearnPCA

# Membuat objek PCA dengan 8 komponen
sklearn_pca = sklearnPCA(n_components=8)

# Melakukan transformasi PCA pada data latihan yang telah dinormalisasi
X_train_pca = sklearn_pca.fit_transform(X_train_scaled)

# Menyimpan objek PCA ke dalam file 'PCA8.pkl' menggunakan modul pickle
dump(sklearn_pca, open('PCA8.pkl', 'wb'))

# Membuka kembali file 'PCA8.pkl' untuk mendapatkan objek PCA yang telah disimpan
with open('PCA8.pkl', 'rb') as pca:
    loadpca = pickle.load(pca)

# Melakukan transformasi PCA pada data uji yang telah dinormalisasi
X_test_pca = loadpca.transform(X_test_scaled)
X_test_pca.shape
```

```
(507, 8)
```

```
# Mengimpor modul KNeighborsClassifier dari scikit-learn
from sklearn.neighbors import KNeighborsClassifier

# Membuat objek klasifikasi K-Nearest Neighbors dengan jumlah tetangga sebanyak 15
classifier = KNeighborsClassifier(n_neighbors=15)

# Melatih model klasifikasi menggunakan data latihan yang telah direduksi dimensinya dengan PCA
classifier.fit(X_train_pca, y_train)

# Melakukan prediksi menggunakan data uji yang telah direduksi dimensinya dengan PCA
y_prediksi = classifier.predict(X_test_pca)

# Mengimpor modul accuracy_score dari scikit-learn
from sklearn.metrics import accuracy_score

# Menghitung akurasi dari prediksi yang telah dilakukan
acc_pca = accuracy_score(y_test, y_prediksi)

# Menampilkan akurasi hasil prediksi
print("Akurasi:", acc_pca)
```

```
Akurasi: 0.6706114398422091
```