Whinta Virginia Putri

Proyek Sains Data

To blah, blah, and blah.

Table of contents

0

Proyek Sains Data

Proyek sains data adalah suatu usaha yang dilakukan untuk mengumpulkan, menganalisis, dan menginterpretasi data dengan tujuan untuk mendapatkan wawasan, mendukung pengambilan keputusan, atau memecahkan masalah tertentu. Proyek ini melibatkan penerapan metode statistik, matematika, dan teknik komputasi untuk mengeksplorasi dan memahami pola atau tren dalam data.

Langkah-langkah umum dalam proyek sains data melibatkan:

1. **Pemahaman Masalah**: Menentukan tujuan proyek dan pemahaman mendalam tentang masalah yang ingin dipecahkan.
2. **Pengumpulan Data**: Mengumpulkan data yang relevan dan diperlukan untuk proyek.
3. **Pembersihan Data**: Membersihkan dan mempersiapkan data untuk analisis dengan mengatasi masalah seperti data yang hilang atau outlier.
4. **Eksplorasi Data**: Mengeksplorasi data untuk mengidentifikasi pola, tren, atau informasi yang dapat diambil.
5. **Pemodelan**: Menerapkan teknik statistik atau pembelajaran mesin untuk membangun model yang dapat meramalkan atau menjelaskan fenomena yang diamati.
6. **Evaluasi Model**: Menilai kinerja model dan memastikan bahwa solusi yang diusulkan sesuai dengan tujuan awal proyek.
7. **Implementasi**: Menerapkan solusi atau rekomendasi berdasarkan hasil analisis data.
8. **Komunikasi Hasil**: Mengkomunikasikan temuan dan rekomendasi kepada pemangku kepentingan melalui laporan atau presentasi.

Proyek sains data dapat diterapkan dalam berbagai konteks, termasuk bisnis, ilmu pengetahuan, kesehatan, dan banyak lagi. Penting untuk mencatat bahwa sains data bukan hanya tentang alat dan teknik analisis, tetapi juga

melibatkan pemahaman domain yang baik untuk menghasilkan hasil yang bermakna.

Author: Nama: Whinta Virginia Putri NIM: 210411100047 Proyek Sains Data A

0

Analisis dan Prediksi pada Data Audio Emosi

Nama: Whinta Virginia Putri

NIM: 210411100047

UTS Proyek Sain Data

```
import numpy as np
import pandas as pd
import scipy.stats
import librosa
import soundfile as sf
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import MinMaxScaler
import joblib
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import requests
from io import StringIO
from sklearn.preprocessing import StandardScaler
import pickle
from joblib import dump
from sklearn.decomposition import PCA as sklearnPCA
import math
from google.colab import drive
import os
from google.colab import files
```

0.1 Deskripsi Dataset :

Ada sekitar 200 kata target yang diucapkan dalam frase pembawa ‘say the word_’ oleh dua aktris (berusia 26 dan 64 tahun) dan rekaman dibuat dari set

tersebut menggambarkan tujuh emosi (marah, jijik, takut, bahagia, terkejut menyenangkan, sedih, dan netral). Terdapat total 2800 titik data (file audio).

Kumpulan data ini diatur sedemikian rupa sehingga setiap dari dua aktris perempuan dan emosi mereka terdapat dalam folder tersendiri. Dan di dalamnya, semua file audio 200 kata target dapat ditemukan. Format file audio ini adalah format WAV.

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

0.2 Pre-processing

Dibuat 11 fitur dari Dataset audio kaggle yaitu Label, Mean, Median, Std_Deviation, Zero_Crossing_Rate, Energy, Spectral_Centroid, Spectral_Bandwidth, Spectral_Rolloff, Chroma, dan MFCCs dari 2191 data, Kemudian di simpan pada format csv. Berikut adalah hasil Audio_features:

```
df = pd.read_csv('https://raw.githubusercontent.com/whintaaa/datapsd/main/audio_fitur.csv')
df
```

	Label	Audio_Name	Mean	Median	Std_Deviation	Zero_Crossing_Rate	Ener
0	disgust	YAF_chief_disgust	-0.000022	-0.000181	0.022747	0.232713	0.017
1	disgust	YAF_chain_disgust	-0.000018	-0.000350	0.020867	0.207239	0.015
2	disgust	YAF_base_disgust	-0.000021	-0.000203	0.022053	0.246661	0.017
3	disgust	YAF_ditch_disgust	-0.000024	-0.000103	0.020262	0.196533	0.015
4	disgust	YAF_germ_disgust	-0.000018	-0.000353	0.025729	0.192057	0.020
...	...	...	...	...	...	...	...
2185	Sad	OAF_wheat_sad	0.000033	0.000606	0.010872	0.086951	0.008
2186	Sad	OAF_puff_sad	-0.000010	0.000292	0.013971	0.078135	0.011
2187	Sad	OAF_raise_sad	-0.000001	0.000284	0.013523	0.086271	0.011
2188	Sad	OAF_sail_sad	0.000032	0.000330	0.010211	0.118340	0.008
2189	Sad	OAF_rat_sad	-0.000013	0.000459	0.010945	0.079590	0.009

```
# Memisahkan fitur (X) dan label (y)
X = df.drop(['Label','Audio_Name'], axis=1)
y = df['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)
```

0.2.1 Normalisasi Scaler

Normalisasi scaler dalam konteks kode yang Anda berikan tampaknya menggunakan `StandardScaler` dari pustaka scikit-learn untuk melakukan normalisasi pada data. Normalisasi adalah proses mengubah nilai-nilai dalam dataset ke skala umum, biasanya dengan mengurangkan rata-rata dan membagi hasilnya dengan deviasi standar.

Mari kita bahas langkah-langkah dalam kode tersebut:

1. **Define and Fit Scaler:**

   ```
   scaler = StandardScaler()
   scaler.fit(X_train)
   ```

   Pada langkah ini, Anda membuat objek `StandardScaler` dari scikit-learn dan kemudian menggunakan metode `fit` untuk menghitung rata-rata dan deviasi standar dari dataset pelatihan (`X_train`).

2. **Save Scaler Using Pickle:**

   ```
   scaler_file_path = r'scaler.pkl'
   with open(scaler_file_path, 'wb') as scaler_file:
       pickle.dump(scaler, scaler_file)
   ```

   Pada langkah ini, Anda menyimpan objek scaler ke dalam file menggunakan modul `pickle`. Ini memungkinkan Anda untuk menggunakan kembali objek scaler tanpa harus melatih ulang model setiap kali Anda menjalankan program.

3. **Transform Training Data Using Scaler:**

   ```
   X_train_scaled = scaler.transform(X_train)
   ```

   Di sini, Anda menggunakan scaler yang telah dilatih untuk melakukan normalisasi pada dataset pelatihan (`X_train`) dan menyimpan hasilnya dalam `X_train_scaled`.

4. **Load Scaler Using Pickle:**

   ```
   with open(r'scaler.pkl', 'rb') as normalisasi:
       loadscal = pickle.load(normalisasi)
   ```

   Langkah terakhir adalah memuat kembali objek scaler dari file yang telah disimpan sebelumnya menggunakan modul `pickle`. Objek scaler yang dimuat disimpan dalam variabel `loadscal`.

Dengan cara ini, Anda dapat menggunakan objek `loadscal` untuk melakukan transformasi normalisasi pada dataset lain dengan menggunakan parameter normalisasi yang sama seperti yang telah dihitung pada dataset pelatihan.

```
# Define and fit the scaler on the training dataset
scaler = StandardScaler()
scaler.fit(X_train)
# Save the scaler using pickle
scaler_file_path = r'scaler.pkl'
with open(scaler_file_path, 'wb') as scaler_file:
    pickle.dump(scaler, scaler_file)

X_train_scaled = scaler.transform(X_train)
with open(r'scaler.pkl', 'rb') as normalisasi:
    loadscal = pickle.load(normalisasi)
```

```
X_test_scaled = loadscal.transform(X_test)
```

```
X_test_scaled
```

```
array([[-0.36438935, -0.01864303,  0.62951586, ...,  1.700305  ,
         1.54756458,  0.50833751],
       [-0.4145327 , -0.16604988,  0.48640237, ...,  0.96098207,
        -0.94090678,  0.60164653],
       [-0.54970173, -0.37708373,  0.00366902, ...,  0.23587569,
        -0.06940957,  0.06954893],
       ...,
       [-0.47993707, -0.22850376, -0.05819755, ...,  0.90422257,
         0.30463648, -0.36001778],
       [ 0.80198861, -0.0505326 , -1.00057672, ...,  0.02687033,
        -0.90359502, -1.22195475],
       [-0.63036712,  0.86737544, -0.17119103, ...,  0.01166204,
        -0.94401292,  1.46828321]])
```

0.3 Modelling

0.3.1 Mencari k terbaik dengan akurasi paling tinggi

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
K = 30
acc = np.zeros((K - 1))

for n in range(1, K, 2):
    knn = KNeighborsClassifier(n_neighbors=n, metric="euclidean").fit(X_train_scaled, y_train)
    y_pred = knn.predict(X_test_scaled)
    acc[n - 1] = accuracy_score(y_test, y_pred)

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

# Tampilkan akurasi terbaik dan nilai k
print('Akurasi terbaik adalah', best_accuracy, 'dengan nilai k =', best_k)
```

```
Akurasi terbaik adalah 0.769406392694064 dengan nilai k = 9
```

0.3.2 Menyimpan model knn dengan k terbaik

```
# Simpan model KNN terbaik
best_knn = KNeighborsClassifier(n_neighbors=best_k, metric="euclidean")
best_knn.fit(X_train_scaled, y_train)
```

```
KNeighborsClassifier(metric='euclidean', n_neighbors=9)
```

```
# Save the best KNN model using pickle
model_file_path = r'model.pkl'
with open(model_file_path, 'wb') as model_file:
    pickle.dump(best_knn, model_file)

with open(r'model.pkl', 'rb') as knn_model:
    load_knn = pickle.load(knn_model)
```

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

```
array(['angry', 'fear', 'pleasantsurprised', 'Sad', 'neutral', 'disgust',
       'pleasantsurprised', 'neutral', 'happy', 'Fear', 'Fear', 'angry',
        'pleasantsurprised', 'disgust', 'happy', 'pleasantsurprised',
        'sad', 'neutral', 'sad', 'happy', 'happy', 'disgust', 'Fear',
        'Sad', 'angry', 'sad', 'sad', 'neutral', 'disgust', 'disgust',
        'angry', 'pleasantsurprised', 'neutral', 'sad', 'Sad',
       'pleasantsurprised', 'Fear', 'disgust', 'Fear', 'happy', 'happy',
      'neutral', 'happy', 'Fear', 'Pleasantsurprise', 'neutral', 'Fear',
        'disgust', 'disgust', 'sad', 'pleasantsurprised', 'neutral',
        'happy', 'neutral', 'neutral', 'disgust', 'Fear',
     'pleasantsurprised', 'sad', 'happy', 'Pleasantsurprise', 'neutral',
        'Fear', 'Sad', 'pleasantsurprised', 'disgust', 'Sad',
      'pleasantsurprised', 'disgust', 'Fear', 'Sad', 'Pleasantsurprise',
        'disgust', 'sad', 'Sad', 'angry', 'Sad', 'happy', 'angry',
        'neutral', 'sad', 'disgust', 'disgust', 'happy',
       'pleasantsurprised', 'fear', 'fear', 'neutral', 'Sad', 'neutral',
        'disgust', 'happy', 'disgust', 'angry', 'Sad', 'happy', 'Sad',
      'disgust', 'Sad', 'Pleasantsurprise', 'Fear', 'Sad', 'sad', 'Sad',
        'Sad', 'fear', 'neutral', 'happy', 'Fear', 'disgust', 'fear',
        'disgust', 'happy', 'Sad', 'sad', 'Pleasantsurprise', 'angry',
        'happy', 'disgust', 'Pleasantsurprise', 'neutral',
        'Pleasantsurprise', 'disgust', 'neutral', 'Pleasantsurprise',
      'pleasantsurprised', 'disgust', 'happy', 'fear', 'sad', 'neutral',
        'Sad', 'pleasantsurprised', 'happy', 'fear', 'disgust', 'Fear',
        'Sad', 'neutral', 'happy', 'neutral', 'neutral', 'neutral',
        'disgust', 'happy', 'Sad', 'neutral', 'disgust', 'disgust',
       'neutral', 'sad', 'neutral', 'Sad', 'angry', 'Pleasantsurprise',
        'sad', 'Sad', 'angry', 'happy', 'angry', 'neutral',
        'pleasantsurprised', 'disgust', 'disgust', 'disgust', 'Fear',
      'happy', 'Sad', 'Pleasantsurprise', 'neutral', 'neutral', 'angry',
       'neutral', 'sad', 'sad', 'Fear', 'angry', 'Fear', 'Sad', 'fear',
        'neutral', 'sad', 'happy', 'disgust', 'sad', 'sad', 'fear',
        'happy', 'Sad', 'disgust', 'disgust', 'pleasantsurprised',
        'pleasantsurprised', 'pleasantsurprised', 'Pleasantsurprise',
        'disgust', 'happy', 'sad', 'angry', 'disgust', 'Fear',
        'pleasantsurprised', 'pleasantsurprised', 'Fear', 'neutral',
        'disgust', 'fear', 'disgust', 'pleasantsurprised', 'fear',
        'disgust', 'Pleasantsurprise', 'pleasantsurprised', 'fear',
        'neutral', 'angry', 'pleasantsurprised', 'sad', 'Sad', 'happy',
       'angry', 'happy', 'happy', 'angry', 'angry', 'pleasantsurprised',
      'neutral', 'sad', 'angry', 'angry', 'pleasantsurprised', 'disgust',
        'fear', 'disgust', 'happy', 'fear', 'happy', 'Fear', 'sad',
       'happy', 'Pleasantsurprise', 'neutral', 'disgust', 'happy', 'Sad',
      'disgust', 'sad', 'disgust', 'neutral', 'angry', 'Fear', 'disgust',
        'Fear', 'neutral', 'happy', 'neutral', 'angry', 'happy',
```

```
 'pleasantsurprised', 'angry', 'happy', 'fear', 'neutral', 'happy',
'Sad', 'happy', 'neutral', 'pleasantsurprised', 'disgust', 'angry',
'angry', 'pleasantsurprised', 'Fear', 'pleasantsurprised', 'Fear',
  'Pleasantsurprise', 'Fear', 'disgust', 'pleasantsurprised',
  'angry', 'disgust', 'happy', 'happy', 'neutral', 'happy',
  'neutral', 'Sad', 'Pleasantsurprise', 'happy', 'happy', 'sad',
  'fear', 'neutral', 'pleasantsurprised', 'disgust', 'Fear',
  'Pleasantsurprise', 'disgust', 'happy', 'neutral', 'Sad',
'disgust', 'fear', 'happy', 'disgust', 'angry', 'disgust', 'happy',
  'fear', 'disgust', 'neutral', 'Pleasantsurprise', 'neutral',
  'neutral', 'angry', 'disgust', 'neutral', 'pleasantsurprised',
'neutral', 'Sad', 'neutral', 'Fear', 'angry', 'disgust', 'neutral',
 'happy', 'Sad', 'happy', 'angry', 'disgust', 'neutral', 'happy',
  'Sad', 'fear', 'sad', 'pleasantsurprised', 'fear',
  'Pleasantsurprise', 'Fear', 'happy', 'happy', 'neutral',
  'pleasantsurprised', 'fear', 'Fear', 'Sad', 'Pleasantsurprise',
  'angry', 'Pleasantsurprise', 'Sad', 'neutral', 'happy', 'Sad',
 'pleasantsurprised', 'Fear', 'pleasantsurprised', 'sad', 'fear',
  'neutral', 'neutral', 'happy', 'neutral', 'neutral', 'angry',
'Pleasantsurprise', 'disgust', 'neutral', 'fear', 'happy', 'angry',
  'Fear', 'fear', 'Sad', 'neutral', 'Sad', 'Fear', 'disgust',
 'happy', 'disgust', 'neutral', 'Sad', 'Pleasantsurprise', 'sad',
 'fear', 'happy', 'neutral', 'happy', 'happy', 'happy', 'neutral',
'neutral', 'neutral', 'disgust', 'happy', 'Sad', 'disgust', 'Fear',
  'neutral', 'angry', 'pleasantsurprised', 'pleasantsurprised',
  'happy', 'angry', 'disgust', 'pleasantsurprised',
  'pleasantsurprised', 'neutral', 'pleasantsurprised',
  'pleasantsurprised', 'angry', 'Sad', 'disgust',
  'pleasantsurprised', 'sad', 'Fear', 'neutral', 'angry',
 'pleasantsurprised', 'happy', 'Sad', 'sad', 'neutral', 'disgust',
  'Pleasantsurprise', 'pleasantsurprised', 'neutral', 'angry',
 'happy', 'disgust', 'angry', 'angry', 'neutral', 'happy', 'fear',
  'disgust', 'pleasantsurprised', 'Sad', 'sad'], dtype=object)
```

```
print('Akurasi KNN dengan data test:')
accuracy = accuracy_score(y_test, y_pred)
print( accuracy)
```

```
Akurasi KNN dengan data test:
0.769406392694064
```

```
# Hitung prediksi label KNN
knn_predictions = load_knn.predict(X_test_scaled)
```

```
# Simpan hasil prediksi KNN ke dalam DataFrame
knn_results_df = pd.DataFrame({'Actual Label': y_test, 'Predicted Label (KNN)': knn_predictions

# Tampilkan tabel prediksi KNN
print("Tabel Prediksi Label KNN")
knn_results_df
```

```
Tabel Prediksi Label KNN
```

	Actual Label	Predicted Label (KNN)
353	angry	angry
1631	fear	fear
414	pleasantsurprised	pleasantsurprised
1827	neutral	Sad
851	neutral	neutral
...	...	...
1613	fear	fear
50	disgust	disgust
453	pleasantsurprised	pleasantsurprised
1822	neutral	Sad
1233	sad	sad

0.3.3 Reduksi PCA

Reduksi PCA (Principal Component Analysis) adalah teknik yang digunakan untuk mengurangi dimensi dari dataset yang kompleks, dengan tetap mempertahankan sebagian besar informasi yang terkandung dalam dataset tersebut. PCA bekerja dengan mentransformasi data asli ke dalam ruang fitur baru yang disebut komponen utama atau principal components. Komponen utama ini diurutkan berdasarkan seberapa besar variansinya, sehingga komponen pertama menyimpan sebagian besar varians, yang diikuti oleh komponen kedua, dan seterusnya.

```
# Lakukan reduksi PCA
sklearn_pca = sklearnPCA(n_components=10)
X_train_pca = sklearn_pca.fit_transform(X_train_scaled)

print("Principal Components 10:")
X_train_pca
```

```
Principal Components 10:

array([[-2.15699059,  0.12088686, -1.05816054, ...,  0.15522723,
        -0.09174985, -0.01146861],
```

```
       [ 2.70852347, -0.64630269,  0.05224175, ...,  0.00458544,
        -0.04659703, -0.08161985],
       [ 2.91065674,  0.09092699, -0.62907659, ...,  0.05744571,
        -0.00876878, -0.01138743],
       ...,
       [ 1.00230967, -0.72522836,  2.26887371, ..., -0.10442859,
         0.15861958, -0.09644085],
       [ 3.10754228, -0.69602145,  0.38281969, ...,  0.05710673,
         0.06719287,  0.07069764],
       [-2.00722733,  0.50821046,  1.9109502 , ..., -0.00369189,
        -0.05388441,  0.01722397]])
```

```
# Save the PCA model
pca_model_file_path = r'PCA10.pkl'
with open(pca_model_file_path, 'wb') as pca_model_file:
    pickle.dump(sklearn_pca, pca_model_file)
# Load the PCA model
with open(pca_model_file_path, 'rb') as pca_model:
    loadpca = pickle.load(pca_model)
```

```
# Transform test data using the loaded PCA model
X_test_pca = loadpca.transform(X_test_scaled)

# Continue with KNN and evaluation as needed
K = 30
acc_pca = np.zeros((K - 1))
for n in range(1, K, 2):
    knn_pca = KNeighborsClassifier(n_neighbors=n, metric="euclidean").fit(X_train_pca, y_train)
    y_pred_pca = knn_pca.predict(X_test_pca)
    acc_pca[n - 1] = accuracy_score(y_test, y_pred_pca)

best_accuracy_pca = acc_pca.max()
best_k_pca = acc_pca.argmax() + 1

# Tampilkan akurasi terbaik dan nilai k dengan PCA
print('Akurasi KNN terbaik dengan PCA adalah', best_accuracy_pca, 'dengan nilai k =', best_k_pca
```

```
Akurasi KNN terbaik dengan PCA adalah 0.769406392694064 dengan nilai k = 9
```

```
# Hitung prediksi label KNN setelah PCA
knn_pca_predictions = knn_pca.predict(X_test_pca)

# Simpan hasil prediksi KNN setelah PCA ke dalam DataFrame
```

```
knn_pca_results_df = pd.DataFrame({'Actual Label': y_test, 'Predicted Label (KNN with PCA)': kn

# Tampilkan tabel prediksi KNN setelah PCA
print("Tabel Prediksi Label KNN dengan PCA")
knn_pca_results_df
```

```
Tabel Prediksi Label KNN dengan PCA
```

	Actual Label	Predicted Label (KNN with PCA)
353	angry	angry
1631	fear	fear
414	pleasantsurprised	neutral
1827	neutral	Sad
851	neutral	neutral
...	...	...
1613	fear	fear
50	disgust	disgust
453	pleasantsurprised	pleasantsurprised
1822	neutral	neutral
1233	sad	sad

0.4 Evaluasi

0.4.1 Prediksi Data Audio

```
import librosa
import numpy as np

# Fungsi untuk menghitung fitur audio
def extract_features(audio_path):
    y, sr = librosa.load(audio_path)

    # Fitur 1: Mean
    mean = np.mean(y)

    # Fitur 2: Median
    median = np.median(y)

    # Fitur 3: Standard Deviation
```

```
    std_deviation = np.std(y)

    # Fitur 4: Zero Crossing Rate
    zero_crossing_rate = np.mean(librosa.feature.zero_crossing_rate(y))

    # Fitur 5: Energy
    energy = np.mean(librosa.feature.rms(y=y))

    # Fitur 6: Spectral Centroid
    spectral_centroid = np.mean(librosa.feature.spectral_centroid(y=y, sr=sr))

    # Fitur 7: Spectral Bandwidth
    spectral_bandwidth = np.mean(librosa.feature.spectral_bandwidth(y=y, sr=sr))

    # Fitur 8: Spectral Roll-off
    spectral_rolloff = np.mean(librosa.feature.spectral_rolloff(y=y, sr=sr))

    # Fitur 9: Chroma Feature
    chroma = np.mean(librosa.feature.chroma_stft(y=y, sr=sr))

    # Fitur 10: Mel-frequency Cepstral Coefficients (MFCCs)
    mfccs = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13))

    return [mean, median, std_deviation, zero_crossing_rate, energy,
            spectral_centroid, spectral_bandwidth, spectral_rolloff, chroma, mfccs]

print('Ekstraksi Fitur Audio')
print('Unggah file audio WAV untuk menghitung fitur statistiknya.')

# Unggah file audio
from google.colab import files
uploaded_audio = files.upload()

if uploaded_audio:
    audio_path = list(uploaded_audio.keys())[0]
    print(f"Audio file: {audio_path}")

    audio_features = extract_features(audio_path)

    feature_names = [
        "Mean", "Median", "Std Deviation", "Zero Crossing Rate", "Energy",
        "Spectral Centroid", "Spectral Bandwidth", "Spectral Rolloff", "Chroma", "MFCCs"
    ]
```

```
    # Tampilkan hasil fitur
    print("### Hasil Ekstraksi Fitur Audio:")
    for i, feature in enumerate(audio_features):
        print(f"{feature_names[i]}: {feature}")

    # Transform audio_features using the loaded scaler
    # (Pastikan loadscal, loadpca, dan load_knn sudah di-load sebelumnya)
    datauji = loadscal.transform(np.array(audio_features).reshape(1, -1))
    datapca = loadpca.transform(datauji)
    # Make predictions using the KNN model
    y_pred_uji = load_knn.predict(datauji)
    # y_pred_pca = load_knn.predict(datapca)

    print("Fitur-fitur setelah di normalisasi: ", datauji)
    print("Data PCA:", datapca)
    # print("Predicted Label (PCA):", y_pred_pca)
    print("Predicted Label (KNN):", y_pred_uji)
```

```
Ekstraksi Fitur Audio
Unggah file audio WAV untuk menghitung fitur statistiknya.
```

```
<IPython.core.display.HTML object>
```

```
Saving OAF_bath_sad.wav to OAF_bath_sad.wav
Audio file: OAF_bath_sad.wav
### Hasil Ekstraksi Fitur Audio:
Mean: -4.950474249199033e-06
Median: 0.0005449416348710656
Std Deviation: 0.013166016899049282
Zero Crossing Rate: 0.07502528599330358
Energy: 0.010921107605099678
Spectral Centroid: 1823.6221088038858
Spectral Bandwidth: 2179.453805174795
Spectral Rolloff: 3287.8509521484375
Chroma: 0.3211115300655365
MFCCs: -30.506229400634766
Fitur-fitur setelah di normalisasi:  [[ 0.27547308  0.03367706 -0.99734606 -1.38292082 -1.01350426 -1.
  -0.13268214 -1.26206853  0.4160591  -0.45044388]]
Data PCA: [[-2.67397594e+00  1.15609678e-01 -2.55621792e-01 -1.74982840e-02
   6.99985866e-01 -2.89191174e-01 -7.49915711e-02 -1.24858203e-02
  -1.94857090e-04  2.58036094e-02]]
Predicted Label (KNN): ['Sad']
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid featu
  warnings.warn(
```

0

Analisis dan Prediksi pada dataset Heart failure clinical records Menggunakan Pycaret.

Nama: Whinta Virginia Putri

NIM: 210411100047

Proyek Sain Data

0.5 Business understanding

0.5.1 Tujuan Analisis:

Analisis pada dataset Heart failure clinical records bertujuan untuk memprediksi kemungkinan terjadinya gagal jantung pada pasien berdasarkan 13 fitur-fitur klinis mereka.

0.6 Data understanding / Memahami Data

0.6.1 Deskripsi Dataset

Dataset "Heart failure clinical records" merupakan kumpulan data yang berisi catatan medis dari 299 pasien yang mengalami gagal jantung. Data ini dikumpulkan selama periode pemantauan pasien-pasien tersebut. Setiap profil pasien dalam dataset ini dilengkapi dengan 13 fitur klinis yang mencerminkan kondisi kesehatan mereka. Di bawah ini, saya akan menjelaskan deskripsi dari dataset ini serta tujuan utamanya:

0.6.1.1 Deskripsi Dataset Dan Tujuan Dataset:

1. Jumlah Sampel: Dataset ini berisi informasi dari 299 pasien yang mengalami gagal jantung dan tidak ada missing values.
2. Fitur Klinis: Setiap pasien dalam dataset ini memiliki 13 fitur klinis

yang mencakup berbagai aspek dari kesehatan mereka. Beberapa contoh fitur klinis yang mungkin termasuk dalam dataset ini adalah Usia, tekanan darah, kadar serum kreatinin, kadar serum natrium, kadar serum kalium, ejection fraction (fraksi ejeksi), jenis kelamin pasien, dan lain sebagainya. dibawah ini adalah fitur pada dataset: 'age', 'anaemia', 'creatinine_phosphokinase', 'diabetes', 'ejection_fraction', 'high_blood_pressure', 'platelets', 'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time', 'DEATH_EVENT'

3. Dataset ini memiliki target pada kolom Death Event. Fitur target "death event" adalah sebuah fitur yang digunakan untuk menunjukkan apakah pasien mengalami kematian selama periode pemantauan atau tidak. Fitur ini bersifat boolean, yang berarti nilainya hanya dapat berupa dua kemungkinan: 1 / True (benar) atau 0 / False (salah). Jumlah Target dengan nilai 1 Berjumlah 96 dan nilai 0 berjumlah 203. jumlah antara setiap target tidak seimbang maka pada processing akan dilakukan oversampling untuk menyeimbangkan setiap target.
4. Tujuan Dataset: Dataset ini memiliki beberapa tujuan utama, antara lain:
    a. Analisis dan Penelitian Kesehatan: Data ini dapat digunakan untuk menganalisis faktor-faktor risiko dan prediksi gagal jantung, serta untuk memahami hubungan antara berbagai fitur klinis dan kondisi pasien.
    b. Pengembangan Model Prediktif: Dataset ini dapat digunakan untuk mengembangkan model prediktif yang dapat memprediksi kemungkinan terjadinya gagal jantung pada pasien berdasarkan fitur-fitur klinis mereka. Hal ini dapat membantu tenaga medis dalam melakukan tindakan pencegahan yang lebih tepat waktu.

Tujuan utama dari dataset ini adalah meningkatkan pemahaman tentang gagal jantung, membantu dalam pengembangan metode prediktif, serta mendukung penelitian dan pengembangan terkait kesehatan jantung. Data ini menjadi dasar penting untuk menjalankan berbagai analisis dan penelitian dalam upaya untuk meningkatkan diagnosis, perawatan, dan pencegahan penyakit gagal jantung.

```
# Import Library
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from pycaret.classification import *
from imblearn.over_sampling import RandomOverSampler
from pycaret.classification import *
import pickle
```

```
# Baca data dari file CSV yang sudah di upload di github
url = "https://raw.githubusercontent.com/whintaaa/datapsd/main/heart_failure_clinical_records_d
df = pd.read_csv(url)
# menampilkan dataset
df
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	pla
0	75.0	0	582	0	20	1	265
1	55.0	0	7861	0	38	0	263
2	65.0	0	146	0	20	0	162
3	50.0	1	111	0	20	0	210
4	65.0	1	160	1	20	0	327
...	...	...	...	...	...	...	...
294	62.0	0	61	1	38	1	155
295	55.0	0	1820	0	38	0	270
296	45.0	0	2060	1	60	0	742
297	45.0	0	2413	0	38	0	140
298	50.0	0	196	0	45	0	395

0.6.1.2 Penjelasan Fitur-fitur:

```
# Print the column names to identify the correct target variable
print(df.columns)
```

```
Index(['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
       'ejection_fraction', 'high_blood_pressure', 'platelets',
       'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time',
       'DEATH_EVENT'],
      dtype='object')
```

Fitur-fitur klinis dalam dataset "Heart Failure Clinical Records" adalah sebagai berikut:

1. **Usia (age)**: Ini adalah usia pasien dalam tahun. Fitur ini memberikan informasi tentang berapa usia pasien yang mengalami gagal jantung. Usia seringkali menjadi faktor penting dalam menilai risiko dan prognosis penyakit jantung. Fitur ini bertipe data numerik.

2. **Anemia (anaemia)**: Ini adalah fitur boolean yang menunjukkan apakah pasien mengalami penurunan jumlah sel darah merah atau kadar hemoglobin. Nilai 1 / "true" menunjukkan kehadiran anemia, sementara 0 / "false" menunjukkan ketiadaan anemia.
3. **Kreatinin Fosfokinase (CPK)**: Ini adalah tingkat enzim CPK dalam darah, diukur dalam mikrogram per liter (mcg/L). Tingkat CPK dalam darah dapat memberikan indikasi adanya kerusakan otot atau jaringan jantung. Ini adalah indikator penting dalam penilaian kondisi jantung. Fitur ini bertipe data numerik.
4. **Diabetes**: Ini adalah fitur boolean yang menunjukkan apakah pasien menderita diabetes atau tidak. Nilai 1 / "true" menunjukkan keberadaan diabetes, sementara 0 / "false" menunjukkan ketiadaan diabetes. Diabetes merupakan faktor risiko yang signifikan dalam perkembangan penyakit jantung.
5. **Fraksi Ejeksi (Ejection Fraction)**: Ini adalah persentase darah yang meninggalkan jantung pada setiap kontraksi. Fraksi ejeksi ini dinyatakan dalam persentase. Ini adalah ukuran penting dalam menilai kemampuan jantung untuk memompa darah dan dapat memberikan informasi tentang fungsi jantung. Fitur ini bertipe data numerik.
6. **Tekanan Darah Tinggi (High Blood Pressure)**: Ini adalah fitur boolean yang menunjukkan apakah pasien memiliki hipertensi atau tidak. Nilai 1 / "true" menunjukkan keberadaan tekanan darah tinggi, sementara 2 / "false" menunjukkan ketiadaan tekanan darah tinggi. Tekanan darah tinggi adalah faktor risiko utama untuk penyakit jantung.
7. **Platelet (platelets)**: Ini adalah jumlah platelet dalam darah, diukur dalam ribu platelet per mililiter (kiloplatelets/mL). Platelet adalah sel darah yang berperan dalam pembekuan darah. Nilai platelet dalam darah dapat memberikan informasi tentang kemampuan darah untuk membeku. Fitur ini bertipe data numerik.
8. **Jenis Kelamin (Sex)**: Ini adalah fitur biner yang menunjukkan jenis kelamin pasien, yaitu perempuan (woman) atau laki-laki (man). Informasi ini dapat digunakan untuk mengevaluasi perbedaan jenis kelamin dalam insiden gagal jantung.
9. **Kreatinin Serum (Serum Creatinine)**: Ini adalah tingkat kreatinin serum dalam darah, diukur dalam miligram per desiliter (mg/dL). Kreatinin adalah produk sisa metabolisme yang dapat memberikan informasi tentang fungsi ginjal. Tingkat kreatinin serum yang tinggi dapat menunjukkan masalah ginjal yang dapat mempengaruhi kondisi jantung. Fitur ini bertipe data numerik.

10. **Natrium Serum (Serum Sodium)**: Ini adalah tingkat natrium serum dalam darah, diukur dalam miliequivalents per liter (mEq/L). Natrium adalah elektrolit penting dalam tubuh dan tingkat natrium serum dapat memberikan informasi tentang keseimbangan elektrolit yang dapat mempengaruhi fungsi jantung. Fitur ini bertipe data numerik.
11. **Merokok (Smoking)**: Ini adalah fitur biner yang menunjukkan apakah pasien merokok atau tidak. Nilai 1 / "true" menunjukkan kebiasaan merokok, sementara 0 / "false" menunjukkan ketiadaan kebiasaan merokok. Merokok adalah faktor risiko yang signifikan dalam perkembangan penyakit jantung.
12. **Waktu (Time)**: Ini adalah periode pemantauan atau follow-up pasien dalam satuan hari (days). Fitur ini mengukur lamanya pasien dipantau dalam penelitian. Ini adalah informasi penting dalam analisis klinis dan penelitian lanjutan. Fitur ini bertipe data numerik.
13. **Kejadian Kematian (Death Event)**: Fitur target "death event" adalah sebuah fitur yang digunakan untuk menunjukkan apakah pasien mengalami kematian selama periode pemantauan atau tidak. Fitur ini bersifat boolean, yang berarti nilainya hanya dapat berupa dua kemungkinan: 1 / True (benar) atau 0 / False (salah). Jika "death event" memiliki nilai True, ini berarti bahwa pasien tersebut mengalami kematian selama periode pemantauan yang dicatat dalam dataset. Sebaliknya, jika "death event" memiliki nilai False, maka ini menunjukkan bahwa pasien tersebut masih hidup atau tidak mengalami kematian selama periode tersebut.

0.6.2 Eksplorasi Dataset / Satistik Dataset:

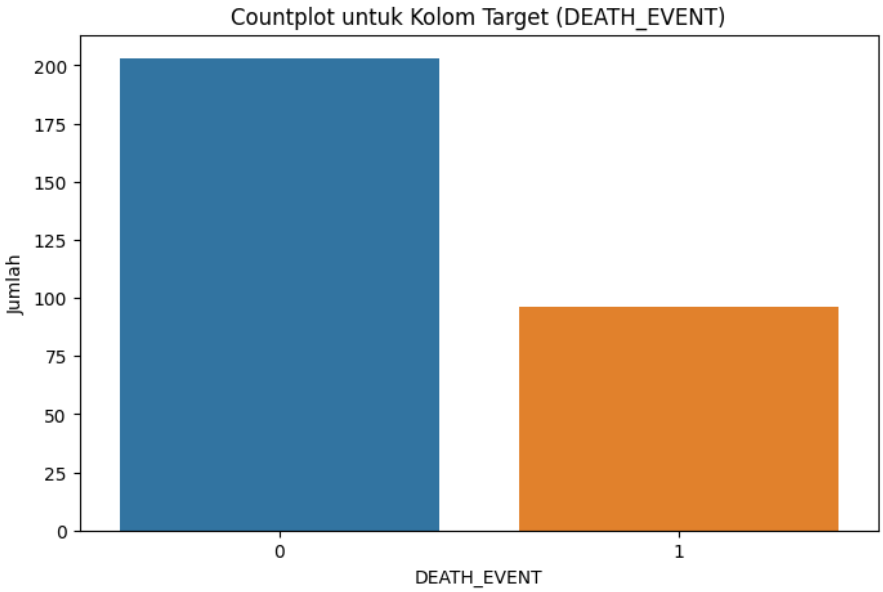
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Baca data dari URL CSV
url = "https://raw.githubusercontent.com/whintaaa/datapsd/main/heart_failure_clinical_records_d
data = pd.read_csv(url)


# Informasi Umum tentang Dataset
info = data.info()

# Statistik Deskriptif untuk Kolom-Kolom Numerik
describe = data.describe()
```

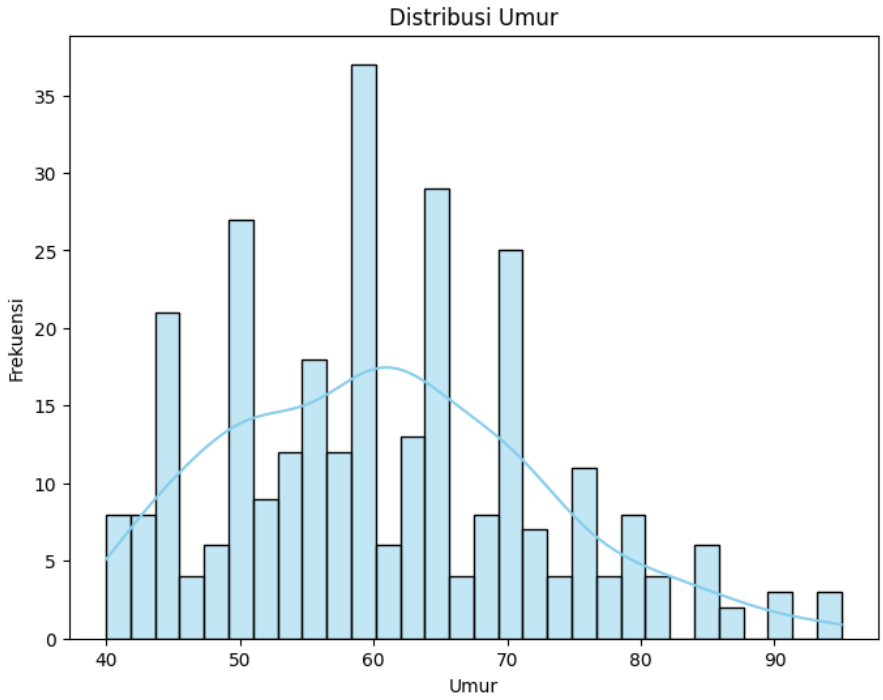
```
# Jumlah Nilai yang Hilang untuk Setiap Kolom
missing_values = data.isnull().sum()

# Beberapa Baris Pertama dari Dataset
head = data.head()

# Jumlah Unik untuk Kolom Target (DEATH_EVENT)
target_counts = data['DEATH_EVENT'].value_counts()

# Korelasi Antar Kolom Numerik
correlation_matrix = data.corr()

# Distribusi Umur (Age)
age_distribution = data['age'].value_counts()

# Visualisasi Data
plt.figure(figsize=(15, 10))

# Countplot untuk Kolom Target (DEATH_EVENT)
plt.figure(figsize=(8, 5))
sns.countplot(x='DEATH_EVENT', data=data)
plt.title('Countplot untuk Kolom Target (DEATH_EVENT)')
plt.xlabel('DEATH_EVENT')
plt.ylabel('Jumlah')
plt.show()

# Grafik distribusi umur (Age)
plt.figure(figsize=(8, 6))
sns.histplot(data['age'], bins=30, kde=True, color='skyblue')
plt.title('Distribusi Umur')
plt.xlabel('Umur')
plt.ylabel('Frekuensi')
plt.show()

# Grafik korelasi antara umur (Age) dan kadar serum kreatinin (Serum Creatinine)
plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='serum_creatinine', data=data, hue='sex', palette='viridis')
plt.title('Korelasi Umur dan Kadar Serum Kreatinin')
plt.xlabel('Umur')
plt.ylabel('Kadar Serum Kreatinin')
plt.legend(title='Jenis Kelamin', loc='upper right', labels=['Female', 'Male'])
plt.show()
```

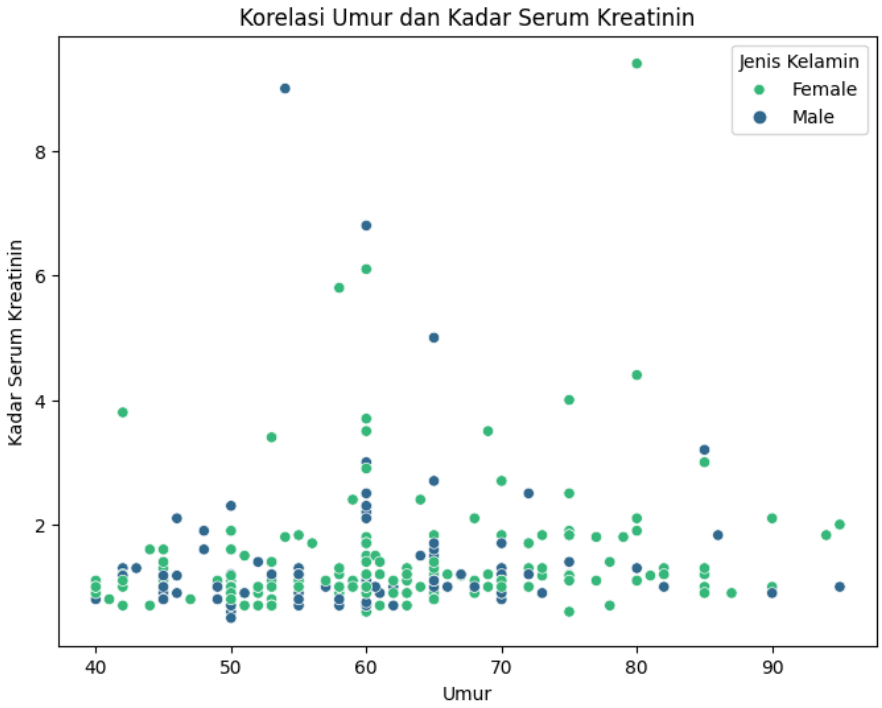
```
# Contoh menampilkan hasil
print("Informasi Umum tentang Dataset:")
print(info)

print("\nJumlah Nilai yang Hilang:")
print(missing_values)

print("\nJumlah data untuk Kolom Target (DEATH_EVENT):")
print(target_counts)

# Statistik deskriptif
statistics = data.describe()

# Menampilkan hasil
print("\nStatistik Deskriptif:")
print(statistics)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    int64
 2   creatinine_phosphokinase  299 non-null    int64
 3   diabetes                  299 non-null    int64
 4   ejection_fraction         299 non-null    int64
 5   high_blood_pressure       299 non-null    int64
 6   platelets                 299 non-null    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    int64
 9   sex                       299 non-null    int64
 10  smoking                   299 non-null    int64
 11  time                      299 non-null    int64
 12  DEATH_EVENT               299 non-null    int64
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

```
<Figure size 1500x1000 with 0 Axes>
```


Countplot untuk Kolom Target (DEATH_EVENT)
Jumlah
200
175
150
125
100
75
50
25
0
0
1
DEATH_EVENT


Distribusi Umur
Frekuensi
35
30
25
20
15
10
5
0
40
50
60
70
80
90
Umur




```
Informasi Umum tentang Dataset:
None

Jumlah Nilai yang Hilang:
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64

Jumlah data untuk Kolom Target (DEATH_EVENT):
0    203
```

```
1     96
Name: DEATH_EVENT, dtype: int64

Statistik Deskriptif:
              age     anaemia  creatinine_phosphokinase    diabetes  \
count  299.000000  299.000000                299.000000  299.000000
mean    60.833893    0.431438                581.839465    0.418060
std     11.894809    0.496107                970.287881    0.494067
min     40.000000    0.000000                 23.000000    0.000000
25%     51.000000    0.000000                116.500000    0.000000
50%     60.000000    0.000000                250.000000    0.000000
75%     70.000000    1.000000                582.000000    1.000000
max     95.000000    1.000000               7861.000000    1.000000

       ejection_fraction  high_blood_pressure      platelets  \
count         299.000000           299.000000     299.000000
mean           38.083612             0.351171  263358.029264
std            11.834841             0.478136   97804.236869
min            14.000000             0.000000   25100.000000
25%            30.000000             0.000000  212500.000000
50%            38.000000             0.000000  262000.000000
75%            45.000000             1.000000  303500.000000
max            80.000000             1.000000  850000.000000

      serum_creatinine  serum_sodium        sex    smoking       time  \
count        299.00000    299.000000  299.000000  299.00000  299.000000
mean           1.39388    136.625418    0.648829    0.32107  130.260870
std            1.03451      4.412477    0.478136    0.46767   77.614208
min            0.50000    113.000000    0.000000    0.00000    4.000000
25%            0.90000    134.000000    0.000000    0.00000   73.000000
50%            1.10000    137.000000    1.000000    0.00000  115.000000
75%            1.40000    140.000000    1.000000    1.00000  203.000000
max            9.40000    148.000000    1.000000    1.00000  285.000000

       DEATH_EVENT
count    299.00000
mean       0.32107
std        0.46767
min        0.00000
25%        0.00000
50%        0.00000
75%        1.00000
max        1.00000
```

0.7 Data Prepocessing

0.7.1 Oversampling

Melihat jumlah masing-masing target pada kolom 'death event':

```
# Menghitung jumlah masing-masing target pada kolom 'death event'
jumlah_death_event_1 = df[df['DEATH_EVENT'] == 1].shape[0]
jumlah_death_event_0 = df[df['DEATH_EVENT'] == 0].shape[0]

# Menampilkan jumlah masing-masing target
print("Jumlah Target 'death event' dengan Nilai 1:", jumlah_death_event_1)
print("Jumlah Target 'death event' dengan Nilai 0:", jumlah_death_event_0)
```

```
Jumlah Target 'death event' dengan Nilai 1: 96
Jumlah Target 'death event' dengan Nilai 0: 203
```

Bisa dilihat jumlah target dengan nilai 1 = 96 dan nilai 0 = 203 ini menandakan bahwa jumlah target pada dataset tidak seimbang. Maka salah satu metode untuk menyeimbangkan target bisa menggunakan teknik oversampling. Teknik oversampling adalah salah satu pendekatan untuk menyeimbangkan dataset yang tidak seimbang dengan meningkatkan jumlah sampel dalam kategori minoritas. Kategori minoritas adalah kelas target yang memiliki frekuensi yang lebih rendah dibandingkan dengan kelas mayoritas. Teknik oversampling dilakukan dengan cara menambahkan lebih banyak contoh dari kategori minoritas agar jumlahnya sebanding dengan kategori mayoritas.

Ada beberapa metode oversampling yang umum digunakan, dan salah satunya adalah RandomOverSampler. Dalam RandomOverSampler, sampel acak dari kategori minoritas ditambahkan kembali ke dataset hingga jumlahnya setara dengan jumlah sampel dalam kategori mayoritas.

Berikut adalah langkah-langkah umum untuk menggunakan teknik oversampling:

1. Identifikasi dataset yang tidak seimbang.
2. Pisahkan fitur (X) dan target (y).
3. Terapkan teknik oversampling pada kategori minoritas.
4. Gabungkan kembali data yang sudah diresampling.
5. Lanjutkan dengan analisis atau pemodelan seperti biasa.

Dengan menggunakan teknik oversampling, kita meningkatkan jumlah sampel di kategori minoritas (DEATH_EVENT = 1) sehingga seimbang dengan kategori mayoritas (DEATH_EVENT = 0).

```
# Print the column names to identify the correct target variable
print(df.columns)

# Pilih kolom-kolom yang perlu dinormalisasi / bertype numerik
numerical_columns = ['age', 'creatinine_phosphokinase', 'ejection_fraction', 'platelets', 'seru

# Langkah 3: Split data menjadi fitur (X) dan target (y)
X = df.drop(columns=['DEATH_EVENT'])
y = df['DEATH_EVENT']

# Menggunakan teknik oversampling dengan RandomOverSampler
oversampler = RandomOverSampler(random_state=42)
X_resampled, y_resampled = oversampler.fit_resample(X, y)

# Membuat dataframe baru setelah oversampling
df_resampled = pd.concat([X_resampled, y_resampled], axis=1)

# Menampilkan jumlah target setelah oversampling
print("Jumlah Target setelah Oversampling:")
print(df_resampled['DEATH_EVENT'].value_counts())
```

```
Index(['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
       'ejection_fraction', 'high_blood_pressure', 'platelets',
       'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time',
       'DEATH_EVENT'],
      dtype='object')
Jumlah Target setelah Oversampling:
1    203
0    203
Name: DEATH_EVENT, dtype: int64
```

0.7.2 Normalisasi Menggunakan Zscore

Normalisasi Z-score adalah teknik normalisasi yang digunakan untuk mengubah setiap nilai dalam suatu variabel ke dalam skala yang memiliki rata-rata nol dan deviasi standar satu. Ini adalah cara umum untuk menormalkan data sehingga nilai-nilai yang berbeda dari variabel yang sama dapat dibandingkan secara langsung.

Proses normalisasi Z-score melibatkan mengurangkan rata-rata dari setiap nilai dalam variabel dan membaginya dengan deviasi standar. Formula normalisasi Z-score untuk suatu nilai (x) dalam variabel (X) adalah sebagai berikut:

$[z = \frac{x - \text{mean}(X)}{\text{std}(X)}]$

di mana: - $(z $) adalah nilai hasil normalisasi (Z-score) dari (x). - $(mean(X) $) adalah rata-rata dari variabel (X). - $(std(X) $) adalah deviasi standar dari variabel (X).

Proses ini menghasilkan distribusi data yang memiliki rata-rata nol dan deviasi standar satu. Normalisasi Z-score sangat berguna dalam beberapa konteks, terutama ketika Anda ingin membandingkan nilai-nilai dari variabel yang memiliki skala yang berbeda.

sebelum di mormalisasi terdapat fitur boolean pada dataset maka harus dipidahkan dengan fitur numerik karena hanya fitur numerik yang akan di normalisasi kemudian normalisasi Z-score diaktifkan dengan parameter `normalize=True` dan `normalize_method='zscore'`. PyCaret akan otomatis menormalisasi fitur-fitur numerik yang ditentukan menggunakan normalisasi Z-score. Lalu secara default pycaret akan membagi dataset menjadi 70% data train dan 30% data test, maka untuk dataset ini 209 menjadi data train dan 90 menjadi data test.

```
# Print the column names to identify the correct target variable
print(df.columns)
# Pilih kolom-kolom yang perlu dinormalisasi / bertype numerik
numerical_columns = ['age', 'creatinine_phosphokinase', 'ejection_fraction', 'platelets', 'serum

# Split data menjadi fitur (X) dan target (y)
X = df.drop(columns=['DEATH_EVENT'])
y = df['DEATH_EVENT']

# Inisialisasi ekperimen PyCaret
exp = setup(data=df, target='DEATH_EVENT', normalize=True, normalize_method='zscore', numeric_f
```

```
Index(['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
       'ejection_fraction', 'high_blood_pressure', 'platelets',
       'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time',
       'DEATH_EVENT'],
      dtype='object')
```

Table 0.5

	Description	Value
0	Session id	8772
1	Target	DEATH_EVENT
2	Target type	Binary
3	Original data shape	(299, 13)
4	Transformed data shape	(299, 13)
5	Transformed train set shape	(209, 13)

	Description	Value
6	Transformed test set shape	(90, 13)
7	Numeric features	6
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Normalize	True
13	Normalize method	zscore
14	Fold Generator	StratifiedKFold
15	Fold Number	10
16	CPU Jobs	-1
17	Use GPU	False
18	Log Experiment	False
19	Experiment Name	clf-default-name
20	USI	1259

0.8 Modelling

0.8.1 Mencari Model Terbaik Menggunakan Pycaret

Pycaret adalah library Python yang menyederhanakan proses pengembangan model machine learning. Dengan fitur otomatis seperti setup data, pemilihan model, optimasi hyperparameter, dan visualisasi hasil, Pycaret memungkinkan pengguna untuk fokus pada inti pemodelan tanpa menulis banyak kode. Berikut implementasi pycaret untuk mencari model terbaik untuk memprediksi resiko gagal jantung pada pasien:

```
# Bandingkan model dan cari yang terbaik
best_model = compare_models()

best_model
```

```
<IPython.core.display.HTML object>
```

Table 0.6

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
rf	Random Forest Classifier	0.8274	0.8750	0.6500	0.8216	0.6962	0.5805	0.6064
lr	Logistic Regression	0.8131	0.8522	0.6476	0.7370	0.6703	0.5442	0.5588
ridge	Ridge Classifier	0.8131	0.0000	0.6476	0.7370	0.6703	0.5442	0.5588

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lda	Linear Discriminant Analysis	0.8131	0.8535	0.6476	0.7370	0.6703	0.5442	0.5588
gbc	Gradient Boosting Classifier	0.8129	0.8676	0.6762	0.7685	0.6904	0.5589	0.5808
xgboost	Extreme Gradient Boosting	0.7938	0.8795	0.6786	0.7238	0.6787	0.5290	0.5449
lightgbm	Light Gradient Boosting Machine	0.7938	0.8655	0.6643	0.7437	0.6740	0.5257	0.5457
et	Extra Trees Classifier	0.7843	0.8326	0.4976	0.7825	0.5760	0.4448	0.4792
ada	Ada Boost Classifier	0.7700	0.8135	0.5500	0.7420	0.5950	0.4430	0.4740
dt	Decision Tree Classifier	0.7650	0.7179	0.6000	0.7023	0.5992	0.4397	0.4670
nb	Naive Bayes	0.7462	0.7793	0.4881	0.6703	0.5337	0.3714	0.3930
qda	Quadratic Discriminant Analysis	0.7367	0.7423	0.4429	0.6314	0.5058	0.3382	0.3544
svm	SVM - Linear Kernel	0.7314	0.0000	0.6214	0.5955	0.5924	0.3960	0.4105
knn	K Neighbors Classifier	0.7124	0.7169	0.2524	0.5750	0.3261	0.2046	0.2372
dummy	Dummy Classifier	0.6795	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000

```
Processing:   0%|          | 0/65 [00:00<?, ?it/s]
```

```
<IPython.core.display.HTML object>
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='sqrt',
                             max_leaf_nodes=None, max_samples=None,
                             min_impurity_decrease=0.0, min_samples_leaf=1,
                          min_samples_split=2, min_weight_fraction_leaf=0.0,
                             n_estimators=100, n_jobs=-1, oob_score=False,
                             random_state=8111, verbose=0, warm_start=False)
```

0.9 Evaluasi

0.9.1 Menyimpan Model terbaik Menggunakan Pycaret

Bisa dilihat diatas bahwa model terbaik salah satunya adalah Random Forest Classifier. maka kita simpan model Random Forest Classifier untuk prediksi nantinya.

0.9.1.1 Random Forest Classifier:

Random Forest adalah algoritma machine learning yang termasuk dalam kategori ensemble learning. Ensemble learning menggabungkan prediksi dari beberapa model untuk meningkatkan performa dan ketahanan terhadap overfitting. Random Forest dapat digunakan untuk tugas klasifikasi (seperti prediksi kategori) dan regresi (prediksi nilai numerik).

0.9.1.2 Cara Kerja:

1. **Pembuatan Banyak Pohon (Trees):**
   - Random Forest terdiri dari sejumlah besar pohon keputusan yang dibuat secara acak. Setiap pohon dalam Random Forest dibuat berdasarkan subset acak dari data pelatihan dan fitur-fiturnya.
2. **Bootstrap Sampling (Bootstrapped Dataset):**
   - Pada setiap langkah pembuatan pohon, dilakukan bootstrap sampling, yaitu pengambilan sampel acak dengan penggantian dari dataset pelatihan. Beberapa data dapat muncul lebih dari sekali, dan beberapa mungkin tidak dipilih.
3. **Pemilihan Fitur Secara Acak:**
   - Pada setiap langkah pembuatan pohon, juga dilakukan pemilihan acak dari fitur-fitur yang tersedia. Ini membantu dalam menciptakan variasi antar pohon.
4. **Pembuatan Pohon Keputusan:**
   - Setiap pohon dibuat menggunakan data sampel dari bootstrap dan fitur yang dipilih secara acak. Pemisahan (split) di setiap node pohon dilakukan berdasarkan kriteria seperti Gini Impurity untuk klasifikasi atau Mean Squared Error untuk regresi.
5. **Voting (Klasifikasi):**
   - Untuk tugas klasifikasi, setelah semua pohon selesai membuat prediksi, hasilnya diambil berdasarkan mayoritas voting. Kelas dengan voting terbanyak dianggap sebagai prediksi akhir.

jika (N) adalah jumlah pohon dalam Random Forest, dan (h_i(x)) adalah hasil prediksi dari pohon ke-i, maka hasil akhir (H(x)) dari Random Forest dapat dihitung sebagai berikut:

Rumus untuk Klasifikasi Random Forest:

$[H(x) = mode(h_1(x), h_2(x), ..., h_N(x)) $]

penjelasan:

- $(H(x))$ adalah prediksi akhir dari ensemble model.
- $(h_i(x))$ adalah prediksi dari model ke-(i).
- (mode) merujuk pada nilai yang paling sering muncul atau kelas yang paling sering diprediksi di antara prediksi model-individu.

Rumus ini, sesuai dengan prinsip mayoritas voting pada ensambel model seperti Random Forest. Model ensambel, seperti Random Forest, cenderung memberikan performa yang baik dalam berbagai jenis dataset, termasuk dataset kesehatan seperti "Heart failure clinical records".

Dalam kasus dataset kesehatan seperti ini, Random Forest bisa menjadi pilihan yang baik karena:

1. Robust terhadap Overfitting: Random Forest mampu mengatasi masalah overfitting yang mungkin muncul pada pohon keputusan tunggal, karena hasil mayoritas dari banyak pohon keputusan.
2. Tidak Sensitif terhadap Outliers: Random Forest dapat menangani data yang tidak seimbang dan keberadaan outlier dalam dataset.
3. Interpretability: Meskipun Random Forest cenderung tidak seinterpretatif pohon keputusan tunggal, tetapi masih memberikan pemahaman yang baik tentang pentingnya fitur dalam membuat keputusan.
4. Handling Fitur Numerik dan Kategorikal: Random Forest dapat menangani baik fitur numerik maupun kategorikal tanpa memerlukan transformasi khusus.
5. Performa yang Baik secara Umum: Random Forest umumnya memberikan performa yang baik tanpa perlu penyesuaian parameter yang terlalu rumit.

```
best_model = create_model('rf')

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

# Load model dari file pickle
with open(model_filename, 'rb') as file:
    loaded_model = pickle.load(file)
```

```
<IPython.core.display.HTML object>
```

Table 0.7

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7619	0.7806	0.5714	0.6667	0.6154	0.4444	0.4472
1	0.8095	0.9184	0.4286	1.0000	0.6000	0.5000	0.5774
2	0.9048	0.9592	0.8571	0.8571	0.8571	0.7857	0.7857
3	0.7619	0.9286	1.0000	0.5833	0.7368	0.5455	0.6124
4	0.7619	0.8061	0.4286	0.7500	0.5455	0.4000	0.4287
5	0.9524	0.9796	0.8571	1.0000	0.9231	0.8889	0.8944
6	0.8571	0.9235	0.7143	0.8333	0.7692	0.6667	0.6708
7	0.8571	0.9222	0.8333	0.7143	0.7692	0.6667	0.6708
8	0.8571	0.8667	0.5000	1.0000	0.6667	0.5882	0.6455

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
9	0.7000	0.8452	0.5000	0.5000	0.5000	0.2857	0.2857
Mean	0.8224	0.8930	0.6690	0.7905	0.6983	0.5772	0.6019
Std	0.0731	0.0622	0.1979	0.1697	0.1294	0.1725	0.1689

```
Processing:   0%|          | 0/4 [00:00<?, ?it/s]
```

```
<IPython.core.display.HTML object>
```

0.9.2 Prediksi Death_Event Menggunakan Random Forest Classifier (rf) dengan Data Baru

Prediksi Menggunakan Random Forest untuk Klasifikasi:

1. **Persiapkan Data Baru:**
   - Masukkan data baru ke dalam model dengan memberikan nilai untuk setiap fitur yang sesuai.
2. **Lakukan Prediksi pada Setiap Pohon:**
   - Setiap pohon dalam Random Forest memberikan prediksi berdasarkan data baru.
3. **Aggregasi Hasil Prediksi:**
   - Hasil prediksi dari setiap pohon diambil dan dihitung mayoritas voting.
4. **Tentukan Kelas Akhir:**
   - Hasil akhir diambil berdasarkan mayoritas voting sebagai kelas prediksi akhir.

Contoh Kode:

```
# Membaca data baru yang akan diprediksi
new_data = pd.DataFrame([[age, anaemia, creatinine_phosphokinase, diabetes, ejection_fraction,

# Memuat model terbaik dari file pickle
loaded_model = load_model('best_model')

# Melakukan prediksi pada data baru
predictions = predict_model(loaded_model, data=new_data)

# Menampilkan hasil prediksi
if predictions['prediction_label'].iloc[0] == 1:
    print("Hasil Prediksi: Pasien berisiko mengalami DEATH_EVENT")
else:
    print("Hasil Prediksi: Pasien tidak berisiko mengalami DEATH_EVENT")
```

Dalam kode ini, `prediction_label` adalah kolom yang berisi prediksi kelas (0 atau 1) dari model Random Forest untuk tugas klasifikasi. Jika nilai Label adalah 1, itu berarti pasien berisiko mengalami DEATH_EVENT; jika 0, itu berarti pasien tidak berisiko.

```
# Membaca data baru yang akan diprediksi
age = float(input("Masukkan nilai Age: "))
anaemia = int(input("Masukkan nilai Anaemia (0 untuk Tidak, 1 untuk Ya): "))
creatinine_phosphokinase = float(input("Masukkan nilai Creatinine Phosphokinase: "))
diabetes = int(input("Masukkan nilai Diabetes (0 untuk Tidak, 1 untuk Ya): "))
ejection_fraction = float(input("Masukkan nilai Ejection Fraction: "))
high_blood_pressure = int(input("Masukkan nilai High Blood Pressure (0 untuk Tidak, 1 untuk Ya)
platelets = float(input("Masukkan nilai Platelets: "))
serum_creatinine = float(input("Masukkan nilai Serum Creatinine: "))
serum_sodium = float(input("Masukkan nilai Serum Sodium: "))
sex = int(input("Masukkan nilai Sex (0 untuk Perempuan, 1 untuk Laki-laki): "))
smoking = int(input("Masukkan nilai Smoking (0 untuk Tidak, 1 untuk Ya): "))
time = float(input("Masukkan nilai Time: "))

# Membuat data baru untuk prediksi
new_data = pd.DataFrame([[age, anaemia, creatinine_phosphokinase, diabetes, ejection_fraction,

# Memuat model terbaik dari file pickle
loaded_model = load_model('best_model')

# Melakukan prediksi pada data baru
predictions = predict_model(loaded_model, data=new_data)

# Menampilkan hasil prediksi
if predictions['prediction_label'].iloc[0] == 1:
    print("Hasil Prediksi: Pasien berisiko mengalami DEATH_EVENT")
else:
    print("Hasil Prediksi: Pasien tidak berisiko mengalami DEATH_EVENT")
```

```
Masukkan nilai Age: 60
Masukkan nilai Anaemia (0 untuk Tidak, 1 untuk Ya): 0
Masukkan nilai Creatinine Phosphokinase: 528
Masukkan nilai Diabetes (0 untuk Tidak, 1 untuk Ya): 0
Masukkan nilai Ejection Fraction: 20
Masukkan nilai High Blood Pressure (0 untuk Tidak, 1 untuk Ya): 1
Masukkan nilai Platelets: 125000
Masukkan nilai Serum Creatinine: 1.4
Masukkan nilai Serum Sodium: 13
Masukkan nilai Sex (0 untuk Perempuan, 1 untuk Laki-laki): 1
Masukkan nilai Smoking (0 untuk Tidak, 1 untuk Ya): 0
```

```
Masukkan nilai Time: 9
Transformation Pipeline and Model Successfully Loaded

<IPython.core.display.HTML object>

Hasil Prediksi: Pasien berisiko mengalami DEATH_EVENT
```

0.10 Deployment

0.10.1 Link Streamlit/Aplikasi Prediksi pada dataset Heart failure clinical records:

https://psd-prediksi-jantung.streamlit.app[1]

[1] https://psd-prediksi-jantung.streamlit.app/