Segmentasi Pelanggan menggunakan Pembelajaran Mesin Tanpa Pengawasan

Segmentasi Pelanggan melibatkan pengelompokan pelanggan berdasarkan karakteristik, perilaku, dan preferensi bersama. Dengan mengelompokkan pelanggan, bisnis dapat menyesuaikan strategi mereka dan menargetkan kelompok tertentu secara lebih efektif dan meningkatkan nilai pasar secara keseluruhan. Hari ini kita akan menggunakan **Pembelajaran Mesin Tanpa Pengawasan** untuk melakukan Segmentasi Pelanggan di Python.

Step 1: Import Libraries

Kita akan mulai dengan mengimpor perpustakaan yang diperlukan seperti Pandas, Numpy, Matplotlib, Seaborn, dan Sklearn.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.cluster import KMeans

import warnings
warnings.filterwarnings('ignore')
```

Step 2: Load the Dataset

Muat himpunan data yang berisi detail pelanggan seperti status perkawinan, pendapatan, jumlah barang yang dibeli, jenis barang yang dibeli, dan lainnya

```
df = pd.read_csv('new.csv')

df.head()
```

Untuk memeriksa bentuk dataset kita dapat menggunakan metode data.shape.

Step 3: Pra-pemrosesan Data
Untuk mendapatkan informasi dataset seperti memeriksa nilai null, jumlah nilai, dll. kita akan menggunakan metode .info().

```
df.info()
```

Menggambarkan dataset
```
df.describe().T
```

Untuk memeriksa nilai null dalam himpunan data.

```
for col in df.columns:
   temp = df[col].isnull().sum()
   if temp > 0:
      print(f'Column {col} contains {temp} null values.')
```

Sekarang, setelah kita memiliki jumlah nilai nol dan kita tahu nilainya sangat sedikit, kita dapat menjatuhkannya karena tidak akan banyak mempengaruhi kumpulan data

```
df = df.dropna()
print("Total values in the dataset after removing the null values:", len(df))
```

Untuk menemukan jumlah total nilai unik di setiap kolom kita dapat menggunakan data.unique() method.

```
df.nunique()
```

Di sini kita dapat mengamati bahwa ada kolom yang berisi nilai tunggal di seluruh kolom sehingga, mereka tidak memiliki relevansi dalam pengembangan model.
Juga dataset memiliki Dt_Customer kolom yang berisi kolom tanggal, kita dapat mengonversi menjadi 3 kolom yaitu hari, bulan, tahun.

```
parts = df["Dt_Customer"].str.split("-", n=3, expand=True)
df["day"] = parts[0].astype('int')
df["month"] = parts[1].astype('int')
df["year"] = parts[2].astype('int')
```

Sekarang kita memiliki semua fitur penting, sekarang kita dapat menjatuhkan fitur seperti **Z_CostContact, Z_Revenue, Dt_Customer.**

```
df.drop(['Z_CostContact', 'Z_Revenue', 'Dt_Customer'],
        axis=1,
        inplace=True)
```

Step 4: Visualisasi dan Analisis Data
Visualisasi data adalah representasi grafis informasi dan data dalam format bergambar atau grafis. Di sini kita akan menggunakan plot batang dan plot hitung untuk visualisasi yang lebih baik.

```
floats, objects = [], []
for col in df.columns:
    if df[col].dtype == object:
        objects.append(col)
    elif df[col].dtype == float:
        floats.append(col)

print(objects)
print(floats)
```

Untuk mendapatkan plot hitungan untuk kolom objek tipe data -, lihat kode di bawah ini.

```
plt.subplots(figsize=(15, 10))
for i, col in enumerate(objects):
    plt.subplot(2, 2, i + 1)
    sb.countplot(df[col])
plt.show()
```

Mari kita periksa value_counts Marital_Status data.

```
df['Marital_Status'].value_counts()
```

Sekarang mari kita lihat perbandingan fitur sehubungan dengan nilai respons.

```
plt.subplots(figsize=(15, 10))
for i, col in enumerate(objects):
   plt.subplot(2, 2, i + 1)

   df_melted = df.melt(id_vars=[col], value_vars=['Response'], var_name='hue')
   sb.countplot(x=col, hue='value', data=df_melted)
plt.show()
```

Pengkodean Label *digunakan untuk mengubah nilai kategoris menjadi nilai numerik sehingga model dapat memahaminya.*

```
for col in df.columns:
   if df[col].dtype == object:
      le = LabelEncoder()
      df[col] = le.fit_transform(df[col])
```

Peta panas adalah cara terbaik untuk memvisualisasikan korelasi antara berbagai fitur kumpulan data. Mari kita berikan nilai 0,8

```
plt.figure(figsize=(15, 15))
sb.heatmap(df.corr() > 0.8, annot=True, cbar=False)
plt.show()
```

Standardisasi adalah metode penskalaan fitur yang merupakan bagian integral dari rekayasa fitur. Ini memperkecil skala data dan memudahkan model pembelajaran mesin untuk belajar darinya. Ini mengurangi rata-rata menjadi '0' dan standar deviasi menjadi '1'.

```
scaler = StandardScaler()
data = scaler.fit_transform(df)
```

Step 5: Segmentasi

Kami akan menggunakan Penyematan Tetangga Stokastik yang didistribusikan T. Ini membantu dalam memvisualisasikan data dimensi tinggi. Ini mengubah kesamaan antara titik data menjadi probabilitas bersama dan mencoba meminimalkan nilai menjadi penyematan dimensi rendah.

```
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0)
tsne_data = model.fit_transform(df)
plt.figure(figsize=(7, 7))
plt.scatter(tsne_data[:, 0], tsne_data[:, 1])
plt.show()
```

Tentu saja ada beberapa cluster yang jelas visual dari representasi 2-D dari data yang diberikan. KMeans Clustering juga dapat digunakan untuk mengelompokkan berbagai titik dalam bidang.

```
error = []
for n_clusters in range(1, 21):
  model = KMeans(init='k-means++',
           n_clusters=n_clusters,
           max_iter=500,
           random_state=22)
  model.fit(df)
  error.append(model.inertia_)
```

Di sini inersia tidak lain adalah jumlah jarak kuadrat di dalam kelompok.

```
plt.figure(figsize=(10, 5))
sb.lineplot(x=range(1, 21), y=error)
```

```
sb.scatterplot(x=range(1, 21), y=error)
plt.show()
```

Di sini dengan menggunakan metode siku kita dapat mengatakan bahwa k = 6 adalah jumlah cluster optimal yang harus dibuat karena setelah k = 6 nilai inersia tidak menurun secara drastis.

```
model = KMeans(init='k-means++',
               n_clusters=5,
               max_iter=500,
               random_state=22)
segments = model.fit_predict(df)
```

Scatterplot akan digunakan untuk melihat semua 6 cluster yang dibentuk oleh KMeans Clustering.

```
plt.figure(figsize=(7, 7))

df_tsne = pd.DataFrame({'x': tsne_data[:, 0], 'y': tsne_data[:, 1], 'segment': segments})

sb.scatterplot(x='x', y='y', hue='segment', data=df_tsne)
plt.show()
```

Di sini kita dapat melihat bahwa kita telah membagi pelanggan menjadi 5 klaster dan berdasarkan klaster ini kita dapat menargetkan pelanggan dengan perilaku pembelian yang sama jauh lebih baik. Kami dapat memberikan iklan yang dipersonalisasi dan dapat membuat keputusan yang tepat tentang bisnis untuk pertumbuhan yang lebih baik.