

PEMBELAJARAN MESIN DAN PEMBELAJARAN MENDALAM

Modul 4

Unsupervised Learning

Yohanes Sigit Purnomo Wuryo Putro, ST., M.Kom., Ph.D.

Aloysius Gonzaga Pradnya Sidhwara, S.T., M.Eng.

Kimmi Caitlyn Ariesta

F Jonathan Putra Prasetyo

Tugas Modul Unsupervised Learning

Dalam rangka meningkatkan layanan dan dukungan kepada nasabah di Bank Ayodya, tim marketing berupaya untuk memperkirakan perilaku penggunaan kartu kredit nasabah berdasarkan rata-rata limit kredit, total kartu kredit yang dimiliki, total kunjungan ke bank oleh nasabah, total kunjungan online, dan total panggilan yang dilakukan oleh nasabah.

Instruksi dari pimpinan tim data scientist yang Anda terima adalah sebagai berikut:

Terima kasih atas kinerja baik Anda dalam proyek-proyek pengembangan model supervised learning. Kali ini kita mendapatkan klien yang berbeda yaitu dengan kasus clustering. Berikut ini adalah instruksi terbaru yang saya berikan untuk membuat perbandingan kinerja model unsupervised learning menggunakan algoritma K-Means, Agglomerative, dan DBSCAN

1. Pahami dataset: Sebelum memulai pembuatan model, pastikan Anda memahami dataset yang akan digunakan. Periksa apakah terdapat missing value, hapus kolom data yang tidak diperlukan, eliminasi outlier, dan jika ada ubah data kategorik dari string menjadi numerik menggunakan One Hot Encoder
2. Tentukan fitur dan jumlah optimal dari cluster. Tentukan fitur-fitur yang digunakan untuk clustering. Gunakan Elbow Method dan Silhouette Score untuk menentukan jumlah optimal cluster untuk segmentasi nasabah
3. Buat masing-masing model algoritme: Untuk nilai parameter random state sesuaikan dengan dua digit terakhir nomor pegawai Anda (**random_state: dua/satu digit terakhir NPM**)
4. Evaluasi Model: Setelah membuat ketiga algoritma clustering, lakukan evaluasi untuk mengetahui performa masing-masing model. Gunakan Silhouette Score untuk mengevaluasi performa model. Supaya memudahkan tim, buatlah grafik hasil clustering dalam bentuk representasi visual tiga dimensi berdasarkan kolom rata-rata limit kredit, total kartu kredit, dan total kunjungan online

Semoga instruksi ini dapat membantu dalam membuat perbandingan kinerja model unsupervised learning menggunakan algoritma K-Means, Agglomerative, dan DBSCAN. Jika ada pertanyaan atau kesulitan, jangan ragu untuk menghubungi saya atau anggota tim lainnya.

Load Data

- Tahap pertama adalah import library yang dibutuhkan
- Load dataset berdasarkan path di mana dataset disimpan
- Karena dataset dalam bentuk file CSV maka kita menggunakan fungsi `read_csv` dari pandas

```
In [ ]: from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN #library untuk algoritma clustering
from sklearn.metrics import silhouette_score #library untuk evaluasi model clustering
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt #library untuk visualisasi data
import logging #library untuk menonaktifkan pesan Log yang dihasilkan matplotlib
logging.getLogger('matplotlib.font_manager').disabled = True #di HPC Library ini digunakan agar visualisasi tidak terganggu dengan pesan Log

#Load dataset menggunakan function read_csv dari pandas dan menampilkan dataset
credit = #gunakan pandas/pd untuk memasukan dataset
```

- Data yang hilang akan membuat model menjadi error, untuk itu kita perlu menanganinya
- Kita dapat melakukan pengecekan pada data apabila ada data yang bernilai null, kosong, atau Nan dengan cara seperti berikut:

```
In [ ]: # gunakan fungsi isnull(), isna(), dan empty dari DataFrame Pandas
```

```
#Lengkapi xx dengan fungsi yang relevan untuk melihat data yang rusak
print("data null \n",credit.xx().sum())
print("data kosong \n",credit.xx())
print("data nan \n",credit.xx().sum())
```

Data Cleansing

- Drop kolom yang tidak diperlukan



In []: #dalam data set ini, kolom Customer Key dan SL_No tidak diperlukan

```
#Drop fitur yang tidak relevan. clue = Customer key, SL_No  
df_credit = credit.xx(yy) #axis=1 digunakan untuk menghapus kolom  
df_credit.head() #gunakan untuk mengecek apakah fitur berhasil di drop
```

out[3]:

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
0	100000	2	1	1	0
1	50000	3	0	10	9
2	50000	7	1	3	4
3	30000	5	1	1	4
4	100000	6	0	12	3
5	20000	3	0	1	8
6	100000	5	0	11	2
7	15000	3	0	1	1
8	5000	2	0	2	2
9	3000	4	0	1	7

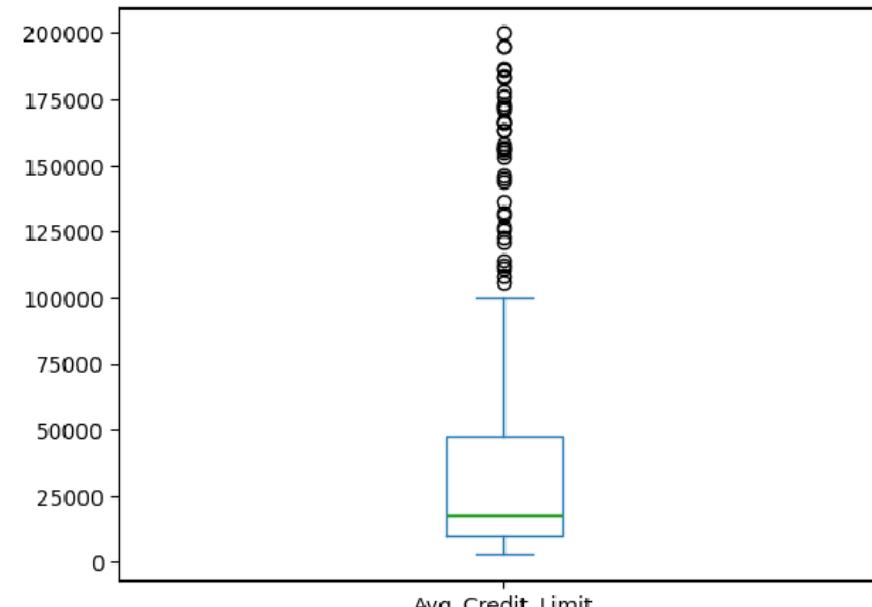
Outlier Detection

- Tampilkan sebaran nilai untuk kolom Avg_Credit_Limit menggunakan boxplot

In []:

```
df_credit['Avg_Credit_Limit'].plot(kind='box')
```

out[4]:



Copy

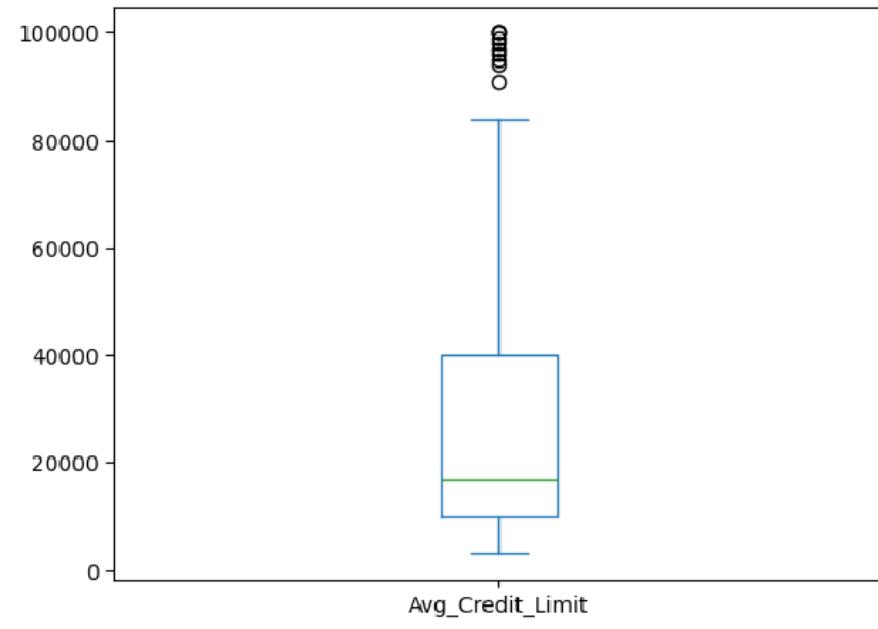
- buat fungsi untuk menghapus outlier menggunakan metode Inter-Quartile Range (IQR)
- Jika suatu nilai bernilai lebih besar dari batas atas (upper bound) dan lebih kecil dari batas bawah (lower bound), maka dihapus dari dataset

Watermarkly

```
In [ ]: from pandas.api.types import is_numeric_dtype  
  
def remove_outlier(data_frame, kolom_outlier):  
  
    #gunakan pembagian quantile 75:25  
    #gunakan fitur Avg_Credit_Limit untuk mendeteksi outlier  
  
    # Menghapus outlier hanya pada kolom "Avg_Credit_Limit"  
    data_bersih = #panggil fungsi yang baru saja dibuat  
  
    print("Jumlah data sebelum dibuang outlier:", df_credit.shape[0])  
    print("Jumlah data sesudah dibuang outlier:", credit_clean.shape[0])  
  
    #menampilkan kembali sebaran nilai dari kolom "Avg_Credit_Limit" setelah dibersihkan  
    data_bersih['Avg_Credit_Limit'].plot(kind='box')
```

Jumlah data sebelum dibuang outlier: 660
Jumlah data sesudah dibuang outlier: 621

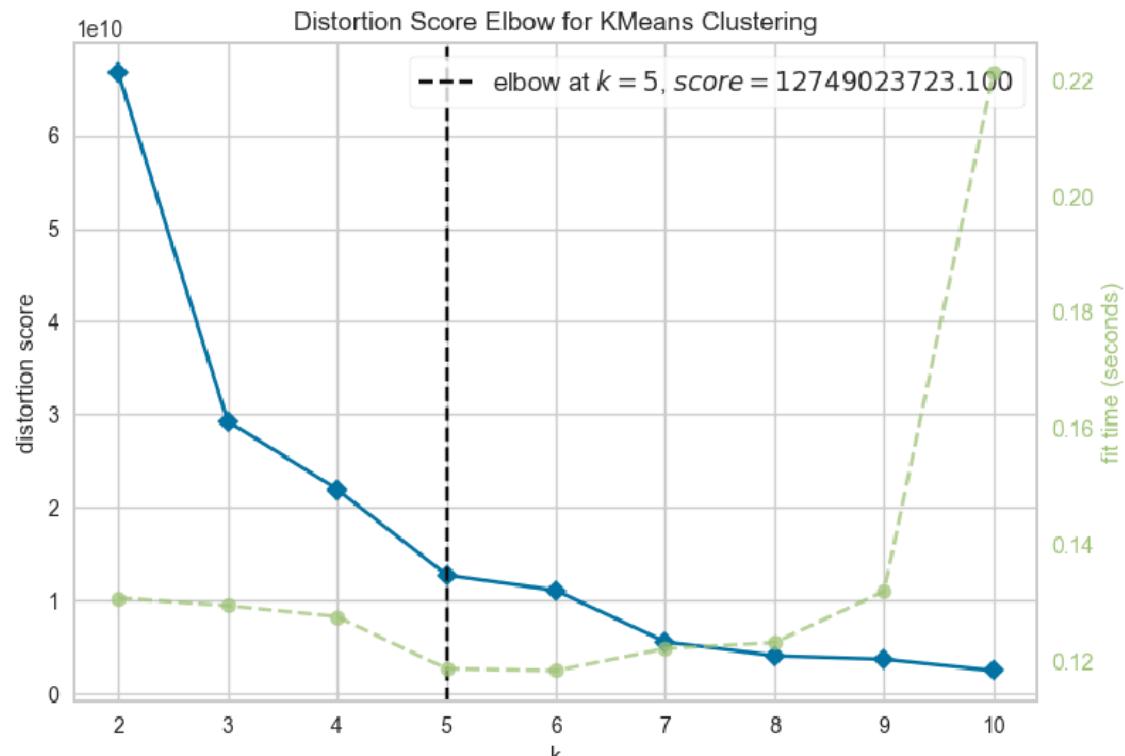
Out[5]: <Axes>



Elbow Method dan Silhouette Score menggunakan K-Means Clustering

- Install library yellowbrick menggunakan perintah pip install yellowbrick pada anaconda prompt
- Import KElbowVisualizer dan buat objek K-Means untuk clustering
- Isikan parameter KElbowVisualize untuk jumlah k antara 2-10 cluster

```
In [ ]: #import library warning untuk mematikan pesan peringatan yang mungkin muncul  
#dan import library KElbowVisualizer dari yellowbrick untuk mencari jumlah cluster yang optimal  
  
import warnings  
warnings.filterwarnings('ignore')  
from yellowbrick.cluster import KElbowVisualizer  
  
#membuat model KMeans dengan random_state=0 untuk memastikan hasil yang sama (konsisten)  
kMeansC = KMeans(random_state=0)  
vis = KElbowVisualizer(kMeansC,k=(2,11)) #untuk menemukan jumlah cluster optimal antara 2 - 10  
#xx = data yang sudah bersih  
vis.fit(xx)  
  
vis.show() #menampilkan plot elbow
```



```
Out[7]: <Axes: title={'center': 'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>
```

- Import SilhouetteVisualizer dan buat grafik untuk menampilkan hasil skor Silhouette Score dari proses clustering dengan nilai k 2-10
- Tampilkan hasil Silhouette Score untuk masing-masing jumlah clustering
- Hasil grafik Elbow Method menunjukkan bahwa jumlah optimal cluster adalah lima (5)

```
In [ ]: #mengimport library yellowbrick untuk visualisasi silhouette score  
from yellowbrick.cluster import SilhouetteVisualizer
```

```
#membuat figure dan subplot dengan ukuran 15x20 untuk menampung 9 plot silhouette score  
fig, ax = plt.subplots(9,1, figsize=(15,20))
```

```
#melakukan Looping untuk menguji model kmeans dengan jumlah cluster dari 2 hingga 10  
for k in np.arange(2,11):  
    kMeansC = KMeans(n_clusters=k, init='k-means++', random_state=0)  
    #Lengkapi Loop ini dengan :  
    #xx = data yang sudah bersih  
    KM_clusters=kMeansC.fit_predict(xx)  
    print("Silhouette Score K-Means for ", k, " clusters :", silhouette_score(xx, KM_clusters))  
  
    #visualisasi silhouette score  
    sil_vis = SilhouetteVisualizer(kMeansC, colors='yellowbrick', ax=ax[k-2])  
    sil_vis.fit(xx) #memasukkan data ke dalam visualizer untuk dihitung dan divisualisasikan
```

```
#menambahkan Label pada setiap plot silhouette  
ax[k-2].set(xlabel="Silhouette Score of each samples", ylabel="Cluster label")
```

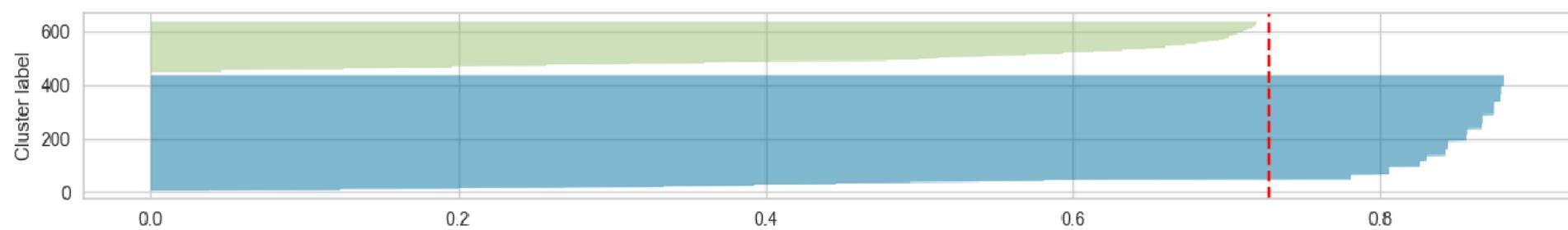
```
Silhouette Score K-Means for 2 clusters : 0.7275517813087928  
Silhouette Score K-Means for 3 clusters : 0.7058100919802311  
Silhouette Score K-Means for 4 clusters : 0.5928895843252326  
Silhouette Score K-Means for 5 clusters : 0.6039773808643728  
Silhouette Score K-Means for 6 clusters : 0.5561748553700107  
Silhouette Score K-Means for 7 clusters : 0.5988247715843502  
Silhouette Score K-Means for 8 clusters : 0.5883936959989691  
Silhouette Score K-Means for 9 clusters : 0.5834209460817851  
Silhouette Score K-Means for 10 clusters : 0.6027794287285431
```

No Copy

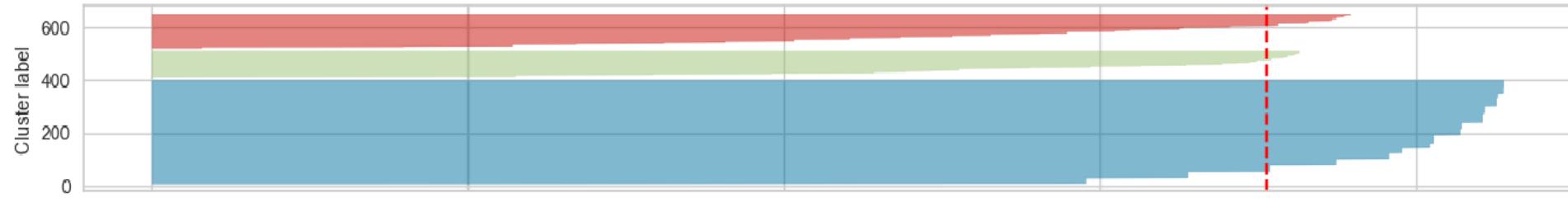


Watermarkly

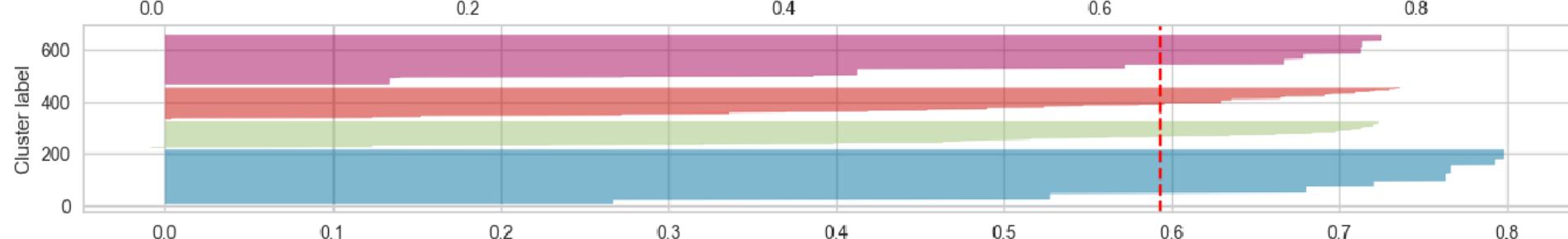
py



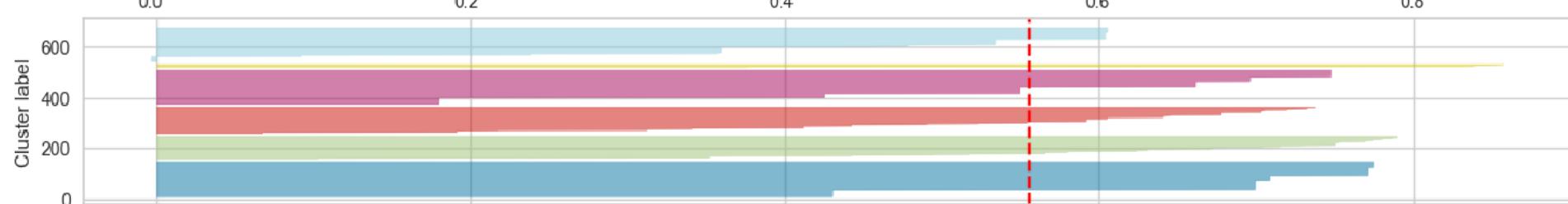
No



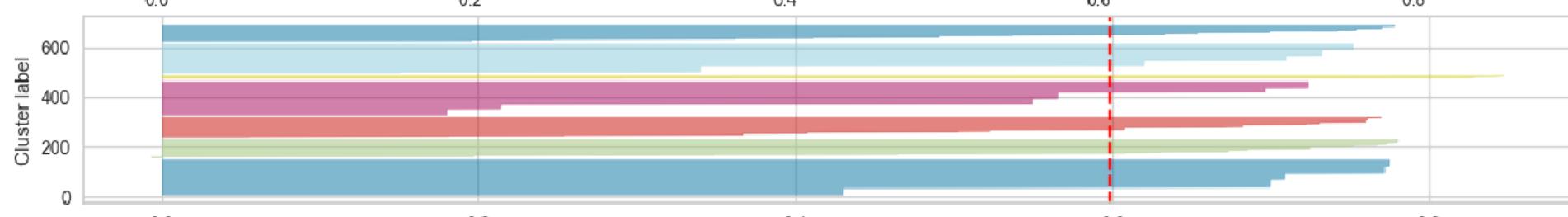
py



No



py



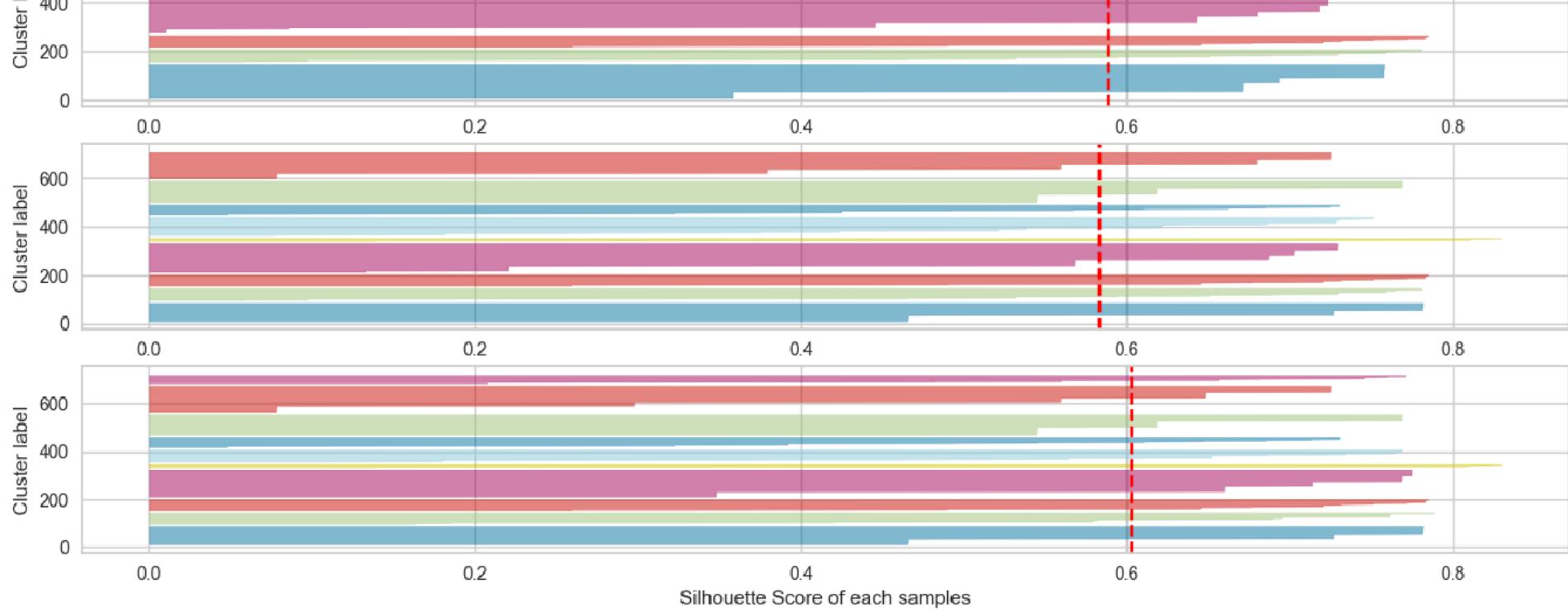
ly

No

No

py

No



- Jika dilihat dari nilai Silhouette Score, jumlah cluster dua (2) memiliki skor paling tinggi
- Namun, jika dilihat dari grafik, jumlah cluster memiliki satu cluster yang tidak memenuhi nilai Silhouette Score
- Sehingga kita akan bereksperimen menggunakan jumlah tiga cluster
- Untuk merepresentasikan pembagian cluster kita dapat menggambarkan dalam grafik tiga dimensi berdasarkan kolom-kolom yang ada pada dataset

```
In [ ]: #melihat nama kolom yang ada dalam DataFrame xx(yang sudah diberishkan)
xx.columns
```

```
Out[9]: Index(['Avg_Credit_Limit', 'Total_Credit_Cards', 'Total_visits_bank',
       'Total_visits_online', 'Total_calls_made'],
       dtype='object')
```

K-Means

- Buat objek K-Means dengan jumlah cluster tiga dan random state == NPM
- Gunakan fungsi fit-predict untuk melakukan clustering terhadap dataset
- Buat fungsi untuk membuat grafik scatter tiga dimensi hasil clustering dengan sumbu-sumbu: Average Credit Limit, Total Credit Cards, dan Total Visit Online
- Tampilkan pula metrik Silhouette Score dari K-Means

```
In [ ]:
#membuat objek kmeans untuk melakukan clustering
#dengan jumlah cluster yang diinginkan (3)
#init='k-means++' untuk membantu menentukan titik awal bagi setiap cluster
#random_state=0 untuk memastikan hasil selalu sama (konsisten) setiap kali code dijalankan
kmeans = KMeans(n_clusters = 3, init='k-means++', random_state = 0)
K_clusters = kmeans.fit_predict(xx) #melakukan clustering pada data "xx" dan menyimpan hasil clustering

#membuat grafik 3D
fig = plt.figure()
ax = plt.axes(projection='3d')

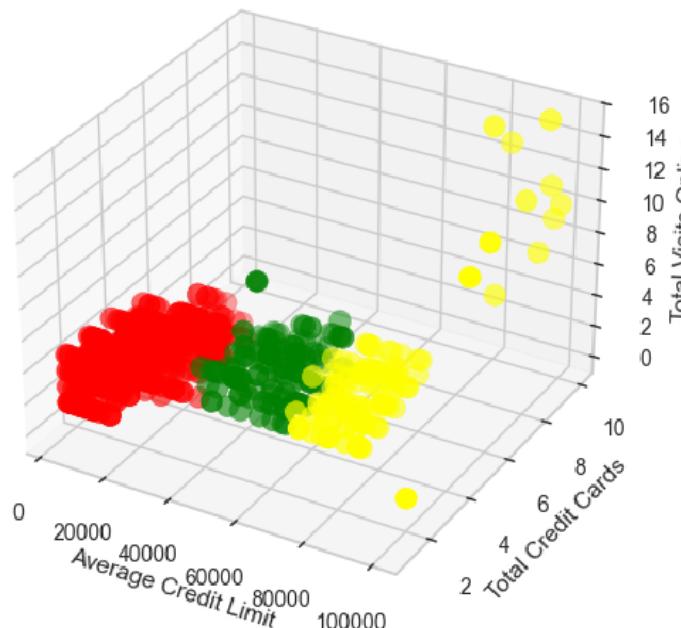
#menentukan warna untuk masing" cluster
#membuat scatter 3D menggunakan warna yang telah di tentukan dengan ukuran titik 100
cluster_colors = {0: 'red', 1: 'yellow', 2: 'green', 3: 'blue', 4:'magenta', 5:'black' ,6:'cyan'}

#xx = data yang sudah bersih
ax.scatter(xx['Avg_Credit_Limit'], xx['Total_Credit_Cards'], xx['Total_visits_online'],
           c=[cluster_colors[i] for i in K_clusters], s=100)

plt.title('Segmentasi Nasabah berdasarkan K-Means')
ax.set_xlabel('Average Credit Limit') #Label sumbu x
ax.set_ylabel('Total Credit Cards') #Label sumbu y
ax.set_zlabel('Total Visits Online') #Label sumbu z
plt.show() #menampilkan grafik

#untuk menghitung dan menampilkan nilai Silhouette Score
print ("Silhouette Score K-Means: %0.3f" % silhouette_score(xx, K_clusters))
```

Segmentasi Nasabah berdasarkan K-Means



Silhouette Score K-Means: 0.706

Agglomerative Clustering

- Buat objek Agglomerative Clustering dengan jumlah cluster 3 dan linkage 'ward'
- Gunakan fungsi fit_predict untuk melakukan clustering terhadap dataset

No Copy
Watermarkly

- Buat fungsi untuk membuat grafik scatter tiga dimensi hasil clustering dengan sumbu-sumbu: Average Credit Limit, Total Credit Cards, dan Total Visit Online
- Tampilkan pula metrik Silhouette Score dari Agglomerative Clustering

In []: #Lakukan instalasi seaborn apabila belum memiliki modul tersebut
#Eksekusi "pip install seaborn" pada anaconda prompt (TANPA "")

```
#mengimpor library seaborn untuk visualisasi data
import seaborn as sns
from matplotlib.colors import ListedColormap #untuk custom warna cluster

AGG = AgglomerativeClustering(n_clusters=3,linkage='ward') #membuat objek AgglomerativeClustering untuk melakukan clustering
#dengan jumlah cluster yang diinginkan (3) dan metode Linkage 'ward'

#xx = data yang sudah bersih
AGG_clusters = AGG.fit_predict(xx) #melakukan clustering pada data "xx" dan menyimpan hasil clustering
cmap = ListedColormap(sns.color_palette("plasma", 256).as_hex()) #membuat colormap khusus menggunakan palet warna "rocket" dari seaborn

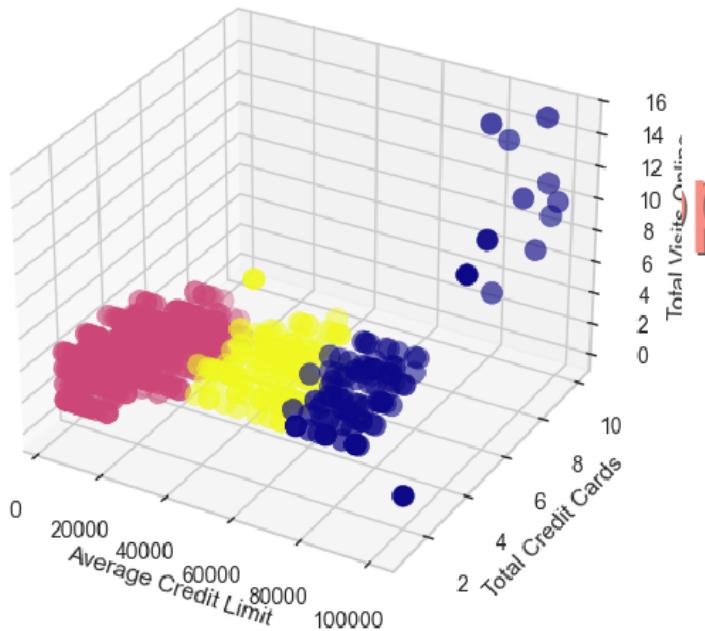
#membuat figure dan axis untuk plot 3D
fig = plt.figure()
ax = plt.axes(projection='3d')

#sumbu x = "Average Credit Limit", sumbu y = "Total Credit Cards", sumbu z = "Total Visits Online"
ax.scatter(xx['Avg_Credit_Limit'], xx['Total_Credit_Cards'], xx['Total_visits_online'],
           c=AGG_clusters, cmap=cmap, s=100)

plt.title('Segmentasi Nasabah berdasarkan Agglomerative')
ax.set_xlabel('Average Credit Limit') #label sumbu x
ax.set_ylabel('Total Credit Cards') #label sumbu y
ax.set_zlabel('Total Visits Online') #label sumbu z
plt.show() #menampilkan plot

#untuk menghitung dan menampilkan nilai Silhouette Score
print("Silhouette Score K-Means: %.3f" % silhouette_score(xx, AGG_clusters))
```

Segmentasi Nasabah berdasarkan Agglomerative



Silhouette Score K-Means: 0.705

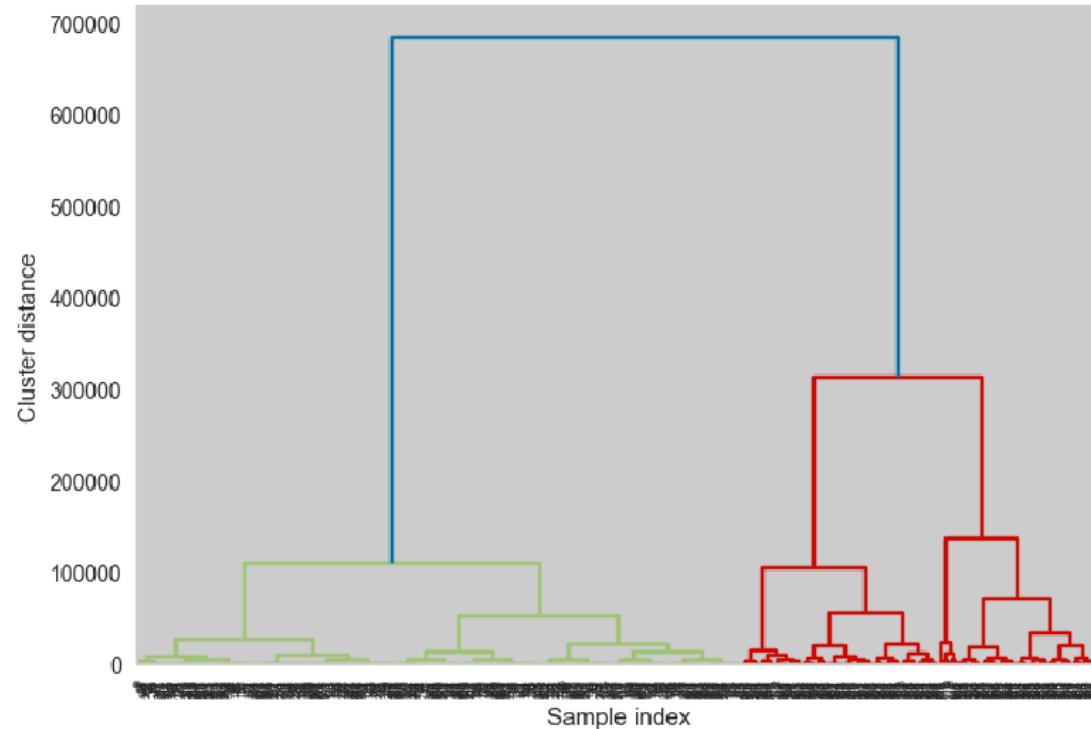


- Buat dendrogram untuk menunjukkan proses clustering dari algoritma Agglomerative Clustering

In []: `from scipy.cluster.hierarchy import dendrogram, linkage`

```
#'ward' digunakan untuk meminimalkan varian dalam tiap cluster
#xx = data yang sudah dibersihkan
linkage_array = linkage(xx, method='ward')
dendrogram(linkage_array) #dendrogram untuk visualisasi

plt.xlabel("Sample index") #Label sumbu x
plt.ylabel("Cluster distance") #Label sumbu y
plt.show() #menampilkan dendrogram
```

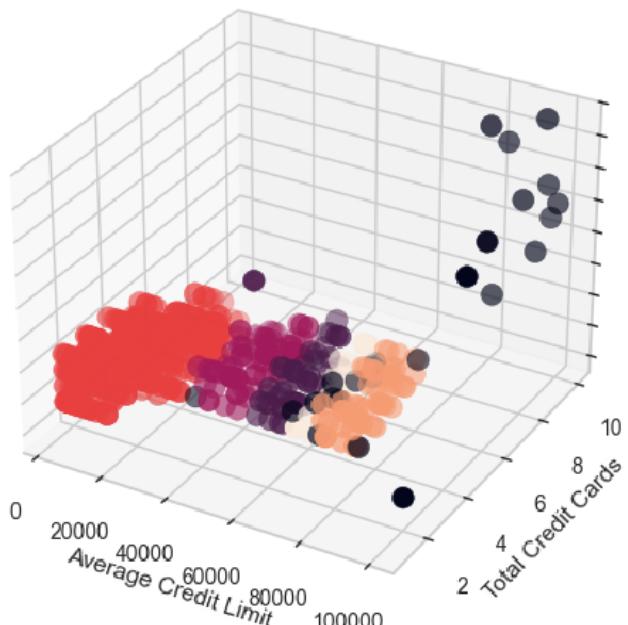


DBSCAN

- Buat objek DBSCAN dengan eksperimen terhadap parameter min_samples dan epsilon
- Gunakan fungsi fit_predict untuk melakukan clustering terhadap dataset
- Buat fungsi untuk membuat grafik scatter tiga dimensi hasil clustering dengan sumbu-sumbu: Average Credit Limit, Total Credit Cards, dan Total Visit Online
- Tampilkan pula metrik Silhouette Score dari DBSCAN

```
In [ ]: #inisialisasi model DBSCAN dengan parameter min_sample dan eps  
DBSC = DBSCAN(min_samples=20,eps=2500)  
#melakukan clustering pada data "xx" dan menyimpan hasil clustering  
  
#xx = data yang sudah dibersihkan  
DBSC_clusters = DBSC.fit_predict(xx)  
  
from matplotlib.colors import ListedColormap #untuk custom warna cluster  
cmap = ListedColormap(sns.color_palette("rocket", 256).as_hex()) #membuat colormap khusus menggunakan palet warna "rocket" dari seaborn  
  
#membuat figure dan axis untuk plot 3D  
fig = plt.figure()  
ax = plt.axes(projection='3d')  
  
#sumbu x = "Average Credit Limit", sumbu y = "Total Credit Cards", sumbu z = "Total Visits Online"  
ax.scatter(xx['Avg_Credit_Limit'], xx['Total_Credit_Cards'], xx['Total_visits_online'],  
          c=DBSC_clusters, cmap=cmap, s=100) #warna berdasarkan cluster hasil DBSCAN dan menggunakan colormap yang telah dibuat tadi  
  
plt.title('Segmentasi Nasabah berdasarkan DBSCAN')  
ax.set_xlabel('Average Credit Limit') #label sumbu x  
ax.set_ylabel('Total Credit Cards') #label sumbu y  
ax.set_zlabel('Total Visits Online') #label sumbu z  
plt.show() #untuk menampilkan plot 3d  
  
#untuk menghitung dan menampilkan nilai Silhouette Score  
print("Silhouette Score K-Means: %0.3f" % silhouette_score(xx, DBSC_clusters))
```

Segmentasi Nasabah berdasarkan DBSCAN



Silhouette Score K-Means: 0.657

- Simpan semua hasil pelatihan menggunakan library Pickle

No Copy
Watermarkly

```
In [14]: import pickle  
  
with open('KMeans_model.pkl','wb') as r:  
    pickle.dump((kmeans),r)  
  
with open('AGG_model.pkl','wb') as r:  
    pickle.dump((AGG),r)  
  
with open('DBSCAN_model.pkl','wb') as r:  
    pickle.dump((DBSC),r)  
  
print("Model berhasil disimpan")
```

Model berhasil disimpan

py

No Copy

No Copy

No Copy

No

Streamlit

- No Copy
- Buat antarmuka Streamlit pada file yang berekstensi Python (.py)
 - Buatlah sesuai dengan code berikut ini

No Copy

No Copy

No Copy

py

No Copy

No Copy

No Copy

No

py

No Copy

No Copy

No Copy

No Copy

py

No Copy

No Copy

No Copy

No



Watermarkly

```
In [ ]: import streamlit as st
import pandas as pd
import pickle
import os
import plotly.express as px #digunakan untuk membuat grafik scatter 3D
import numpy as np
from sklearn.metrics import pairwise_distances #menghitung jarak antar data
import plotly.graph_objects as go #digunakan untuk membuat grafik

#fungsi untuk membuat scatter olot 3D dan menentukan cluster dari titik baru
def scatter(model, model_name, data, new_point, features, color_scale, title):
    clusters = model.fit_predict(data[features]) #untuk memprediksi cluster untuk setiap titik pada data menggunakan model
    data[f'{model_name}_Cluster'] = clusters

    #menentukan cluster untuk titik baru
    if model_name == "KMeans_model":
        #pada k-means, dilakukan prediksi langsung
        new_cluster = model.predict(new_point[features])[0]
    else:
        #pada agglomeratif dan dbscan dilakukan dengan cara menghitung jarak titik terdekat
        distances = pairwise_distances(new_point[features], data[features])
        nearest_index = distances.argmin()
        new_cluster = clusters[nearest_index]

    #membuat grafik 3D menggunakan Plotly Express
    fig = px.scatter_3d(data, x='Avg_Credit_Limit', y='Total_Credit_Cards', z='Total_visits_online',
                        color=f'{model_name}_Cluster', title=title, color_continuous_scale=color_scale)

    #menambahkan titik baru pada grafik
    fig.add_trace(
        go.Scatter3d(
            x=new_point['Avg_Credit_Limit'],
            y=new_point['Total_Credit_Cards'],
            z=new_point['Total_visits_online'],
            mode='markers',
            marker=dict(size=10, color='red'),
            name='New Point'
        )
    )
    return fig, new_cluster

st.set_page_config(
    page_title="XXXXX - Unsupervised Learning", #XXXXX diisi dengan 5 digit NPM
    page_icon="📊",
    layout="wide",
    initial_sidebar_state="expanded",
)

uploaded_file = st.sidebar.file_uploader("Upload your input CSV file", type=["csv"])

if uploaded_file is not None:
    input_data = pd.read_csv(uploaded_file)
    st.markdown("<h1 style='text-align: center;'>Unsupervised Learning - YYYYYY</h1>", unsafe_allow_html=True) #YYYYYY diisi dengan nama panggilan
    st.dataframe(input_data)

    #direktori tempat penyimpanan ketiga model yang telah di dump sebelumnya
    model_directory = r'D:\Kuliah\Materi Tugas\Semester 5\Asdos\modul\4 Unsupervised Learning\Unsupervised Learning'
    model_path = {
        "AGG_model" : os.path.join(model_directory, r'AGG_model.pkl'),
        "KMeans_model" : os.path.join(model_directory, r'KMeans_model.pkl'),
        "DBSCAN_model" : os.path.join(model_directory, r'DBSCAN_model.pkl'),
    }

    #Load model ketiga model ke dalam dictionary
    models = {}
```

```

for model_name, path in model_path.items():
    if os.path.exists(path):
        with open(path, 'rb') as f:
            models[model_name] = pickle.load(f)
    else:
        st.write(f"Model {model_name} tidak ditemukan di path : ", path)

#sidebar untuk memasukkan (menginputkan) nilai untuk titik baru yang akan diprediksi clusternya
avg_CL = st.sidebar.number_input("Average Credit Limit", 0, 100000)
sum_CC = st.sidebar.number_input("Total Credit Cards", 0, 10)
sum_VO = st.sidebar.number_input("Total Visits Online", 0, 16)

if st.sidebar.button("Prediksi !"):
    #fitur yang digunakan untuk memprediksi
    features = ['Avg_Credit_Limit', 'Total_Credit_Cards', 'Total_visits_online']
    #memasukkan data titik baru ke dalam DataFrame
    new_point = pd.DataFrame({
        'Avg_Credit_Limit': [avg_CL],
        'Total_Credit_Cards': [sum_CC],
        'Total_visits_online': [sum_VO]
    })

    #model clustering yang digunakan dan warna grafik scatternya
    #warna dapat diubah sesuai keinginan
    cluster_method = [
        ("KMeans_model", models["KMeans_model"], "KMeans Clustering", px.colors.sequential.Cividis),
        ("AGG_model", models["AGG_model"], "Agglomerative Clustering", px.colors.sequential.Mint),
        ("DBSCAN_model", models["DBSCAN_model"], "DBSCAN Clustering", px.colors.sequential.Plasma)
    ]

    #membuat tiga kolom untuk menampilkan grafik
    col1, col2, col3 = st.columns(3)
    cols = [col1, col2, col3]

    for i, (model_name, model, title, color_scale) in enumerate(cluster_method):
        fig, new_cluster = scatter(model, model_name, input_data, new_point, features, color_scale, title)
        with cols[i]:
            st.plotly_chart(fig)
            st.markdown(f"<p style='text-align: center;'>Titik Dari Data yang baru masuk ke dalam cluster : {new_cluster}</p>", unsafe_allow_html=True)

```