MACHINE LEARNING

UTS Case Method

Nama: Muhammad irsyad Dimas Abdillah

NIM: 2341720088

Prodi: D4-Teknik Informatika

Jl. Soekarno Hatta No.9, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur 65141

Phone : (0341) 404424, 404425

E-email : Polinema.ac.id

TUGAS STUDI KASUS PEMBELAJARAN MESIN

Clustering dan Approximate Nearest Neighbor (ANN)

LINK Colab: https://colab.research.google.com/drive/1aQX1PWWwvddA-1f2gZygTn9AfX8r33vw?usp=sharing

1. Preprocessing Data:
   a. Cek jenis data pada setiap kolom

      Berdasarkan output gambar di bawah, dapat dilihat tipe-tipe data yang ada dalam dataset. Output yang ditampilkan dalam dataset hanya mencakup fitur numerik—khususnya yang bertipe int64 dan float64. Jadi dataset tidak memiliki data fitur yang bertipe kategorikal (lebih tepatnya jika melalui penjelasan dataset, fitur kategorikal telah di encoding sebelum dataset dibagikan).

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  float64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  float64
 13  BILL_AMT2                   30000 non-null  float64
 14  BILL_AMT3                   30000 non-null  float64
 15  BILL_AMT4                   30000 non-null  float64
 16  BILL_AMT5                   30000 non-null  float64
 17  BILL_AMT6                   30000 non-null  float64
 18  PAY_AMT1                    30000 non-null  float64
 19  PAY_AMT2                    30000 non-null  float64
 20  PAY_AMT3                    30000 non-null  float64
 21  PAY_AMT4                    30000 non-null  float64
 22  PAY_AMT5                    30000 non-null  float64
 23  PAY_AMT6                    30000 non-null  float64
 24  default.payment.next.month  30000 non-null  int64
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
None
```

b. Cek missing values berupa NaN, inf, dan null

Karena tidak terdapat missing values, proses imputasi tidak perlu dilakukan. Hal ini bisa dibuktikan dengan menbuat kode untuk memeriksa keberadaan missing values (NaN, Inf, dan null) atau dengan membuka halaman dataset dan mengecek pada halaman column yang sudah menyediakan ringkasan detailnya.

```
Tidak ada nilai hilang.
Tidak ada nilai inf / -inf.
Tidak ada nilai hilang (isnull).
```

c. Melakukan feature engineering

Dalam tahap ini melakukan feature engineering untuk memperkaya informasi dataset dengan informasi turunan baru yang merepresentasikan profil finansial dan perilaku pembayaran nasabah. Pemilihan fitur dalam kode ini divalidasi secara konseptual oleh penelitian [1] dalam artikel berjudul:

"The comparisons of data mining techniques for the predictive accuracy of credit card clients default"

Penelitian tersebut menegaskan bahwa kapasitas kredit dan perilaku pembayaran merupakan faktor utama yang membedakan karakteristik antar nasabah, sehingga relevan digunakan sebagai dasar pembentukan segmen (cluster).

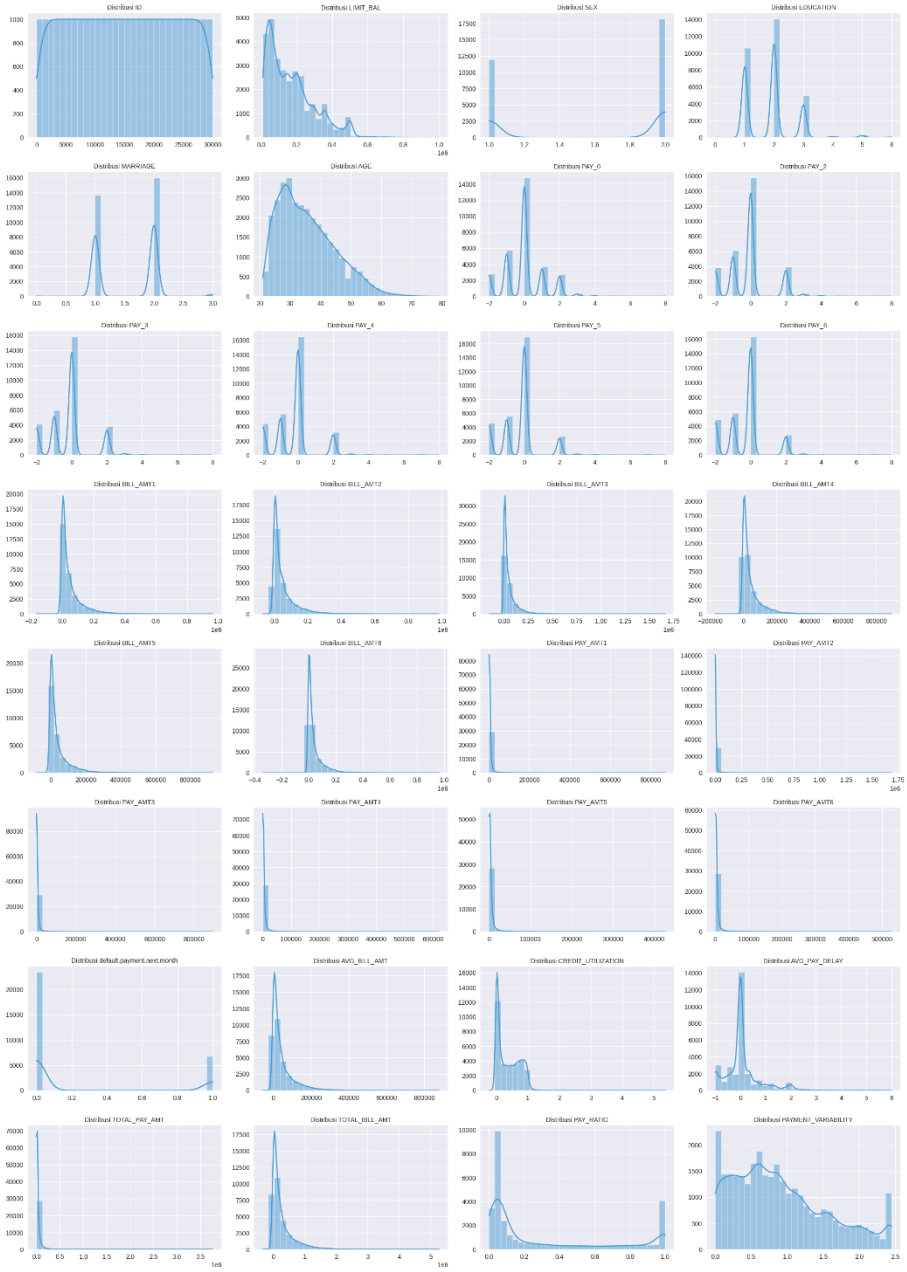
```
Feature engineering selesai!

Preview hasil:
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	PAY_AMT5	PAY_AMT6	default.payment.next.month	AVG_BILL_
0	1	20000.0	2	2	1	24	2	2	-1	-1	...	0.0	0.0	1	1284.00
1	2	120000.0	2	2	2	26	-1	2	0	0	...	0.0	2000.0	1	2846.16
2	3	90000.0	2	2	2	34	0	0	0	0	...	1000.0	5000.0	0	16942.16
3	4	50000.0	2	2	1	37	0	0	0	0	...	1069.0	1000.0	0	38555.66
4	5	50000.0	1	2	1	57	-1	0	-1	0	...	689.0	679.0	0	18223.16
5	6	50000.0	1	1	2	37	0	0	0	0	...	1000.0	800.0	0	39685.66
6	7	500000.0	1	1	2	29	0	0	0	0	...	13750.0	13770.0	0	454099.16
7	8	100000.0	2	2	2	23	0	-1	-1	0	...	1687.0	1542.0	0	2247.66
8	9	140000.0	2	3	1	28	0	0	2	0	...	1000.0	1000.0	0	10868.66
9	10	20000.0	1	3	2	35	-2	-2	-2	-2	...	1122.0	0.0	0	4486.50

10 rows × 32 columns

d. Mengecek Distrubusi data

Pengecekan distribusi tiap kolom dataset digunakan untuk menentukan keputusan akan menggunakan scaling apa yang cocok untuk dataset yang digunakan.



e. Pemilihan Fitur yang digunakan berdasarkan artikel rujukan

```python
# === DAFTAR FITUR TERPILIH ===
selected_features = [
    # Fitur Asli Penting
    "LIMIT_BAL",              # Kapasitas kredit
    "PAY_0",                  # Status pembayaran terakhir
    "AGE",                    # Umur nasabah
    "SEX",                    # Jenis kelamin
    "EDUCATION",              # Pendidikan
    "MARRIAGE",               # Status pernikahan

    # Semua Fitur Hasil Feature Engineering
    "AVG_BILL_AMT",           # Rata-rata tagihan (proxy BALANCE)
    "AVG_PAY_AMT",            # Rata-rata pembayaran (proxy PURCHASES
    "BALANCE_PURCHASE_RATIO", # Rasio antara BALANCE dan PURCHASES
    "CREDIT_UTILIZATION",     # Rasio pemakaian kredit terhadap limi
    "AVG_PAY_DELAY",          # Keterlambatan rata-rata pembayaran
    "TOTAL_PAY_AMT",          # Total pembayaran 6 bulan terakhir
    "TOTAL_BILL_AMT",         # Total tagihan 6 bulan terakhir
    "PAY_RATIO",              # Rasio bayar terhadap total tagihan
    "PAYMENT_VARIABILITY"     # Fluktuasi pembayaran (stabilitas per
]
```

f. Scaling data

Pada tahap ini, untuk memperbaiki distribusi data numerik yang memiliki skew tinggi atau varians besar, serta agar algoritma berbasis jarak bekerja lebih baik, saya menerapkan Yeo–Johnson power-transform terlebih dahulu untuk menyeimbangkan distribusi, kemudian StandardScaler untuk menyamakan skala antar fitur [2].

```
=== Preprocessing Selesai ===
Shape akhir: (30000, 15)
Total missing values: 0

=== Statistik Kolom Setelah Transformasi ===
```

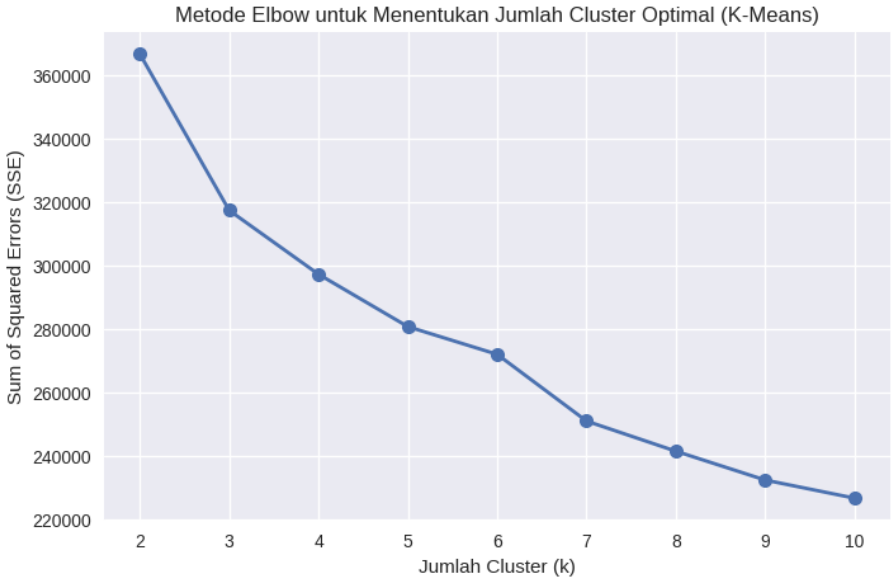
	min	max	mean	std
LIMIT_BAL	-2.098927	3.065845	1.212660e-16	1.000017
PAY_0	-2.005396	5.303146	1.326346e-17	1.000017
AGE	-2.155527	2.813880	8.374930e-16	1.000017
SEX	-1.234323	0.810161	-4.395891e-16	1.000017
EDUCATION	-4.435645	3.080580	-1.023182e-16	1.000017
MARRIAGE	-3.302841	2.603498	4.035883e-16	1.000017
AVG_BILL_AMT	-37.759105	9.732235	-3.031649e-17	1.000017
AVG_PAY_AMT	-2.436105	7.406569	-7.579123e-18	1.000017
BALANCE_PURCHASE_RATIO	-67.476609	76.026264	9.710751e-18	1.000017
CREDIT_UTILIZATION	-2.786437	3.439213	-1.061077e-16	1.000017
AVG_PAY_DELAY	-2.181219	3.994203	1.894781e-17	1.000017
TOTAL_PAY_AMT	-2.467220	7.619392	6.063298e-17	1.000017
TOTAL_BILL_AMT	-35.628004	10.191323	-7.768601e-17	1.000017
PAY_RATIO	-1.226889	1.520924	2.385055e-16	1.000017
PAYMENT_VARIABILITY	-1.773086	1.872095	3.713770e-16	1.000017

2. Clustering

A. K-MEANS

1. Elbow Method

Setelah scaling dan encoding, untuk menentukan jumlah kluster optimal pada metode K-Means, digunakan Elbow Method. Metode ini mengevaluasi nilai inertia dari berbagai jumlah kluster K K , lalu memilih titik di mana penurunan inertia mulai melambat (titik siku grafik). Titik “siku” (elbow point) pada grafik menunjukkan jumlah kluster yang optimal, karena setelah titik tersebut peningkatan jumlah kluster tidak memberikan perbaikan signifikan pada hasil klastering.



=== Nilai SSE Tiap K ===

k = 2: SSE = 366635.13

k = 3: SSE = 317373.46

k = 4: SSE = 297251.09

k = 5: SSE = 280761.26

k = 6: SSE = 272029.35

k = 7: SSE = 251025.20

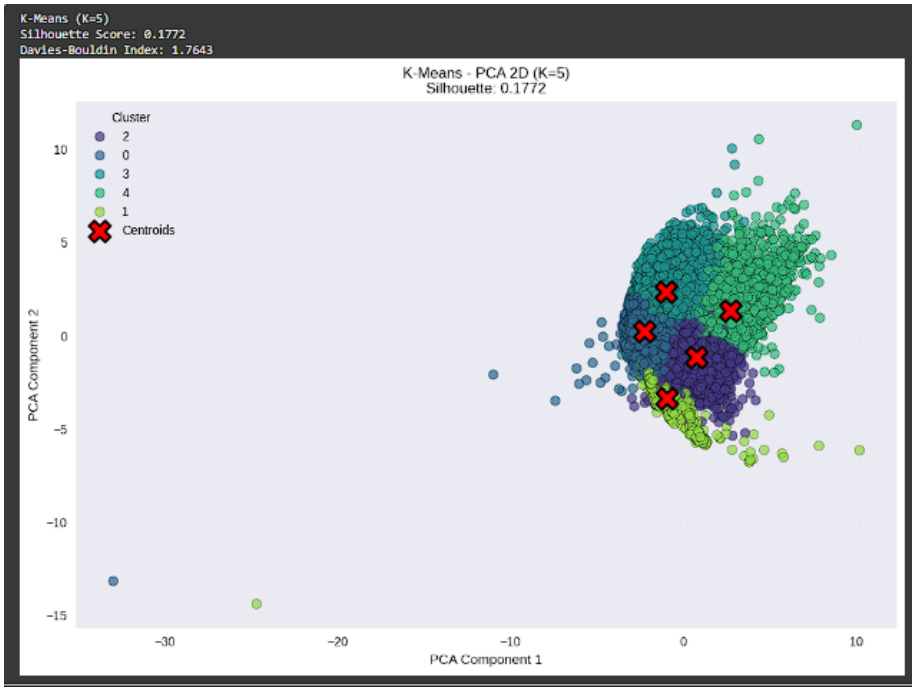
k = 8: SSE = 241531.34

k = 9: SSE = 232446.83

k = 10: SSE = 226749.57

2. Klasterisasi dan Visualisasi

Dari hasil output dibawah ini dapat diketahui dengan k = 5, didapatkan Sihouette Score: 0.1769, yang menunjukkan bahwa pemisahan cluster masih belum sempurna, namun struktur yang terbentuk sudah cukup untuk mencerminkan pola data. Sementara Davis-Bouldin Index: 1.7696, mengindikasikan bahwa antar cluster masih memiliki kemiripan.



3. Analisis kontribusi PCA 2D

Hasil PCA 2D menunjukkan bahwa dua komponen utama menangkap 45.34% variasi data. PCA 1 dipengaruhi oleh fitur terkait penggunaan dan pembayaran kredit, sedangkan PCA 2 lebih mencerminkan kapasitas dan volume transaksi. Ini membantu menyederhanakan struktur data untuk analisis lanjutan.

```
================================================
INTERPRETASI PCA 2D
================================================

Variance Explained:
PCA 1: 24.68%
PCA 2: 20.66%
Total: 45.34%

Kontribusi Fitur pada Komponen PCA 2D:
------------------------------------------------

Top 5 fitur untuk PCA 1:
```

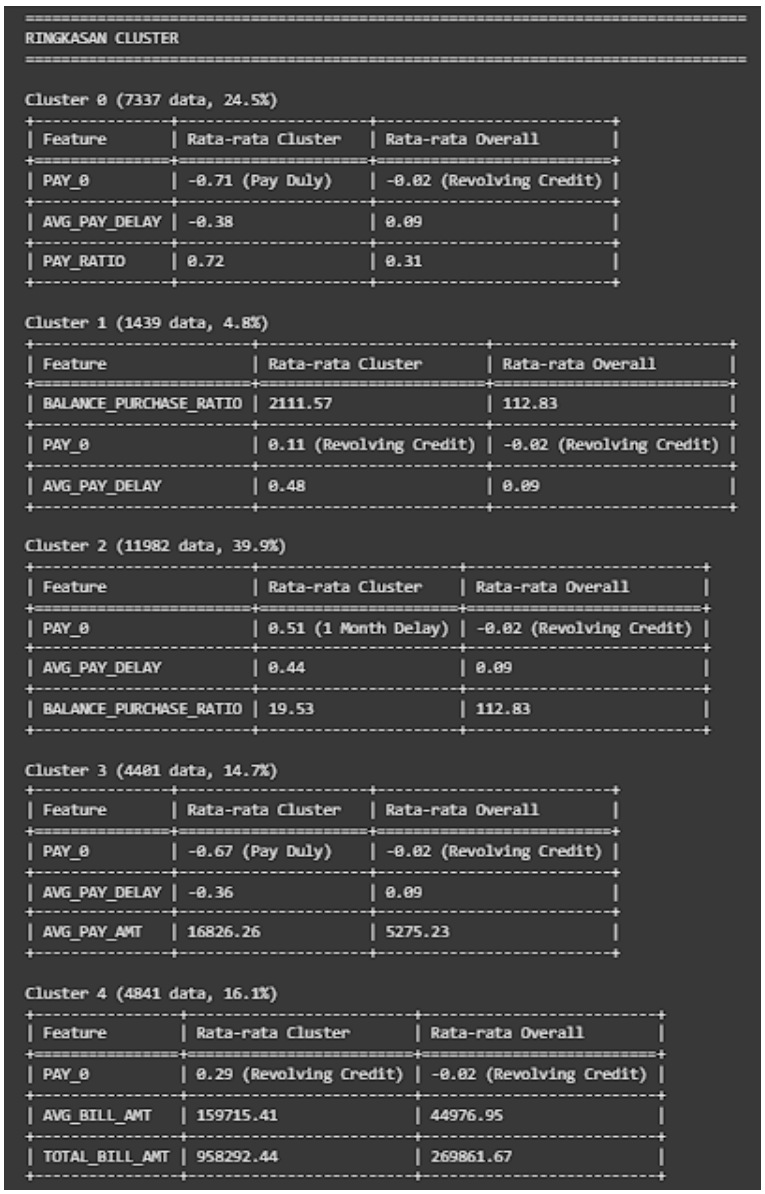
	PCA 1
CREDIT_UTILIZATION	0.449015
AVG_BILL_AMT	0.435335
TOTAL_BILL_AMT	0.433326
PAY_RATIO	-0.381251
AVG_PAY_DELAY	0.307318

```
Top 5 fitur untuk PCA 2:
```

	PCA 2
TOTAL_PAY_AMT	0.492465
AVG_PAY_AMT	0.492313
LIMIT_BAL	0.374127
PAY_RATIO	0.287203
AVG_PAY_DELAY	-0.272332

4. Tambahan Fitur untuk menjelaskan tiap cluster

Fitur tambahan pada tabel Ringkasan Cluster menggambarkan segmentasi perilaku finansial pengguna dengan karakteristik berbeda: Cluster 0 (24,5%) berisi pengguna disiplin yang membayar tepat waktu, Cluster 1 (4,8%) memiliki rasio pembelian terhadap saldo sangat tinggi, Cluster 2 (39,9%) didominasi keterlambatan sekitar satu bulan, Cluster 3 (14,7%) menunjukkan keterlambatan lebih lama dengan nilai pembayaran besar, dan Cluster 4 (16,1%) menampilkan pengguna dengan tagihan serta pembayaran jauh di atas rata-rata. Segmentasi ini memberikan gambaran jelas untuk merancang strategi kredit, promosi, maupun manajemen risiko yang lebih tepat sasaran.

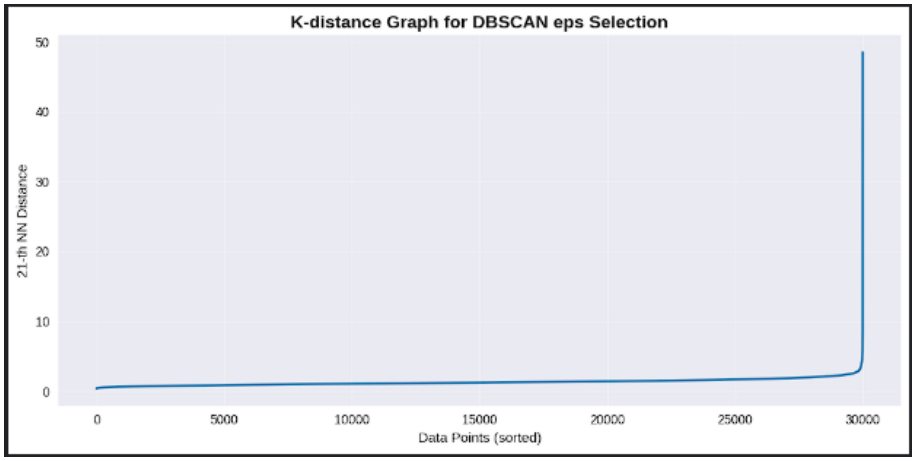


```
================================================================================
RINGKASAN CLUSTER
================================================================================

Cluster 0 (7337 data, 24.5%)
+--------------+--------------------+--------------------------+
| Feature      | Rata-rata Cluster  | Rata-rata Overall        |
+==============+====================+==========================+
| PAY_0        | -0.71 (Pay Duly)   | -0.02 (Revolving Credit) |
+--------------+--------------------+--------------------------+
| AVG_PAY_DELAY | -0.38             | 0.09                     |
+--------------+--------------------+--------------------------+
| PAY_RATIO    | 0.72               | 0.31                     |
+--------------+--------------------+--------------------------+

Cluster 1 (1439 data, 4.8%)
+-----------------------+-------------------------+--------------------------+
| Feature               | Rata-rata Cluster       | Rata-rata Overall        |
+=======================+=========================+==========================+
| BALANCE_PURCHASE_RATIO | 2111.57                | 112.83                   |
+-----------------------+-------------------------+--------------------------+
| PAY_0                 | 0.11 (Revolving Credit) | -0.02 (Revolving Credit) |
+-----------------------+-------------------------+--------------------------+
| AVG_PAY_DELAY         | 0.48                    | 0.09                     |
+-----------------------+-------------------------+--------------------------+

Cluster 2 (11982 data, 39.9%)
+-----------------------+---------------------+--------------------------+
| Feature               | Rata-rata Cluster   | Rata-rata Overall        |
+=======================+=====================+==========================+
| PAY_0                 | 0.51 (1 Month Delay) | -0.02 (Revolving Credit) |
+-----------------------+---------------------+--------------------------+
| AVG_PAY_DELAY         | 0.44                | 0.09                     |
+-----------------------+---------------------+--------------------------+
| BALANCE_PURCHASE_RATIO | 19.53              | 112.83                   |
+-----------------------+---------------------+--------------------------+

Cluster 3 (4401 data, 14.7%)
+--------------+--------------------+--------------------------+
| Feature      | Rata-rata Cluster  | Rata-rata Overall        |
+==============+====================+==========================+
| PAY_0        | -0.67 (Pay Duly)   | -0.02 (Revolving Credit) |
+--------------+--------------------+--------------------------+
| AVG_PAY_DELAY | -0.36             | 0.09                     |
+--------------+--------------------+--------------------------+
| AVG_PAY_AMT  | 16826.26           | 5275.23                  |
+--------------+--------------------+--------------------------+

Cluster 4 (4841 data, 16.1%)
+---------------+-------------------------+--------------------------+
| Feature       | Rata-rata Cluster       | Rata-rata Overall        |
+===============+=========================+==========================+
| PAY_0         | 0.29 (Revolving Credit) | -0.02 (Revolving Credit) |
+---------------+-------------------------+--------------------------+
| AVG_BILL_AMT  | 159715.41               | 44976.95                 |
+---------------+-------------------------+--------------------------+
| TOTAL_BILL_AMT | 958292.44              | 269861.67                |
+---------------+-------------------------+--------------------------+
```

B. DBSCAN

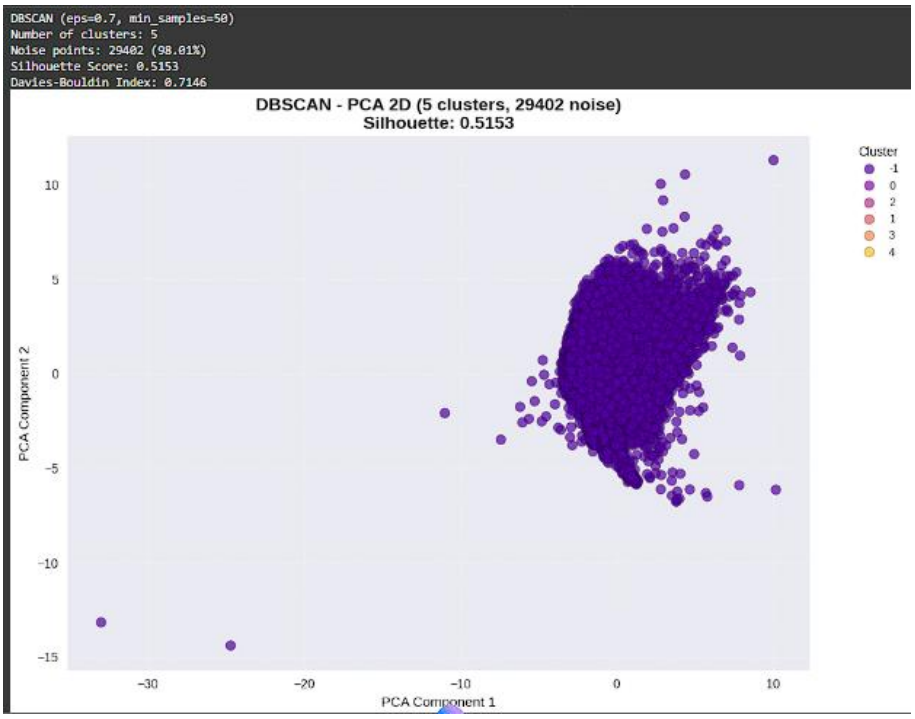
1. Mengukur K-Distance graph

   Grafik K-distance menunjukkan titik elbow yang menandai transisi dari area padat ke area jarang dalam data. Titik ini digunakan untuk menentukan nilai eps yang optimal dalam algoritma DBSCAN.



2. Melakukan Klasterisasi dan Visualisasi PCA

   Clustering dengan DBSCAN (eps=0.7, min_samples=50) menghasilkan 5 cluster dan 29.402 data (98,01%) sebagai noise. Silhouette Score 0.5153 dan DB Index 0.7146 menunjukkan pemisahan antar cluster cukup baik. Visualisasi PCA 2D memperlihatkan mayoritas data terkonsentrasi di satu area, menandakan pola perilaku keuangan yang seragam, dengan sedikit outlier yang berbeda signifikan.



3. Approximate Nearest Neighbor (ANN)
   a. Menyiapkan data untuk Annoy

```
Jumlah data: 30000
Jumlah fitur: 15
Shape data: (30000, 15)
```

   b. Membuat Annoy Index

      Dalam tahap ini Pembangunan index Annoy menggunakan 10 trees dengan metrik Euclidean memerlukan waktu 0.258s. dan ini menunjukkan efisiensi dalam menyiapkan struktur pencarian tetangga terdekat untuk data berskala besar atau berdimensi tinggi.

```
Building Annoy index...
- Trees: 10
- Metric: euclidean
✓ Build selesai dalam 0.258 detik
```

   c. Test Query Exact vs Annoy

      Kali ini adalah perbandingan hasil pencaria tetangga terdekat (NN) menunjukkan bahwa metode Annoy mampu menemukan 5 tetangga yang identik, jadi akurasinya 100%. Perbedaan kedua metode ada pada kecepatan: Exact NN butuh sekitar 3.8273ms, sedangkan Annoy NN hanya 0,1473 ms. Artinya Annoy jauh lebih cepat, daripada Exact NN.

```
Query index: 19794
Mencari 5 tetangga terdekat...

Exact NN
Index: [19794 11633  7604 23163 19125]
Jarak: [0.         0.57609624 0.65243477 0.7856165  0.82574594]
Waktu: 3.8273 ms

Annoy NN
Index: [19794, 11633, 7604, 23163, 19125]
Jarak: [0.0, 0.57609623670578, 0.652434766292572, 0.7856165170669556, 0.8257459482084351]
Waktu: 0.1473 ms

Perbandingan
Speedup: 25.98x
Akurasi: 5/5 sama
```

   d. Eksperimen Annoy dengan berbagai nilai trees

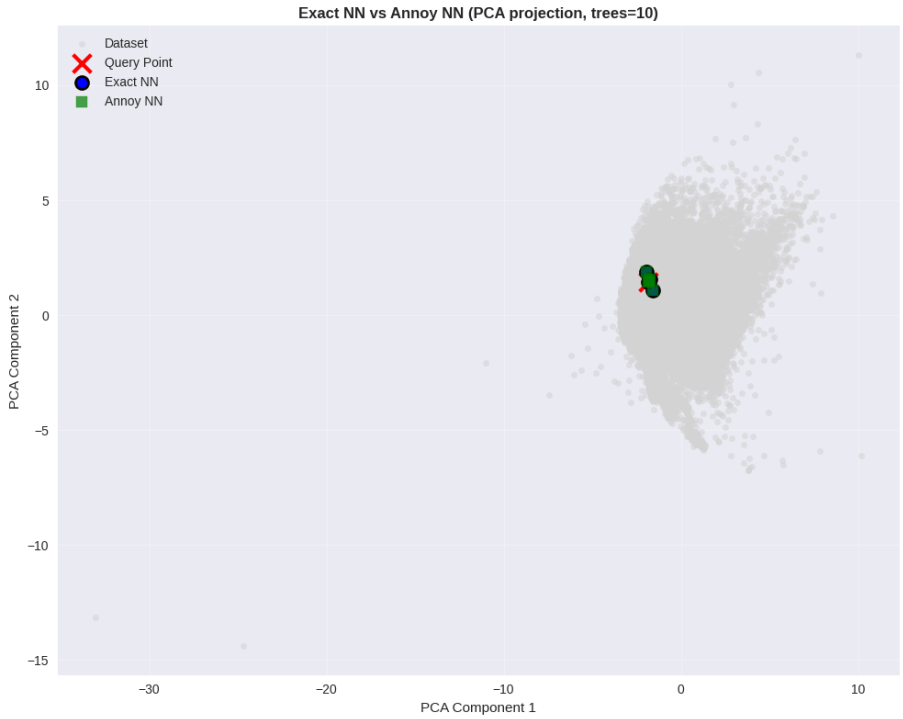
      Dalam percobaan kali ini, dapat diketahui bahwa semakin banyak jumlah trees maka membutuhkan waktu membangun model yang semakin lama. Waktu query juga lebih lama. Dari akurasi model trees 3 memiliki akurasi hanya 40%, sedangkan mulai 8 trees keatas memiliki akurasi 100%. Jadi penambahan trees meningkatkan

akurasi, namun harus dibayarkan dengan waktu membangun model dan query yang jadi lebih berat dan lama.

```
Eksperimen dengan berbagai jumlah trees...

 Trees  Build Time (s)  Query Time (ms)  Accuracy (%)
     3          0.0622           0.0317          60.0
     8          0.1567           0.0505          80.0
    10          0.1705           0.0491         100.0
    20          0.5082           0.0918         100.0
    50          1.7859           0.2248         100.0
```

e. Visualisasi Annoy dengan PCA 2D

Titik merah adalah query point yang dicari tetangga terdekatnya. Metode Exact NN (biru) dan Annoy NN (hijau) sama-sama menemukan tetangga yang tepat di posisi yang sama. Artinya, meskipun Annoy bekerja lebih cepat, hasilnya tetap akurat seperti metode exact. Visualisasi ini menunjukkan bahwa pendekatan approximate bisa menghemat waktu tanpa mengorbankan kualitas hasil.



f. Mencari 5 NN untuk semua data

Beberapa titik query dipilih secara acak, lalu dicari 5 tetangga terdekat untuk masing-masing. Hasilnya menampilkan index tetangga beserta nilai jaraknya, yang menunjukkan seberapa dekat tiap data dengan query. Proses ini berjalan sangat cepat, hanya rata-rata 0,0167 ms per query, sehingga efisien meski diterapkan ke seluruh dataset.

```
Mencari 5 nearest neighbors untuk semua data...
✓ Selesai dalam 0.502 detik
Rata-rata 0.0167 ms per query

Contoh neighbors untuk 5 data pertama:
Data 0: [0, 9093, 9339, 19506, 15732]...
Data 1: [1, 12877, 383, 15187, 1327]...
Data 2: [2, 20054, 1684, 2207, 17245]...
Data 3: [3, 17486, 13595, 19068, 22720]...
Data 4: [4, 14198, 11237, 10299, 26987]...
```

g. Perbandingan berbagai distance metrics

Eksperimen ini membandingkan tiga distance metrics: Euclidean, Angular, dan Manhattan. Ketiganya menghasilkan tetangga terdekat yang mirip, namun dengan urutan berbeda. Waktu komputasi juga hampir sama, sekitar 0,05–0,08 ms per query. Artinya, pemilihan metrik akan memengaruhi hasil tetangga yang didapat, meski dari sisi kecepatan tidak jauh berbeda.

```
Membandingkan berbagai distance metrics...

euclidean : 0.0813 ms | [19794, 11633, 7604, 23163, 19125]
angular   : 0.0701 ms | [19794, 11633, 7604, 17565, 23163]
manhattan : 0.0546 ms | [19794, 7604, 11633, 23163, 5661]
```

4. TULISKAN KESIMPULAN SINGKAT:

a. Perbedaan hasil KMeans dan DBSCAN, mana yang lebih baik diantara kedua model ini dan jelaskan jawaban anda

Jawab: K-Means dengan Silhouette 0.1772 dan DBI 1.7643 menunjukkan pemisahan cluster yang lemah, meski mampu membentuk struktur dasar. Sebaliknya, DBSCAN dengan Silhouette 0.5153 dan DBI 0.7146 menghasilkan kualitas cluster lebih baik, namun menganggap 98% data sebagai noise. Meski demikian, DBSCAN lebih unggul karena mampu menangani outlier sesuai prinsip robust statistics, di mana metode konvensional sering gagal saat data mengandung outlier (Hinkley, 1975), terdapat dalam artikel [2].

b. Nilai metrik terbaik (Silhouette, DBI).

Jawab: Pada evaluasi ini, K-Means (K=5) memiliki Silhouette Score 0.1769 dan DBI Score 1.7696, sedangkan DBSCAN (eps=0.7, min_samples=50) mencapai Silhouette Score 0.5153 dan DBI Score 0.7146. Hasil tersebut menunjukkan bahwa DBSCAN menghasilkan pemisahan cluster yang lebih baik.

c. Hasil query Annoy: apakah tetangga yang ditemukan termasuk dalam cluster yang sama? Jelaskan jawaban anda.

   Jawab: Tidak selalu. Titik terdekat yang ditemukan oleh metode Annoy belum tentu berada dalam satu cluster yang sama menurut DBSCAN, karena keduanya mengartikan "kedekatan data" secara berbeda. Annoy melihat jarak dengan kesamaan numerik, sedangkan DBSCAN menilai kepadatan area dengan titik dianggap satu cluster jika berada di wilayah dengan cukup banyak tetangga dalam radius tertentu.

   Hal ini sejalan dengan konsep statistika robust [2]:

   "For the subset of size m we order all observations by closeness to the fitted model; the residuals determine closeness."

   Artinya, bukan hanya jarak yang penting, tapi juga kecocokan terhadap pola utama. Jadi, titik yang dekat menurut Annoy bisa saja dianggap noise oleh DBSCAN jika tidak termasuk area padat. Kesimpulannya, Annoy menjawab "siapa yang paling dekat," sedangkan DBSCAN menjawab "siapa yang termasuk kelompok."

REFERENSI

[1] I. C. Yeh and C. hui Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Syst. Appl.*, vol. 36, no. 2 PART 1, pp. 2473–2480, 2009, doi: 10.1016/j.eswa.2007.12.020.

[2] M. Riani, A. C. Atkinson, and A. Corbellini, "Automatic robust Box–Cox and extended Yeo–Johnson transformations in regression," *Stat. Methods Appl.*, vol. 32, no. 1, pp. 75–102, Mar. 2023, doi: 10.1007/s10260-022-00640-7.