

# LAPORAN TUGAS BESAR I : CLUSTERING

Disusun oleh Ryan Abdurohman (1301191171)

S-1 Informatika, IF-43-10

## A. Formulasi Masalah

Diberikan dataset mengenai data pelanggan di *dealer*, kita diminta mengelompokkan pelanggan tanpa memperhatikan label kelas apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak. Untuk mengelompokkan data-data tersebut, kita harus memahami datanya, lalu menentukan berapa banyak kluster hingga mengukurnya dengan metrik performansi tertentu. Untuk pengerjaan, semuanya menggunakan bahasa pemrograman python [1].

## B. Eksplorasi dan Persiapan Data

Sebelum melakukan eksplorasi dan persiapan, dataset diunduh terlebih dahulu lalu diproses ke tipe Data Frame menggunakan pandas [2].

```
[ ] !gdown --id 1QeNUIIr6VAMQORrGjK0BmZUCB4qBA10w  
[ ] df = pd.read_csv("kendaraan_train.csv")
```

**1. Melihat cuplikan pada data**, dilakukan untuk mengetahui gambaran kasar pada data, serta untuk menentukan langkah-langkah selanjutnya. Misal, kita bisa melihat bahwa ada data yang bernilai NaN, sehingga kita harus melakukan strategi tertentu untuk mengatasi hal ini.

```
[ ] df.head()
```

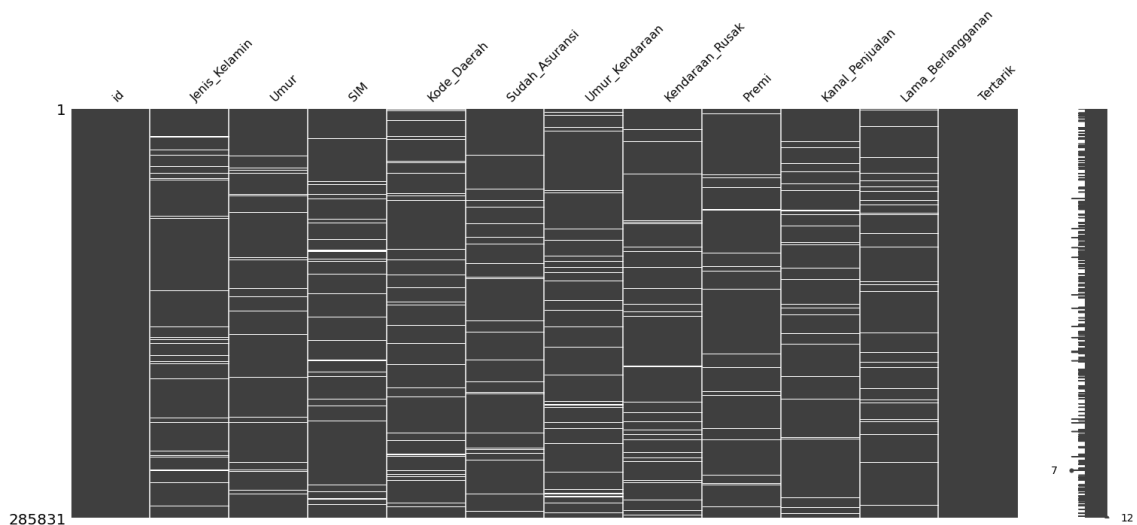
	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	1	Wanita	30.0	1.0	33.0	1.0	< 1 Tahun	Tidak	28029.0	152.0	97.0	0
1	2	Pria	48.0	1.0	39.0	0.0	> 2 Tahun	Pernah	25800.0	29.0	158.0	0
2	3	NaN	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	32733.0	160.0	119.0	0
3	4	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.0	0
4	5	Pria	50.0	1.0	35.0	0.0	> 2 Tahun	NaN	34857.0	88.0	194.0	0

**2. Melihat informasi pada data**, dilakukan untuk mengetahui secara spesifik karakteristik dari data. Dapat kita lihat bahwa ada 285831 baris dengan 12 kolom, namun data yang Non-Null tidak setara dengan jumlah baris data, sehingga bisa kita dapatkan intuisi bahwa banyak data yang bernilai NaN atau kosong.

```
[ ] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285831 entries, 0 to 285830
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   id                     285831 non-null  int64  
1   Jenis_Kelamin          271391 non-null  object  
2   Umur                   271617 non-null  float64 
3   SIM                    271427 non-null  float64 
4   Kode_Daerah            271525 non-null  float64 
5   Sudah_Asuransi         271602 non-null  float64 
6   Umur_Kendaraan         271556 non-null  object  
7   Kendaraan_Rusak        271643 non-null  object  
8   Premi                  271262 non-null  float64 
9   Kanal_Penjualan        271532 non-null  float64 
10  Lama_Berlangganan      271839 non-null  float64 
11  Tertarik               285831 non-null  int64  
dtypes: float64(7), int64(2), object(3)
memory usage: 26.2+ MB
```

**3. Melakukan visualisasi terhadap *missing value* (NaN),** hal ini dilakukan untuk mengetahui persebaran dari data yang kosong. Bisa kita lihat bahwa persebarannya merata, tidak berkumpul di satu kolom. Sehingga kita perlu mempertimbangkan jumlah dari *missing value* itu sendiri di setiap kolomnya. Visualisasi ini menggunakan Pustaka missingno [3].



**4. Melihat jumlah *missing value* pada setiap kolomnya,** hal ini dilakukan untuk mendapatkan informasi yang akurat terhadap jumlah data yang kosong pada setiap kolomnya. Bisa kita lihat bahwa rata-rata ada data yang kosong sebanyak 14000-an di setiap kolom, namun pada langkah sebelumnya data kosong ini pun tersebar secara acak di setiap kolomnya, sehingga setiap baris bisa saja ada satu *feature* yang kosong atau lebih.

```
df.isna().sum()
```

id	0
Jenis_Kelamin	14440
Umur	14214
SIM	14404
Kode_Daerah	14306
Sudah_Asuransi	14229
Umur_Kendaraan	14275
Kendaraan_Rusak	14188
Premi	14569
Kanal_Penjualan	14299
Lama_Berlangganan	13992
Tertarik	0

dtype: int64

**5. Drop data pada kolom label dan data yang memiliki sedikit kategori**, hal ini disebabkan kita akan menggunakan k-Means clustering yang pada dasarnya menggunakan jarak. Data kategori yang dihapus adalah data yang kategorinya sedikit. Misal, ada data jenis kelamin yang hanya ada dua kategori laki-laki dan perempuan (0 dan 1). Nah, ketika data nanti dinormalisasi dengan rentang 0-1, maka data jenis kelamin ini akan mengganggu penghitungan jarak dengan fitur yang lain. Sedangkan data label didrop karena memang kita tidak menggunakan label pada proses *clustering*.

```
df.nunique()
```

id	285831
Jenis_Kelamin	2
Umur	66
SIM	2
Kode_Daerah	53
Sudah_Asuransi	2
Umur_Kendaraan	3
Kendaraan_Rusak	2
Premi	45114
Kanal_Penjualan	151
Lama_Berlangganan	290
Tertarik	2

dtype: int64

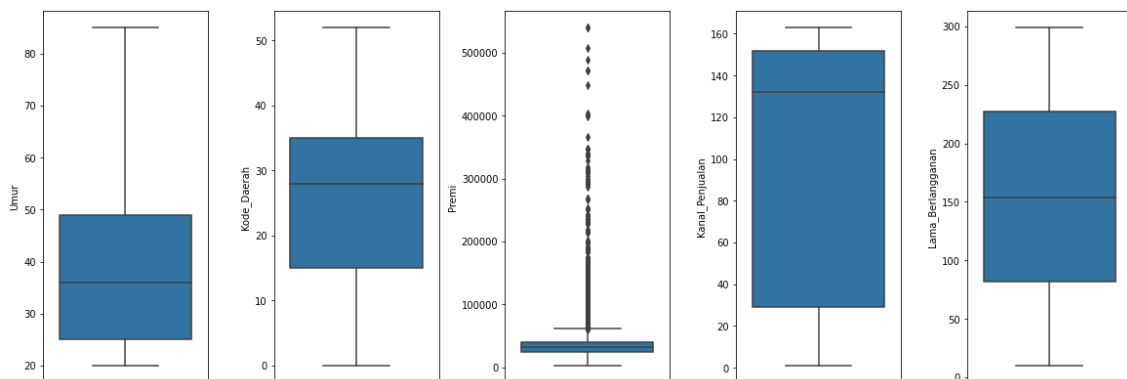
Kita bisa melihat data id itu sebagai *primary key* sehingga data ini tidak memiliki pengaruh pada dataset secara keseluruhan. Namun, id tidak akan didrop pada saat ini karena id bisa digunakan untuk membantu memudahkan indeks klasterisasi. Lalu, kita bisa lihat juga ada beberapa kolom yang memiliki sedikit kategori, maka kita bisa drop kolom-kolom tersebut. Data hasil drop kolom: [https://docs.google.com/spreadsheets/d/1A\\_ucQCu0OfwLJAVQ8Jom5lxeEe8aW6UB/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true](https://docs.google.com/spreadsheets/d/1A_ucQCu0OfwLJAVQ8Jom5lxeEe8aW6UB/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true)

```
[ ] numdf = df.drop(['Jenis_Kelamin', 'SIM', 'Sudah_Asuransi', 'Umur_Kendaraan', 'Kendaraan_Rusak', 'Tertarik'], axis=1)
```

```
[ ] numdf.head()
```

	id	Umur	Kode_Daerah	Premi	Kanal_Penjualan	Lama_Berlangganan
0	1	30.0	33.0	28029.0	152.0	97.0
1	2	48.0	39.0	25800.0	29.0	158.0
2	3	21.0	46.0	32733.0	160.0	119.0
3	4	58.0	48.0	2630.0	124.0	63.0
4	5	50.0	35.0	34857.0	88.0	194.0

**6. Deteksi anomali/outlier pada data**, pertama kita lakukan visualisasi menggunakan boxplot dengan Pustaka seaborn [4] , lalu melihat statistika deskripif pada setiap kolom. Hal ini dilakukan untuk melihat distribusi suatu data dan melihat secara kasar apakah ada outlier.



```
[ ] numdf.describe()
```

	id	Umur	Kode_Daerah	Premi	Kanal_Penjualan	Lama_Berlangganan
count	285831.000000	271617.000000	271525.000000	271262.000000	271532.000000	271839.000000
mean	142916.000000	38.844336	26.405410	30536.683472	112.021567	154.286302
std	82512.446734	15.522487	13.252714	17155.000770	54.202457	83.694910
min	1.000000	20.000000	0.000000	2630.000000	1.000000	10.000000
25%	71458.500000	25.000000	15.000000	24398.000000	29.000000	82.000000
50%	142916.000000	36.000000	28.000000	31646.000000	132.000000	154.000000
75%	214373.500000	49.000000	35.000000	39377.750000	152.000000	227.000000
max	285831.000000	85.000000	52.000000	540165.000000	163.000000	299.000000

Pada data yang kita olah, **tidak terdapat outlier**. Walaupun data premi pada boxplot menunjukkan banyak titik-titik di luar plot, namun kita pahami bersama bahwa premi merupakan jumlah premi yang harus dibayarkan per tahun. Tentu saja bisa terjadi banyak data yang distribusinya beda, karena memang bisa saja setiap orang memiliki banyak produk asuransi yang berbeda, bahkan perbedaannya bisa signifikan. Oleh karena itu, bisa dinyatakan bahwa karakteristik data premi itu sendiri di dunia nyata merupakan sesuatu yang bukan anomali/outlier.

**7. Handling missing value,** bagian ini penting karena data yang hilang akan menyebabkan proses klasterisasi tidak bisa dilakukan. Oleh karena itu, saya melakukan tiga eksperimen terkait penanganan pada data yang hilang. Eksperimen yang dilakukan akan dijelaskan lebih lengkap pada subbab eksperimen di laporan ini. Pada eksperimen pertama saat ini, kita akan melakukan drop pada missing value. Kita bisa lihat bahwa data yang ada awalnya terdapat 285831, setelah didrop itu berkurang menjadi 221199. Ini artinya data yang akan diproses adalah sekitar 77% dari data awal.

```
[ ] dfna = numdf.dropna()
    id = np.array(dfna['id'])
    dfna = dfna.drop(['id'], axis=1)

dfna.shape

(221199, 5)
```

Data                      hasil                      drop                      missing                      value:  
[https://docs.google.com/spreadsheets/d/1GTvPjOCzrh\\_m70foHLiFzWk0GcpcC0bt/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true](https://docs.google.com/spreadsheets/d/1GTvPjOCzrh_m70foHLiFzWk0GcpcC0bt/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true)

**8. Normalisasi data,** bagian ini digunakan untuk melakukan scaling terhadap semua data pada rentang nol sampai satu. Hal ini dilakukan karena model yang akan kita pakai adalah k-Means clustering yang memakai perhitungan jarak, sehingga data yang akan diproses haruslah memiliki rentang yang sama agar mudah dalam perhitungan dan dalam visualisasi. Adapun teknik normalisasi yang digunakan menggunakan Min-Max scaler pada pustaka scikit-learn [5].

```
[ ] dfna[dfna.columns] = MinMaxScaler().fit_transform(dfna)

[ ] dfna.head()
```

	Umur	Kode_Daerah	Premi	Kanal_Penjualan	Lama_Berlangganan
0	0.153846	0.634615	0.047251	0.932099	0.301038
1	0.430769	0.750000	0.043104	0.172840	0.512111
2	0.015385	0.884615	0.056002	0.981481	0.377163
3	0.584615	0.923077	0.000000	0.759259	0.183391
4	0.461538	0.673077	0.059953	0.537037	0.636678

Data                                      hasil                                      normalisasi:  
[https://docs.google.com/spreadsheets/d/10\\_6yc0CaezRzxvPRBgNCV3-Aty1ll-4e/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true](https://docs.google.com/spreadsheets/d/10_6yc0CaezRzxvPRBgNCV3-Aty1ll-4e/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true)

**9. Reduksi dimensi dengan Principal Component Analysis**, dalam machine learning, terdapat istilah “distance concentration” pada data yang memiliki feature tinggi/banyak. Hal ini mengacu pada pengukuran jarak antara dua titik yang akan cenderung konvergen di tempat yang sama seiring dengan banyaknya data [6]. Maka, akan dilakukan feature extraction pada data dengan mereduksinya menjadi dua dimensi. Keuntungan lainnya adalah data yang diproses akan mudah dalam hal visualisasi dan menurunkan kompleksitas ruang dan waktu.



Data direduksi dengan teknik PCA menggunakan library scikit-learn [5]. Secara sederhana PCA akan melakukan proyeksi data yang dimensinya tinggi ke dalam data yang berdimensi rendah. Data hasil proyeksi tersebut memiliki titik pusat (0,0) sehingga ada kemungkinan datanya negative. Oleh karena itu dilakukan normalisasi Kembali terhadap data hasil reduksi. Berikut datanya: <https://docs.google.com/spreadsheets/d/106F2VVChL4oU521xsB4Iqq-MwbR1YCWb/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true>. Adapun hasil PCA nya menghasilkan dua principal component dengan cumulative variance explained sebanyak 69.73%. Artinya, angka tersebut adalah ukuran seberapa representatif principal component yang dibentuk dari data sebelumnya.

## C. Pemodelan

Model yang digunakan pada data hanyalah k-Means Clustering. Eksperimen yang dilakukan akan dijelaskan pada subbab eksperimen, namun yang pasti tidak ada eksperimen pada model lainnya.

### 1. k-Means Clustering

Algoritma k-Means Clustering diprogram dengan class KMeans. Implementasi k-Means yang dibuat bisa digunakan untuk k berapapun dan juga bisa melakukan inisialisasi centroid secara manual atau tidak random. Dalam implementasinya, langkah-langkah dalam k-Means Clustering yang diterapkan adalah

1. Inisialisasi centroid
2. Menghitung jarak setiap titik data ke masing-masing centroid
3. Data akan dimasukkan ke cluster terdekat
4. Memperbarui titik centroid berdasarkan rata-rata dari masing-masing cluster
5. Mengecek konvergensi dengan melihat ada tidaknya perubahan posisi centroid sebelumnya. Jika ada perubahan, maka ulang ke tahap 2

```
def run(self):
    # Inisialisasi centroid
    if self.use_init_random:
        self.cent = np.array([self.init_centroid(i) for i in range(self.k)])

    cek = False
    while not cek:
        # hitung jarak setiap titik ke masing-masing centroid
        jarak_cent = np.array([])
        jarak = []
        for c in self.cent:
            # perhitungan menggunakan bantuan library scipy untuk euclidean
            # hal ini dilakukan untuk mendapatkan benefit dari vectorization (supaya lebih cepat)
            jarak.append(cdist(self.data, np.array([c]), 'euclidean').reshape(-1,))
        jarak_cent = np.array(jarak)

        # klaster data ke centroid terdekat
        self.klaster = self.assign_cluster(jarak_cent)

        # update centroid dengan rata-rata masing-masing klaster
        old = self.cent
        self.update_centroid()

        # cek konvergensi
        cek = self.is_converged(old, self.cent)
```

Selain implementasi utama dari k-Means Clustering, method-method pembantu juga dibuat, seperti method plot cluster, method konversi hasil cluster ke dataframe, juga method pendukung lainnya.



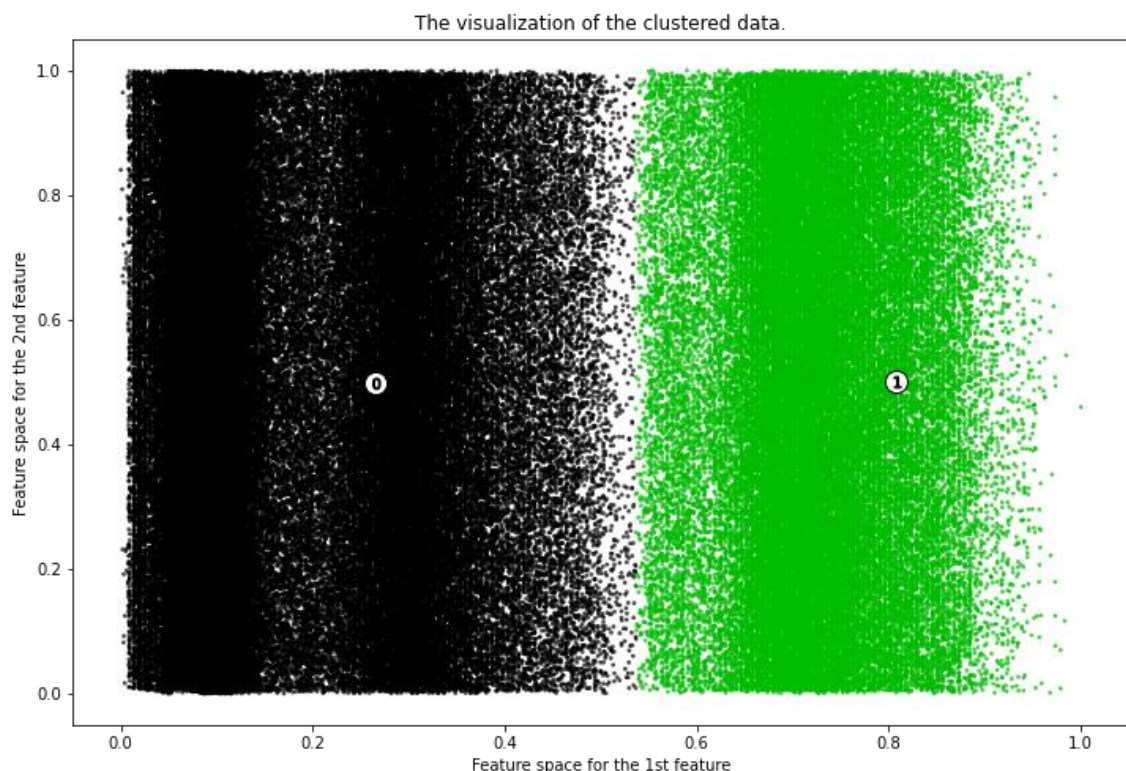
## 2. Setup Hyperparameter

Beberapa nilai hyperparameter yang digunakan pada tugas kali ini adalah

1. Nilai  $k = 2$ . Nilai ini diperoleh dari hasil visualisasi data yang sudah direduksi (step 9 pada B) bahwa kita bisa langsung melihat bahwa kluster yang sebaiknya dihasilkan adalah dua kluster. Sehingga tidak dilakukan pengujian nilai  $k$  yang lebih lanjut.
2. Nilai centroid awal yang dipakai adalah centroid nol (0.5, 0.5) dan centroid satu (0.99, 0.5). Nilai ini diperoleh dari observasi hasil visualisasi reduksi dimensi. Observasi yang dimaksud adalah melakukan estimasi dari hasil visualisasi tadi (step 9 pada B), estimasi yang dilakukan hanya berdasarkan pengamatan secara kasat mata, lalu memilih titik yang sekiranya cocok. Selain itu, dengan inisialisasi centroid yang ditentukan menyebabkan hasil perbandingan eksperimen yang lebih adil dan hasil yang bisa diproduksi ulang.

## 3. Hasil

Berdasarkan hasil klasterisasi, centroid nol berpindah ke (0.26502203, 0.49914071) dan centroid satu ke (0.80676677, 0.50042053). Berikut hasil visualisasi dari klasterisasi yang dilakukan. Dapat kita analisis secara pengamatan visual bahwa terdapat dua kluster dari data yang bisa dibedakan dan dipisahkan. Untuk data hasilnya, dapat dilihat di





## D. Evaluasi

Metode evaluasi yang digunakan adalah silhouette analysis untuk mengukur kualitas dari hasil klusterisasi.

### 1. Silhouette Analysis

Untuk setiap  $i \in X$  di mana  $X$  adalah himpunan titik setiap sampel data, maka dihitung  $a(i)$  yaitu rata-rata jarak dari setiap titik  $i$  ke semua titik yang ada dalam klaster. Lalu  $b(i)$  yaitu rata-rata jarak terhadap titik  $i$  ke semua tetangga klaster terdekat. Idealnya,  $a(i)$  akan kurang dari  $b(i)$ . Akan dihitung pula silhouette coefficient  $[s(i)]$  yaitu:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

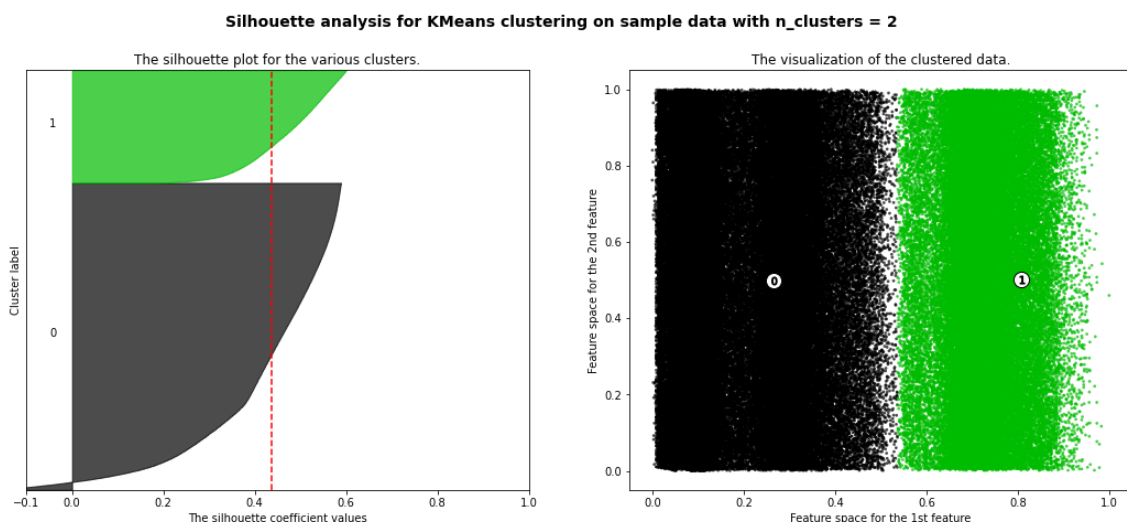
Untuk menghitungnya, kita cukup menggunakan bantuan Pustaka scikit-learn dengan memanggil `sklearn.metrics.silhouette_samples`. Data hasil pengukuran silhouette coefficient untuk setiap titik dilampirkan di [https://docs.google.com/spreadsheets/d/1PwHcCMbvKjKi1gH1c\\_NuxoQ8fo4HazoT/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true](https://docs.google.com/spreadsheets/d/1PwHcCMbvKjKi1gH1c_NuxoQ8fo4HazoT/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true)

Kelebihan silhouette coefficient yaitu skornya yang terikat dari -1 sampai 1, di mana lebih tinggi lebih bagus, namun nol mengindikasikan klaster yang tumpang tindih. Skor ini mencerminkan semakin tinggi skornya, maka klaster semakin rapat dan terpisah dengan baik, tentu ini sesuai dengan konsep pada klusterisasi itu sendiri.

Adapun rerata dari silhouette coefficient untuk setiap sampelnya pada eksperimen ini adalah 0.4361822845384469.

### 2. Analisis dan Evaluasi

Untuk mempermudah analisis, maka dilakukan visualisasi terhadap setiap titik silhouette coefficient pada masing-masing klaster.



Berdasarkan hasil plotting, terdapat nilai koefisien silhouette yang negatif. Secara umum, jika ada nilai koefisien silhouette yang negatif, maka terdapat outlier atau ada data yang salah kluster. Namun, pada kasus kali ini, nilai koefisien silhouette yang negatif cenderung sedikit, pun masalah yang diangkat adalah klasterisasi dari data pelanggan. Tujuan dari klasterisasi ini hanyalah untuk mendapatkan insight, tidak begitu harus mendapatkan presisi dan akurasi yang tinggi, misal pada kasus klasterisasi penyakit yang menyangkut hidup-mati seseorang. Kita juga bisa melihat pertimbangan pada plotting klasterisasi di sebelah kanan yang secara kasat mata sudah terklasterisasi dengan baik. Sehingga bisa dikatakan model yang kita bangun sudah efektif dalam mengklasterisasi data pelanggan. Namun, ada satu masalah. Kita diminta untuk mengkluster semua data pelanggan yang ada, akan tetapi sebelumnya kita melakukan drop pada missing value yang seyogianya dapat menghilangkan data itu sendiri. Sehingga diperlukan strategi lanjut untuk mengisi missing value supaya semua data bisa terklasterisasi dengan baik. Berikut hasil klasterisasi pada eksperimen ke satu kali ini: <https://docs.google.com/spreadsheets/d/1DAmGYdaXZy0eSlGgqvtRobxkrWTU8oYb/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true>

## E. Eksperimen

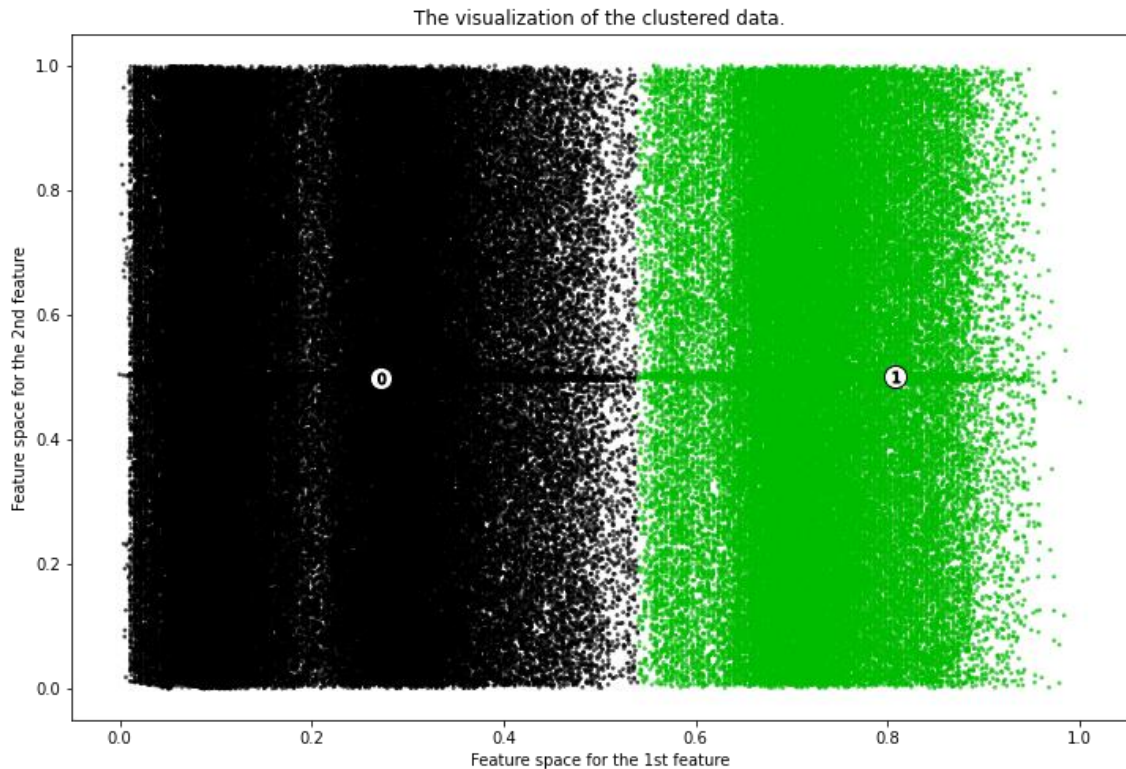
Adapun eksperimen yang dilakukan melibatkan eksperimen pada proses preprocessing data, yaitu penanganan pada missing value.

### 1. Mengisi Missing Value dengan Rerata

Data yang diisi missing value nya adalah data pada langkah ke-5 bagian B. Berikut tahapannya:

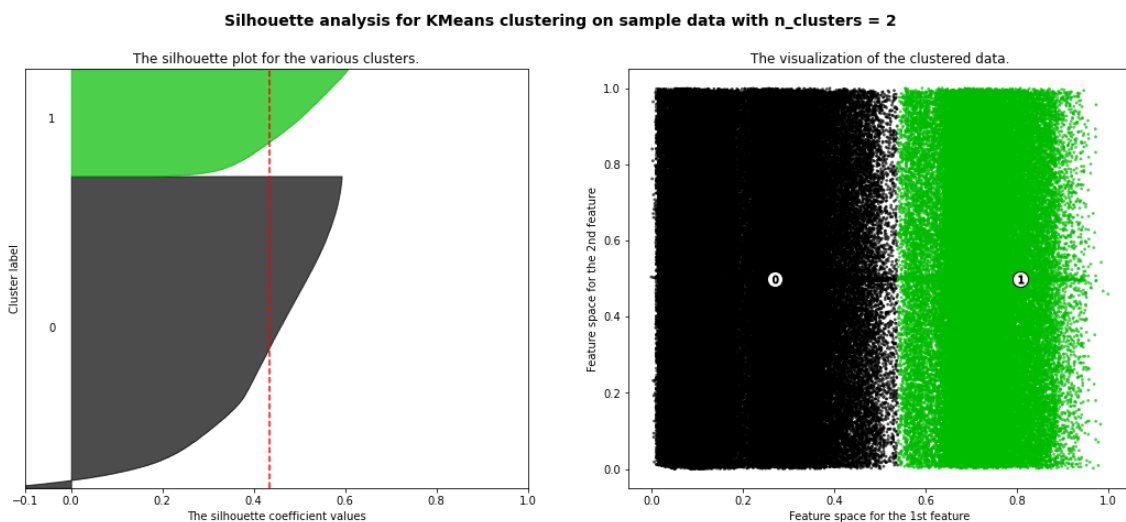
- Isi semua missing value dengan rerata tiap kolomnya. Hasil: <https://docs.google.com/spreadsheets/d/1pmIxX5CJagsO7fbjvd5Xg8JCchoyX8j1/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true>
- Normalisasi data. [https://docs.google.com/spreadsheets/d/1gO7fVHwEig3Ig-9AEVFezXsU\\_D77CvtK/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true](https://docs.google.com/spreadsheets/d/1gO7fVHwEig3Ig-9AEVFezXsU_D77CvtK/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true)
- Reduksi dimensi dengan PCA dan normalisasi ulang. <https://docs.google.com/spreadsheets/d/1DphtXZcwBDb4TShc3YtyRbZqYo5CCpc9/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true>

Secara umum, tahapan dan justifikasinya sama dengan yang sudah dijelaskan pada bagian B. Hasil dari reduksi dimensi menghasilkan cumulative variance explained sebanyak 69.13%. Setelah dilakukan proses pemodelan yang sama dengan bagian C, berikut hasil dari klasterisasi. <https://docs.google.com/spreadsheets/d/1e2VySE-B0EuvMhElyufSYoVVUxheUurm/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true>



Lalu, kita akan evaluasi kualitas klasterisasi dengan tahapan yang sama pada bagian D. Berikut hasil silhouette coefficient untuk setiap sampel. Adapun rerata silhouette coefficientnya adalah 0.4343952418495413.

[https://docs.google.com/spreadsheets/d/1SeANxx3Vjx\\_fNMle\\_cAN5NoytEi\\_O2DO/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true](https://docs.google.com/spreadsheets/d/1SeANxx3Vjx_fNMle_cAN5NoytEi_O2DO/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true)



## 2. Mengisi Missing Value dengan kNNImputer

### Cara Kerja

**Pertama**, data yang didalamnya terdapat missing value (dalam hal ini NaN) akan dilakukan perhitungan jarak nan euclidean ke semua titik. Secara sederhana, nan euclidean didefinisikan sebagai

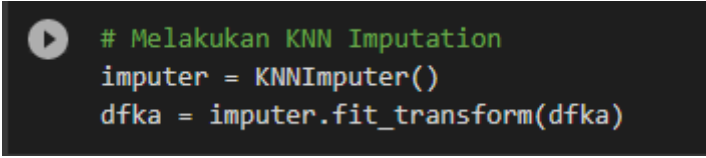
$$dist(x, y) = \sqrt{\frac{a}{b}}(euclid(x, y))$$

$x$  merupakan titik acuan yang akan kita isi missing valuenya, sedangkan  $y$  adalah titik lain yang akan kita hitung jaraknya. Prinsip nan euclidean adalah memberikan bobot pada perhitungan jarak. Bobot ini didefinisikan di atas dengan  $a$  adalah total/Panjang dari titik koordinat, sedangkan  $b$  adalah banyaknya koordinat yang bukan NaN pada titik  $x$ .

**Kedua**, setelah dihitung seluruh nan euclidean dari titik  $x$  ke semua titik yang ada, maka akan diurutkan dari jarak yang paling dekat. Di sinilah teknik k-Nearest Neighbor (kNN) diterapkan. Data yang sudah terurut akan diambil sebanyak  $n$  neighbors teratas. Pada kasus ini  $n$  yang dipilih adalah sebanyak 5.

Data yang diisi missing value nya adalah data pada langkah ke-5 bagian B, namun data tersebut dinormalisasi terlebih dahulu. Lalu selanjutnya, berikut tahapannya:

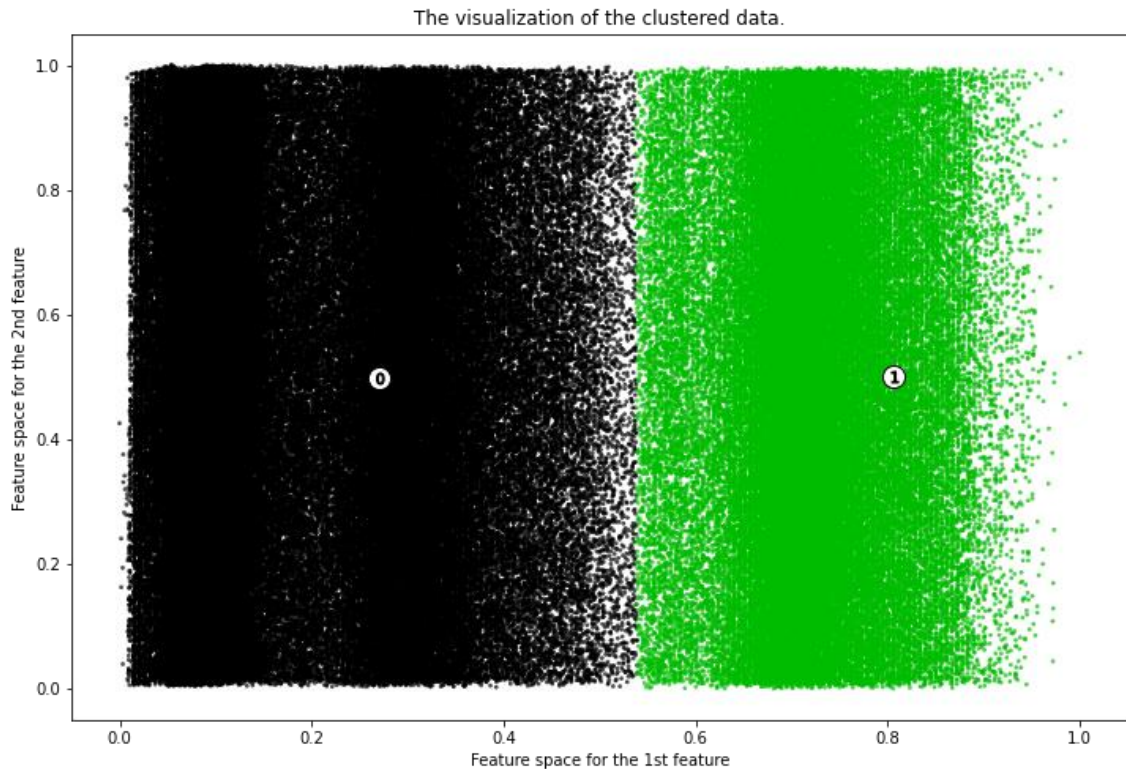
- Isi semua missing value dengan hasil kNNImputation dengan bantuan Pustaka scikit-learn. Hasil: <https://docs.google.com/spreadsheets/d/18-nGsiJX7SrekazgP5ui0pBEdz6X1iNq/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true>



```
# Melakukan KNN Imputation
imputer = KNNImputer()
dfka = imputer.fit_transform(dfka)
```

- Reduksi dimensi dengan PCA dan normalisasi ulang. [https://docs.google.com/spreadsheets/d/1W08pI4YtoVVn06OddtmL8Pzgyy\\_QUb8W/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true](https://docs.google.com/spreadsheets/d/1W08pI4YtoVVn06OddtmL8Pzgyy_QUb8W/edit?usp=sharing&oid=100206322009227195695&rtpof=true&sd=true)

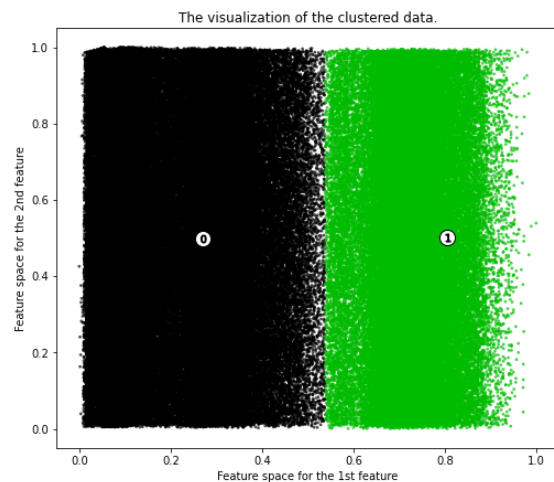
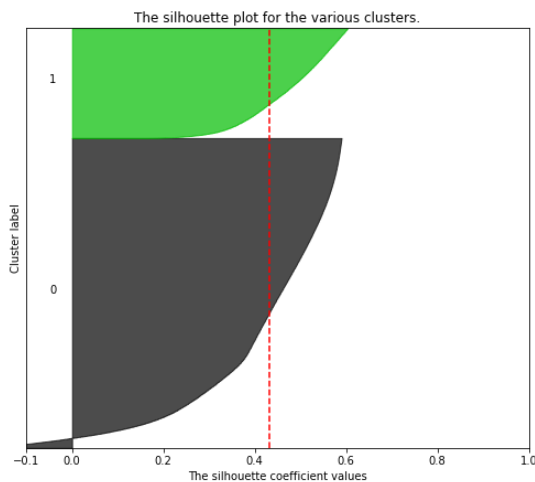
Secara umum, tahapan dan justifikasinya sama dengan yang sudah dijelaskan pada bagian B. Hasil dari reduksi dimensi menghasilkan cumulative variance explained sebanyak 69.76%. Setelah dilakukan proses pemodelan yang sama dengan bagian C, berikut hasil dari klasterisasi.



Lalu, kita akan evaluasi kualitas klasterisasi dengan tahapan yang sama pada bagian D. Berikut hasil silhouette coefficient untuk setiap sampel. Adapun rerata silhouette coefficientnya adalah 0.4330090848189424.

[https://docs.google.com/spreadsheets/d/1qX2BjCj4IqXGP6gyI41X0\\_xr1ts5IGH-/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true](https://docs.google.com/spreadsheets/d/1qX2BjCj4IqXGP6gyI41X0_xr1ts5IGH-/edit?usp=sharing&ouid=100206322009227195695&rtpof=true&sd=true)

**Silhouette analysis for KMeans clustering on sample data with n\_clusters = 2**



### 3. Analisis Hasil Eksperimen

Berikut rekapitulasi hasil eksperimen:

No.	Eksperimen	Jumlah Baris	CEV* PCA	Avg. Silhouette Coeff.	Posisi Centroid Nol	Posisi Centroid Satu
1.	Drop missing value	221199	69,73%	0.436	(0.26502203, 0.49914071)	(0.80676677, 0.50042053)
2.	Isi missing value dengan rerata	285831	69,13%	0.434	(0.27119488, 0.49875994)	(0.80702568, 0.4998344)
3.	Isi missing value dengan kNNImputer	285831	69,76%	0.433	(0.27044053, 0.49964398)	(0.80565176, 0.50067356)

*\*Cumulative Explained Variance*

Dari tabel tersebut walaupun pada eksperimen I memiliki rerata silhouette coeff. yang paling tinggi, namun jumlah barisnya berkurang drastis. Tentu saja ini tidak sesuai tujuan kita untuk mengklasterisasi seluruh data pelanggan. Adapun eksperimen II nilai cumulative explained variancanya adalah yang terkecil, sehingga hasil klasterisasinya bisa dikatakan sedikit belum menggambarkan/merepresentasikan data latih dibanding dua eksperimen yang lain. Sehingga hasil ini tidak bisa dipilih. Maka, hasil akhir yang dipilih adalah hasil dari eksperimen III. Walaupun Avg. silhouette coeff. nya yang terkecil, namun perbedaannya tidak signifikan, sehingga masih wajar untuk dipilih. Cumulative explained variancanya hasil PCA nya pun yang tertinggi, sehingga data ini adalah data yang paling representatif terhadap feature data sebelum direduksi dimensinya.

### G. Simpulan

Hasil eksplorasi menunjukkan data pelanggan yang diberikan berisi data kategorial, ordinal, dan numerik, lalu terdapat pula missing value. Model yang digunakan yaitu k-Means Clustering yang sangat sensitif dengan data kategorial karena ada melalui mekanisme penghitungan jarak. Sehingga, data kategorial yang datanya memiliki sedikit kategori didrop. Selanjutnya dilakukan normalisasi dan reduksi dimensi menggunakan PCA. Adapun eksperimen yang dilakukan berkaitan dengan penanganan missing value. Hasil akhir yang dipilih adalah hasil eksperimen III yaitu mengisi missing value dengan teknik kNN Imputation. Ketika dilakukan reduksi, hasil daripada reduksinya bisa merepresentasikan 69,76% dari data sebelum direduksi. Hasil rerata silhouette coefficient nya relatif sama dengan eksperimen yang lain, tidak berbeda secara signifikan. Berdasarkan hasil plotting, seluruh eksperimen menghasilkan plot dengan adanya nilai koefisien silhouette yang negatif. Secara umum, jika ada nilai koefisien silhouette yang negatif, maka terdapat outlier atau ada data yang salah klaster. Namun, pada kasus kali ini, nilai koefisien silhouette yang negatif cenderung sedikit, pun masalah yang diangkat adalah klasterisasi dari data pelanggan. Tujuan dari klasterisasi ini hanyalah untuk mendapatkan insight, tidak begitu harus mendapatkan presisi dan akurasi yang tinggi, misal pada kasus klasterisasi penyakit yang menyangkut hidup-mati seseorang. Kita juga



bisa melihat pertimbangan pada plotting klasterisasi di sebelah kanan yang secara kasat mata sudah terklasterisasi dengan baik. Sehingga bisa dikatakan model yang kita bangun sudah efektif dalam mengklasterisasi data pelanggan.

## LAMPIRAN

Video Youtube: <https://youtu.be/7kPm6-xOmLk>

## DAFTAR PUSTAKA

- [1] Van Rossum, G. & Drake, F.L., 2009. *Python 3 Reference Manual*, Scotts Valley, CA: CreateSpace.
- [2] Wes McKinney 2010 . Data Structures for Statistical Computing in Python . In *Proceedings of the 9th Python in Science Conference* (pp. 56 - 61 ).
- [3] Bilogur, A., 2018. Missingno: a missing data visualization suite. *The Journal of Open Source Software*, 3(22), p.547.
- [4] Michael L. Waskom 2021. Seaborn: Statistical Data Visualization. *Journal of Open Source Software*, 6(60), p.3021.
- [5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, D., Brucher, M., Perrot, M., and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, p.2825–2830.
- [6] GreatLearning Blog: Free Resources what Matters to shape your Career!. 2021. *What is Curse of Dimensionality in Machine Learning?*. [online] Available at: <<https://www.mygreatlearning.com/blog/understanding-curse-of-dimensionality/>> [Accessed 2 November 2021].
- [7] Hunter, J. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science Engineering*, 9(3), p.90-95.
- [8] Harris, C., Millman, S., Gommers, P., Cournapeau, E., Taylor, J., Berg, N., Kern, R., Picus, S., Kerkwijk, M., Haldane, J., Wiebe, P., Gérard-Marchant, K., Reddy, T., Weckesser, H., and Gohlke, T. 2020. Array programming with NumPy. *Nature*, 585, p.357–362.
- [9] Virtanen, P., Gommers, R., Oliphant, M., Reddy, T., Cournapeau, E., Peterson, P., Weckesser, J., Walt, M., Wilson, J., Millman, N., Nelson, A., Jones, R., Larson, E., Carey, ., Feng, Y., Moore, J., Laxalde, D., Perktold, R., Henriksen, I., Quintero, C., Archibald, A., Pedregosa, P., and SciPy 1.0 Contributors 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, p.261–272.