

Formulasi Masalah

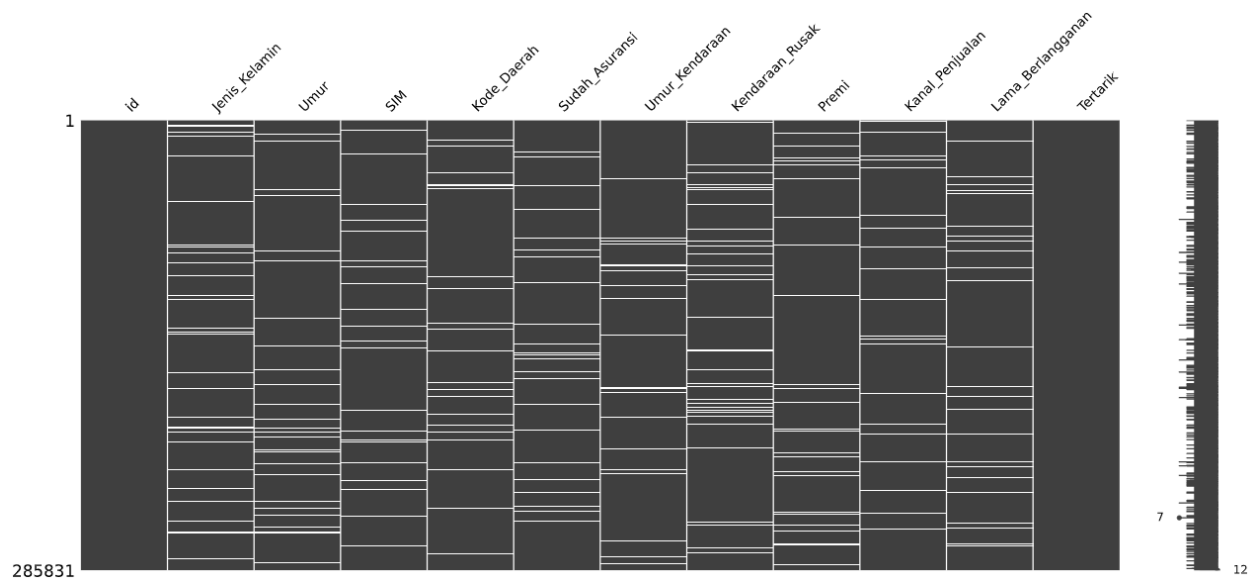
Memprediksi ketertarikan pelanggan dalam membeli kendaraan baru atau tidak berdasarkan data dari dealer

Data Exploration

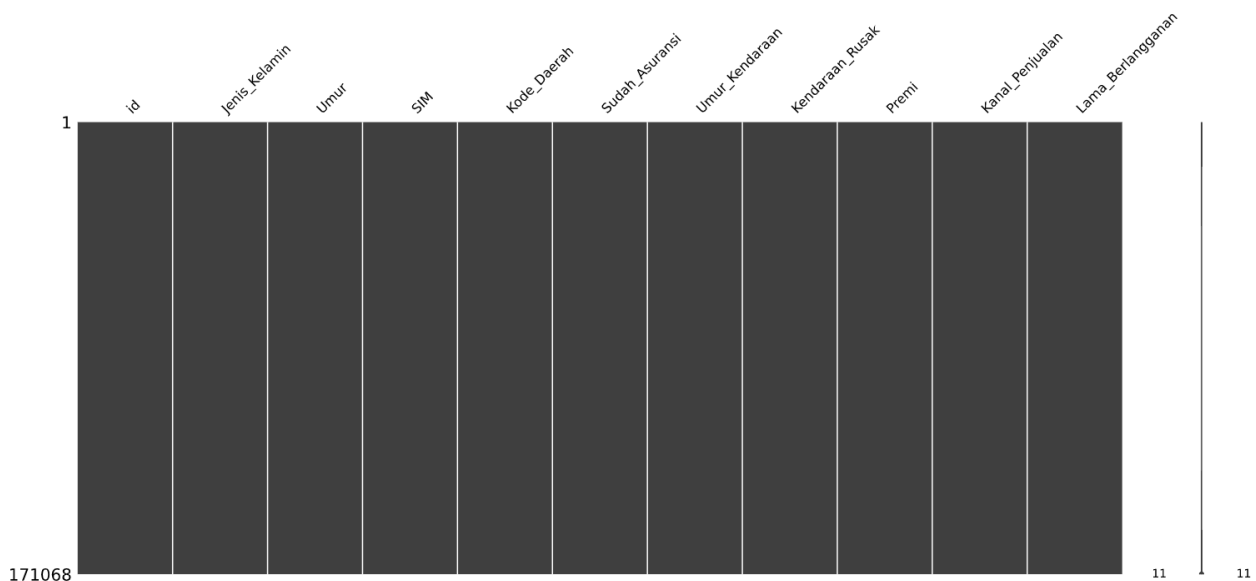
1. Impute Missing Values

Karena terdapat missing value yang sangat banyak sehingga jika semua row yang terdapat missing value dihapus, maka sekitar 40% dari total dataset akan hilang. Angka tersebut terlalu besar sehingga saya menggunakan metode imputation, yaitu mengisi value yang kosong.

Data jika row missing value tidak di delete:



Dataset jika row missing value di delete:



Untuk missing value yang terdapat di kolom Jenis_Kelamin, saya menggunakan random.choice() untuk pemilihan secara acak antara Pria dan Wanita dengan probabilitasnya masing-masing. Pada awalnya Pria mendominasi sekitar 51% data dan Wanita 43% data. Lalu agar dapat menjalankan fungsi random.choice() harus memiliki jumlah probabilitas 1. Sehingga probabilitasnya: Pria 54% dan Wanita 46%.

Untuk missing value yang terdapat di kolom umur, saya menggunakan mean untuk mencari nilai rata rata dari total value umur yg ada.

Mean Umur:

```
data["Umur"].mean()
38.802348240743655
```

Untuk missing value yang terdapat di kolom SIM, saya menggunakan random.choice() untuk pemilihan secara acak antara 1 dan 0 dengan probabilitasnya masing-masing: 99.56% dan 0.44%.

Value SIM:

```
a = data["SIM"]
unique, counts = np.unique(a, return_counts=True)
dict(zip(unique, counts))

{0.0: 584,
 1.0: 270843,}
```

Untuk missing value yang terdapat di kolom Kode_Daerah, saya menggunakan mode untuk mencari nilai modus dari total value Kode_Daerah yg ada. Modusnya yaitu "28".

Untuk missing value yang terdapat di kolom Sudah_Asuransi, saya menggunakan random.choice() untuk pemilihan secara acak antara 1 dan 0 dengan probabilitasnya masing-masing: 45.83% dan 54.17%.

```
a = data["Sudah_Asuransi"]
unique, counts = np.unique(a, return_counts=True)
dict(zip(unique, counts))

{0.0: 146997,
 1.0: 124605,}
```

Untuk missing value yang terdapat di kolom Umur_Kendaraan, saya menggunakan random.choice() untuk pemilihan secara acak antara < 1 Tahun, 1-2 Tahun, > 2 Tahun dengan probabilitasnya masing-masing: 43.1%, 52.7%, 4.2%.

Untuk missing value yang terdapat di kolom Kendaraan_Rusak, saya menggunakan random.choice() untuk pemilihan secara acak antara Pernah dan Tidak dengan probabilitasnya masing-masing: 53.55%, 46.45%.

Untuk missing value yang terdapat di kolom Premi, saya menggunakan mean untuk mencari nilai rata rata dari total value Premi yg ada.

Untuk missing value yang terdapat di kolom Kanal_Penjualan, saya menggunakan mode untuk mencari nilai modus dari total value Kode_Daerah yg ada. Modusnya yaitu "152".


Untuk missing value yang terdapat di kolom Lama_Berlangganan, saya menggunakan mean untuk mencari nilai rata rata dari total value Lama_Berlangganan yg ada.

Note: Untuk kolom yang merupakan categorical values, seperti: Jenis_Kelamin, SIM, Umur_Kendaraan dan Kendaraan_Rusak. Melalui beberapa tahap :

1. Perhitungan persen suatu value dari total data
2. Jika dijumlahkan maka tidak mencapai 100% karena terdapat data yang kosong
3. Persenan data kosong tersebut disubstitusi kedalam persenan value dengan pembagian berdasarkan persenan awal.
4. Maka total probabilitasnya mencapai 100% untuk mengisi missing value

```
#input missing value
data["Jenis_Kelamin"] = data["Jenis_Kelamin"].replace(np.NaN, np.random.choice(a=["Pria", "Wanita"],p=[0.54, 0.46]))
data["Umur"] = data["Umur"].replace(np.NaN, data["Umur"].mean())
data["SIM"] = data["SIM"].replace(np.NaN, np.random.choice(a=[1, 0],p=[0.9956, 0.0044]))
data["Kode_Daerah"] = data["Kode_Daerah"].replace(np.NaN, statistics.mode(data["Kode_Daerah"]))
data["Sudah_Asuransi"] = data["Sudah_Asuransi"].replace(np.NaN, np.random.choice(a=[0, 1],p=[0.5417, 0.4583]))
data["Umur_Kendaraan"] = data["Umur_Kendaraan"].replace(np.NaN, np.random.choice(a=["< 1 Tahun", "1-2 Tahun", "> 2 Tahun"],p=[0.431, 0.527, 0.042]))
data["Kendaraan_Rusak"] = data["Kendaraan_Rusak"].replace(np.NaN, np.random.choice(a=["Pernah", "Tidak"],p=[0.5355, 0.4645]))
data["Premi"] = data["Premi"].replace(np.NaN, data["Premi"].mean())
data["Kanal_Penjualan"] = data["Kanal_Penjualan"].replace(np.NaN, 152)
data["Lama_Berlangganan"] = data["Lama_Berlangganan"].replace(np.NaN, data["Lama_Berlangganan"].mean())
```

Setelah semua missing values sudah terisi, maka saya lakukan perubahan tipe data. Untuk kolom yang bertipe data float akan saya ubah menjadi integer dengan maksud penyederhanaan angka.

data.dtypes				data = data.astype(convert_data_types) data.dtypes	
id	int64			id	int32
Jenis_Kelamin	object			Jenis_Kelamin	object
Umur	float64			Umur	int32
SIM	float64			SIM	int32
Kode_Daerah	float64			Kode_Daerah	int32
Sudah_Asuransi	float64			Sudah_Asuransi	int32
Umur_Kendaraan	object			Umur_Kendaraan	object
Kendaraan_Rusak	object			Kendaraan_Rusak	object
Premi	float64			Premi	int32
Kanal_Penjualan	float64			Kanal_Penjualan	int32
Lama_Berlangganan	float64			Lama_Berlangganan	int32
dtype: object				dtype: object	

2. Outliers

Menurut pemahaman saya terhadap data yang diberikan, dari semua kolom yang paling rentan untuk terjadi Outlier, mungkin karena kesalahan input, dsb, dan yang paling fatal jika terjadi kesalahan, yaitu kolom Premi. Di kolom Premi terdapat data yang paling kecil yaitu sekitar 2000 dan yang paling besar 500.000 . Angka tersebut sangatlah jauh, Lalu saya mencoba mendeteksi outlier menggunakan fungsi `detect_outlier`. Dari hasilnya terdapat sekitar 1626 data yang dideteksi sebagai outlier, angka tersebut setidaknya mewakili sekitar 1% dari total data. Tapi untuk memastikan apakah itu merupakan outlier, saya membandingkannya dengan dataset `kendaraan_test` yang dianggap bebas dari outlier sebagai konfirmasi bahwa benar terdapat outlier di dataset `kendaraan_train`. Dari hasilnya terdapat 253 data yang dideteksi sebagai outlier, angka tersebut setidaknya mewakili sekitar 1% dari total dataset `kendaraan_test`. Hal ini menarik karena sama sama terdeteksi 1% dari total dataset terdeteksi sebagai outlier. Sehingga saya menarik kesimpulan bahwa tidak terdapat outlier di dataset. Dan saya rasa tidak diperlukan operasi lanjutan mengenai outlier.

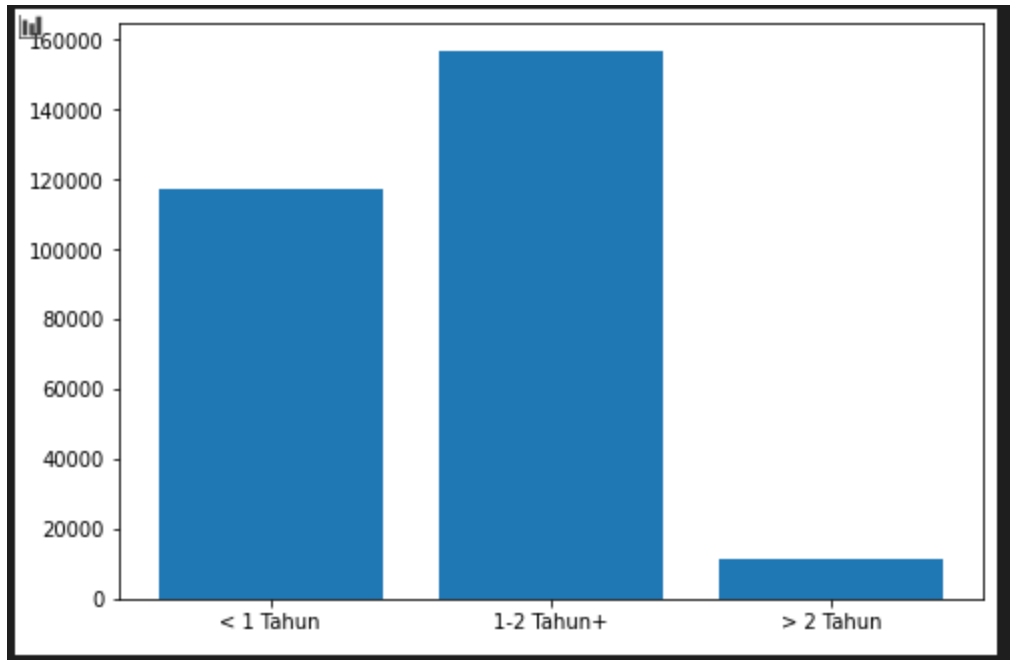
```
Data Train (Lowest: 80682 ; Highest 540165): 1626 -> 1.0 %  
Data Test (Lowest: 82876; Highest: 540165): 253 -> 1.0 %
```

Untuk penjelasan fungsi `detect_outlier` adalah sebagai berikut: Dalam fungsi ini saya menggunakan metode `z_score`. `Z_score` sendiri merupakan hasil kalkulasi dari $(\text{value} - \text{mean}) / \text{Standard Deviation}$. Lalu saya tetapkan `thresholdnya`, yaitu 3. Jika hasil `z_score` nya melebihi `threshold` maka terdeteksi sebagai outlier. Lakukan hal tersebut untuk semua dataset.

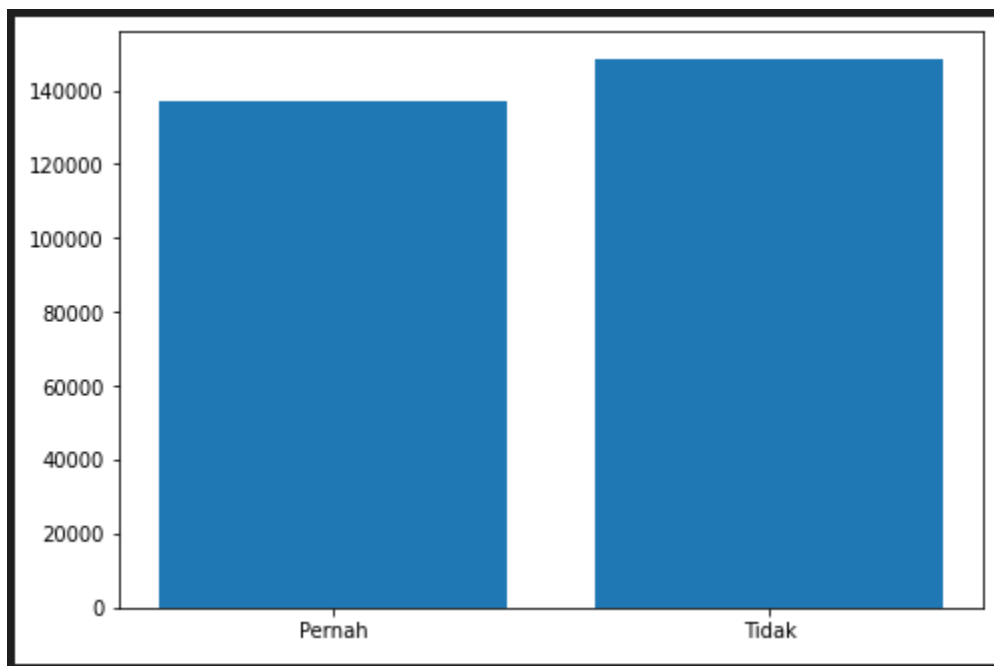
3. Data Distributions

Saya melakukan pengecekan distribusi data untuk kolom `Umur_Kendaraan` dan `Kendaraan_Rusak`. Dari hasil yang didapat terdapat ketidak seimbangan untuk kolom `Umur_Kendaraan`. Untuk umur kendaraan yang > 2 tahun sangat sedikit datanya.

`Umur_Kendaraan`:



Kendaraan_Rusak:



4. Scaling

Saya melakukan scaling terhadap kolom Umur_Kendaraan dan Kendaraan_Rusak, untuk memudahkan dalam proses pengukuran jarak antar baris data di K-Means nanti. Untuk kolom Umur_kendaraan saya mengganti kategori "< 1 tahun", "1-2 Tahun", dan "> 2 Tahun" menjadi 0, 0.5, dan 1. Untuk kolom Kendaraan_Rusak saya mengganti kategori "Pernah" dan "Tidak" menjadi 1 dan 0.

```
▶ ▶≡ ML

#Scaling kolom Umur_Kendaraan untuk diubah nilainya menjadi range 0 - 1
scale_Umur_Kendaraan = {"< 1 Tahun":0, "1-2 Tahun":0.5, "> 2 Tahun":1}

data["Umur_Kendaraan"] = data["Umur_Kendaraan"].replace(scale_Umur_Kendaraan)

print(data["Umur_Kendaraan"][:5])

0    0.0
1    1.0
2    0.0
3    0.5
4    1.0
Name: Umur_Kendaraan, dtype: float64

▶ ▶≡ ML

#Scaling kolom Kendaraan_Rusak untuk diubah nilainya menjadi range 0 - 1
scale_Kendaraan_Rusak = {"Tidak":0, "Pernah":1}

data["Kendaraan_Rusak"] = data["Kendaraan_Rusak"].replace(scale_Kendaraan_Rusak)

print(data["Kendaraan_Rusak"][:5])

0    0
1    1
2    0
3    0
4    0
Name: Kendaraan_Rusak, dtype: int64
```

5. Normalization

Saya melakukan normalisasi untuk kolom Premi. Metode normalisasi yang saya gunakan adalah min max normalization. Min max normalization adalah metode yang dimana mulai dari nilai terkecil ke nilai terbesar akan diubah menjadi 0 dan 1.

```
#Normalisasi kolom Premi sehingga nilainya berada di range 0 - 1
def min_max_normalization (value, data2):
    return (value - data2.min()) / (data2.max() - data2.min())

tic = time.perf_counter()

for i in range(len(data["Premi"])):
    scale_Premi = min_max_normalization (data["Premi"].iloc[i], data["Premi"])
    data["Premi"].iloc[i] = scale_Premi

toc = time.perf_counter()
print(f"Time: {toc - tic:0.4f} seconds")
print(data["Premi"][:5])

C:\Users\shidq\AppData\Roaming\Python\Python39\site-packages\pandas\core\indexing.py:1637: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
self._setitem_single_block(indexer, value, name)
Time: 486.8930 seconds
0    0.047251
1    0.047763
2    0.060598
3    0.004869
4    0.064530
Name: Premi, dtype: float64
```

6. Dimensionality Reductions

Saya melakukan pengurangan fitur sehingga hanya tersisa 3 fitur yaitu: “Umur_kendaraan”, “Kendaraan_Rusak”, dan “Premi. Karena saya rasa fitur tersebut yang menjadi acuan utama apakah pelanggan ingin membeli kendaraan baru atau tidak.

Lalu Terakhir untuk data exploration adalah saya mengubah tipe dataset dari dataframe menjadi list agar mempermudah saya dalam melakukan pemodelan data.

```
#Yang diambil 3 kolom lalu diubah menjadi list
#[Umur_kendaraan, Kendaraan_Rusak, Premi]
dataList = data[["Umur_Kendaraan", "Kendaraan_Rusak", "Premi"]].values.tolist()
len(dataList)
dataList[:10]

[[0.0, 0.0, 0.047250876687099444],
 [1.0, 1.0, 0.047763100174894095],
 [0.0, 0.0, 0.06059806839101972],
 [0.5, 0.0, 0.0048687956072277425],
 [1.0, 0.0, 0.06453027398181872],
 [0.0, 0.0, 0.04208898269256069],
 [0.5, 1.0, 0.06004645881084174],
 [0.0, 0.0, 0.04974219983410439],
 [0.0, 0.0, 0.056993687870733965],
 [1.0, 1.0, 0.16454786210205088]]
```

Data Modelling

K_Means

Untuk pemodelan data saya menggunakan metode K-Means untuk melakukan clustering. Untuk menjalankan K-Means setidaknya saya membutuhkan 5 fungsi pembantu, yaitu: `init_centroid`, `dist`, `cluster_determination`, `compute_centroid`, `measure_change`.

Fungsi `init_centroid` untuk melakukan inisiasi centroid yang didapatkan dari random datapoints. Untuk random datapoint, saya lakukan menggunakan random integer sebagai index datapoints yang akan menjadi initial centroid. Hasilnya akan di assign ke array. Lakukan hal tersebut hingga sebanyak cluster yang diminta.

Fungsi `dist` untuk melakukan kalkulasi jarak antar data. Rumus yang digunakan adalah euclidean distance.

Fungsi `cluster_determination` untuk menentukan tiap data termasuk cluster yang mana melalui jarak terdekat untuk tiap 1 datapoint ke semua cluster yang ada. Hasil dari jarak 1 datapoint ke tiap cluster akan dimasukan ke array lalu akan dipilih index yang dimana terdapat angka yang paling kecil yang menandakan termasuk ke dalam cluster tersebut. Lakukan hal tersebut untuk semua datapoint.

Fungsi `compute_centroid` untuk melakukan perubahan lokasi centroid berdasarkan rata rata semua datapoints yang termasuk kedalam cluster tersebut. Tiap cluster dilakukan looping ke semua datapoints, lalu yang termasuk ke dalam cluster tersebut akan dimasukkan kedalam array. Setelah terkumpul semua datapoints yang termasuk kedalam cluster tersebut, dilakukan rata rata nilai datapoints tersebut dan hasilnya akan di assign ke dalam `cg_arr`. Setelah semua cluster selesai maka `cg_arr` menjadi cikal bakal dari `cg_new` yang merupakan lokasi centroid yang baru.

Fungsi `measure_change` untuk melakukan monitor terhadap `cg_prev` (lokasi centroid sebelumnya) dan `cg_new` (lokasi centroid yang baru). Jika hanya terjadi perubahan kurang dari 0.001 maka iterasi akan dihentikan.

Semua fungsi diatas akan digunakan kedalam fungsi `KMeans` yang dimana fungsi ini untuk melakukan metode K-Means. Pertama tama, dilakukan initial centroid yang dilakukan oleh fungsi `init_centroid` lalu hasilnya akan di assign ke variable `cg_prev`. lalu dibuat cluster yang berupa array sepanjang datapoints nya. Lalu dilakukan looping, lakukan fungsi `cluster_determination` untuk menentukan semua datapoints termasuk ke cluster yang mana dan hasilnya akan disimpan di variable `cluster`. Dari hasil cluster tersebut akan dilakukan fungsi `compute_centroid` untuk dilakukan perubahan lokasi centroid yang baru dan hasilnya akan di simpan di variable `cg_new`. Lalu akan dilakukan pengukuran perbedaan jarak antar lokasi cluster yang dulu dengan cluster yang baru dan hasilnya akan di assign ke variable `cg_change`. Dan untuk melakukan iterasi selanjutnya maka `cg_prev` akan di update dengan `cg_new`.

Evaluasi

Dikarenakan hasil clustering sangat berpengaruh pada initial centroid yang dilakukan. Oleh karena itu saya melakukan ke-Means sebanyak 5 kali dan yang paling terakhir yang di pilih untuk dijadikan clustering final.

Kesimpulan

Pada tugas kali ini adalah melakukan clustering pada dataset yang diberikan untuk memberi gambaran apakah orang tersebut memiliki minat untuk membeli kendaraan yang baru atau tidak.

Untuk Data Exploration, dilakukan 6 tahapan, yaitu: Impute Missing Value, Outliers, Data Distribution, Scaling, Normalization, Dimensionality Reduction. Impute Missing value untuk melakukan input value ke kolom yang terdapat missing value. Outliers untuk mengecek data yang jauh dari semua data. Data Distribution untuk melihat persebaran data, semakin jumlah data yang tersebar di tiap kolom semakin baik. Scaling untuk merubah value dengan tipe kategorikal menjadi numerik, sehingga bisa dilakukan perhitungan jarak. Normalization untuk merubah semua nilai pada kolom tersebut menjadi antara range 0 - 1. Dimensionality Reduction untuk memfokuskan fitur tertentu yang akan dilakukan pemodelan, Hal tersebut dilakukan agar pemodelan dilakukan menjadi lebih simple dan hasil yang didapat menjadi lebih akurat.

Pada tahap pemodelan, Saya menggunakan metode K-Means, yang dimana saya menggunakan $k = 2$ karena formulasi masalahnya memiliki 2 kemungkinan: minat membeli kendaraan baru atau tidak berminat. Karena hasil dari pemodelan K-Means bergantung pada pola persebaran data dan initial centroid, oleh karena itu saya melakukan 5 kali pemodelan. Dari hasil yang didapat setidaknya terdapat 2 variasi clustering, sebagai berikut:

```
[1 0 1 ... 1 1 0]
1 iteration 9.0108 seconds
[0 1 0 ... 0 0 1]
2 iteration 9.2308 seconds
[1 0 1 ... 1 1 0]
3 iteration 9.0654 seconds
[1 0 1 ... 1 1 0]
4 iteration 18.4826 seconds
[1 0 1 ... 1 1 0]
5 iteration 18.1538 seconds
```

Link Video Presentasi: <https://youtu.be/IA9IrPbBEJc>