

LAPORAN TUGAS BESAR TAHAP 2
PEMBELAJARAN MESIN
CLASSIFICATION



Oleh :

Rafly Athalla (1301194216)

Zahra Fadiah Putri (1301194212)

PROGRAM STUDI INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2021

1. Formulasi Masalah

1.1 Identifikasi Masalah

Dari dataset yang diberikan akan dilakukan classfication yaitu memprediksi data dengan ketentuan-ketentuan tertentu. Agar dapat melakukan classification, dataset yang diberikan akan diterapkan model *supervised learning* agar mempermudah selama proses classification. Dataset yang diberikan ada sekitar kurang lebih 200.000 data.

1.2 Identifikasi Sumber Data

Dataset yang digunakan yaitu kendaraan.csv yang berupa data numerik dan data kategorikal. Dataset ini adalah dataset yang dipilihkan oleh dosen saya. Jumlah keseluruhan data , sebelum dilakukan pra pemrosesan 285831. Pada data kendaraan yang saya dapatkan dalam file kendaraan.csv terdapat kolom Jenis_Kelamin, umur, sim, Kode_Daerah, Sudah_asuransi, Umur_Kendaraan, Kendaraan_Rusak, Premi, Kanal_Penjualan dan Lama_Berlangganan. Agar dapat melakukan clustering pada data, maka dataset harus dieksplorasi terlebih dahulu.

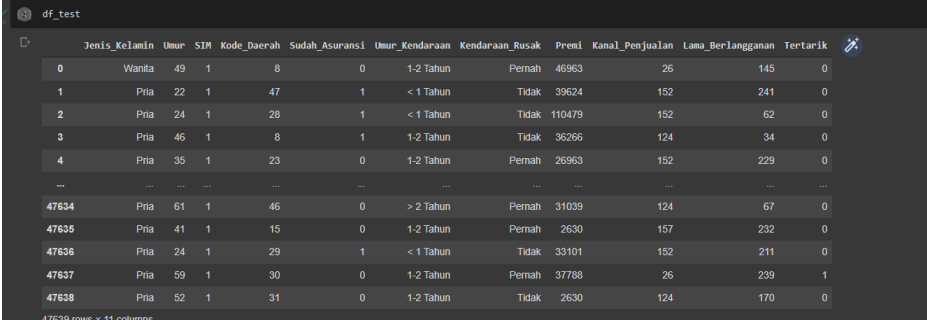
2. Eksplorasi Data

Terdapat beberapa tahap yang kami lakukan dalam eksplor data Kendaraan, antara lain sebagai berikut :

2.1 Upload Dataset yang Akan Digunakan

Upload dataset yang akan digunakan pada proses ini, dataset yang digunakan adalah dataset kendaraan, yang dibagi menjadi dataset test dan dataset train.

```
[3] df_test = pd.read_csv('kendaraan_test.csv')
     df_train = pd.read_csv('kendaraan_train.csv')
```



	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	Wanita	49	1	8	0	1-2 Tahun	Pernah	46963	26	145	0
1	Pria	22	1	47	1	< 1 Tahun	Tidak	39624	152	241	0
2	Pria	24	1	28	1	< 1 Tahun	Tidak	110479	152	62	0
3	Pria	46	1	8	1	1-2 Tahun	Tidak	36266	124	34	0
4	Pria	35	1	23	0	1-2 Tahun	Pernah	26963	152	229	0
...
47634	Pria	61	1	46	0	> 2 Tahun	Pernah	31039	124	67	0
47635	Pria	41	1	15	0	1-2 Tahun	Pernah	2630	157	232	0
47636	Pria	24	1	29	1	< 1 Tahun	Tidak	33101	152	211	0
47637	Pria	59	1	30	0	1-2 Tahun	Pernah	37788	26	239	1
47638	Pria	52	1	31	0	1-2 Tahun	Tidak	2630	124	170	0

2.2 Keluaran dari Seluruh Dataset

```
[4] df_test
```



	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	Wanita	49	1	8	0	1-2 Tahun	Pernah	46963	26	145	0
1	Pria	22	1	47	1	< 1 Tahun	Tidak	39624	152	241	0
2	Pria	24	1	28	1	< 1 Tahun	Tidak	110479	152	62	0
3	Pria	46	1	8	1	1-2 Tahun	Tidak	36266	124	34	0
4	Pria	35	1	23	0	1-2 Tahun	Pernah	26963	152	229	0
...
47634	Pria	61	1	46	0	> 2 Tahun	Pernah	31039	124	67	0
47635	Pria	41	1	15	0	1-2 Tahun	Pernah	2630	157	232	0
47636	Pria	24	1	29	1	< 1 Tahun	Tidak	33101	152	211	0
47637	Pria	59	1	30	0	1-2 Tahun	Pernah	37788	26	239	1
47638	Pria	52	1	31	0	1-2 Tahun	Tidak	2630	124	170	0

5) df_train

	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	0.0
1	2	Pria	48.0	1.0	39.0	0.0	> 2 Tahun	Pernah	25800.0	29.0	158.0	0.0
2	3	NaN	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	32733.0	160.0	119.0	0.0
3	4	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.0	0.0
4	5	Pria	50.0	1.0	35.0	0.0	> 2 Tahun	NaN	34857.0	88.0	194.0	0.0
...
20092	20093	Wanita	42.0	1.0	28.0	1.0	1-2 Tahun	Tidak	45878.0	26.0	211.0	0.0
20093	20094	Wanita	51.0	1.0	50.0	0.0	> 2 Tahun	Pernah	24710.0	8.0	61.0	0.0
20094	20095	Pria	24.0	1.0	15.0	1.0	< 1 Tahun	Tidak	54678.0	152.0	40.0	0.0
20095	20096	Wanita	43.0	1.0	18.0	1.0	1-2 Tahun	Tidak	30408.0	NaN	189.0	0.0
20096	20097	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

20097 rows x 12 columns

2.3 Menggabungkan Data

Menggabungkan data menjadi satu dataframe untuk memudahkan proses preprocessing pada tahap selanjutnya

[6] # Menggabungkan dataframe testing dan training
dataframe = pd.concat([df_test, df_train]).drop_duplicates().reset_index(drop=True)
dataframe

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik	id
0	Wanita	49.0	1.0	8.0	0.0	1-2 Tahun	Pernah	46963.0	26.0	145.0	0.0	NaN
1	Pria	22.0	1.0	47.0	1.0	< 1 Tahun	Tidak	39624.0	152.0	241.0	0.0	NaN
2	Pria	24.0	1.0	28.0	1.0	< 1 Tahun	Tidak	110479.0	152.0	62.0	0.0	NaN
3	Pria	46.0	1.0	8.0	1.0	1-2 Tahun	Tidak	36266.0	124.0	34.0	0.0	NaN
4	Pria	35.0	1.0	23.0	0.0	1-2 Tahun	Pernah	26963.0	152.0	229.0	0.0	NaN
...
67728	Wanita	42.0	1.0	28.0	1.0	1-2 Tahun	Tidak	45878.0	26.0	211.0	0.0	20093.0
67729	Wanita	51.0	1.0	50.0	0.0	> 2 Tahun	Pernah	24710.0	8.0	61.0	0.0	20094.0
67730	Pria	24.0	1.0	15.0	1.0	< 1 Tahun	Tidak	54678.0	152.0	40.0	0.0	20095.0
67731	Wanita	43.0	1.0	18.0	1.0	1-2 Tahun	Tidak	30408.0	NaN	189.0	0.0	20096.0
67732	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	20097.0

67733 rows x 12 columns

3. Preprocessing

Pada tahap Data Preprocessing, kami melakukan beberapa tahap yaitu, drop kolom id, lalu clear missing value / NaN, lalu mengubah data kategorikal mejadi numerical.

3.1 Check Empty

Persiapan yang pertama dilakukan yaitu memastikan bahwa tidak ada *missing value* atau data yang kosong, karena jika terdapat data yang kosong maka akan berpengaruh terhadap tingkat akurasi dari suatu data.

[10] dataframe.isnull().any()

Jenis_Kelamin	False
Umur	False
SIM	False
Kode_Daerah	False
Sudah_Asuransi	False
Umur_Kendaraan	False
Kendaraan_Rusak	False
Premi	False
Kanal_Penjualan	False
Lama_Berlangganan	False
Tertarik	False
dtype: bool	

```
[9] dataframe.isnull().sum()

Jenis_Kelamin      0
Umur               0
SIM               0
Kode_Daerah        0
Sudah_Asuransi     0
Umur_Kendaraan     0
Kendaraan_Rusak    0
Premi             0
Kanal_Penjualan    0
Lama_Berlangganan  0
Tertarik          0
dtype: int64
```

Setelah melakukan pengecekan data kosong dan memastikan tidak terdapat data yang kosong, selanjutnya yaitu menampilkan statistic data set.

3.2 Drop Kolom ID

```
Drop kolom yang tidak digunakan

[25] #kendaraan_test
dropKolom = ['SIM']
df_test = df_test.drop(dropKolom, axis = 1)

#kendaraan_train
dropKolom = ['SIM']
df_train = df_train.drop(dropKolom, axis = 1)
```

3.3 Clear Missing Value

Selanjutnya membuang data nilai yang hilang

```
# Membuang data yang missing value / NaN
dataframe.dropna(inplace=True)
dataframe
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	Wanita	49.0	1.0	8.0	0.0	1-2 Tahun	Pernah	46963.0	26.0	145.0	0.0
1	Pria	22.0	1.0	47.0	1.0	< 1 Tahun	Tidak	39624.0	152.0	241.0	0.0
2	Pria	24.0	1.0	28.0	1.0	< 1 Tahun	Tidak	110479.0	152.0	62.0	0.0
3	Pria	46.0	1.0	8.0	1.0	1-2 Tahun	Tidak	36266.0	124.0	34.0	0.0
4	Pria	35.0	1.0	23.0	0.0	1-2 Tahun	Pernah	26963.0	152.0	229.0	0.0

3.4 Mengubah Data Kategorikal Menjadi Numerical

Tahap ini dilakukan agar data tersebut dapat di plotting, karena apabila ada data yang bertipe kategorikal dan numerical, maka proses plotting akan sulit dilakukan.

```
[11] # Mengubah tipe data kategorikal menjadi numerikal
dataframe = dataframe.apply(lambda series: pd.Series(LabelEncoder().fit_transform(series[series.notnull()]), index=series[series.notnull()].index))
dataframe
```

3.5 Normalisasi Data

Untuk normalisasi data yang kami gunakan adalah Min Max

```
[15] # Normalisasi dataframe dengan MinMax range [0,1]
new_dataframe = dataframe.copy()
new_dataframe = (dataframe - dataframe.min(axis=0)) / (dataframe.max(axis=0) - dataframe.min(axis=0))
new_dataframe
```

Normalisasi data ini dilakukan untuk membuat semua value yang ada pada dataframe berkisar antara 0-1.

4. Pemodelan

Pada tahap ini metode classification yang kami gunakan yaitu metode Naïve Bayes dan Decision Tree. Naïve Bayes merupakan sebuah metode klasifikasi berdasarkan probabilitas sederhana dan dirancang untuk dipergunakan dengan asumsi bahwa antar satu kelas dengan kelas yang lain tidak saling tergantung (independen). Pada klasifikasi Naïve Bayes, proses pembelajaran lebih ditekankan pada mengestimasi probabilitas. Sedangkan Decision Tree merupakan metode klasifikasi menggunakan struktur pohon atau struktur berhirarki. Konsep dari pohon keputusan adalah mengubah data menjadi decision tree dan aturan-aturan keputusan.

5. Eksperimen

X = Feature , Y = Class Target

```
[20] # Data Train
      x_train = df_train.drop(['Tertarik'], axis=1)
      y_train = df_train['Tertarik'].values

[21] x_train = x_train.values

[22] #Data test

      x_test = df_test.drop(['Tertarik'], axis=1)
      y_test = df_test['Tertarik'].values

[23] x_test = x_test.values
```

Untuk eksperimen, kami membagi data train dan data test menjadi 2 bagian masing-masingnya, yaitu feature dan class target untuk data train, feature dan class target untuk data test.

6. Evaluasi

Hasil evaluasi metode sebagai berikut

6.1 Naïve Bayes

```
[ ] def confusion_metrics (y_test,y_pred):  
    # Untuk menentukan True Positive, False Positive, True Negative dan False Negative  
    cm = metrics.confusion_matrix(y_test, y_pred)  
    cm_df = pd.DataFrame(cm,  
                          columns = ['Predicted Negative', 'Predicted Positive'],  
                          index = ['Actual Negative', 'Actual Positive'])  
  
    # Memasukkan nilai True Positive, False Positive, True Negative dan False Negative ke variable  
    TN = cm_df.loc['Actual Negative','Predicted Negative']  
    FN = cm_df.loc['Actual Positive','Predicted Negative']  
    FP = cm_df.loc['Actual Negative','Predicted Positive']  
    TP = cm_df.loc['Actual Positive','Predicted Positive']  
  
    # Menampilkan jumlah masing-masing True Positive, False Positive, True Negative dan False Negative  
    print('True Negative : ',TN)  
    print('False Negative : ',FN)  
    print('False Positive : ',FP)  
    print('True Positive : ',TP)  
    print('')  
  
    # Menampilkan hasil perhitungan Precision, Recall dan F1-Score  
    print(classification_report(y_test,y_pred))
```

```
[ ] confusion_metrics(y_test, y_pred_nb)
```

```
True Negative : 6208  
False Negative : 38  
False Positive : 4258  
True Positive : 1433
```

	precision	recall	f1-score	support
0.0	0.99	0.59	0.74	10466
1.0	0.25	0.97	0.40	1471
accuracy			0.64	11937
macro avg	0.62	0.78	0.57	11937
weighted avg	0.90	0.64	0.70	11937

6.2 Decision Tree

```
[ ] def confusion_metrics (y_test,y_pred):  
    # Untuk menentukan True Positive, False Positive, True Negative dan False Negative  
    cm = metrics.confusion_matrix(y_test, y_pred)  
    cm_df = pd.DataFrame(cm,  
                          columns = ['Predicted Negative', 'Predicted Positive'],  
                          index = ['Actual Negative', 'Actual Positive'])  
  
    # Memasukkan nilai True Positive, False Positive, True Negative dan False Negative ke variable  
    TN = cm_df.loc['Actual Negative','Predicted Negative']  
    FN = cm_df.loc['Actual Positive','Predicted Negative']  
    FP = cm_df.loc['Actual Negative','Predicted Positive']  
    TP = cm_df.loc['Actual Positive','Predicted Positive']  
  
    # Menampilkan jumlah masing-masing True Positive, False Positive, True Negative dan False Negative  
    print('True Negative : ',TN)  
    print('False Negative : ',FN)  
    print('False Positive : ',FP)  
    print('True Positive : ',TP)  
    print('')  
  
    # Menampilkan hasil perhitungan Precision, Recall dan F1-Score  
    print(classification_report(y_test,y_pred))
```

```
[ ] confusion_metrics(y_test, y_pred_dtree)
```

```
True Negative : 7622
False Negative : 384
False Positive : 2844
True Positive : 1087
```

	precision	recall	f1-score	support
0.0	0.95	0.73	0.83	10466
1.0	0.28	0.74	0.40	1471
accuracy			0.73	11937
macro avg	0.61	0.73	0.61	11937
weighted avg	0.87	0.73	0.77	11937

Dapat dilihat bahwa Algoritma Decision Tree memiliki nilai akurasi lebih tinggi yaitu 0.73 jika dibandingkan dengan nilai akurasi yang dimiliki oleh algoritma Naïve Bayes yaitu 0.64

Namun walaupun nilai dari masing-masing precision, recall dan f1-score berbeda yang menghasilkan sebuah akurasi yang berbeda, namun prediksi kedua metode menunjukkan kesamaan. Bisa dilihat dari dataframe dengan kolom yang ditambahkan yaitu kolom prediksi.

7. Kesimpulan

Berdasarkan hasil eksperimen klasifikasi terhadap dataset kendaraan_train dan dataset kendaraan_test yang telah kami lakukan mendapatkan hasil yaitu nilai akurasi klasifikasi data dengan menggunakan Decision Tree lebih tinggi 73% dibandingkan dengan nilai akurasi yang dimiliki Naïve Bayes yaitu 64%. Preprocessing yang dilakukan membantu klasifikasi memiliki akurasi yang lebih tinggi.