LAPORAN OBSERVASI
TUGAS BESAR 02 – KLASIFIKASI

diajukan untuk memenuhi tugas mata kuliah (CII3C3) Pembelajaran Mesin

oleh:
Otniel Abiezer (NIM 1301180469)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2021

1. FORMULASI MASALAH

Masalah yang akan diselesaikan adalah pada toko dealer mobil. Toko dealer tersebut memiliki data-data pelanggan yang mencakup ID, jenis kelamin, umur, sudah memiliki SIM atau belum, kode daerah, sudah pernah asuransi atau belum, umur kendaraan, kendaran pernah rusak atau tidak, premi, kanal penjualan, lama berlangganan dan tertarik atau tidak.

Untuk itu, dilakukan prediksi untuk mengetahui pelanggan pada dealer mobil tersebut tertarik untuk membeli mobil baru atau tidak berdasarkan data-data pelanggan yang dimiliki oleh toko dealer mobil. Untuk itu dibangun model klasifikasi Decision Tree yang merupakan contoh dari Supervised Learning. Data-data tersebut memiliki label kelas tertarik atau tidak yang akan diprediksi.

2. EKSPLORASI DAN PERSIAPAN DATA

2.1. Eksplorasi Data Train

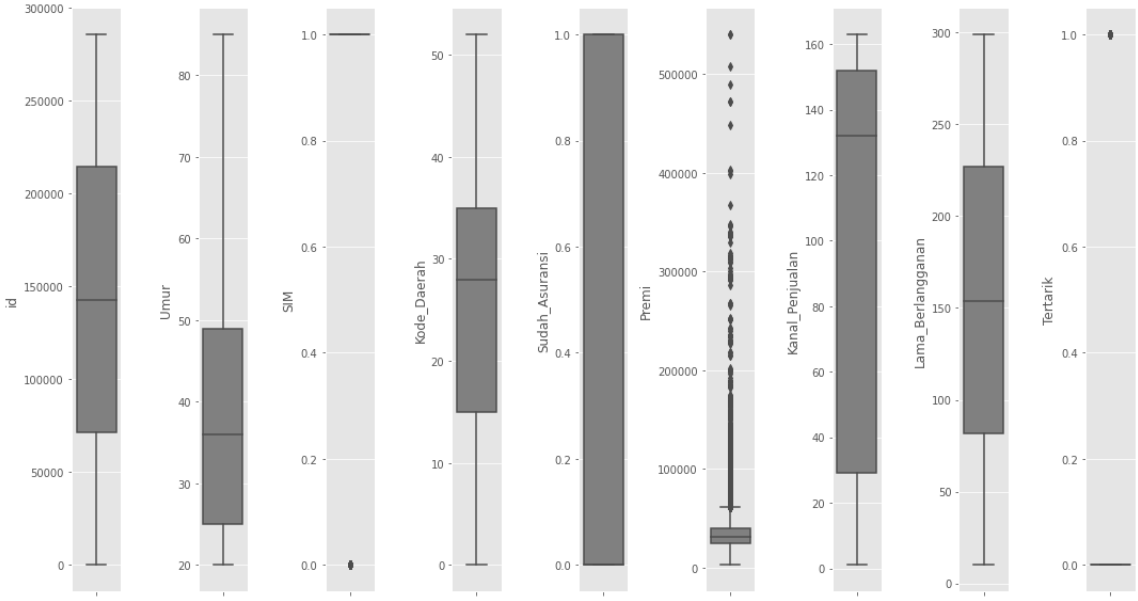
a. Mengetahui ukuran data train (kendaraan_train.csv), yaitu 285831 x 12

b. Melihat banyaknya missing value dan tipe data setiap kolom

```
<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
```

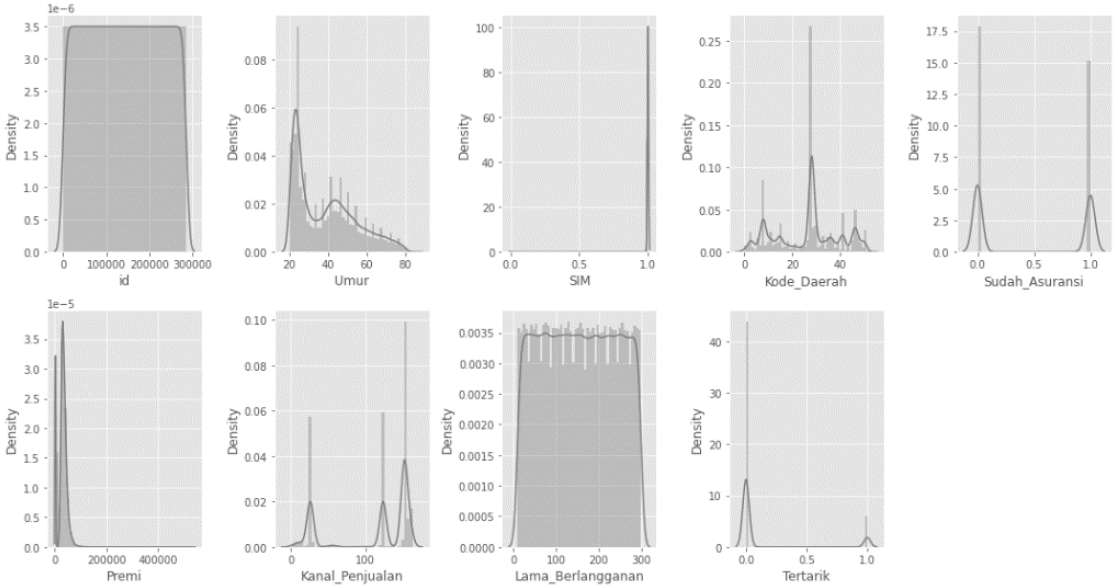
c. Melakukan pembagian kolom yang menjadi kategorikal (object) dan numerikal (int64, float64). Untuk kategorikal adalah Jenis_Kelamin, Umur_Kendaraan, Kendaraan_Rusak. Sisanya adalah numerikal.

d. Memvisualsasikan box plot untuk melihat outlier



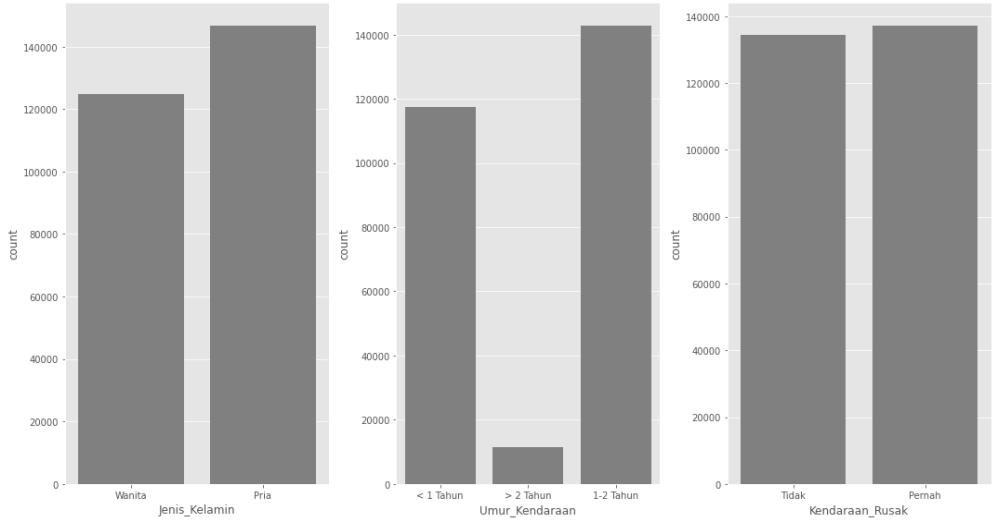
Selain Premi, tidak ada outlier pada masing-masing kolom.

e. Memvisualisasikan persebaran data



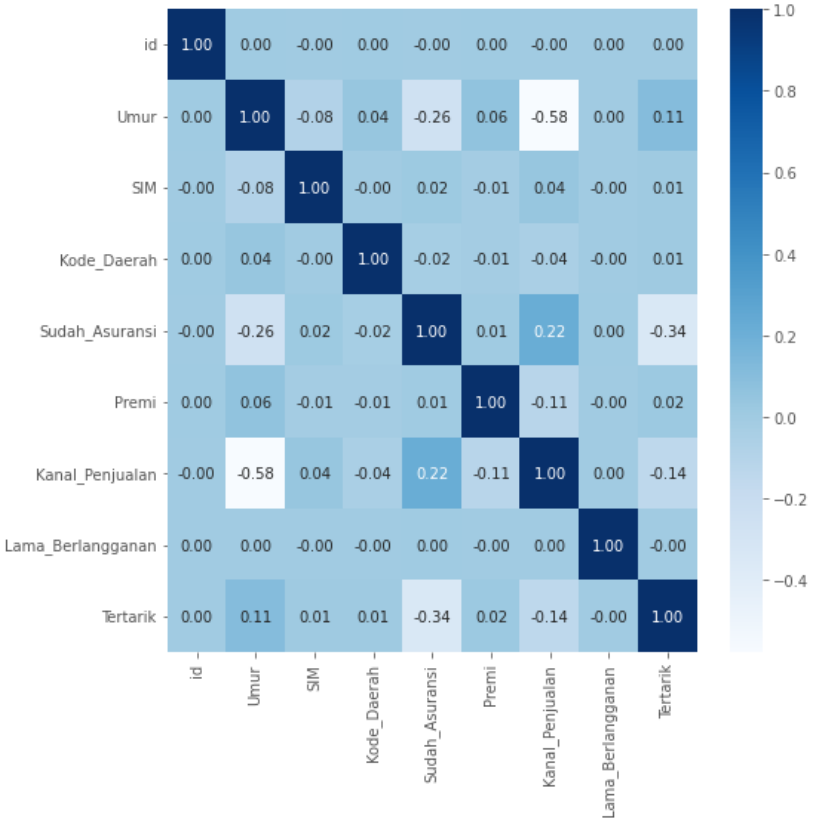
Untuk kolom SIM data imbalance sehingga tidak akan dipakai ke depannya, dan untuk Premi data skew (berat sebelah) serta skala angkanya terlalu besar. Selain itu, sudah cukup baik.

f. Visulasasi data kategorikal untuk melihat kemunculan



Banyak kategori hanya berkisar dari 2 sampai 3

g. Melihat korelasi



2.2. Eksplorasi Data Test

a. Mengetahui ukuran data test (kendaraan_test.csv), yaitu 47639x11

b. Mengetahui banyaknya missing value dan tipe data setiap kolom

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47639 entries, 0 to 47638
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Jenis_Kelamin     47639 non-null  object
 1   Umur              47639 non-null  int64
 2   SIM               47639 non-null  int64
 3   Kode_Daerah       47639 non-null  int64
 4   Sudah_Asuransi    47639 non-null  int64
 5   Umur_Kendaraan    47639 non-null  object
 6   Kendaraan_Rusak   47639 non-null  object
 7   Premi             47639 non-null  int64
 8   Kanal_Penjualan   47639 non-null  int64
 9   Lama_Berlangganan 47639 non-null  int64
 10  Tertarik          47639 non-null  int64
dtypes: int64(8), object(3)
memory usage: 4.0+ MB
```

Tidak ada missing value pada data test

2.3. Persiapan (Praproses) Data

a. Categorical Encoding

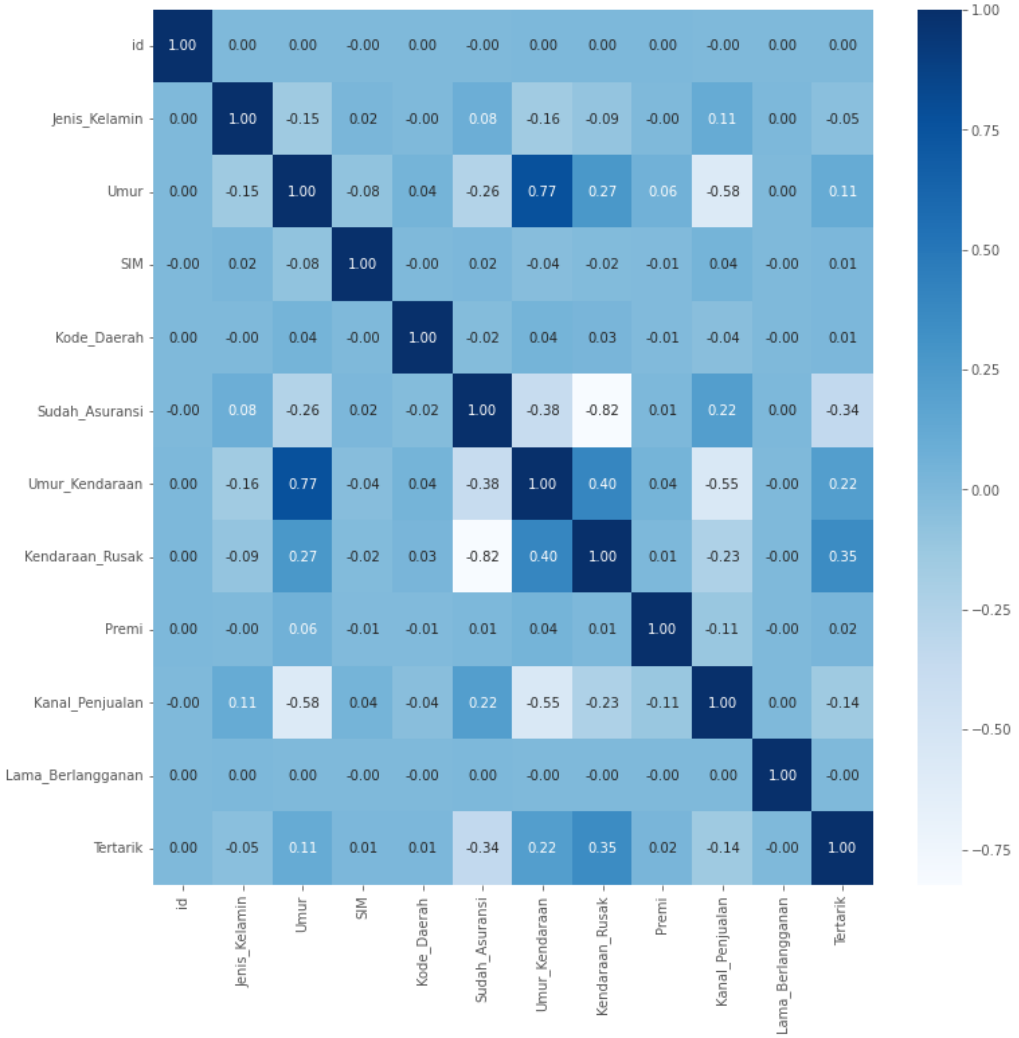
Mengubah 3 kolom tipe kategorikal menjadi numerikal, agar bisa dilakukan pembelajaran dan juga dapat dilihat korelasinya. Karena kategorinya tidak terlalu banyak (2 sampai 3 masing-masing), maka bisa dengan mendefinisikan angka untuk setiap kategori dengan menggunakan Label Encoding sebagai berikut:

Jenis_Kelamin : 0 untuk Pria dan 1 untuk Wanita
Kendaraan_Rusak : 0 untuk Tidak dan 1 untuk Pernah
Umur_Kendaraan : 0 untuk < 1 Tahun,
0.5 untuk 1 – 2 Tahun,
1 untuk > 2 Tahun

Sehingga tipe data setiap kolom menjadi seperti ini.

```
<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  float64
 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  float64
 7   Kendaraan_Rusak   271643 non-null  float64
 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(10), int64(2)
memory usage: 26.2 MB
```

b. Feature Selection



Menggunakan Korelasi Pearson, untuk melihat kolom (fitur) mana yang berpengaruh terhadap Tertarik. Dapat dilihat ada 5 kolom yang memiliki nilai korelasi $\geq |0.1|$ yaitu Umur, Sudah_Asuransi, Umur_Kendaraan, Kendaraan_Rusak, dan Kanal_Penjualan. Semua fitur diambil kecuali Kanal_Penjualan karena walaupun memiliki korelasi, tetapi tidak memberikan sebab-akibat karena Kanal_Penjualan hanyalah kode kontak pelanggan.

c. Handling Missing Value

Missing value di-handling dengan melakukan drop baris yang kosong nilainya pada 4 kolom yang sudah diseleksi di atas. Sehingga, jumlah data setelah dilakukan drop adalah 80.5%.

d. Undersampling

Karena data dengan nilai Tertarik 0 dan 1 memiliki perbedaan yang jauh, maka dilakukan undersampling, yaitu mengambil sebagian data-data bernilai 0 agar menyesuaikan dengan bernilai 1

Data dengan nilai Tertarik = 1 semuanya dipertahankan, sehingga mengambil sampel dari nilai yang besar (Tertarik = 0). Hasil akhir dari persentase tidak tertarik dengan tertarik menjadi 75% dan 25% masing-masing atau data yang tersisa adalah 113756.

e. Data Clean

Data yang sudah dilakukan praproses dan siap untuk dilakukan pembelajaran adalah sebagai berikut

	id	Umur	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Tertarik
0	1	30.0	1.0	0.0	0.0	0
7	8	23.0	1.0	0.0	0.0	0
8	9	20.0	1.0	0.0	0.0	0
9	10	54.0	0.0	1.0	1.0	1
14	15	66.0	1.0	0.5	0.0	0

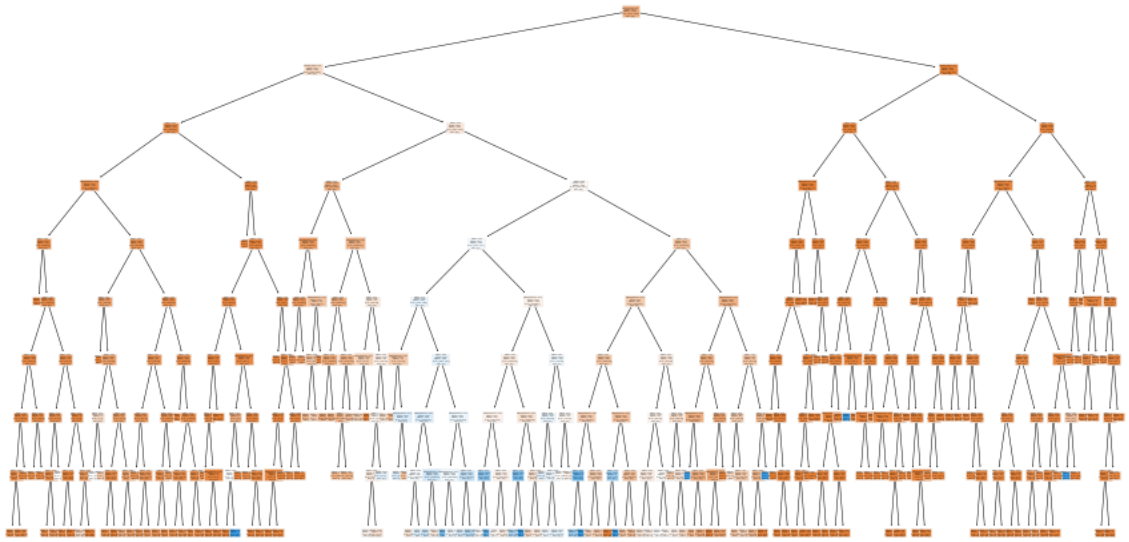
3. PEMODELAN

Pemodelan dilakukan dengan menggunakan Decision Tree karena Decision Tree efektif dalam melakukan pembelajaran dengan data kategorikal. Data-data yang didapatkan dari feature selection kebanyakan bernilai 2 atau 3 sehingga Decision Tree yang dihasilkan tidak akan terlalu rumit.

```
[64] X_train = model[['Umur', 'Sudah_Asuransi', 'Umur_Kendaraan', 'Kendaraan_Rusak']]
     y_train = model['Tertarik']

     pohon = tree.DecisionTreeClassifier(criterion="entropy")
     pohon = pohon.fit(X_train,y_train)
```

Untuk penghitungan nilai menggunakan Entropy, sehingga mencari fitur dengan nilai Information Gain yang besar sebagai root.



Kedalaman maksimal dari pohon tersebut adalah 9 yang ditandai dengan warna coklat memprediksi sebagai kelas 0 (tidak tertarik) dan warna biru sebagai kelas 1 (tertarik).

4. EVALUASI

Model yang dibangun dari data latih selanjutnya dilakukan validasi dengan data uji. Evaluasi dilakukan untuk menentukan model apakah model dapat melakukan klasifikasi dengan baik atau tidak dengan menggunakan berbagai metrik evaluasi.

```
[68] X_test = uji[['Umur', 'Sudah_Asuransi', 'Umur_Kendaraan', 'Kendaraan_Rusak']]
     y_test = uji['Tertarik']

     predik = pohon.predict(X_test)
```

Metriknya adalah akurasi, recall, precision, dan f1 score. Alasan pemilihan karena akurasi merupakan metrik yang paling umum dipakai untuk menilai kebenaran dalam melakukan klasifikasi. Sementara recall, precision, dan f1 score untuk meninjau kebenaran dari kelas positif. Kelas positifnya adalah nilai 1 pada Tertarik.

```
from sklearn import metrics
print("DecisionTrees's Accuracy : ", metrics.accuracy_score(y_test, predik))
print("DecisionTrees's Precision: ", metrics.precision_score(y_test, predik))
print("DecisionTrees's Recall   : ", metrics.recall_score(y_test, predik))
print("DecisionTrees's F1 Score : ", metrics.f1_score(y_test, predik))
```

Lebih lanjut, digunakan juga confusion matrix agar dapat diamati secara visualisasi dari hasil klasifikasi yang dilakukan.

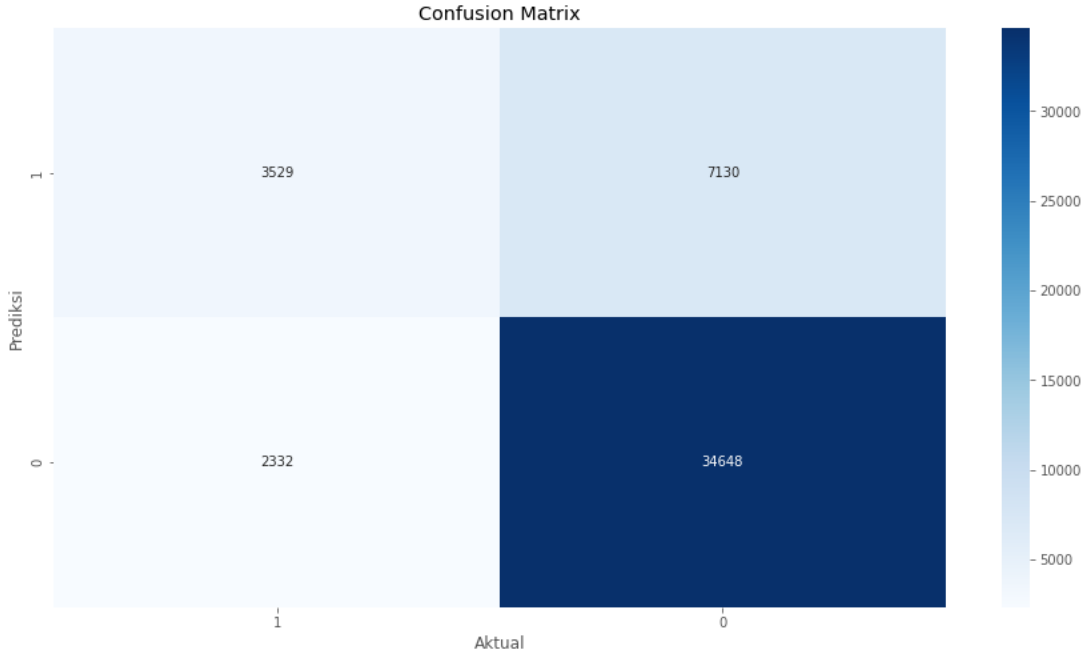
```
cf_matrix = metrics.confusion_matrix(y_test, predik, labels=[1,0]).transpose()

ax = plt.subplot()
sns.heatmap(cf_matrix, annot=True, fmt='g', cmap='Blues', ax=ax)
ax.set_xlabel('Aktual')
ax.set_ylabel('Prediksi')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['1', '0'])
ax.yaxis.set_ticklabels(['1', '0'])
```

Hasil yang didapatkan adalah sebagai berikut.

Metrik	Nilai
Akurasi	80,1%
Precision	33,1%
Recall	60,2%
F1 Score	41,5%

Dapat juga diamati confusion matrix sebagai berikut



5. EKSPERIMEN

5.1. Hyperparameter Tuning

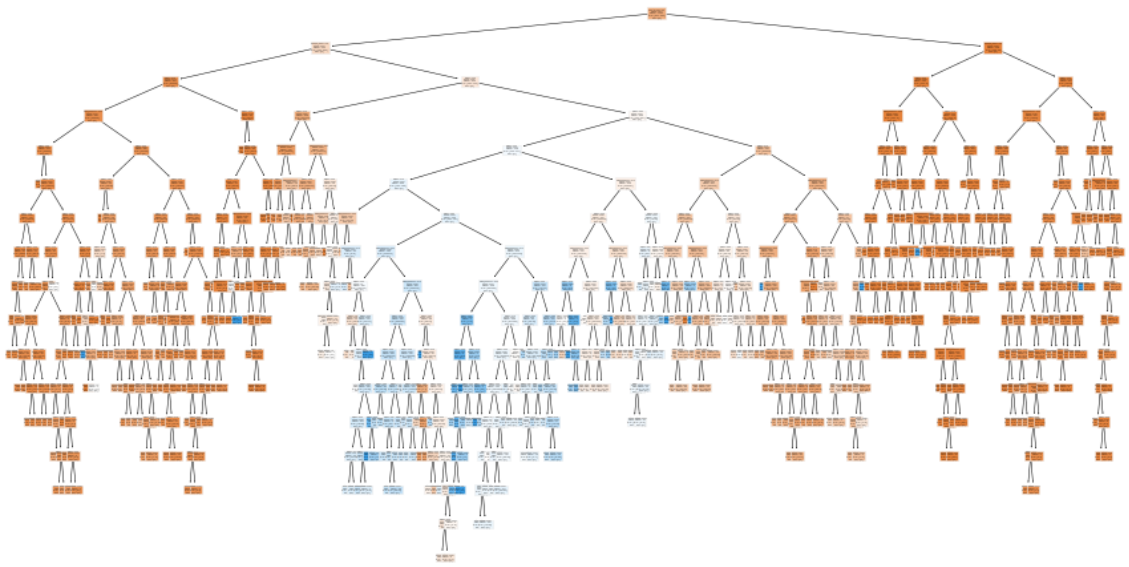
Pada setiap eksperimen, dilakukan hyperparameter tuning untuk mencari nilai-nilai parameter yang optimal pada model sehingga menghasilkan prediksi yang lebih akurat.

Pencarian parameter optimal dilakukan dengan menggunakan Grid Search Cross-Validation (Grid Search CV). Nilai parameter yang digunakan dapat dilihat pada tabel berikut.

```
[339] dt = tree.DecisionTreeClassifier(criterion='entropy')
      parameters = {
              'max_depth': [3, 4, 5, 6, 7, 8, 9]
              }

[340] search = GridSearchCV(dt,
                           parameters,
                           scoring='accuracy',
                           cv = 5,
                           verbose=3)
      search.fit(X_train, y_train)
```

Parameter yang di-tuning adalah max_depth atau kedalaman pohon maksimal. Karena pada percobaan tanpa melakukan tuning pada max_depth, maka pohon memiliki kedalaman maksimal 16 dengan gambar sebagai berikut.

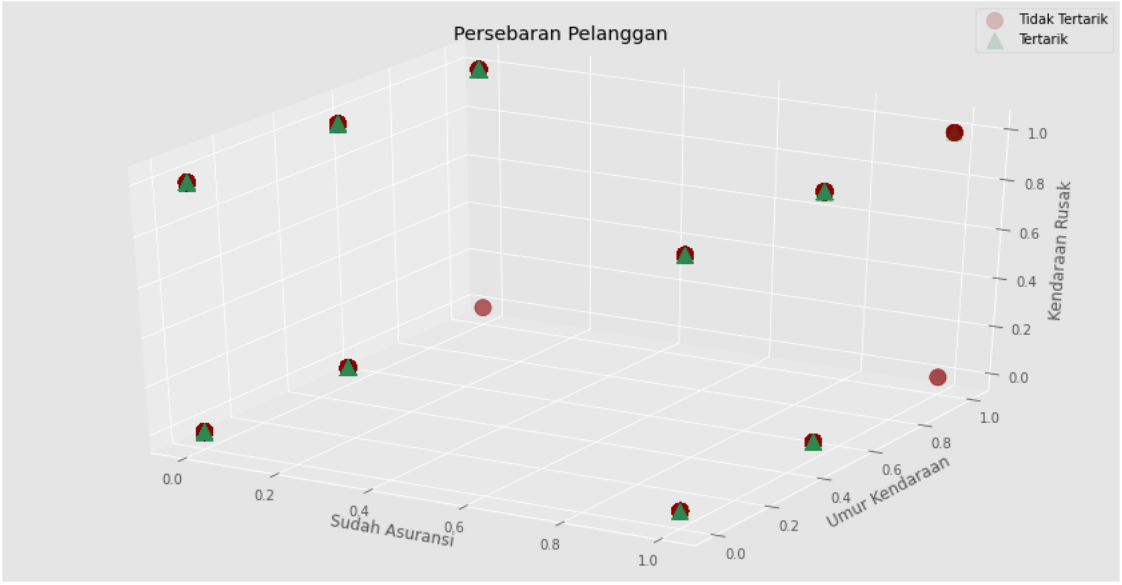


Pohon dengan kedalaman yang tinggi cenderung mengalami *overfit* atau model terlalu spesifik untuk data train, tetapi kurang bisa memprediksi untuk data test. Hasil klasifikasi dengan pohon tersebut adalah sebagai berikut

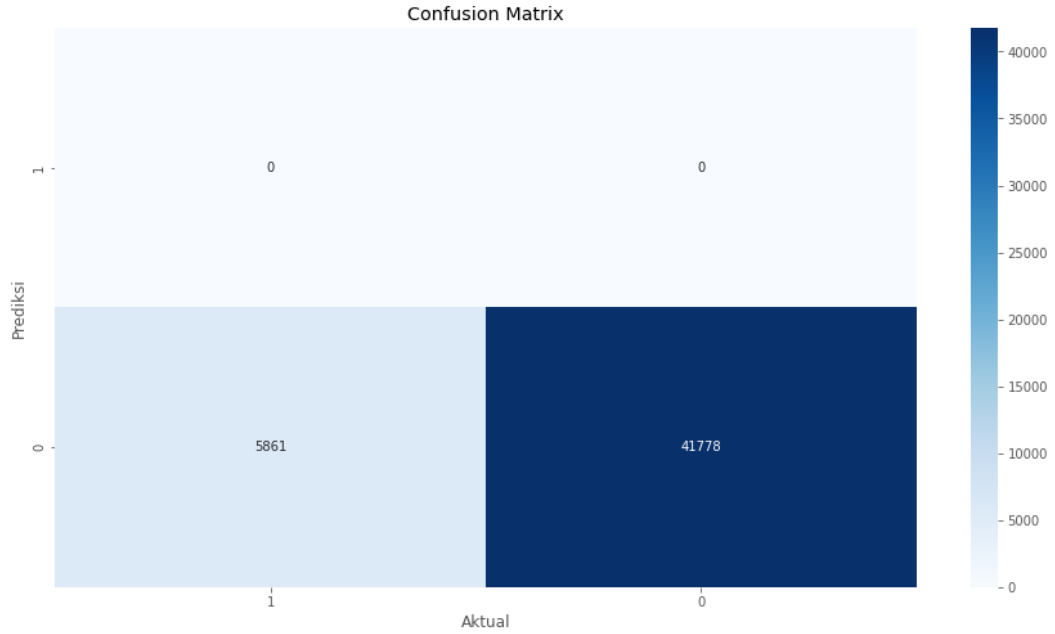
Metrik	Nilai
Akurasi	81%
Precision	33,4%
Recall	54,8%
F1 Score	41,5%

5.2. Seleksi menggunakan 3 fitur

Sebelumnya dilakukan seleksi fitur dengan memilih 3 fitur karena memiliki nilai korelasi lebih tinggi lagi, yaitu > | 0.2 |, tetapi dihasilkan persebaran data sebagai berikut.



Sudah_Asuransi, Umur_Kendaraan, dan Kendaraan_Rusak masing-masing hanya memiliki 2, 3, 2 sehingga menghasilkan 2x3x2=12 titik saja. Setiap titik memiliki nilai ketiga fitur yang sama, tetapi menghasilkan nilai Tertarik yang berbeda. Jika dilanjutkan melakukan pembelajaran, maka didapatkan hasil berikut akurasi 87,76%, tetapi untuk precision, recall, dan f1 score semuanya 0%. Dapat dilihat juga dengan *confusion matrix* sebagai berikut.



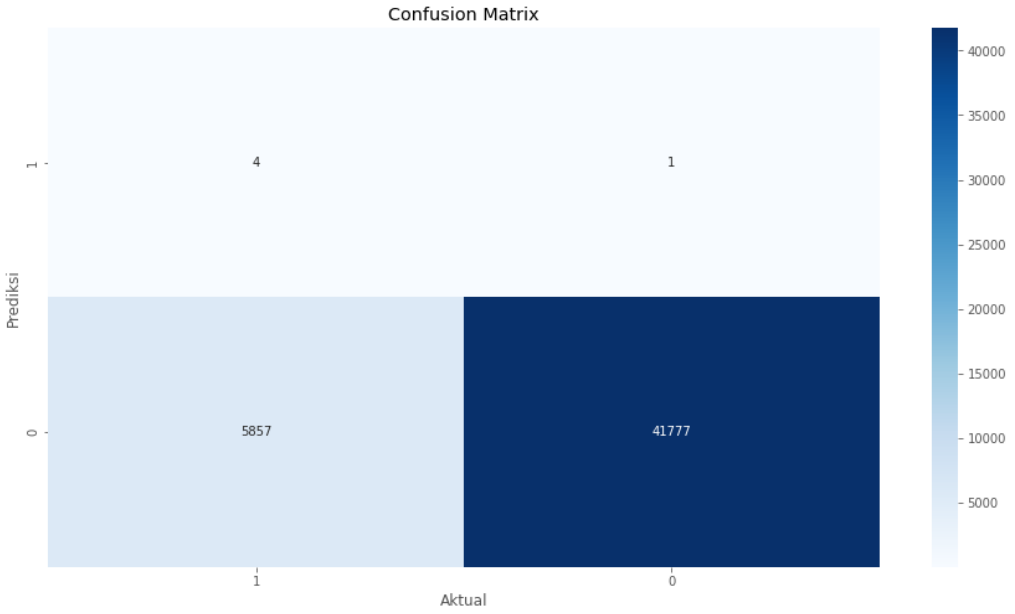
Model sama sekali tidak dapat memprediksi Tertarik = 1.

5.3. Mengatur Undersampling

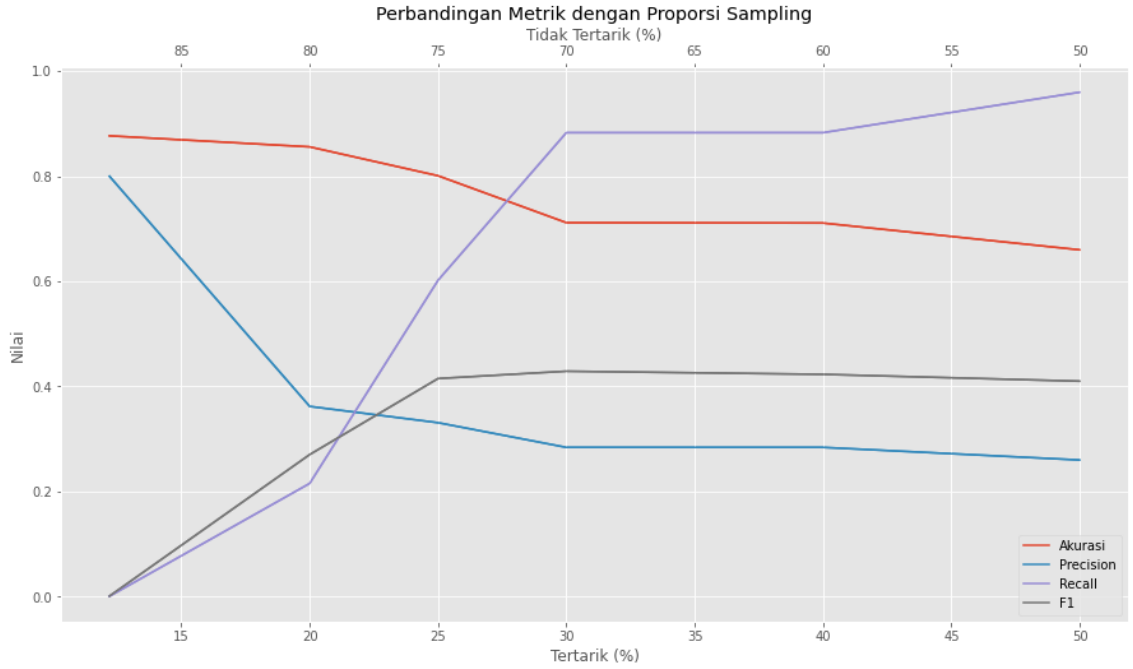
Karena data memiliki perbedaan antara tertarik dan tidak tertarik yang cukup jauh (12,8% dengan 87,2%) maka dilakukan sampling. Jika dilanjutkan melakukan pembelajaran dengan data tersebut, maka dihasilkan pengujian sebagai berikut.

Metrik	Nilai
Akurasi	87,77%
Precision	80%
Recall	0,06%
F1 Score	0,1%

Dapat juga dilihat confusion matrix sebagai berikut.



Model kurang baik dalam memprediksi tertarik karena data terlalu banyak memiliki Tertarik = 0, sehingga model cenderung menghasilkan klasifikasi bernilai 0. Untuk itu dilakukan sampling dengan teknik undersampling, yaitu tetap mempertahankan semua data dengan nilai Tertarik = 1, tetapi mengambil sebagian sampel dari data kelas dominan (Tertarik = 0). Hasil yang optimal adalah dengan menggunakan proporsi 75% tidak tertarik dan 25% tertarik. Hasil pengujian juga menggunakan bermacam-macam nilai proporsi yang dapat dilihat pada grafik berikut.



Dari ketiga eksperimen yang dilakukan, dapat dilihat keseluruhan hasil eksperimen pada tabel berikut

Jenis Eksperimen		Akurasi	Precision	Recall	F1 Score
Banyak Fitur	Sampling				
3	Tanpa sampling	87,76%	0%	0%	0%
4	Tanpa sampling	**87,77%**	**80%**	0,06%	0,1%
4	20% dan 80%	85,6%	36,2%	21,54%	27,03%
4	25% dan 75%	80,1%	33,1%	60,2%	41,5%
4	25% dan 75% (tanpa tuning)	81%	33,4%	54,8%	41,5%
4	30% dan 70%	71,17%	28,4%	88,3%	**42,9%**
4	40% dan 60%	71,11%	28,4%	88,3%	42,3%
4	50% dan 50%	66%	26%	**96%**	41%

6. PRESENTASI

https://youtu.be/skGCBnNzlwM

7. KESIMPULAN

Algoritma Decision Tree terbukti baik dalam melakukan klasifikasi dan didapatkan hasil akurasi, precision, recall, dan f1 score masing-masing 80,1%, 33,1%, 60,2%, dan 41,5%. Untuk mendapatkan hasil yang optimal, perlu dilakukan tuning baik dari parameter yang digunakan, maupun jumlah sampling. Hyperparameter tuning dilakukan untuk menghindari overfit dan sampling dilakukan agar hasil pembelajaran tidak hanya cenderung menghasilkan klasifikasi menjadi kelas dominan.