LAPORAN TUGAS SUPERVISED MACHINE LEARNING

Rizki Syafaat Amardita (1301184032)
Vijay Cheza Pangestu (1301180351)
IF 42-12

Pembelajaran Mesin

Formulasi Masalah

Diberikan sebuah file excel kendaraan_test.xls dan kendaraan_train.xls yang berisi data pelanggan yang memiliki kendaraan dengan atribut yaitu atribut ID, jenis kelamin, umur, SIM, kode_daerah, Sudah_Asuransi, Umur_kendaraan, Kendaraan_rusak, Premi, Kanal_Penjualan, lama_berlangganan, Tertarik. Yang akan diproses menggunakan komparasi dari berbagai metode algoritma supervised untuk melakukan klasifikasi.

Persiapan dan Eksplorasi Data

```
#menyiapkan dataset
kendaraan_train_df = pd.read_csv("kendaraan_train.csv")
kendaraan_test_df = pd.read_csv("kendaraan_test.csv")

#mendrop atribut ID pada dataset train, karena pada dataset test tidak ada atribut ID
kendaraan_train_df.drop('id', axis=1, inplace=True)
```

```
#Menghitung jumlah data yang bernilai null pada setiap kolom dataset train
kendaraan_train_df.isnull().sum()
```

```
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
```

Sebelum data diproses kita perlu mendrop kolom ”ID” yang nantinya tidak akan digunakan pada pemrosesan agar mempermudah klasifikasi, dan juga menganalisa apakah ada data yang bertipe null, jika terdapat data yang bertipe null maka harus dilakukannya preprocessing agar data tersebut valid untuk diproses, setelah semua data null terisi.

Mengatasi Missing Value

```
# UNTUK DATASET TRAIN
#menghindari missing values
#mengisi nilai null dari atribut numeric dengan nilai median dari atribut tersebut

kendaraan_train_df['Umur'] = kendaraan_train_df['Umur'].fillna(kendaraan_train_df['Umur'].median())
kendaraan_train_df['SIM'] = kendaraan_train_df['SIM'].fillna(kendaraan_train_df['SIM'].median())
kendaraan_train_df['Kode_Daerah'] = kendaraan_train_df['Kode_Daerah'].fillna(kendaraan_train_df['Kode_Daerah'].median())
kendaraan_train_df['Sudah_Asuransi'] = kendaraan_train_df['Sudah_Asuransi'].fillna(kendaraan_train_df['Sudah_Asuransi'].median())
kendaraan_train_df['Premi'] = kendaraan_train_df['Premi'].fillna(kendaraan_train_df['Premi'].median())
kendaraan_train_df['Kanal_Penjualan'] = kendaraan_train_df['Kanal_Penjualan'].fillna(kendaraan_train_df['Kanal_Penjualan'].median())
kendaraan_train_df['Lama_Berlangganan'] = kendaraan_train_df['Lama_Berlangganan'].fillna(kendaraan_train_df['Lama_Berlangganan'].median())
kendaraan_train_df['Tertarik'] = kendaraan_train_df['Tertarik'].fillna(kendaraan_train_df['Tertarik'].median())

#mengisi nilai null1 dari atribut categhorical dengan nilai modus dari atribut tersebut
temp_jenis_kelamin = kendaraan_train_df['Jenis_Kelamin'].mode()
kendaraan_train_df['Jenis_Kelamin'].fillna(temp_jenis_kelamin[0], inplace=True)

temp_Umur_kendaraan = kendaraan_train_df['Umur_Kendaraan'].mode()
kendaraan_train_df['Umur_Kendaraan'].fillna(temp_Umur_kendaraan[0], inplace=True)

temp_kendaraan_Rusak = kendaraan_train_df['Kendaraan_Rusak'].mode()
kendaraan_train_df['Kendaraan_Rusak'].fillna(temp_kendaraan_Rusak[0], inplace=True)
```

Disini dilakukannya proses replace data null, untuk data numerikal diisi dengan nilai median dari atribut itu sendiri, dan untuk data kategorikal diisi dengan nilai modus dari atribut itu sendiri.

Mengecek Tipe Data

```
#mengecek data untuk melihat tipe object
kendaraan_train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285831 entries, 0 to 285830
Data columns (total 11 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   Jenis_Kelamin       285831 non-null  object
 1   Umur                285831 non-null  float64
 2   SIM                 285831 non-null  float64
 3   Kode_Daerah         285831 non-null  float64
 4   Sudah_Asuransi      285831 non-null  float64
 5   Umur_Kendaraan      285831 non-null  object
 6   Kendaraan_Rusak     285831 non-null  object
 7   Premi               285831 non-null  float64
 8   Kanal_Penjualan     285831 non-null  float64
 9   Lama_Berlangganan   285831 non-null  float64
 10  Tertarik            285831 non-null  int64
dtypes: float64(7), int64(1), object(3)
memory usage: 24.0+ MB
```

Sebelum klasifikasi dilakukan, maka perlu dilakukannya encode data yang artinya pengubahan tipe data yang asalnya bertipe kategorikal menjadi tipe data numerikal, untuk itu perlu dicek mana saja data yang bertipe object/kategorikal.

Encode Data

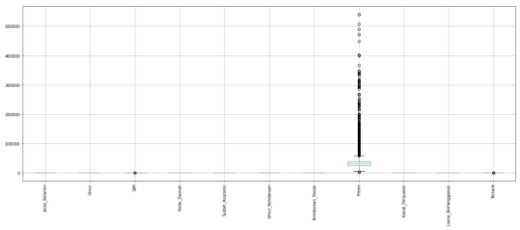
```
#menggunakan library label encoder, agar mempermudah proses clustering
#meng encode label categori menjadi numeric
encoder = LabelEncoder()

kendaraan_train_df['Jenis_Kelamin'] = encoder.fit_transform(kendaraan_train_df['Jenis_Kelamin'])
kendaraan_train_df['Umur_Kendaraan'] = encoder.fit_transform(kendaraan_train_df['Umur_Kendaraan'])
kendaraan_train_df['Kendaraan_Rusak'] = encoder.fit_transform(kendaraan_train_df['Kendaraan_Rusak'])

kendaraan_test_df['Jenis_Kelamin'] = encoder.fit_transform(kendaraan_test_df['Jenis_Kelamin'])
kendaraan_test_df['Umur_Kendaraan'] = encoder.fit_transform(kendaraan_test_df['Umur_Kendaraan'])
kendaraan_test_df['Kendaraan_Rusak'] = encoder.fit_transform(kendaraan_test_df['Kendaraan_Rusak'])
```

Selanjutnya mengubah tipe data yang non numerik menjadi numerik (untuk mengkonversi label kata menjadi angka) label encode data mengacu pada proses transformasi label kata menjadi bentuk numerik.

Menghandle Outlier



```
kendaraan_test_df['Premi'].loc[kendaraan_test_df['Premi'] <= 16263.6] = 16263.6
kendaraan_test_df['Premi'].loc[kendaraan_test_df['Premi'] >= 59211.75] = 59211.75

kendaraan_train_df['Premi'].loc[kendaraan_train_df['Premi'] <= 16619.3] = 16619.3
kendaraan_train_df['Premi'].loc[kendaraan_train_df['Premi'] >= 43267.5] = 43267.5
```

Outlier merupakan data yang nilainya jauh berbeda dengan nilai dari data yang lain pada suatu kelompok. Untuk melakukan deteksi dari outlier, dapat dilakukan dengan menggunakan boxplot.

Handling outliers dilakukan dengan cara mereplace data yang berada di atas nilai maksimum dan berada di bawah nilai minimum. Pada data nilai outlier diatas maximum maka diganti dengan data nilai maximum (premi 75% x 1,5). pada data nilai outlier dibawah minimum maka diganti dengan data nilai minimum (premi 25% / 1.5).

Normalisasi Data

```
kendaraan_train_normalisasi = kendaraan_train_df.loc[:,["Umur", "Kode_Daerah", "Umur_Kendaraan",
                                                        "Premi", "Kanal_Penjualan", "Lama_Berlangganan"]].apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
```

```
.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
```

Normalisation

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

kendaraan_train_normalisasi

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	1	0.153846	1.0	0.634615	1.0	0.5	1	0.428160	0.932099	0.301038	0
1	0	0.430769	1.0	0.750000	0.0	1.0	0	0.344515	0.172840	0.512111	0
2	0	0.015385	1.0	0.884615	1.0	0.5	1	0.604682	0.981481	0.377163	0
3	1	0.584615	1.0	0.923077	0.0	0.0	1	0.000000	0.759259	0.183391	0
4	0	0.461538	1.0	0.673077	0.0	1.0	0	0.684388	0.537037	0.636678	0

Disini dilakukan normalisasi data yang dimaksud adalah untuk memodifikasi nilai dalam atribut sehingga dapat mengukur dalam skala umum. Normalisasi data menggunakan metode min-max.

Pemodelan

```
#atribut biasa atau selain kelas target
x_train = kendaraan_train_normalisasi[["Jenis_Kelamin", "Umur", "SIM"
x_test = kendaraan_test_normalisasi[["Jenis_Kelamin", "Umur", "SIM",

#atribut label atau kelas target
y_train = kendaraan_train_normalisasi.loc[:,["Tertarik"]]
y_test = kendaraan_test_normalisasi.loc[:,["Tertarik"]]
```

Variabel x berisi atribut selain kelas target, dan variabel y berisi atribut kelas target.

```
#import library KNN dari package scikit-learn
from sklearn.neighbors import KNeighborsClassifier

#menginisialisasi jumlah data terdekat dan memanggil fungsi klasifikasi KNN
knn = KNeighborsClassifier(n_neighbors=7)

#Memasukkan data training pada fungsi klasifikasi KNN
knn.fit(x_train, y_train)

#Menentukan hasil prediksi dari x_test yang disebut dengan y prediksi
y_pred = knn.predict(x_test)
```

```
K NEIREST NEIGHBOUR

F1-SCORE   0.5565516147523906
ACCURACY   0.8616049875102332
PRECISION  0.6131062475595782
RECALL 0.5474924558991785
```

	precision	recall	f1-score	support
0	0.89	0.96	0.92	41778
1	0.34	0.13	0.19	5861
accuracy			0.86	47639
macro avg	0.61	0.55	0.56	47639
weighted avg	0.82	0.86	0.83	47639

```
from sklearn.naive_bayes import GaussianNB

#mengimport fungsi GaussianNB dari package scikit-learn.
naive = GaussianNB()

#Memasukkan data training pada fungsi klasifikasi Naive Baiyes
naive.fit(x_train, y_train)

#Menentukan hasil prediksi dari x_test yang disebut dengan y prediksi NB
y_pred_NB = naive.predict(x_test)
```

```
NAIVE BAYES

F1-SCORE   0.5710898974822878
ACCURACY   0.6395600243497974
PRECISION   0.6227893513996916
RECALL 0.7839359057542794
```

	precision	recall	f1-score	support
0	0.99	0.59	0.74	41778
1	0.25	0.98	0.40	5861
accuracy			0.64	47639
macro avg	0.62	0.78	0.57	47639
weighted avg	0.90	0.64	0.70	47639

```
from sklearn.ensemble import RandomForestClassifier

#Model
r_forest=RandomForestClassifier(n_estimators=100)

#Memasukkan data training pada fungsi klasifikasi Random Forest
r_forest.fit(x_train,y_train)

#Menentukan hasil prediksi dari x_test yang disebut dengan y prediksi RF
y_pred_RF = r_forest.predict(x_test)
```

```
RANDOM FOREST

F1-SCORE   0.546362548235812
ACCURACY   0.8661810701316148
PRECISION   0.6207185605037415
RECALL 0.5402003614640605
```

	precision	recall	f1-score	support
0	0.89	0.97	0.93	41778
1	0.36	0.11	0.17	5861
accuracy			0.87	47639
macro avg	0.62	0.54	0.55	47639
weighted avg	0.82	0.87	0.83	47639

```
from sklearn.tree import DecisionTreeClassifier

#Model
d_tree = DecisionTreeClassifier(random_state=0)

#Memasukkan data training pada fungsi klasifikasi Decision Tree
d_tree.fit(x_train, y_train)

#Menentukan hasil prediksi dari x_test yang disebut dengan y prediksi DT
y_pred_DT = d_tree.predict(x_test)
```

```
DECISION TREE

F1-SCORE   0.5933578372755066
ACCURACY   0.8218476458363946
PRECISION   0.5920151646426735
RECALL 0.5947929599266535
```

	precision	recall	f1-score	support
0	0.90	0.90	0.90	41778
1	0.28	0.29	0.29	5861
accuracy			0.82	47639
macro avg	0.59	0.59	0.59	47639
weighted avg	0.82	0.82	0.82	47639

Evaluasi

Eksperimen 1 yaitu, menggunakan metode algoritma K-nearest neighbour dan mendapatkan hasil sebagai berikut:

Akurasi: 0.8616049875102332 (86%)

Presisi: 0.6131062475595782 (61%)

Eksperimen 2 yaitu, menggunakan metode algoritma Naive Bayes dan mendapatkan hasil sebagai berikut:

Akurasi: 0.6395600243497974 (64%)

Presisi: 0.6227893513996916 (62%)

Eksperimen 3 yaitu, menggunakan metode algoritma Random Forest dan mendapatkan hasil sebagai berikut:

Akurasi: 0.8661810701316148 (87%)

Presisi: 0.6207185605037415 (62%)

Eksperimen 4 yaitu, menggunakan metode algoritma Decision Tree dan mendapatkan hasil sebagai berikut:

Akurasi: 0.8218476458363946 (82%)

Presisi: 0.5920151646426735 (59%)

Evaluasi

Kesimpulannya adalah pada pemrosesan klasifikasi dataset kendaraan, metode yang paling optimal dalam melakukan klasifikasi pada dataset kendaraan adalah metode **random forest.** Dengan peroleh akurasi 87%, presisi sebesar 62%.