LAPORAN TUGAS BESAR MK PEMBELAJARAN MESIN
TAHAP 2 : CLASSIFICATION

Oleh :

Hilman Bayu Aji - 1301180397
Ramawaldi Putra - 1301164506

IFX-44-GAB

S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2021/2022

Daftar Isi

1. Formulasi Masalah dan Dataset

Pada Tugas Besar Tahap 2 ini, kami diberikan dataset kendaraan yang akan diselesaikan dengan *classification (supervised learning)*, dalam kolom tersebut. Tugas *classification (supervised learning)* adalah memprediksi apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak berdasarkan data pelanggan di dealer. Kemudian, klasifikasi dengan membandingkan nilai korelasi yang berbeda.

```
df = pd.read_csv('https://raw.githubusercontent.com/hilmanpbayu/tubes-malin-tahap-2-classification/main/Dataset/kendaraan_train.csv')

#Mengubah None menjadi NaN
df.replace("", np.NaN, inplace = True)
df[df.isna().any(axis = 1)]
df
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
285826	285827	Wanita	23.0	1.0	4.0	1.0	< 1 Tahun	Tidak	25988.0	152.0	217.0	0
285827	285828	Wanita	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	44686.0	152.0	50.0	0
285828	285829	Wanita	23.0	1.0	50.0	1.0	< 1 Tahun	Tidak	49751.0	152.0	226.0	0
285829	285830	Pria	68.0	1.0	7.0	1.0	1-2 Tahun	Tidak	30503.0	124.0	270.0	0
285830	285831	Pria	45.0	1.0	28.0	0.0	1-2 Tahun	Pernah	36480.0	26.0	44.0	0

285831 rows × 12 columns

Figure 1. Dataset kendaraan_train

Dataset yang digunakan adalah data kendaraan yang memiliki 285831 *rows* dan 12 *columns* pada file kendaraan_train. Dataset tersebut akan digunakan untuk mengetahui *classification* yang terdapat pada dataset train tersebut dan kemudian digunakan untuk memprediksi dataset test pelanggan tertarik untuk membeli kendaraan baru atau tidak.

2. Eksplorasi dan Persiapan Data

2.1 Eksplorasi

```
#Melihat tipe data
df.dtypes
```

```
id                    int64
Jenis_Kelamin        object
Umur                float64
SIM                 float64
Kode_Daerah         float64
Sudah_Asuransi      float64
Umur_Kendaraan       object
Kendaraan_Rusak      object
Premi               float64
Kanal_Penjualan     float64
Lama_Berlangganan   float64
Tertarik              int64
dtype: object
```

Figure 2. Tipe data kendaraan_train

Dataset tersebut memiliki nilai atribut yang bervariasi, untuk tipe attribute Nominal terdapat pada kolom ‘Jenis_Kelamin’, ‘SIM’, ‘Kode_Daerah’,

‘Kanal_Penjualan’, ‘Sudah_Asuransi’, ‘Kendaraan_Rusak’, ‘Tertarik’. Untuk tipe atribut Ordinal terdapat pada kolom ‘Umur_Kendaraan’. Untuk tipe data Interval terdapat pada kolom ‘Umur’, ‘Lama_Berlangganan’, dan untuk tipe atribut Rasio terdapat pada kolom ‘Premi’. Perbedaan tipe atribut tersebut berfungsi untuk menentukan properti matematika yang tepat untuk proses perhitungan.

Kemudian, mengecek *noise* atau kesalahan input pada atribut Nominal pada kolom ‘Jenis_Kelamin’, ‘Umur_Kendaraan’, dan ‘Kendaraan_Rusak’ dan didapat bahwa kolom tersebut tidak mengandung nilai yang *noise*.

```
#Cek data noise/outlier pada data object/categorical
print(df['Jenis_Kelamin'].value_counts(),'\n', df['Jenis_Kelamin'].isnull().value_counts())
print('----------------------')
print(df['Umur_Kendaraan'].value_counts(),'\n', df['Umur_Kendaraan'].isnull().value_counts())
print('----------------------')
print(df['Kendaraan_Rusak'].value_counts(),'\n', df['Kendaraan_Rusak'].isnull().value_counts())
```

```
Pria      146678
Wanita    124713
Name: Jenis_Kelamin, dtype: int64
 False    271391
True      14440
Name: Jenis_Kelamin, dtype: int64
----------------------
1-2 Tahun    142761
< 1 Tahun    117378
> 2 Tahun     11417
Name: Umur_Kendaraan, dtype: int64
 False    271556
True      14275
Name: Umur_Kendaraan, dtype: int64
----------------------
Pernah    137123
Tidak     134520
Name: Kendaraan_Rusak, dtype: int64
 False    271643
True      14188
Name: Kendaraan_Rusak, dtype: int64
```

Figure 3. Mengecek noise pada kolom tertentu

Pada Dataset yang digunakan terdapat *missing value* sebanyak 142 916, tetapi tidak terdapat duplikasi pada dataset karena terdapat kolom ‘id’ sebagai identitas pelanggan. Berikut adalah nilai *missing value* pada dataset di masing-masing kolom dan data duplikasi cek:

```
# Cleaning data
# Mengecek nilai null
print(df.isna().sum(),'\n')

#Total missing value
print('Total NaN = ',df.isna().sum().sum())

#Data duplicate
print('Data Duplicate = ',df.duplicated().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

Total NaN =  142916
Data Duplicate =  0
```

Figure 4. Missing value dan data duplikat

Dataset tersebut memiliki nilai *outlier* pada kolom ‘Premi’, kolom premi adalah data yang menginformasikan jumlah premi yang harus dibayar pertahun oleh pelanggan. Berikut adalah visualisasi menggunakan plot-box untuk melihat nilai *outlier*:

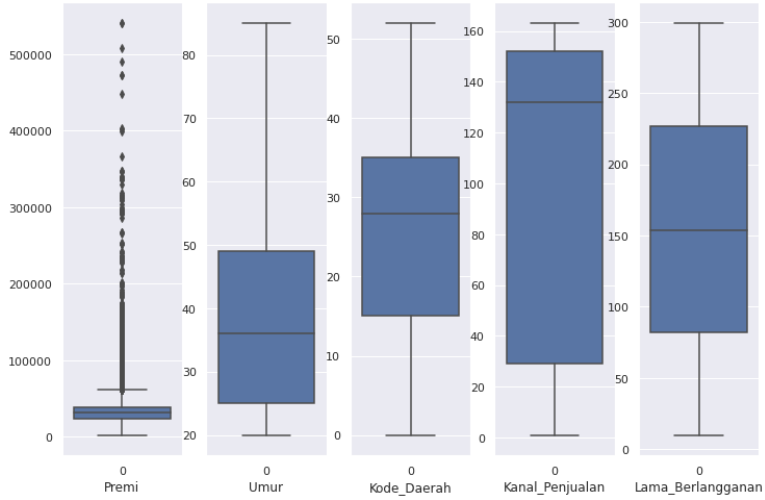


Figure 5. Outlier

2.2 Persiapan Data

Pada hasil eksplorasi data dapat dilihat bahwa terdapat gangguan pada dataset yang digunakan yaitu *missing values* dan *outlier*. Untuk menangani masalah *missing value* pada dataset menggunakan nilai modus(mode). Kemudian, untuk kolom ‘Premi’ dibiarkan karena data tersebut menginformasikan jumlah yang harus dibayar pertahun oleh pelanggan, dimana data tersebut penting untuk menjawab formulasi permasalahan yang dibahas sebelumnya.

```
#Missing value handler with modus(mode)
for col in df.columns:
  df[col].fillna(df[col].mode()[0], inplace=True)
```

Figure 6. Missing value handling dengan modus

Untuk kolom ‘Jenis_Kelamin’, ‘Umur_Kendaraan’, dan ‘Kendaraan_Rusak’ akan ditransformasikan nilainya menjadi Numerik agar pada proses selanjutnya dapat diperhitungkan. Kolom tersebut dianggap penting untuk menjawab formulasi permasalahan yang dibahas sebelumnya.

```
#Transform
#[Nominal to Numeric] Jenis_Kelamin 0 : Wanita | 1 : Pria
df.replace({"Wanita": 0, "Pria": 1}, inplace=True)
df_test.replace({"Wanita": 0, "Pria": 1}, inplace=True)

#[Nominal to Numeric] Umur_Kendaraan --> 1 : 1-2 Tahun | 0 : < 1 Tahun | 2 : > 2 Tahun
df.replace({"< 1 Tahun": 0, "1-2 Tahun": 1, "> 2 Tahun": 2}, inplace=True)
df_test.replace({"< 1 Tahun": 0, "1-2 Tahun": 1, "> 2 Tahun": 2}, inplace=True)

#[Nominal to Numeric] Kendaraan_Rusak --> 0 : Tidak | 1 : Pernah
df.replace({"Tidak": 0, "Pernah": 1}, inplace=True)
df_test.replace({"Tidak": 0, "Pernah": 1}, inplace=True)
```

Figure 7. Transformasi data

Untuk kolom ‘id’ akan di drop karena kolom tersebut hanya identitas dari pelanggan. Kemudian, melakukan handling pada kolom yang memiliki nilai *outlier* pada kolom ‘Premi’. Meskipun outlier menambah variasi dalam data, *outlier* ini dapat menurunkan kekuatan statistik. Sehingga *outlier* perlu dihapus agar hasil dari data menjadi signifikan secara statistik. *Outlier* ini dihapus dengan cara *remove*.

```
#Mendeteksi outlier menggunakan Inter Quantile Range(IQR)
def ROutlier(df,column):
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    interq = q3-q1
    batasBawah = q1-1.5*interq
    batasAtas = q3+1.5*interq
    df_out =  df.loc[(df[column] > batasBawah) & (df[column] < batasAtas)]
    return df_out

#Remove outlier 2x (1 remove handling, 1 recheck)
df = ROutlier(df,'Premi')
df = ROutlier(df,'Premi')
```

Figure 8. Remove outlier

Setelah berhasil, tampilan dari kolom “Premi” akan berubah yaitu sudah tidak ada lagi *outlier* di dalamnya.

```
#Outlier check
sns.boxplot(df['Premi'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f4
```

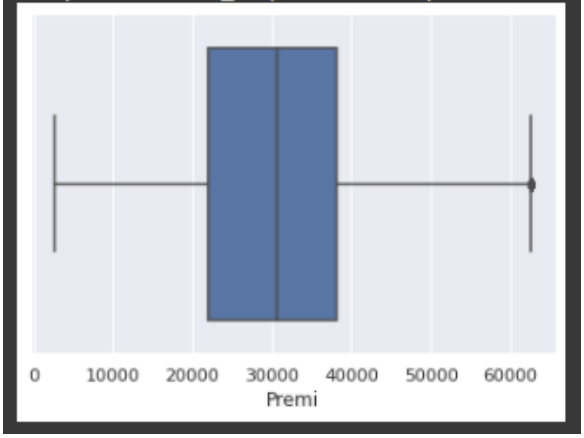
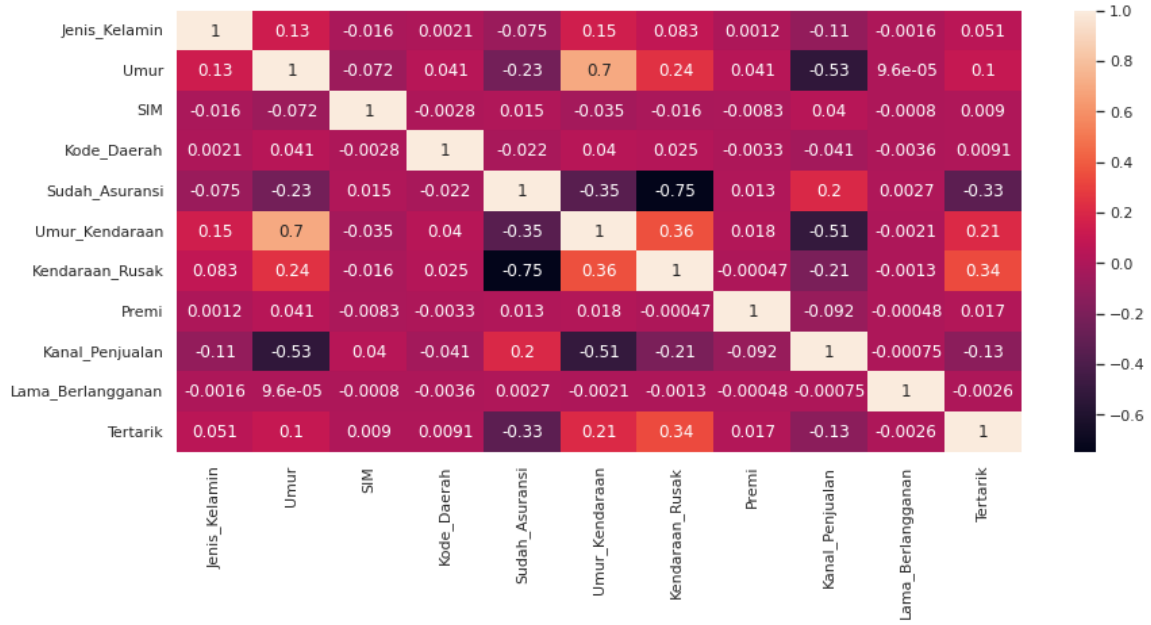


Figure 9. Kolom “Premi” tanpa outlier

3. Pemodelan

Pembangunan model menggunakan metode *k-NN Classification* dengan atribut yang dipilih berdasarkan korelasi kolom x dengan kolom ‘Tertarik’. Pemilihan atribut dilakukan karena atribut dalam dataset belum diketahui akan memberikan hasil yang signifikan. Menambahkan atribut yang tidak relevan ke dalam model akan berdampak pada model yang dihasilkan buruk dan proses pembuatan model menjadi lambat.

Pemilihan atribut dengan melihat heatmap korelasi dengan ketentuan nilai korelasi kolom x dengan kolom ‘Tertarik’ dengan nilai > 0.1. Berdasarkan hasil heatmap, kolom yang akan digunakan adalah ‘Sudah_Asuransi’, ‘Umur_Kendaraan’, dan ‘Kendaraan_Rusak’.



Umur	0.104804
Sudah_Asuransi	0.325695
Umur_Kendaraan	0.212195
Kendaraan_Rusak	0.337629
Kanal_Penjualan	0.134361

Figure 10. Heatmap korelasi atribut x dengan atribut ‘Tertarik’

Untuk pembuatan model menggunakan parameter k = 5, *weight* = ‘uniform’ yang artinya semua titik di setiap k diberi bobot yang sama, dan kalkulasi perhitungan jarak menggunakan *Minkowski distance*, didapat bahwa akurasi pada model tersebut sebesar 0.88 untuk training dan 0.88 pada testing.

```
#k-NN -> k = 5
knn = KNeighborsClassifier(weights = 'uniform', n_neighbors = 5, metric='minkowski')
knn.fit(X_train, y_train)
print('Accuracy of k-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of k-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
```

```
Accuracy of k-NN classifier on training set: 0.88
Accuracy of k-NN classifier on test set: 0.88
```

Figure 11. Akurasi model

4. Evaluasi

Di tahap evaluasi digunakan penghitungan F1 Score. F1 Score sendiri adalah nilai rata-rata antara *precision* dan *recall*. F1 Score dikatakan terbaik apabila menghasilkan nilai 1, dan dikatakan terburuk apabila menghasilkan nilai 0. Parameter *average* yang digunakan adalah 'micro' dan 'macro', dan hasilnya terlihat berbeda namun tidak terlalu jauh. F1 Score dengan parameter *average*='micro' mendapatkan hasil yang lebih besar daripada 'macro'.

Hasil tersebut bisa berbeda karena dengan menggunakan 'micro', penghitungan metrik dilakukan secara global dengan menghitung total TP (*True Positive*), FN (*False Negative*), dan FP (*False Positive*). Sedangkan untuk 'macro', penghitungan metrik dilakukan untuk setiap label, kemudian *unweighted average*. Ini tidak memperhitungkan ketidakseimbangan label.

```
#F1 Score
ck_predict = knn.predict(X_test)
print("F1-SCORE ",f1(y_test,ck_predict,average='micro'))
print("F1-SCORE ",f1(y_test,ck_predict,average='macro'))
```

```
F1-SCORE  0.8769705493398267
F1-SCORE  0.46722659002203165
```

Figure 12. F1 Score Model

5. Eksperimen

Eksperimen dibagi menjadi 2 tahap dengan perbedaan pada parameter *weight*, metode perhitungan jarak dan akan dibandingkan hasilnya berdasarkan nilai akurasi yang didapat dengan eksplorasi nilai k = [1..20].

5.1 Tahap 1

Tahap 1 menggunakan atribut yang sudah ditentukan sebelumnya dengan parameter *weight* = 'uniform', dan kalkulasi jarak menggunakan *Minkowski* dan didapat hasilnya $1 < k \leq 20$ memiliki tingkat akurasi sebesar 0.88.

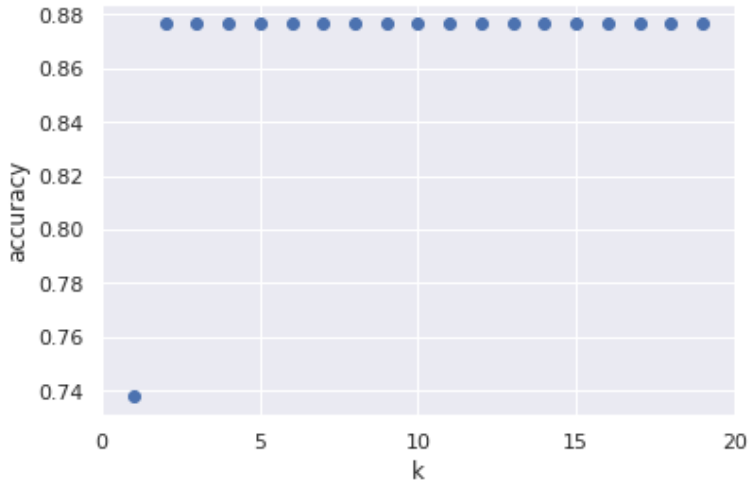


Figure 13. Akurasi model tahap 1

5.2 Tahap 2

Tahap 2 menggunakan atribut dengan nilai korelasinya sebesar > 0.1 dengan parameter *weight* = 'distance', dan kalkulasi jarak menggunakan *Euclidean* dan didapat hasilnya bahwa k = 18 memiliki tingkat akurasi sebesar 0.875.

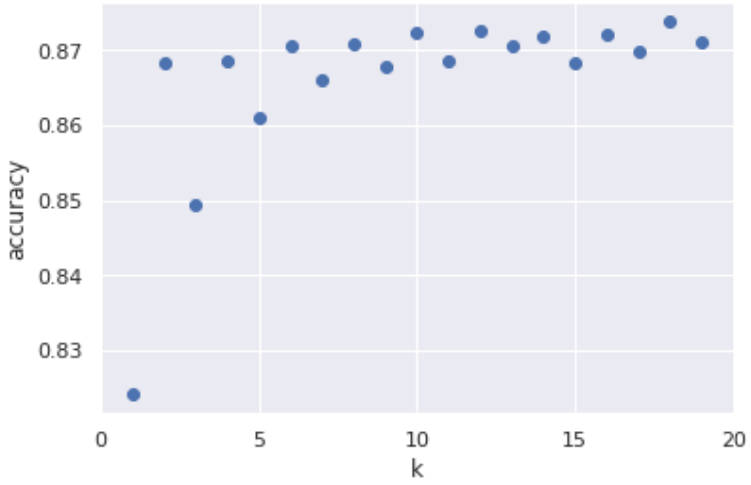


Figure 14. Akurasi model tahap 2

6. Kesimpulan

Pembentukan model menggunakan *k-NN Classification* dengan mentransformasikan setiap atribut yang non-numerik ke dalam numerik agar atribut tersebut bisa diketahui nilai korelasinya. *k-NN* menentukan class dengan menghitung jarak antar objek, maka metode ini sangat sensitif terhadap nilai *outlier*, dan juga *k-NN* tidak memiliki algoritma untuk menangani *missing value*, maka dilakukan *data cleaning* untuk menangani masalah tersebut.

Atribut yang digunakan pada model *k-NN* ditentukan dari nilai korelasinya, pada dataset ini atribut yang akan digunakan harus memenuhi nilai korelasi yang ditentukan. Untuk model tahap 1, atribut yang digunakan dengan kondisi nilai korelasinya sebesar > 0.2 didapat nilai akurasi sebesar 0.88 pada $1 < k \leq 20$. Pada tahap 2, atribut yang digunakan dengan kondisi nilai korelasinya sebesar > 0.1 didapat nilai akurasi sebesar 0.875 pada k = 18.

Akurasi akan meningkat dengan meningkatnya kompleksitas model, untuk *k-NN* kompleksitas model ditentukan oleh nilai k yang diinisialisasikan. Nilai k menyatakan berapa banyak jumlah *neighbor* atau data yang terdekat dengan suatu objek. Nilai k yang berbeda akan mempengaruhi hasil *classification* terhadap model yang dibangun.

7. Link Source Code (Telkom University Email)

https://drive.google.com/file/d/1yEQTlT0epWApQkcuGa26tMQgxYyAMGDJ/view?usp=sharing