PEMBELAJARAN MESIN
LAPORAN TUGAS BESAR TAHAP 2
KLASIFIKASI

Laporan

Disusun untuk memenuhi Tugas Besar Mata Kuliah Pembelajaran Mesin

Oleh:

Muhammad Aziz Pratama (1301180018)
Muhammad Aziz Al-assad (1301180044)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

BANDUNG

2021

A. FORMULASI MASALAH

Tugas kali ini adalah memprediksi apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak berdasarkan data pelanggan di dealer.
Pada dataset :

1. Pelanggan yang tertarik akan ditandai atau memiliki label : 1
2. Pelanggan yang tidak tertarik akan ditandai atau memiliki label : 0

Untuk permasalahan ini, akan menggunakan metode supervised learning dengan beberapa algoritma machine learning dan nantinya akan dapat memprediksi apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak.

B. EKSPLORASI DAN PERSIAPAN DATA

- **Eksplorasi Data**

1. Import data set yang terdiri dari data train dan data test yang sudah disediakan

```
#import data set
data_train = pd.read_csv('kendaraan_train.csv')
data_test = pd.read_csv('kendaraan_test.csv')
```

2. Kita bisa melihat info column beserta tipenya dari dataset, dapat dilihat bahwa data train memiliki data null/nan, sedangkan data test tidak.

```
#info data train
data_train.info()
```

```
<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
```

```
#info data test
data_test.info()
```

```
<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
```

3. Persentase data null/nan yang ada pada data train

```
#menghitung data nan/null dari tiap kolom (dalam %) di data train
data_train.isnull().sum().sort_values(ascending=False)/len(data_train)*100
```

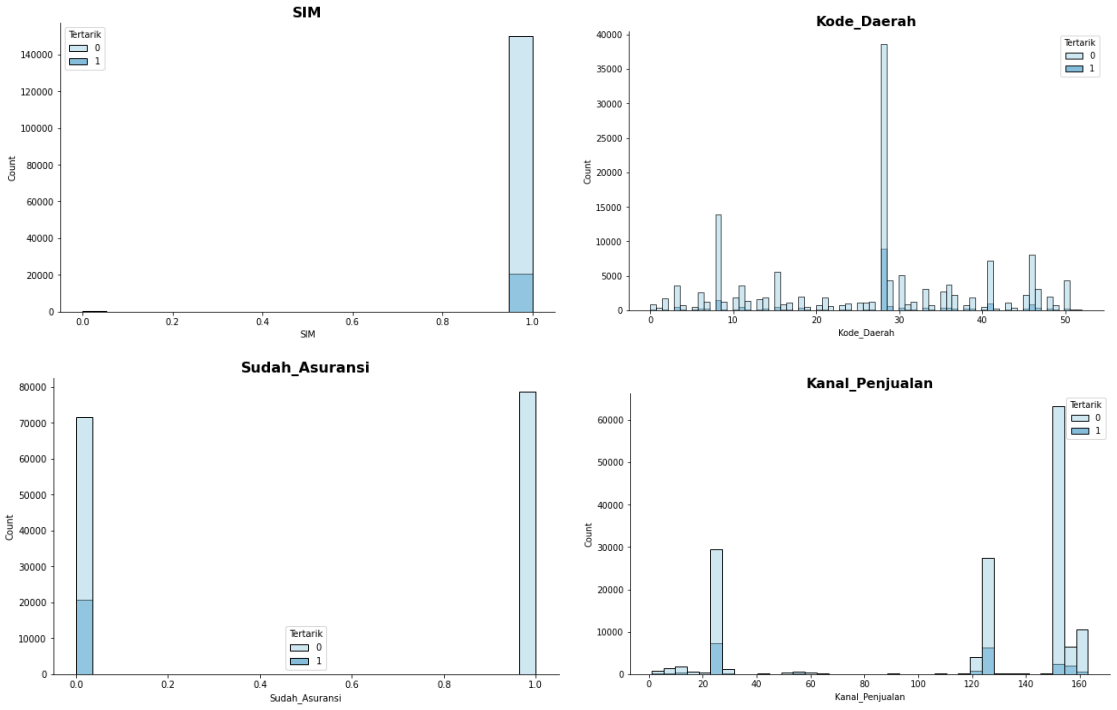
```
Premi                5.097068
Jenis_Kelamin        5.051936
SIM                  5.039341
Kode_Daerah          5.005055
Kanal_Penjualan      5.002606
Umur_Kendaraan       4.994210
Sudah_Asuransi       4.978116
Umur                 4.972869
Kendaraan_Rusak      4.963772
Lama_Berlangganan    4.895200
Tertarik             0.000000
id                   0.000000
dtype: float64
```

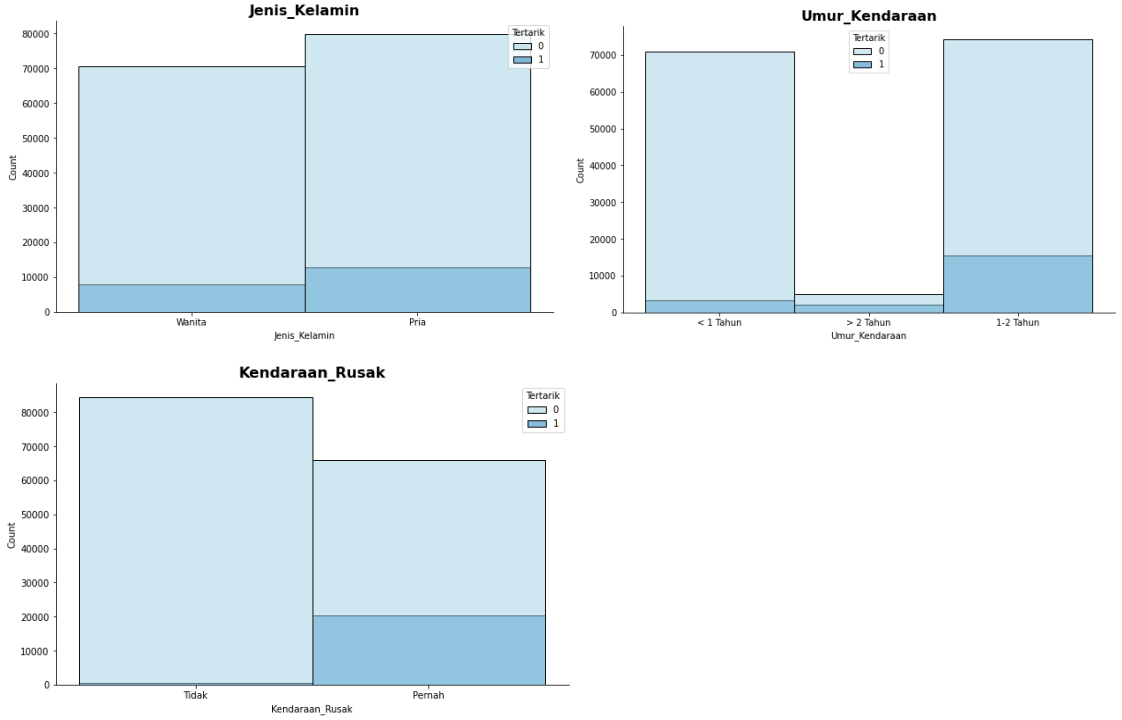
4. Drop kolom ‘ID’ karena merupakan indeks baris data yang tidak mendukung perhitungan akurasi dan drop data dengan nilai null/nan data train, karena di data test tidak terdapat nilai null/nan.

```
#drop kolom yang tidak dipakai
data_train = data_train.drop(columns=['id'])

# drop data dengan nilai nan/null
data_train = data_train.dropna()

data_train['Tertarik'].value_counts()
```

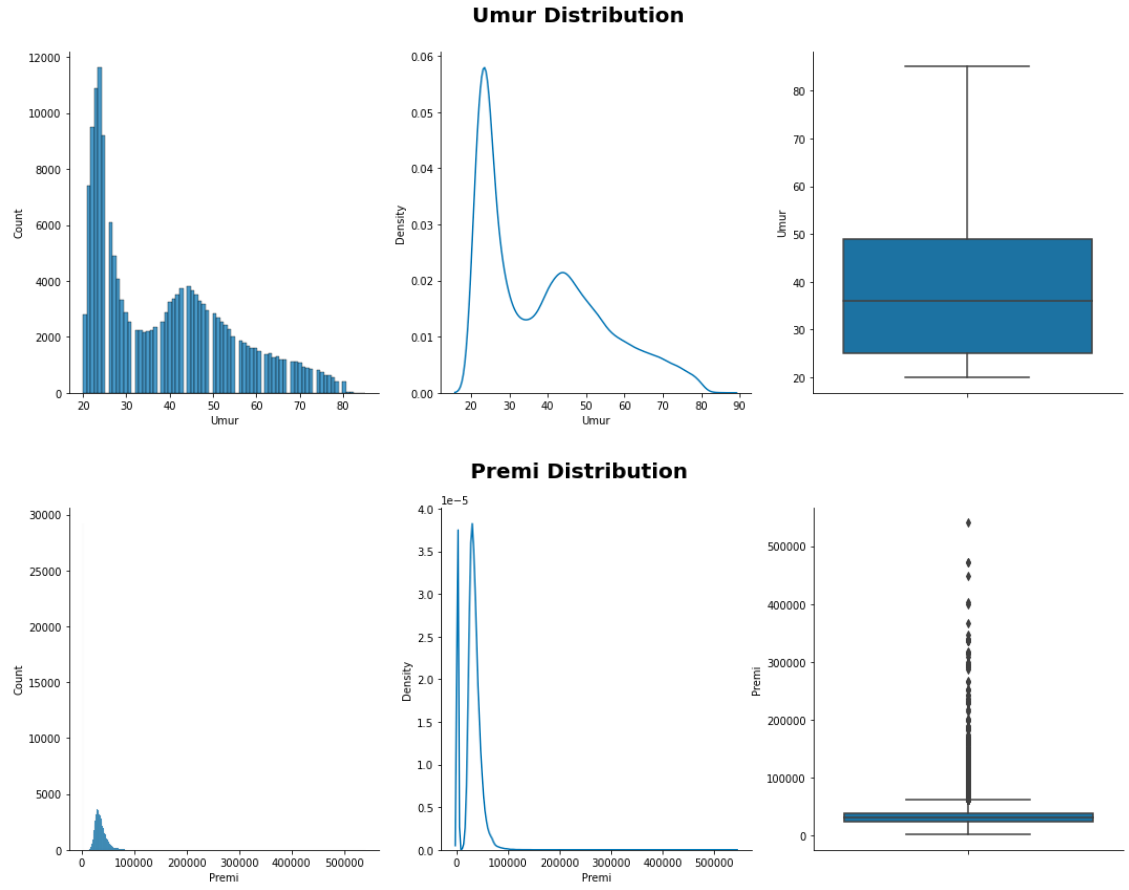
```
0    150270
1     20798
Name: Tertarik, dtype: int64
```

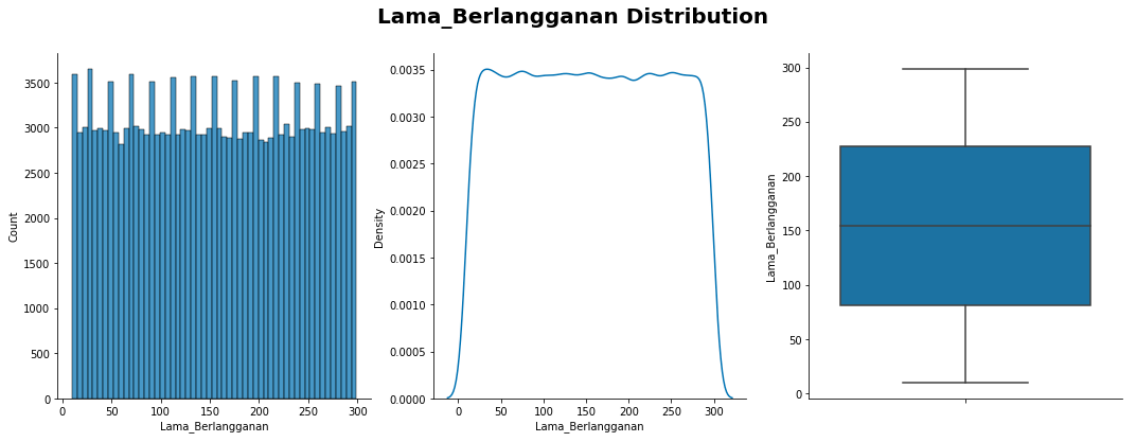
5. Plot data *categorical* untuk menampilkan distribusi data dari data set.



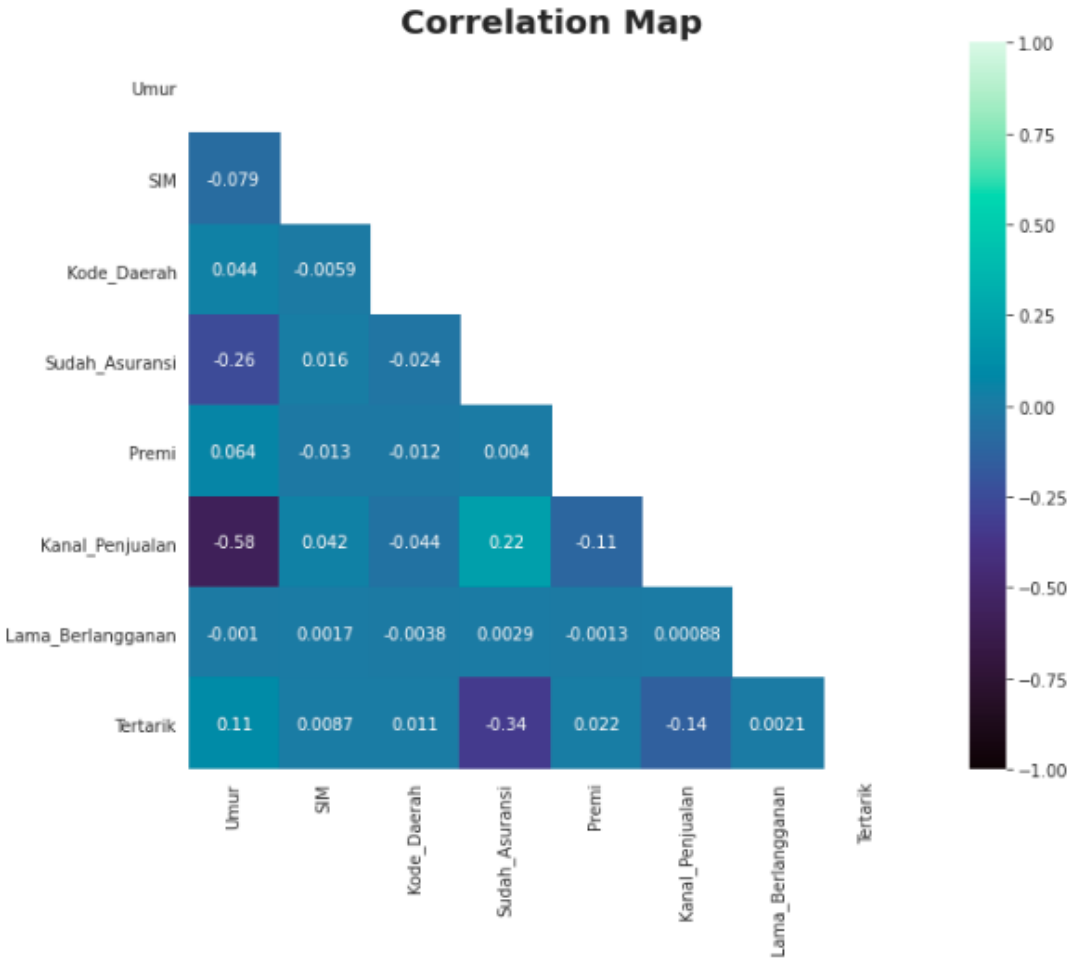


6. Plot data *numerical* untuk menampilkan distribusi data dari data train





7. *Correlation Map* untuk melihat korelasi antar data



8. Agar data yang ada tidak terganggu, maka copy data train dan data test ke variabel baru yang akan digunakan karena nantinya akan dilakukan percobaan undersampling data dan oversampling data.

```
data_train_ori = data_train.reset_index(drop=True).copy()
data_test_ori = data_test.copy()
```

- **Persiapan Data**

Untuk Feature Engineering yang dilakukan pada Undersampling dan Oversampling adalah sama, tetapi urutan langkah nya berbeda. Berikut tiap urutan langkah nya:

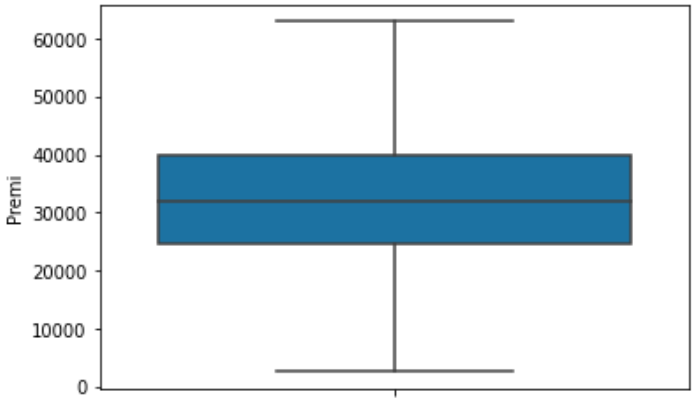
- Undersampling :
    1. Random undersampling majority class target(0) menjadi 1 : 1 dengan minority class target
    2. Handle outlier (pada feature *Premi*)
    3. Gabungkan data train dan test, lalu memberi label untuk data kategorikal
    4. Pisah kembali data train dan test, lalu melakukan MinMax scaling pada data numerik
    5. Setelah itu dijadikan data X untuk data feature dan y untuk target (Tertarik) dari masing-masing data train dan data test
- Oversampling :
    1. Handle outlier (pada feature *Premi*)
    2. Gabungkan data train dan test, lalu memberi label untuk data kategorikal
    3. Pisah kembali data train dan test, lalu melakukan MinMax scaling pada data numerik
    4. Oversampling menggunakan Smote untuk minority class target(1)
    5. Setelah itu dijadikan data X untuk data feature dan y untuk target (Tertarik) dari masing-masing data train dan data test

Catatan : untuk penggunaan library smote, sudah diberikan guide comment agar dapat dijalankan dengan baik

1. Handle outlier
   Untuk feature numerikal yang memiliki data pencilan(outlier) berdasarkan data yang sudah dieksplorasi terdapat di feature *Premi*. Nilai outlier akan diproses dengan cara mengganti data outlier tersebut dengan menggunakan nilai *upper bound* dan *lower bound* dari feature itu sendiri. Jika nilai outlier berada dibawah lower bound, maka akan diganti menjadi nilai lower bound, sedangkan jika nilai berada diatas upper bound maka akan diganti dengan nilai upper bound.

```
#handle outlier data
def handle_outliers(df, column):
  q1 = df[column].quantile(0.25)
  q3 = df[column].quantile(0.75)
  iqr = q3-q1
  lower_bound = q1-1.5*iqr
  upper_bound = q3+1.5*iqr

  for i in range(len(df)):
      if df[column].iloc[i] > upper_bound:
          df[column].iloc[i] = upper_bound
      if df[column].iloc[i] < lower_bound:
          df[column].iloc[i] = lower_bound
```

Sebaran feature *Premi* setelah outlier di handle:



2. Labeling (label encoder and one hot encoder)
   Karena machine learning tidak dapat mengenali string, maka kita perlu me-labelkan feature kategorikal. Feature kategorikal terdapat 2 bagian yaitu *ordinal* : Umur_Kendaraan dan *nominal* : jenis_kelamin, SIM, Sudah_asuransi, kendaraan_rusak, kode_daerah, kanal_penjualan. Feature ordinal dilabel dengan LabelEncoder, sedangkan feature nominal dilabel dengan teknik onehot encoder memakai pandas dummies.
   *Sebelum labeling, data train dan data test digabung terlebih dahulu.

```
#labeling data categorical ordinal
le = LabelEncoder()
data_combined['Umur_Kendaraan'] = le.fit_transform(data_combined['Umur_Kendaraan'])

#labeling data categorical nominal
dummies = pd.get_dummies(data_combined['Jenis_Kelamin'], prefix='jenkel')
data_combined = pd.concat([data_combined, dummies], axis=1)
data_combined = data_combined.drop(['Jenis_Kelamin'], axis = 1)

dummies = pd.get_dummies(data_combined['SIM'], prefix='sim')
data_combined = pd.concat([data_combined, dummies], axis=1)
data_combined = data_combined.drop(['SIM'], axis = 1)

dummies = pd.get_dummies(data_combined['Sudah_Asuransi'], prefix='asuransi')
data_combined = pd.concat([data_combined, dummies], axis=1)
data_combined = data_combined.drop(['Sudah_Asuransi'], axis = 1)

dummies = pd.get_dummies(data_combined['Kendaraan_Rusak'], prefix='kendaraan_rusak')
data_combined = pd.concat([data_combined, dummies], axis=1)
data_combined = data_combined.drop(['Kendaraan_Rusak'], axis = 1)

le_kd = LabelEncoder()
data_combined['Kode_Daerah'] = le_kd.fit_transform(data_combined['Kode_Daerah'])

le_kp = LabelEncoder()
data_combined['Kanal_Penjualan'] = le_kp.fit_transform(data_combined['Kanal_Penjualan'])
```

Catatan : walau kode_daerah dan kanal_penjualan merupakan data kategorikal nominal, karena memiliki kardinalitas yang tinggi, maka dilabel dengan LabelEncoder, agar tidak membuat banyak sekali column nantinya. Contoh setelah dilabel :

	Umur	Kode_Daerah	Umur_Kendaraan	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik	jenkel_Pria	jenkel_Wanita	sim_0.0	sim_1.0
0	52.0	28	2	2630.0	108	11.0	1	1	0	0	1
1	21.0	16	1	28559.0	138	241.0	1	0	1	0	1
2	34.0	36	0	2630.0	134	214.0	1	1	0	0	1
3	35.0	46	0	27249.0	108	241.0	1	1	0	0	1
4	26.0	35	1	28727.0	130	299.0	1	1	0	0	1

asuransi_0.0	asuransi_1.0	kendaraan_rusak_Pernah	kendaraan_rusak_Tidak
1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0

3. Scaling (normalisasi)

   Scaling dilakukan pada feature numerikal seperti Premi, Umur, Lama_Berlangganan. Saling/normalisasi ini dilakukan dengan method minmax scaling, sehingga membuat range feature berada direntang yang sama yaitu [0.1]. Proses normalisasi ini dapat mempercepat perhitungan pada machine learning.

   Minmax scaling data train dan data test dilakukan terpisah, scaling pada data test menggunakan parameter nilai min dan max yang berada di data train. Hal ini dilakukan agar tidak terjadi data leakage.

```
#scaling data dengan minmax scaler
min_premi = data_train['Premi'].min()
max_premi = data_train['Premi'].max()

min_umur = data_train['Umur'].min()
max_umur = data_train['Umur'].max()

min_lb = data_train['Lama_Berlangganan'].min()
max_lb = data_train['Lama_Berlangganan'].max()

#scaling data train
premi_scaler = MinMaxScaler()
umur_scaler = MinMaxScaler()
lama_berlangganan_scaler = MinMaxScaler()

data_train['Premi'] = premi_scaler.fit_transform(data_train[['Premi']])
data_train['Umur'] = umur_scaler.fit_transform(data_train[['Umur']])
data_train['Lama_Berlangganan'] = lama_berlangganan_scaler.fit_transform(data_train[['Lama_Berlangganan']])
```

```
#scaling data test
def MinMaxScalerTest(x,xmin,xmax):
    return((x-xmin) / (xmax-xmin))

data_test['Premi'] = MinMaxScalerTest(data_test['Premi'], min_premi, max_premi)
data_test['Umur'] = MinMaxScalerTest(data_test['Umur'], min_umur, max_umur)
data_test['Lama_Berlangganan'] = MinMaxScalerTest(data_test['Lama_Berlangganan'], min_lb, max_lb)
```

	Umur	Kode_Daerah	Umur_Kendaraan	Premi	Kanal_Penjualan	Lama_Berlangganan
0	0.492308	28	2	0.000000	108	0.003460
1	0.015385	16	1	0.430396	138	0.799308
2	0.215385	36	0	0.000000	134	0.705882
3	0.230769	46	0	0.408651	108	0.799308
4	0.092308	35	1	0.433185	130	1.000000

4. Undersampling dan Oversampling data

   Urutan untuk undersampling dan oversampling data sudah dijelaskan sebelumnya, undersampling dilakukan dengan cara mengambil berikut adalah cara undersampling dan oversampling:

```
data_train = data_train_ori.copy()
data_test = data_test_ori.copy()

#random undersampling
#class majority diturunkan jumlahnya hanya sebanyak 1:1 dengan class minority

data_train_1 = data_train[data_train.Tertarik == 1]

data_train_0 = data_train[data_train.Tertarik == 0]
data_train_0 = data_train_0.sample((len(data_train_1)))

data_train = pd.concat([data_train_1, data_train_0])
len_data_train = len(data_train)

data_train = data_train.reset_index(drop=True)
data_train
```

```
from imblearn.over_sampling import SMOTE

#oversampling dengan library smote, class minority jumlah nya dinaikkan agar 1:1 dengan class majority
smo = SMOTE()
X_train_ovr, y_train_ovr = smo.fit_resample(X_train, y_train)
```

```
data_train = X_train_ovr
data_train['Tertarik'] = y_train_ovr
data_train
```

5. Memisahkan kolom feature dan target yaitu “Tertarik” pada data train dan data test sebagai berikut :

```
X_train = data_train.drop(columns=['Tertarik'])
y_train = data_train.Tertarik
X_test = data_test.drop(columns=['Tertarik'])
y_test = data_test.Tertarik
```

C. PEMODELAN

Pemodelan dilakukan dengan 3 algoritma ditambah dengan 1 menggunakan voting dari ketiga algoritma tersebut.

Walaupun ada Undersampling dan Oversampling, tetapi classifier/algoritma machine learning yang digunakan sama.

1. Pemodelan pertama menggunakan algoritma Random Forest.

```
#classification using random forest
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print(classification_report(y_test, y_pred))
```

Hasil nya :

- Undersampling

	precision	recall	f1-score	support
0	0.97	0.69	0.81	41778
1	0.28	0.87	0.43	5861
accuracy			0.71	47639
macro avg	0.63	0.78	0.62	47639
weighted avg	0.89	0.71	0.76	47639

- Oversampling

	precision	recall	f1-score	support
0	0.91	0.88	0.90	41778
1	0.32	0.42	0.37	5861
accuracy			0.82	47639
macro avg	0.62	0.65	0.63	47639
weighted avg	0.84	0.82	0.83	47639

2. Pemodelan kedua menggunakan algoritma Logistic Regression.

```
#classification using logistic regression
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print(classification_report(y_test, y_pred))
# report = classification_report(y_test, y_pred, output_dict=True)
# print(report)
```

Hasil nya :

- Undersampling

	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

- Oversampling

	precision	recall	f1-score	support
0	0.95	0.72	0.82	41778
1	0.28	0.76	0.40	5861
accuracy			0.73	47639
macro avg	0.62	0.74	0.61	47639
weighted avg	0.87	0.73	0.77	47639

3. Pemodelan ketiga menggunakan algoritma Naive Bayes (gaussian)

```
#classification using naive bayes (gaussian)
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred = nb.predict(X_test)
print(classification_report(y_test, y_pred))
```

hasil nya :

- Undersampling

	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

- Oversampling

	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

4. Pemodelan keempat menggunakan algoritma Voting, dari 3 algoritma yang lain ada

```
#classification using voting (random forest, logistic regression, naive bayes)
from sklearn.ensemble import VotingClassifier

eclf = VotingClassifier(estimators=[('rf', rf), ('lr', lr), ('nb',nb)],voting='hard')
eclf.fit(X_train, y_train)
y_pred = eclf.predict(X_test)
print(classification_report(y_test, y_pred))
```

Hasil nya:

- Undersampling

	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

- Oversampling

	precision	recall	f1-score	support
0	0.96	0.71	0.82	41778
1	0.28	0.80	0.41	5861
accuracy			0.72	47639
macro avg	0.62	0.75	0.61	47639
weighted avg	0.88	0.72	0.77	47639

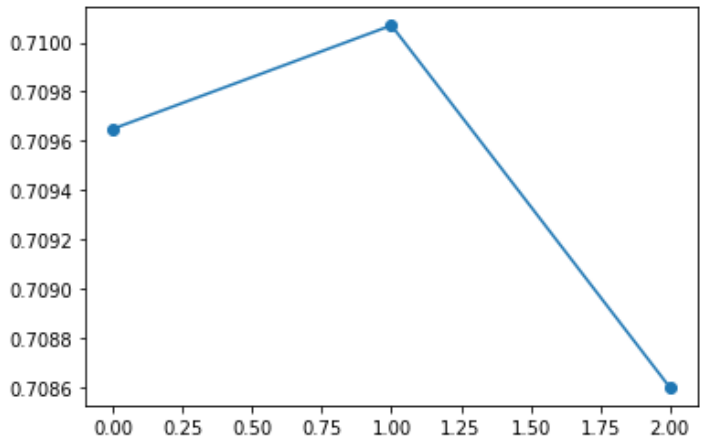
D. EVALUASI

Pada evaluasi ini, dicoba dan digunakan hyper parameter yang kemungkinan dapat menghasilkan model yang lebih baik dari hyper parameter tiap model.

- Random Forest
  - Pada model ini ditambahkan parameter criterion *'entropy'*, karena defaultnya menggunakan *'gini'*
  - Menggunakan max features *'log2'*
  - N_estimators yaitu menentukan banyaknya tree dicoba dari 100, 200, dan 400

```
#classification using random forest
from sklearn.ensemble import RandomForestClassifier
scoreRF = []
n_estimators = [100, 200, 400]
for i in range(len(n_estimators)):
  rf = RandomForestClassifier(criterion = 'entropy', max_features='log2', n_estimators= n_estimators[i])
  rf.fit(X_train, y_train)
  y_pred = rf.predict(X_test)
  scoreRF.append(accuracy_score(y_test, y_pred))
```

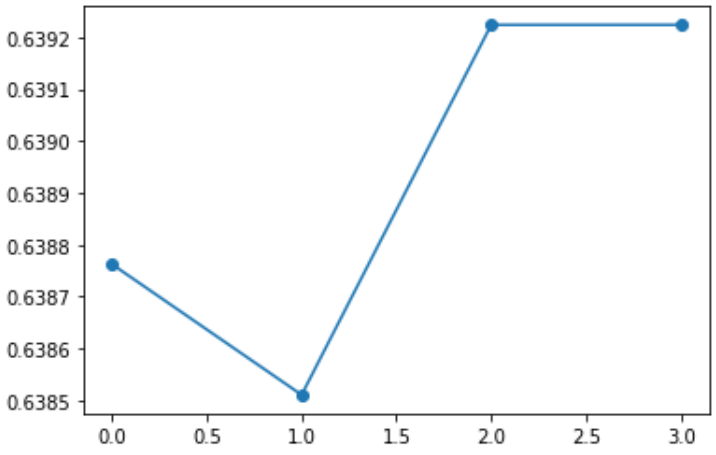
```
#accuracy random forest dengan urutan n_estimators = [100, 200, 400]
plt.plot(scoreRF)
plt.scatter([0,1,2], scoreRF)
plt.show()
```



- Logistic Regression
    - Pada model ini, ditambahkan parameter max_iter yaitu mengatur jumlah iterasi yang akan dicoba dari rentang 100, 200, 300, 400

```
#classification using logistic regression
from sklearn.linear_model import LogisticRegression
scoreLR = []

max_iter = [100, 200, 300, 400]
for i in range(len(max_iter)):
  lr = LogisticRegression(random_state=1, max_iter=max_iter[i])
  lr.fit(X_train, y_train)
  y_pred = lr.predict(X_test)
  scoreLR.append(accuracy_score(y_test, y_pred))
```

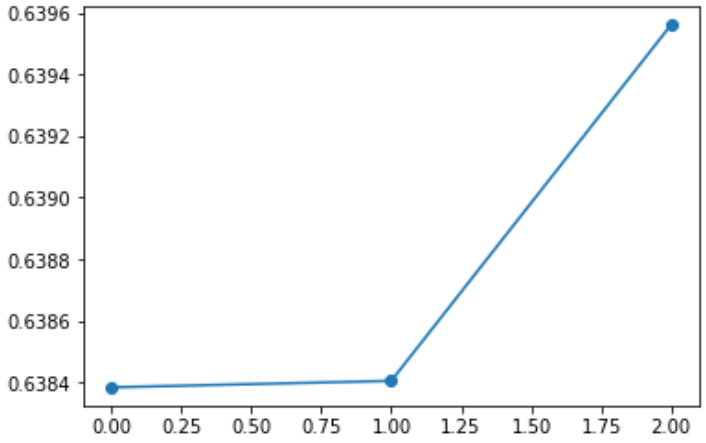
```
#accuracy logistic regression dengan urutan max_iter = [100, 200, 300, 400]
plt.plot(scoreLR)
plt.scatter([0,1,2,3], scoreLR)
plt.show()
```



- Naive Bayes
  - Pada model ini, ditambahkan parameter var_smoothing yaitu porsi variance yang akan dicoba dari rentang 0.0001, 0.00001, 0.000001

```
#classification using naive bayes (gaussian)
from sklearn.naive_bayes import GaussianNB
scoreNB = []

var_smoothing = [0.0001, 0.00001, 0.000001]
for i in range(len(var_smoothing)):
  nb = GaussianNB(var_smoothing = var_smoothing[i])
  nb.fit(X_train, y_train)
  y_pred = nb.predict(X_test)
  scoreNB.append(accuracy_score(y_test, y_pred))
```

```
#accuracy naive bayes dengan urutan var_smoothing = [0.0001, 0.00001, 0.000001]
plt.plot(scoreNB)
plt.scatter([0,1,2], scoreNB)
plt.show()
```



E. EKSPERIMEN

Dari parameter yang sudah didapat pada bagian evaluasi, kita dapat langsung memakai nya dan melihat detail dari hasil prediksi nya seperti precision, recall, f1-score. Berikut percobaanya :

Undersampling :

1. Random Forest

```
#classification using random forest
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(criterion = 'entropy', max_features = 'log2', n_estimators= 200)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.69      0.81     41778
           1       0.28      0.87      0.43      5861

    accuracy                           0.71     47639
   macro avg       0.63      0.78      0.62     47639
weighted avg       0.89      0.71      0.76     47639
```

2. Logistic Regression

```
#classification using logistic regression
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr = LogisticRegression(random_state=1, max_iter=400)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              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
```

3. Naive Bayes

```
#classification using naive bayes (gaussian)
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB(var_smoothing = 0.000001)
nb.fit(X_train, y_train)
y_pred = nb.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              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
```

4. Voting Classifier

```
#classification using voting (random forest, logistic regression, naive bayes)
from sklearn.ensemble import VotingClassifier

eclf = VotingClassifier(estimators=[('rf', rf), ('lr', lr), ('nb',nb)],voting='soft', weights = [2,1,1])
eclf.fit(X_train, y_train)
y_pred = eclf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.60      0.75     41778
           1       0.25      0.97      0.40      5861

    accuracy                           0.65     47639
   macro avg       0.62      0.79      0.58     47639
weighted avg       0.90      0.65      0.71     47639
```

Oversampling :

1. Random Forest

```
#classification using random forest
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(criterion = 'entropy', max_features='log2', n_estimators= 200)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.88      0.90     41778
           1       0.32      0.42      0.37      5861

    accuracy                           0.82     47639
   macro avg       0.62      0.65      0.63     47639
weighted avg       0.84      0.82      0.83     47639
```

2. Logistic Regression

```
#classification using logistic regression
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(solver='lbfgs', max_iter=2000)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.72      0.82     41778
           1       0.27      0.76      0.40      5861

    accuracy                           0.72     47639
   macro avg       0.61      0.74      0.61     47639
weighted avg       0.87      0.72      0.77     47639
```

3. Naive Bayes

```
#classification using naive bayes (gaussian)
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB(var_smoothing = 0.000001)
nb.fit(X_train, y_train)
y_pred = nb.predict(X_test)
print(classification_report(y_test, y_pred))
```

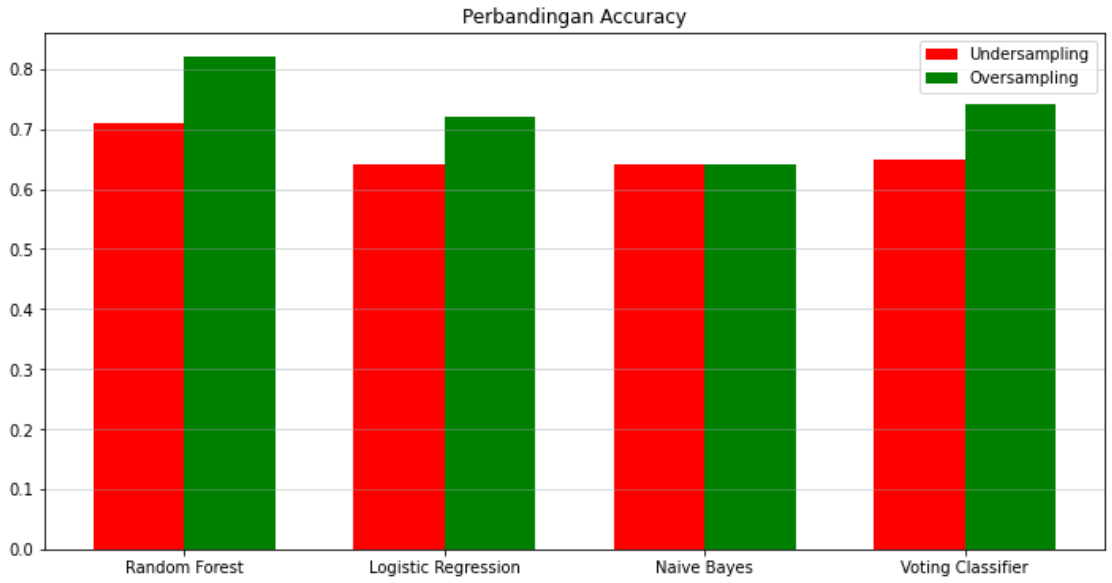
```
              precision    recall  f1-score   support

           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
```

4. Voting Classifier

```
#classification using voting (random forest, logistic regression, naive bayes)
from sklearn.ensemble import VotingClassifier

eclf = VotingClassifier(estimators=[('rf', rf), ('lr', lr), ('nb',nb)],voting='hard')
eclf.fit(X_train, y_train)
y_pred = eclf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.70      0.81     41778
           1       0.28      0.80      0.41      5861

    accuracy                           0.72     47639
   macro avg       0.62      0.75      0.61     47639
weighted avg       0.88      0.72      0.76     47639
```

KESIMPULAN



Setelah melakukan beberapa percobaan, dapat disimpulkan bahwa :

- Dari percobaan yang telah dilakukan, mau itu undersampling atau oversampling, **random forest** memiliki akurasi yang paling baik setelah diuji dengan data_test.
- Akurasi yang dihasilkan oleh machine learning lebih besar saat dilakukan oversampling daripada undersampling dengan dilakukan treatment yang sama terhadap datanya
- Walaupun akurasi machine learning pada oversampling data lebih besar, tetapi saat dilihat pada f1 score, pada target class yang awalnya minority(Tertarik = 1), tetap saja susah mengenali kelas tersebut, dapat kita lihat bahwa f1 score kelas Tertarik = 1 berada dikisaran 40%.
- Mengganti sedikit hyperparameter di machine learning juga berpengaruh, walau tidak terlalu besar
- Untuk data imbalance yang sangat besar perbedaanya, kita perlu melakukan undersampling maupun oversampling karena jika tidak, kemungkinan besar machine learning akan mengalami overfit dan hanya mengenali data yang mayoritas saja.