LAPORAN PEMBELAJARAN MESIN

Disusun Untuk Memenuhi Tugas Besar Tahap II Mata Kuliah Pembelajaran Mesin

Dosen Pengampu: Agus Hartoyo, Ph.D

Disusun Oleh :

Hanvito Michael Lee (1301190090)
Naufal Haritsah Luthfi (1301194073)

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2021/2022

Daftar Isi

1. **Formulasi Masalah**
   a. Melakukan eksplorasi dan persiapan data untuk data test dan data train agar siap digunakan untuk pemodelan classification.
   b. Melihat akurasi dari model klasifikasi Logistic Regression terhadap data train dan data test.
   c. Melihat akurasi dari model klasifikasi Decision Tree terhadap data train dan data test.
   d. Melihat akurasi dari model klasifikasi K-Nearest Neighbors terhadap data train dan data test.
   e. Melihat akurasi dari model klasifikasi Linear Discriminant Analysis terhadap data train dan data test.
   f. Melihat akurasi dari model klasifikasi Gaussian Naive Bayes terhadap data train dan data test.
   g. Melakukan evaluasi terhadap model klasifikasi dengan nilai akurasi terbaik.
   h. Melakukan eksperimen terhadap dataset.

2. **Eksplorasi dan Persiapan Data**
   a. Import Library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Library yang diimport yaitu pandas, numpy, matplotlib.pyplot, dan seaborn untuk melakukan eksplorasi dan persiapan data pada dataset.

b. Data Train

i. Data Preprocessing

```
df_train.shape
```

```
(285831, 12)
```

Pertama, dipanggil fungsi .shape untuk mengetahui ukuran dari dataset yang diberikan. Pada dataset ini terdapat 12 kolom dan 285.831 baris.

```
df_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
```

Kemudian dipanggil fungsi .info() untuk mengetahui informasi lebih lanjut mengenai dataset yang diberikan. Terdapat berbagai jenis tipe data pada tiap tiap kolom yang terdapat dalam dataset.

```
df_train.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
dtype: int64
```

Selanjutnya, dipanggil fungsi .isnull().sum() untuk mengecek missing value yang terdapat pada set. Terdapat banyak nilai yang kosong pada dataset sehingga harus dilakukannya handling missing values.

ii. Handling Missing Value

```
df_train['SIM'] = df_train['SIM'].replace(np.NaN, 1.0)
df_train['Kode_Daerah'] = df_train['Kode_Daerah'].replace(np.NaN, 28.0)
df_train['Jenis_Kelamin'] = df_train['Jenis_Kelamin'].replace(np.NaN, "Pria")
df_train['Umur'] = df_train['Umur'].replace(np.NaN, df_train['Umur'].mean())
df_train['Sudah_Asuransi'].fillna(method = 'bfill', inplace = True)
df_train['Umur_Kendaraan'] = df_train['Umur_Kendaraan'].replace(np.NaN, "1-2 Tahun")
df_train['Kendaraan_Rusak'] = df_train['Kendaraan_Rusak'].replace(np.NaN, "Pernah")
df_train['Premi'] = df_train['Premi'].replace(np.NaN, df_train['Premi'].mean())
df_train['Kanal_Penjualan'] = df_train['Kanal_Penjualan'].replace(np.NaN, 152.0)
df_train['Lama_Berlangganan'] = df_train['Lama_Berlangganan'].replace(np.NaN, df_train['Lama_Berlangganan'].mean())
```

Handling missing values pada dataset diisi dengan berbagai metode yaitu mengisi dengan modus, mean, dan mengisi dengan neighbor values. Nilai modus diisi pada kolom Jenis_Kelamin, SIM, Kode_Daerah, Umur_Kendaraan, Kendaraan_Rusak, dan Kanal_Penjualan. Nilai mean diisi pada kolom Umur, Premi, dan Lama_Berlangganan. Nilai neighbor value diisi pada kolom Sudah_Asuransi.

iii. Data Normalization

Terdapat berbagai tipe data pada dataset yang dapat dilihat pada poin (ii). Sehingga untuk mencari nilai korelasi antara tiap-tiap kolom akan didapatkan hasil yang kurang maksimal. Untuk itu, digunakan metode Z

score untuk melakukan normalisasi untuk tiap tiap kolom pada dataset. Sebelum itu, kolom yang berbentuk kategorikal seperti Jenis_Kelamin, Kendaraan_Rusak, dan Umur_Kendaraan diubah terlebih dahulu menjadi numerikal.

```
df_train['Jenis_Kelamin'] = df_train['Jenis_Kelamin'].replace("Pria", 1.0)
df_train['Jenis_Kelamin'] = df_train['Jenis_Kelamin'].replace("Wanita", 0)

df_train['Kendaraan_Rusak'] = df_train['Kendaraan_Rusak'].replace("Pernah", 1.0)
df_train['Kendaraan_Rusak'] = df_train['Kendaraan_Rusak'].replace("Tidak", 0)

df_train['Umur_Kendaraan'] = df_train['Umur_Kendaraan'].replace("< 1 Tahun", 0)
df_train['Umur_Kendaraan'] = df_train['Umur_Kendaraan'].replace("1-2 Tahun", 1.0)
df_train['Umur_Kendaraan'] = df_train['Umur_Kendaraan'].replace("> 2 Tahun", 2.0)
```

Pada kolom Jenis_Kelamin, data yang bernilai “Pria” akan diubah menjadi 1, dan data yang bernilai “Wanita” akan diubah menjadi 0. Pada kolom Kendaraan_Rusak, data yang bernilai “Pernah” akan diubah menjadi 1, dan data yang bernilai “Tidak” akan diubah menjadi 0. Pada kolom Umur_Kendaraan, data yang bernilai “< 1 Tahun” akan diubah menjadi 0, data yang bernilai “1-2 Tahun” akan diubah menjadi 1, dan data yang bernilai “> 2 Tahun” akan diubah menjadi 2.

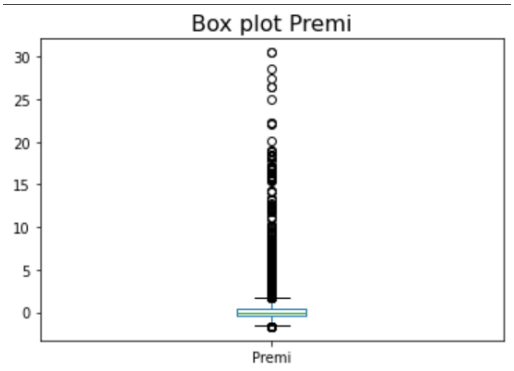
```
#Z score
for column in df_train.columns:
    df_train[column] = (df_train[column] -
                        df_train[column].mean()) / df_train[column].std()

df_train.head(5)
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan
0	-1.136620	-0.584494	0.045248	0.504182	1.085734	-1.124496	-1.060574	-0.150052	0.709303	-0.701860
1	0.879798	0.605069	0.045248	0.968525	-0.921033	2.449372	0.942882	-0.283429	-1.587901	0.045499
2	0.879798	-1.179276	0.045248	1.510259	1.085734	-1.124496	-1.060574	0.131421	0.858715	-0.432321
3	-1.136620	1.265937	0.045248	1.665040	-0.921033	0.662438	-1.060574	-1.669851	0.186362	-1.118421
4	0.879798	0.737242	0.045248	0.658963	-0.921033	2.449372	0.942882	0.258515	-0.485990	0.486564

Perhitungan Z score diimplementasikan dalam kodingan diatas dan diterapkan ke semua kolom yang ada pada dataset. Fungsi .head dipanggil untuk melihat normalisasi data yang telah berhasil dilakukan.

iv. Handling Outliers

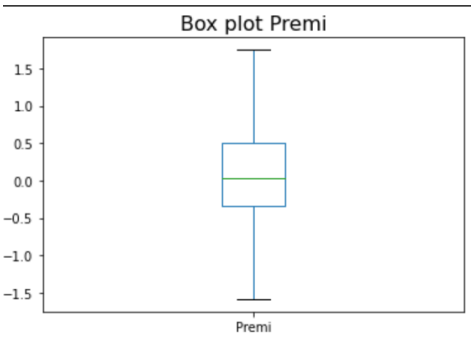
Setelah dilakukan pengecekan terhadap data numerikal, terdapat outliers pada kolom Premi dengan boxplot sebagai berikut.



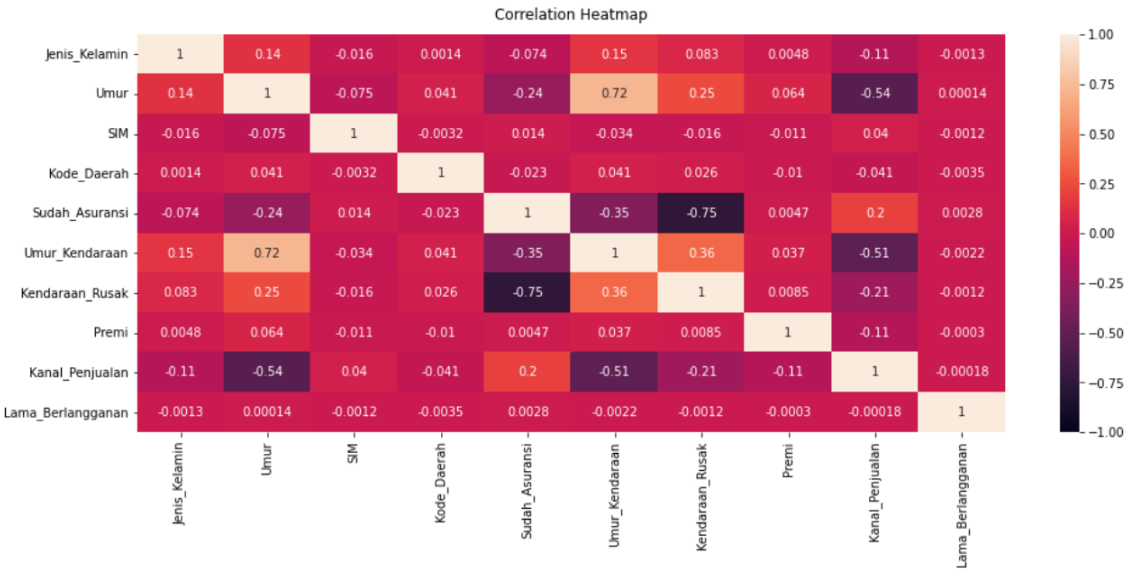
Terdapat banyak pencilan pada kolom premi sehingga pencilan ini harus dihilangkan terlebih dahulu. Digunakan metode IQR untuk menghilangkan pencilan yang ada.

```
data = df_train['Premi']
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)

IQR = q3-q1
lwr_bound = q1-(1.5*IQR)
upr_bound = q3+(1.5*IQR)

for i in range(len(df_train['Premi'])):
   if df_train['Premi'][i] < lwr_bound:
     df_train['Premi'][i] = lwr_bound
   if df_train['Premi'][i] > upr_bound:
     df_train['Premi'][i] = upr_bound
```

Setelah membersihkan pencilan yang ada pada kolom premi, didapatkan boxplot sebagai berikut.



v. Heatmap



Pada heatmap, nilai korelasi yang paling tinggi positif yaitu 0.72 antara kolom Umur dengan kolom Umur_Kendaraan. Serta nilai korelasi yang paling tinggi negatif yaitu -0.75 antara kolom Kendaraan_Rusak dengan Sudah_Asuransi.

c. Data Test

i. Data Preprocessing

```
df_test.shape
```

```
(47639, 11)
```

Pertama, dipanggil fungsi .shape untuk mengetahui ukuran dari dataset yang diberikan. Pada dataset ini terdapat 11 kolom dan 47.639 baris.

```
df_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
```

Selanjutnya .info akan menampilkan Non-Null, Count, dan Dtype pada data tersebut.

```
df_test.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
```

Selanjutnya, dipanggil fungsi .isnull().sum() untuk mengecek missing value yang terdapat pada set. Karena tidak terdapat nilai yang kosong pada dataset maka tidak dilakukan handling missing values.

ii. Data Normalization

```
# Normalisasi Data dengan mengubah tipe data dari kolom Jenis_Kelamin,
# Kendaraan_Rusak dan Umur_Kendaraan
df_test['Jenis_Kelamin'] = df_test['Jenis_Kelamin'].replace("Pria", 1.0)
df_test['Jenis_Kelamin'] = df_test['Jenis_Kelamin'].replace("Wanita", 0)

df_test['Kendaraan_Rusak'] = df_test['Kendaraan_Rusak'].replace("Pernah", 1.0)
df_test['Kendaraan_Rusak'] = df_test['Kendaraan_Rusak'].replace("Tidak", 0)

df_test['Umur_Kendaraan'] = df_test['Umur_Kendaraan'].replace("< 1 Tahun", 0)
df_test['Umur_Kendaraan'] = df_test['Umur_Kendaraan'].replace("1-2 Tahun", 1.0)
df_test['Umur_Kendaraan'] = df_test['Umur_Kendaraan'].replace("> 2 Tahun", 2.0)
```

Proses diatas adalah merubah kategori data kategorikal menjadi numerikal.

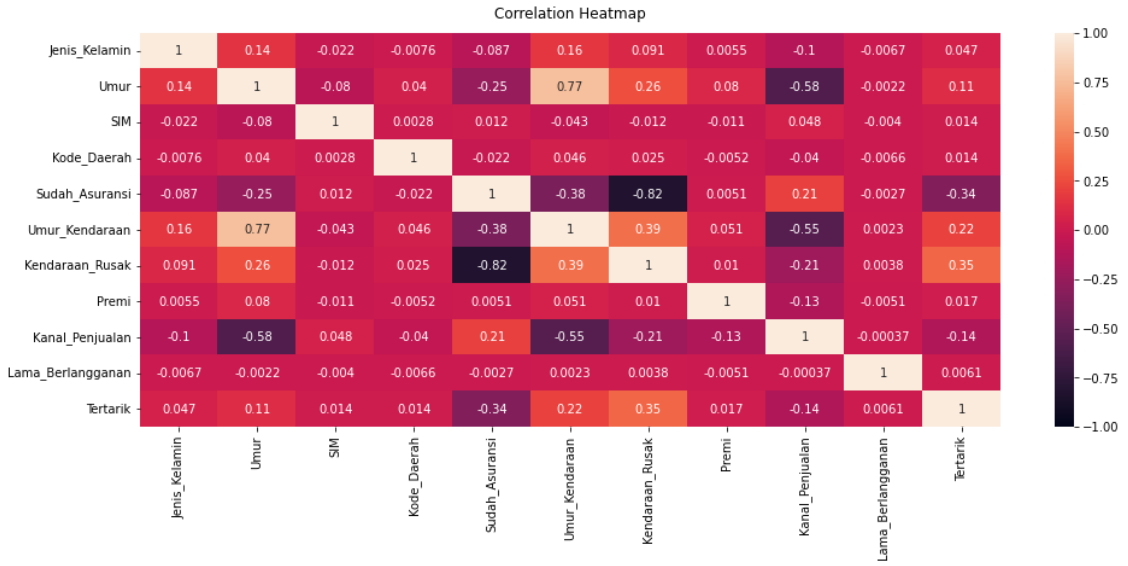
- Untuk kolom Jenis_Kelamin, jika Wanita di set dengan angka 1, Pria di set dengan angka 0.
- Untuk kolom Kendaraan_Rusak, jika Pernah di set dengan angka 1, Tidak di set dengan angka 0.
- Untuk kolom Umur_Kendaraan, jika < 1 tahun di set dengan angka 1, 1-2 tahun di set dengan angka 2, > 2 tahun di set dengan angka 3

```
for column in df_test.columns:
  if column != "Tertarik" :
    df_test[column] = (df_test[column] -
                       df_test[column].mean()) / df_test[column].std()

df_test.head(5)
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	-1.090120	0.655004	0.045633	-1.387348	-0.918514	0.692510	0.990733	0.938622	-1.591142	-0.112952	0
1	0.917311	-1.081463	0.045633	1.564212	1.088692	-1.070649	-1.009332	0.516956	0.735182	1.034795	0
2	0.917311	-0.952836	0.045633	0.126273	1.088692	-1.070649	-1.009332	4.587962	0.735182	-1.105275	0
3	0.917311	0.462063	0.045633	-1.387348	1.088692	0.692510	-1.009332	0.324021	0.218221	-1.440035	0
4	0.917311	-0.245386	0.045633	-0.252132	-0.918514	0.692510	0.990733	-0.210487	0.735182	0.891326	0

Perhitungan Z score diimplementasikan dalam kodingan diatas dan diterapkan ke semua kolom yang ada pada dataset. Fungsi .head dipanggil untuk melihat normalisasi data yang telah berhasil dilakukan.

iii. Heatmap



Pengecekan korelasi menggunakan heatmap diperlukan untuk mengecek variabel-variabel yang memiliki korelasi kuat terhadap variabel-variabel tersebut. Nilai korelasi yang paling tinggi positif yaitu 0.77 antara variabel Umur dengan variabel Umur_Kendaraan. Serta nilai korelasi yang paling tinggi negatif yaitu -0.82 antara variabel Kendaraan_Rusak dengan Sudah_Asuransi.

3. Pemodelan

a. Import Library

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.model_selection import train_test_split
```

Library yang diimport yaitu :

- LogisticRegression digunakan untuk melihat model klasifikasi logistic regression.
- DecisionTreeClassifier digunakan untuk melihat model klasifikasi decision tree classifier.
- KNeighborsClassifier digunakan untuk melihat model klasifikasi KNeighbors classifier.
- LinearDiscriminantAnalysis digunakan untuk melihat model klasifikasi linear discriminant analysis.

- GaussianNB digunakan untuk melihat model klasifikasi gaussianNB.
- metrics digunakan untuk evaluasi model
- classification_report digunakan untuk evaluasi model
- confusion_matrix digunakan untuk evaluasi model
- roc_auc_score digunakan untuk evaluasi model
- roc_curve digunakan untuk evaluasi model
- train_test_split digunakan untuk eksperimen

b. Logistic Regression

```
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
print('Accuracy of Logistic regression classifier on training set: {:.2f}'
     .format(logreg.score(x_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'
     .format(logreg.score(x_test, y_test)))
```

Pada source code Logistic Regression, langkah pertama yaitu memanggil library LogisticRegression yang disimpan ke variabel logreg. Kemudian dimasukkan data x_train dan y_train ke dalam variabel logreg.fit. Selanjutnya dicetak akurasi skor dari data train dan data set.

```
Accuracy of Logistic regression classifier on training set: 0.88
Accuracy of Logistic regression classifier on test set: 0.88
```

Output yang dihasilkan berupa akurasi model logistic regression dengan nilai pada data train yaitu 0.88 dan nilai pada data set yaitu 0.88.

c. Decision Tree

```
clf = DecisionTreeClassifier().fit(x_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(x_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(x_test, y_test)))
```

Pada source code Decision Tree, langkah pertama yaitu memanggil library Decision Tree yang disimpan ke variabel clf. Kemudian dimasukkan data x_train dan y_train ke dalam variabel clf.score. Selanjutnya dicetak akurasi skor dari data train dan data set.

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.81
```

Output yang dihasilkan berupa akurasi model decision tree dengan nilai pada data train yaitu 1.00 dan nilai pada data set yaitu 0.81.

d. K-Nearest Neighbors

```
knn = KNeighborsClassifier()
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)))
```

Pada source code K-Nearest Neighbors, langkah pertama yaitu memanggil library KNeighborsClassifier yang disimpan ke variabel knn. Kemudian dimasukkan data x_train dan y_train ke dalam variabel knn.score. Selanjutnya dicetak akurasi skor dari data train dan data set.

```
Accuracy of K-NN classifier on training set: 0.90
Accuracy of K-NN classifier on test set: 0.86
```

Output yang dihasilkan berupa akurasi model K-Nearest Neighbors dengan nilai pada data train yaitu 0.90 dan nilai pada data set yaitu 0.86.

e. Linear Discriminant Analysis

```
lda = LinearDiscriminantAnalysis()
lda.fit(x_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(x_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(x_test, y_test)))
```

Pada source code Linear Discriminant Analysis, langkah pertama yaitu memanggil library LinearDiscriminantAnalysis yang disimpan ke variabel lda. Kemudian dimasukkan data x_train dan y_train ke dalam variabel lda.fit. Selanjutnya dicetak akurasi skor dari data train dan data set.

```
Accuracy of LDA classifier on training set: 0.88
Accuracy of LDA classifier on test set: 0.88
```

Output yang dihasilkan berupa akurasi model Linear Discriminant Analysis dengan nilai pada data train yaitu 0.88 dan nilai pada data set yaitu 0.88.

f. Gaussian Naive Bayes

```
gnb = GaussianNB()
gnb.fit(x_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'
     .format(gnb.score(x_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'
     .format(gnb.score(x_test, y_test)))
```

Pada source code Gaussian Naive Bayes, langkah pertama yaitu memanggil library GaussianNB yang disimpan ke variabel gnb. Kemudian dimasukkan data x_train dan y_train ke dalam variabel gnb.fit. Selanjutnya dicetak akurasi skor dari data train dan data set.

```
Accuracy of GNB classifier on training set: 0.70
Accuracy of GNB classifier on test set: 0.70
```

Output yang dihasilkan berupa akurasi model Gaussian Naive Bayes dengan nilai pada data train yaitu 0.70 dan nilai pada data set yaitu 0.70.

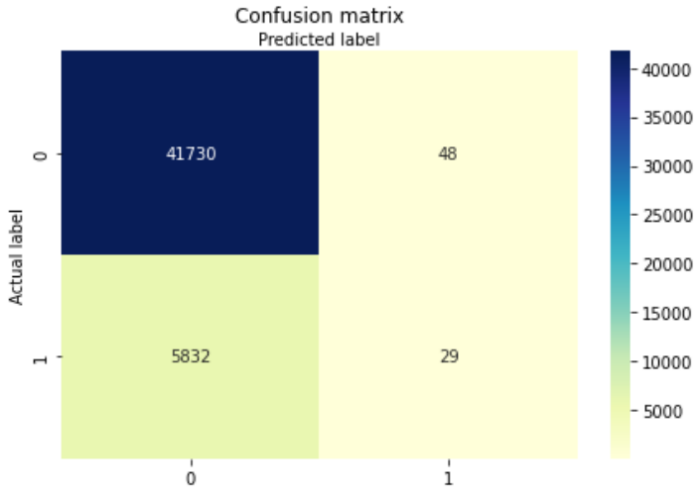
4. Evaluasi

Berdasarkan permodelan yang telah dilakukan sebelumnya, nilai akurasi paling tinggi yaitu ketika menggunakan model Logistic Regression dan Linear Discriminant Analysis dengan nilai akurasi untuk kedua dataset sebesar 0.88. Sehingga evaluasi ini akan memakai model Logistic Regression.

```
pred = logreg.predict(x_test)
cnf = confusion_matrix(y_test, pred)
cnf
```

```
array([[41730,    48],
       [ 5832,    29]])
```

Dilakukan prediksi logistic regression pada x_test yang dimasukkan ke dalam variabel pred. Kemudian dilihat confusion matrix pada y_test dan pred yang dimasukkan ke dalam variabel cnf. Berikut merupakan visualisasi pada confusion matrix.




```
print("Accuracy:",metrics.accuracy_score(y_test, pred))
print("Precision:",metrics.precision_score(y_test, pred))
print("Recall:",metrics.recall_score(y_test, pred))
```

```
Accuracy: 0.8765717164508071
Precision: 0.36231884057971014
Recall: 0.0042654837058522434
```

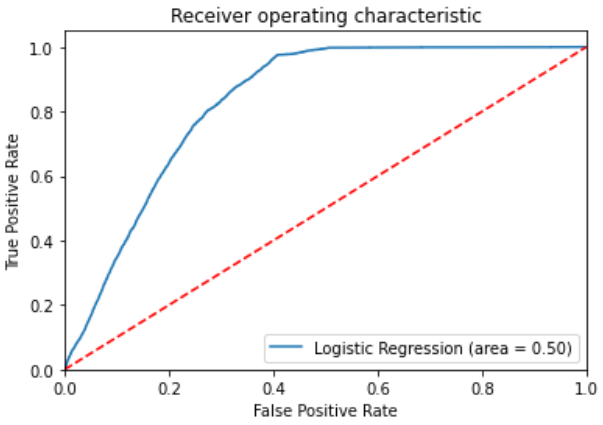
Source code ini untuk melihat akurasi, presisi, dan recall pada y_test dan pred dengan nilai :

- Accuracy : 0.88
- Precision : 0.36
- Recall : 0.004

```
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	0.88	1.00	0.93	41778
1	0.36	0.00	0.01	5861
accuracy			0.88	47639
macro avg	0.62	0.50	0.47	47639
weighted avg	0.81	0.88	0.82	47639

Selanjutnya dilihat juga classification report dari y_test dan pred dan diperoleh hasil yang sama dengan output sebelumnya. Selanjutnya akan dilihat kurva ROC seperti yang terlihat dibawah ini.



Kurva Receiver Operating Characteristic (ROC) merupakan alat umum yang digunakan dengan pengklasifikasi biner. Garis putus-putus mewakili kurva ROC dari pengklasifikasi acak, pengklasifikasi yang baik tetap sejauh mungkin dari garis ROC (menuju sudut kiri atas). Terlihat pada kurva ROC diatas, garis biru menuju sudut kiri atas sehingga pengklasifikasian data test menggunakan Logistic Regression tergolong klasifikasi yang baik.

5. Eksperimen

```
x = df_train[["Jenis_Kelamin", "Umur", "SIM", "Kode_Daerah", "Sudah_Asuransi", "Umur_Kendaraan", "Kendaraan_Rusak", "Premi", "Kanal_Penjualan", "Lama_Berlangganan"]]
y = df_train.loc[:,["Tertarik"]]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

Eksperimen dilakukan dengan melakukan klasifikasi hanya menggunakan data train dengan memanfaatkan library sklearn train_test_split. Data train dibagi menjadi 70% untuk x_train dan y_train, 30% untuk x_test dan y_test. Kemudian dilakukan 5 permodelan klasifikasi untuk melihat akurasi terbaik.

a. Logistic Regression

```
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
print('Accuracy of Logistic regression classifier on training set: {:.2f}'
     .format(logreg.score(x_train, y_train)))
print('Accuracy of Logistic regression classifier on test set: {:.2f}'
     .format(logreg.score(x_test, y_test)))
```

Pada source code Logistic Regression, langkah pertama yaitu memanggil library LogisticRegression yang disimpan ke variabel logreg. Kemudian dimasukkan data x_train dan y_train ke dalam variabel logreg.fit. Selanjutnya dicetak akurasi skor dari data train dan data set.

```
Accuracy of Logistic regression classifier on training set: 0.88
Accuracy of Logistic regression classifier on test set: 0.88
```

Output yang dihasilkan berupa akurasi model logistic regression dengan nilai pada data train yaitu 0.88 dan nilai pada data set yaitu 0.88.

b. Decision Tree

```
clf = DecisionTreeClassifier().fit(x_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(x_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(x_test, y_test)))
```

Pada source code Decision Tree, langkah pertama yaitu memanggil library Decision Tree yang disimpan ke variabel clf. Kemudian dimasukkan data x_train dan y_train ke dalam variabel clf.score. Selanjutnya dicetak akurasi skor dari data train dan data set.

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.82
```

Output yang dihasilkan berupa akurasi model decision tree dengan nilai pada data train yaitu 1.00 dan nilai pada data set yaitu 0.82.

c. K-Nearest Neighbors

```
knn = KNeighborsClassifier()
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)))
```

Pada source code K-Nearest Neighbors, langkah pertama yaitu memanggil library KNeighborsClassifier yang disimpan ke variabel knn. Kemudian dimasukkan data x_train dan y_train ke dalam variabel knn.score. Selanjutnya dicetak akurasi skor dari data train dan data set.

```
Accuracy of K-NN classifier on training set: 0.90
Accuracy of K-NN classifier on test set: 0.86
```

Output yang dihasilkan berupa akurasi model K-Nearest Neighbors dengan nilai pada data train yaitu 0.90 dan nilai pada data set yaitu 0.86.

d. Linear Discriminant Analysis

```
lda = LinearDiscriminantAnalysis()
lda.fit(x_train, y_train)
print('Accuracy of LDA classifier on training set: {:.2f}'
     .format(lda.score(x_train, y_train)))
print('Accuracy of LDA classifier on test set: {:.2f}'
     .format(lda.score(x_test, y_test)))
```

Pada source code Linear Discriminant Analysis, langkah pertama yaitu memanggil library LinearDiscriminantAnalysis yang disimpan ke variabel lda. Kemudian dimasukkan data x_train dan y_train ke dalam variabel lda.fit. Selanjutnya dicetak akurasi skor dari data train dan data set.

```
Accuracy of LDA classifier on training set: 0.88
Accuracy of LDA classifier on test set: 0.88
```

Output yang dihasilkan berupa akurasi model Linear Discriminant Analysis dengan nilai pada data train yaitu 0.88 dan nilai pada data set yaitu 0.88.

e. Gaussian Naive Bayes

```
gnb = GaussianNB()
gnb.fit(x_train, y_train)
print('Accuracy of GNB classifier on training set: {:.2f}'
     .format(gnb.score(x_train, y_train)))
print('Accuracy of GNB classifier on test set: {:.2f}'
     .format(gnb.score(x_test, y_test)))
```

Pada source code Gaussian Naive Bayes, langkah pertama yaitu memanggil library GaussianNB yang disimpan ke variabel gnb. Kemudian dimasukkan data x_train dan y_train ke dalam variabel gnb.fit. Selanjutnya dicetak akurasi skor dari data train dan data set.

```
Accuracy of GNB classifier on training set: 0.70
Accuracy of GNB classifier on test set: 0.70
```

Output yang dihasilkan berupa akurasi model Gaussian Naive Bayes dengan nilai pada data train yaitu 0.70 dan nilai pada data set yaitu 0.70.

f. Evaluasi

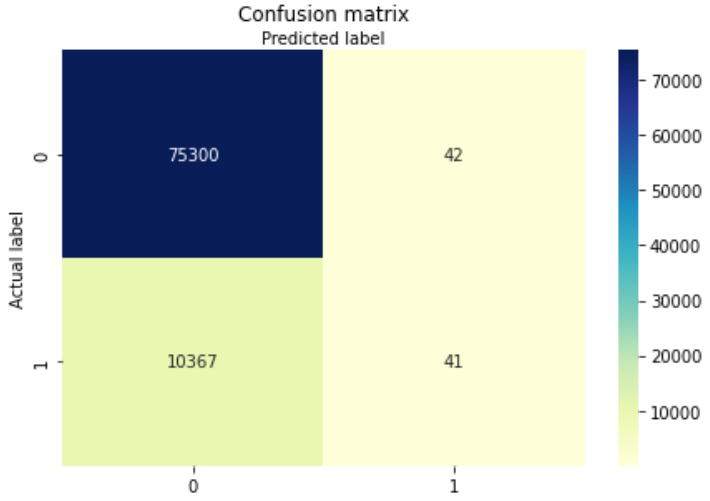
Pada bagian evaluasi diketahui bahwa nilai akurasi paling tinggi yaitu ketika menggunakan model Logistic Regression dan Linear Discriminant Analysis

dengan nilai akurasi untuk kedua dataset sebesar 0.88. Sehingga evaluasi ini akan memakai model Logistic Regression.

```
pred = logreg.predict(x_test)
cnf = confusion_matrix(y_test, pred)
cnf
```

```
array([[75300,    42],
       [10367,    41]])
```

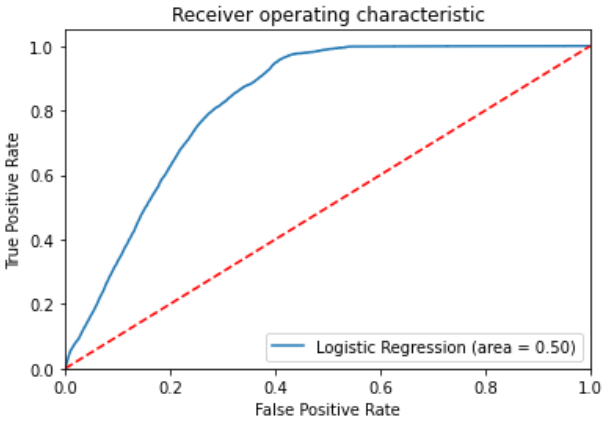
Dilakukan prediksi logistic regression pada x_test yang dimasukkan ke dalam variabel pred. Kemudian dilihat confusion matrix pada y_test dan pred yang dimasukkan ke dalam variabel cnf. Berikut merupakan visualisasi pada confusion matrix.




```
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	0.88	1.00	0.94	75342
1	0.49	0.00	0.01	10408
accuracy			0.88	85750
macro avg	0.69	0.50	0.47	85750
weighted avg	0.83	0.88	0.82	85750

Selanjutnya dilihat juga classification report dari y_test dan pred dan diperoleh hasil yang sama dengan output sebelumnya. Selanjutnya akan dilihat kurva ROC seperti yang terlihat dibawah ini.



Kurva Receiver Operating Characteristic (ROC) merupakan alat umum yang digunakan dengan pengklasifikasi biner. Garis putus-putus mewakili kurva ROC dari pengklasifikasi acak, pengklasifikasi yang baik tetap sejauh mungkin dari garis ROC (menuju sudut kiri atas). Terlihat pada kurva ROC diatas, garis biru menuju sudut kiri atas sehingga pengklasifikasian data test menggunakan Logistic Regression tergolong klasifikasi yang baik.

6. Kesimpulan

- Berdasarkan 5 permodelan klasifikasi yang telah dilakukan terhadap data train dan data test, model Logistic Regression dan Linear Discriminant analysis memiliki nilai akurasi paling tinggi untuk kedua dataset yaitu sebesar 0.88.
- Berdasarkan hasil evaluasi untuk confusion matrix untuk data test menunjukkan true positive sebesar 41730, false positive sebesar 48, false negative sebesar 5832, dan true negative sebesar 29.
- Hasil dari classification report menunjukkan akurasi sebesar 0.88, presisi sebesar 0.38, dan recall sebesar 0.004.
- Berdasarkan kurva ROC, pengklasifikasian data test menggunakan Logistic Regression tergolong klasifikasi yang baik karena garis biru menuju sudut kiri atas menjauhi garis putus-putus (garis ROC).

7. Daftar Pustaka

https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

8. Lampiran

- Google Colab : https://colab.research.google.com/drive/1QFsgRNS4mhjqZxtY18fLA5-DAwJutB--?usp=sharing

- Google Drive : https://drive.google.com/drive/folders/1zPIdvF-ELb7XacDih19I0R4kJ0NS3BS2?usp=sharing