# LAPORAN TAHAP 2 : CLASSIFICATION

Disusun Untuk Memenuhi Tugas Besar Mata Kuliah Pembelajaran Mesin


Disusun Oleh:

1. Kurniadi Ahmad Wijaya (1301194024)
2. Michael Putera Wardana (1301194056)

**PRODI S1 INFORMATIKA**

**FAKULTAS INFORMATIKA**

**UNIVERSITAS TELKOM**

**BANDUNG**

**2021**

# DAFTAR ISI

## 1. Formulasi Masalah

Diberikan dataset kendaraan_train.csv yang berisikan data-data pelanggan pada suatu dealer. Diberikan tugas untuk melakukan classification (*supervised Learning*) dari dataset tersebut dengan mengelompokkan pelanggan berdasarkan data pelanggan di dealer dengan memperhatikan label kelas apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak.

## 2. Eksplorasi dan Persiapan Data

Di Dalam dataset terdapat beberapa macam jenis kolom yang akan digunakan untuk proses clustering yang diantaranya adalah:

| Nama Kolom | Deskripsi |
|---|---|
| SIM | 0: Tidak Punya Sim<br>1: Punya Sim |
| Kode_Daerah | Kode Area Tempat Tinggal Pelanggan |
| Sudah_Asuransi | 1: Pelanggan Sudah Memiliki Asuransi Kendaraan<br>0: Pelanggan Belum Memiliki Asuransi Kendaraan |
| Umur_Kendaraan | Umur Dari Kendaraan |
| Kendaraan_Rusak | 1: Kendaraan Pernah Rusak Sebelumnya.<br>0: Kendaraan Belum Pernah Rusak. |
| Premi | Jumlah Premi Yang Harus Dibayarkan Per Tahun. |
| Kanal_Penjualan | Kode Kanal Untuk Menghubungi Pelanggan |
| Lama_Berlangganan | Lama Pelanggan Menjadi Klien Perusahaan |

Untuk memastikan lebih jauh lagi mengenai jenis data yang akan kita tangani kita dapat melakukan proses EDA (Exploratory Data Analysis) terlebih dahulu.

### 2.1 Proses EDA (Exploratory Data Analysis)

Pada bagian ini kita akan mencari insight dari dataset yang diberikan seluas-luasnya untuk memahami dataset yang diberikan. Adapun beberapa langkah yang dilakukan yaitu:

- **Pengecekan Tipe Data Setiap Kolom**

  Untuk mengenal lebih jauh setiap kolom baik dari segi tipe data, jangkauan data, nilai maximum serta minimumnya kita dapat menggunakan fungsi pada library pandas yaitu dtypes**.** Fungsi ini akan menampilkan berbagai macam detail mengenai keseluruhan column yang akan kita analisis kemudian.

```
df_train = pd.read_csv("Dataset/kendaraan_train.csv")
df_train.drop(['id'], axis=1, inplace=True)

print("Total Dataset :", len(df_train))
df_train.sample(5)
```

✓ 0.6s

```
Total Dataset : 285831
```

| | Jenis_Kelamin | Umur | SIM | Kode_Daerah | Sudah_Asuransi | Umur_Kendaraan | Kendaraan_Rusak | Premi | Kanal_Penjualan | Lama_Berlangganan |
|---|---|---|---|---|---|---|---|---|---|---|
| 81532 | Pria | 72.0 | 1.0 | 28.0 | 0.0 | 1-2 Tahun | Pernah | 46584.0 | 122.0 | 229.0 |
| 215527 | Pria | 25.0 | 1.0 | 28.0 | 0.0 | < 1 Tahun | Pernah | 47206.0 | 163.0 | 48.0 |
| 141356 | Wanita | 24.0 | 1.0 | 2.0 | 0.0 | < 1 Tahun | Pernah | 18940.0 | 152.0 | 115.0 |
| 41521 | Pria | 24.0 | 1.0 | 8.0 | 1.0 | < 1 Tahun | Tidak | 43505.0 | 152.0 | 28.0 |
| 78676 | Pria | 60.0 | NaN | 46.0 | 0.0 | 1-2 Tahun | Pernah | 25347.0 | 124.0 | 50.0 |

```
df_train.dtypes
```

✓ 0.4s

```
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
```

Pada program diatas kita dapat melihat 5 sampel dari isi setiap kolom dataset menggunakan fungsi sample() serta hasil dari pemanggilan fungsi dtypes yaitu jenis tipe data dari setiap kolom yang ada. Diketahui bahwasannya terdapat 7 kolom yang bertipe float yaitu kolom Umur, SIM, Kode_Daerah, Sudah_Asuransi, Premi, Kanal_Penjualan, dan Lama_Berlangganan serta 3 kolom bertipe object atau string yaitu kolom Jenis_Kelamin, Umur_Kendaraan dan Kendaraan_Rusak. Disini terdapat juga kolom Tertarik akan tetapi tidak kita gunakan pada proses klasifikasi.

- **Detail Value Unik Kategorikal Dan Value Numerik Dari Setiap Kolom**

Setelah melakukan pengecekan jenis-jenis tipe data untuk setiap kolom, tentunya untuk menambah pemahaman pada dataset terutama pada data bertipe object kita harus melakukan pengecekan jenis value apa saja yang ada pada setiap kolom. Hal ini dapat kita lakukan pengecekan dengan fungsi value_counts().

Kemudian, meskipun memiliki tipe data float beberapa kolom seperti SIM dan Sudah_Asuransi hanya memiliki value 1 dan 0 yang menandakan sudah atau belum sehingga kolom tersebut termasuk jenis data kategorikal ordinal.

```
df_train['Jenis_Kelamin'].value_counts()
```

✓ 0.5s

```
Pria      146678
Wanita    124713
Name: Jenis_Kelamin, dtype: int64
```

```
df_train['Umur_Kendaraan'].value_counts()
```

✓ 0.6s

```
1-2 Tahun    142761
< 1 Tahun    117378
> 2 Tahun     11417
Name: Umur_Kendaraan, dtype: int64
```

```
df_train['Kendaraan_Rusak'].value_counts()

```

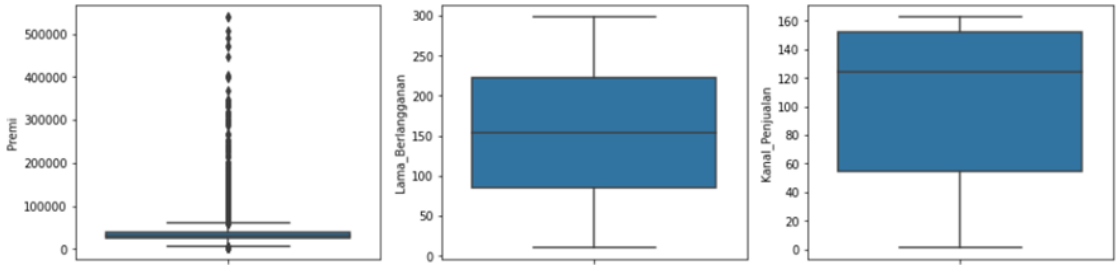
✓ 0.6s

```
Pernah    137123
Tidak     134520
Name: Kendaraan_Rusak, dtype: int64
```

Selanjutnya setelah melakukan detail dari isi data kategorikal kita juga dapat melakukan pengecekan terhadap sebaran isi data numerik dengan menggunakan fungsi describe(). Dapat kita lihat penyebaran data mulai dari jumlah data (count), rerataan (mean), standar deviasi (std), quartil bawah (25%), median (50%), quartil atas (75%) serta nilai maksimal (max) untuk setiap kolom return function tersebut.

- **Pengecekan Distribusi Data Numerik**

  Hal lain yang dapat kita lakukan dalam eksplorasi data yaitu adalah mengecek pendistribusian data pada data numerik. Salah satu hal yang bisa kita dapatkan pada proses ini yaitu mengetahui apakah terdapat outlier pada data atau tidak. Berikut merupakan visualisasi pendistribusian data menggunakan boxplot.




  Dari hasil visualisasi boxplot diatas dapat kita lihat bahwasannya value dari atribut Premi memiliki jangkauan data yang sangat jauh sehingga outlier pada data sangat terlihat. Untuk klasifikasi ini outlier tidak akan dihapus karena outlier sendiri bukanlah suatu noise data. [] Disamping itu data yang berkurang ketika penghapusan outlier cukup banyak yaitu sekitar 55000 data sehingga dapat mengurangi keakuratan dari data yang akan dilatih.

## 2.2 Pra-Pemprosesan Data

Setelah mengetahui bagaimana kondisi dari data yang diberikan maka langkah selanjutnya yang kita lakukan adalah pra-pemrosesan data. Pada proses ini dilakukan beberapa jenis pra-pemrosesan data yaitu pembersihan data (data cleaning) dan transformasi data (data transformation). Proses ini dilakukan untuk menghilangkan data yang tidak lengkap, terdapat noise, dan tidak konsisten sehingga memiliki data yang berkualitas.

- **Pembersihan Data**

  Langkah awal yang harus kita lakukan dahulu dalam proses pembersihan data adalah menghapus data duplikat. Disamping dapat meringankan proses komputasi, penghapusan data duplikasi juga dapat mereduksi noise sehingga dapat meningkatkan kualitas data yang akan di analisis [1]. Untuk menghapus data duplikat kita dapat menggunakan fungsi drop_duplicates pada pandas.

```
1  duplicate = list(df_train.duplicated())
2  print("Data Duplikasi :", duplicate.count(True))
```

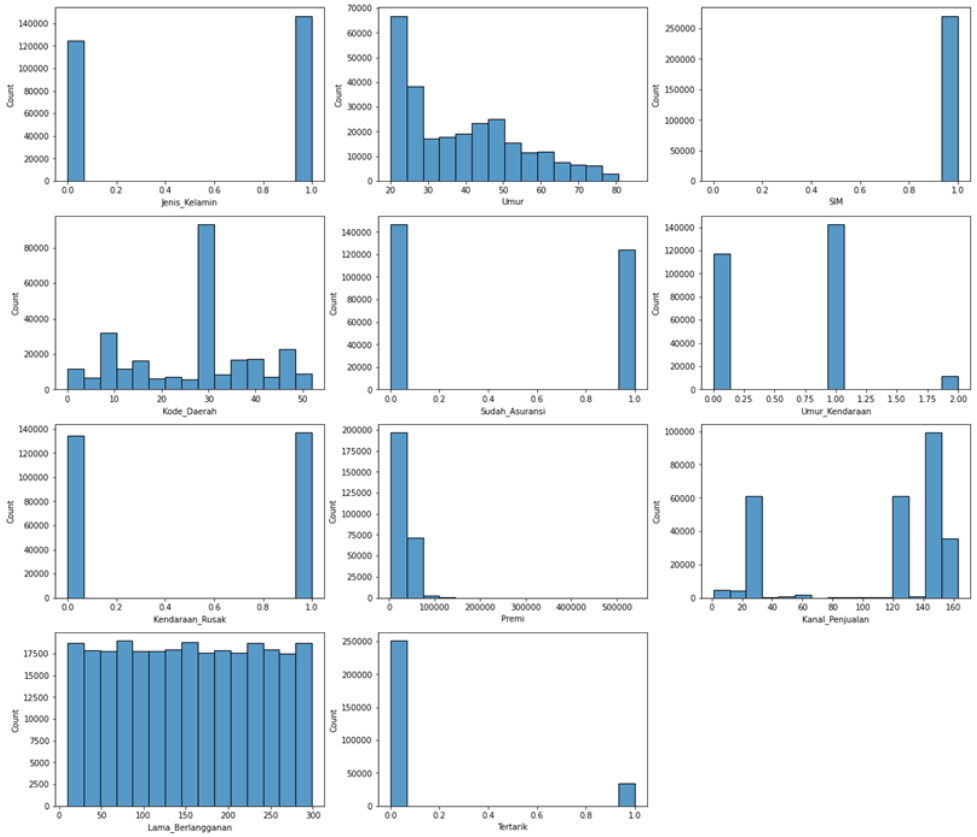
```
Data Duplikasi : 169
```

```
1  df_train.drop_duplicates(inplace=True)
```

```
1  duplicate = list(df_train.duplicated())
2  print("Data Duplikasi :", duplicate.count(True))
```

```
Data Duplikasi : 0
```

Setelah menghapus data yang duplikat selanjutnya kita akan melakukan pengisian data kosong dengan beberapa ukuran pemusatan data seperti mean, median dan mode adapun pengisian data tersebut dilakukan dengan melihat bentuk pendistribusian serta jenis data terlebih dahulu. Berikut merupakan gambar pendistribusian untuk setiap kolom pada data menggunakan distplot.




Dapat kita lihat pada gambar diatas, beberapa kolom memiliki pendistribusian data yang berbeda-beda sehingga penangan selanjutnya pada fase pengisian data untuk setiap kolom akan berbeda-beda. Adapun berdasarkan aturan ketidakseimbangan data (skewness) data kosong berbentuk interval / rasio akan diisi dengan mean, data yang seimbang berbentuk interval / rasio akan diisi dengan Median, sedangkan data nominal akan diisi dengan modus [2]. Adapun kolom SIM, kode_daerah, dan Kanal_Penjualan diisi dengan modus, kolom Lama_Berlangganan diisi dengan mean dan selain itu diisi dengan median. Berikut merupakan pengimplementasian kodenya.

```
# Pengisian Data Sesuai Pendistribusiannya
df_train['SIM'].fillna(float(df_train['SIM'].mode()), inplace=True)
df_train['Kode_Daerah'].fillna(float(df_train['Kode_Daerah'].mode()), inplace=True)
df_train['Kanal_Penjualan'].fillna(float(df_train['Kanal_Penjualan'].mode()), inplace=True)
df_train['Lama_Berlangganan'].fillna(float(df_train['Lama_Berlangganan'].mean()), inplace=True)

# Data Nominal Kategorikal Diisi Menggunakan Modus
for i in df_train.columns:
    df_train[i].fillna(float(df_train[i].median()), inplace=True)
```

- **Transformasi Data**

Transformasi data merupakan salah satu cara menormalkan data dengan merubah skala pengukuran data asli menjadi bentuk lain yang masih memiliki nilai sama sehingga data dapat memenuhi kriteria uji asumsi klasik [3].

Pada transformasi data projek ini dilakukan proses label encoding pada beberapa kolom yang memiliki data kategorikal seperti kolom Jenis_Kelamin, Umur_Kendaraan, dan Kendaraan_Rusak. Proses transformasi ini dilakukan sebelum pengisian data kosong sehingga kedepannya data kategorikal kosong dapat diisi dengan data yang telah ditransformasi. Penggantian label pada setiap kolom tersebut dilakukan dengan menggunakan fungsi pada pandas yaitu replace. Adapun hasil dan kode label encoding dijabarkan sebagai berikut.

```
def label_encoding(df):
    df['Jenis_Kelamin'] = df['Jenis_Kelamin'].replace(['Wanita', 'Pria'], [0, 1])
    df['Umur_Kendaraan'] = df['Umur_Kendaraan'].replace(['< 1 Tahun', '1-2 Tahun', '> 2 Tahun'], [0, 1, 2])
    df['Kendaraan_Rusak'] = df['Kendaraan_Rusak'].replace(['Tidak', 'Pernah'], [0, 1])

label_encoding(df_train)
label_encoding(df_test)

df_train.head()
```

✓ 0.5s

| | Jenis_Kelamin | Umur | SIM | Kode_Daerah | Sudah_Asuransi | Umur_Kendaraan | Kendaraan_Rusak |
|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 30.0 | 1.0 | 33.0 | 1.0 | 0.0 | 0.0 |
| 1 | 1.0 | 48.0 | 1.0 | 39.0 | 0.0 | 2.0 | 1.0 |
| 2 | NaN | 21.0 | 1.0 | 46.0 | 1.0 | 0.0 | 0.0 |
| 3 | 0.0 | 58.0 | 1.0 | 48.0 | 0.0 | 1.0 | 0.0 |
| 4 | 1.0 | 50.0 | 1.0 | 35.0 | 0.0 | 2.0 | NaN |

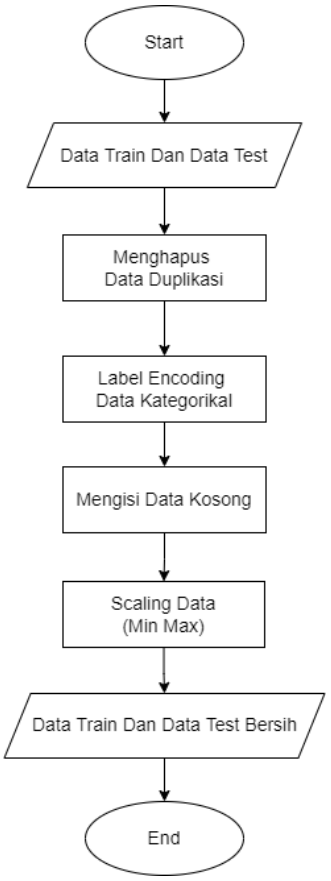
Disamping melakukan label encoding hal lain yang dilakukan pada transformasi data adalah normalisasi. Adapun pada projek ini akan dilakukan penggunaan 2 normalisasi yaitu standard scaler dan min-max scaler sebagai acuan untuk eksperimen mendapatkan hasil maksimal yang terbaik.

```
numerical = ['Lama_Berlangganan', 'Umur', 'Kode_Daerah', 'Kanal_Penjualan', 'Premi']

scaler = MinMaxScaler()

df_train[numerical] = scaler.fit_transform(df_train[numerical].values)
df_train.sample(5)
```

| | Jenis_Kelamin | Umur | SIM | Kode_Daerah | Sudah_Asuransi | Umur_Kendaraan | Kendaraan_Rusak | Premi | Kanal_Penjualan | Lama_Berlangganan | Tertarik |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1378 | 1.0 | 0.430769 | 1.0 | 0.538462 | 0.0 | 1.0 | 1.0 | 0.826262 | 0.154321 | 0.570934 | 1 |
| 144048 | 1.0 | 0.030769 | 1.0 | 0.884615 | 1.0 | 1.0 | 0.0 | 0.552619 | 0.932099 | 0.608997 | 0 |
| 164744 | 0.0 | 0.061538 | 1.0 | 0.326923 | 1.0 | 0.0 | 0.0 | 0.283713 | 0.932099 | 0.342561 | 0 |
| 152517 | 1.0 | 0.230769 | 1.0 | 0.538462 | 0.0 | 1.0 | 1.0 | 0.585480 | 1.000000 | 0.948097 | 0 |
| 226566 | 1.0 | 0.000000 | 1.0 | 0.153846 | 0.0 | 0.0 | 1.0 | 0.609239 | 0.981481 | 0.861592 | 0 |

### 2.3 Keseluruhan Alur Pra-Pemrosesan

Adapun keseluruhan dari alur pemrosesan data pada projek ini dijabarkan melalui flowchart sebagai berikut.




## 3. Pemodelan

Dalam menyelesaikan masalah klasifikasi tugas besar ini akan digunakan beberapa macam model pembelajaran mesin serta metode deep learning untuk menghasilkan model terbaik pada kasus dataset kendaraan ini. Beberapa metode pembelajaran mesin yang digunakan tersebut dijelaskan sebagai berikut.

### 3.1 Naive Bayes

Naive Bayes merupakan sebuah metode klasifikasi yang berakar pada teorema Bayes . Metode pengklasifikasian dengan menggunakan metode probabilitas dan statistik yg dikemukakan oleh ilmuwan Inggris Thomas Bayes, yaitu memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya sehingga dikenal sebagai Teorema Bayes.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

### 3.2 Stochastic Gradient Descent

Stochastic Gradient Descent merupakan salah satu metode yang memiliki optimasi iteratif dalam menemukan titik yang meminimumkan suatu fungsi yang dapat diturunkan.

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

### 3.4 Adaptive Boosting

Adaptive Boosting merupakan salah satu teknik ensemble dengan menggunakan loss function fungsi exponensial untuk memperbaiki tingkat akurasi dari prediksi yang akan dibuat.

$$w(x_i, y_i) = \frac{1}{N}, \; i = 1, 2, \ldots . n$$

$$\frac{1}{2}\log\frac{1 - Total\ Error}{Total\ Error}$$

### 3.4 Random Forest

Random Forest merupakan salah satu metode klasifikasi yang berisi kumpulan dari pohon keputusan (ensemble).

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

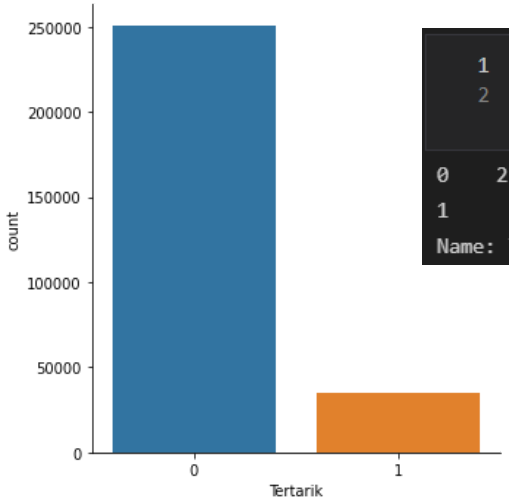
### 3.5 Decision Tree

Decision Tree adalah salah satu metode klasifikasi yang paling populer, karena mudah untuk diinterpretasi oleh manusia. Decision tree adalah model prediksi menggunakan struktur pohon atau struktur berhirarki

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

### 3.1 Proses Pemodelan

Proses pemodelan yang akan dilakukan dimulai dengan melakukan pengecekan target variabel (kolom tertarik) pada dataset train yang ada dengan menggunakan count plot untuk melihat dengan visualisasi yang lebih baik.




```
1  print(df['Tertarik'].value_counts())
2  sns.catplot(x='Tertarik', data=df, kind='count')
```

```
0    250661
1     35001
Name: Tertarik, dtype: int64
```

Dapat terlihat dari hasil visualisasi diatas bahwasannya target variable yang kita miliki dalam dataset train imbalance atau tidak seimbang yaitu value 0 (tidak tertarik) sebanyak 250661 data sedangkan value 1 (tertarik) sebanyak 35001 data. Hal ini akan menyebabkan data yang kita latih pasti akan condong ke value 0 sehingga dibutuhkan penanganan khusus untuk menghasilkan model yang seimbang terhadap kedua target variable tersebut.

Selanjutnya, setelah mengobservasi target variable yang dimiliki langkah selanjutnya yang harus dilakukan sebelum melakukan training model adalah melakukan pemisahan antara feature variable dan target variable pada dataset train. Pemisahan ini dilakukan dengan membuat variabel baru dengan mengisikan dataframe train yang telah dipanggil sebelumnya dengan mendrop target variabel serta dan hanya mengambil kolom target variabel saja.

```
1  x_data = df.drop("Tertarik", axis=1)
2  y_data = df["Tertarik"]
```
✓ 0.5s

```
1  x_data.head()
```
✓ 0.1s

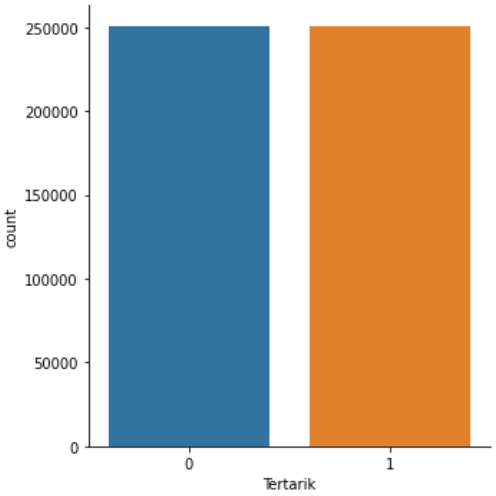
| | Jenis_Kelamin | Umur | SIM | Kode_Daerah | Sudah_Asuransi | Umur_Kendaraan | Kendaraan_Rusak | Premi | Kanal_Penjualan | Lama_Berlangganan |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.153846 | 1.0 | 0.634615 | 1.0 | 0.0 | 0.0 | 0.047251 | 0.932099 | 0.301038 |
| 1 | 1.0 | 0.430769 | 1.0 | 0.750000 | 0.0 | 2.0 | 1.0 | 0.043104 | 0.172840 | 0.512111 |
| 2 | 1.0 | 0.015385 | 1.0 | 0.884615 | 1.0 | 0.0 | 0.0 | 0.056002 | 0.981481 | 0.377163 |
| 3 | 0.0 | 0.584615 | 1.0 | 0.923077 | 0.0 | 1.0 | 0.0 | 0.000000 | 0.759259 | 0.183391 |
| 4 | 1.0 | 0.461538 | 1.0 | 0.673077 | 0.0 | 2.0 | 1.0 | 0.059953 | 0.537037 | 0.636678 |

```
1  y_data.sample(5)
```
✓ 0.6s

```
130685    0
251945    0
86601     0
130677    1
82937     0
Name: Tertarik, dtype: int64
```

Setelah membagi dataset menjadi feature variable (variabel x_data) dan target variable (variabel y_data), dikarenakan adanya imbalance data maka akan lebih baik jika kita lakukan proses oversampling terhadap kelas target paling sedikit. Oversampling merupakan metode sampling yang menambahkan jumlah data pada kelas minoritas sehingga dapat mengimbangi atau mendekati jumlah data pada kelas mayoritas. Pada tugas ini akan digunakan pendekatan Synthetic Minority Over-sampling Technique (SMOTE) untuk melakukan proses oversampling. Adapun proses tersebut dilakukan sebagai berikut.

```
# Melakukan Oversampling
# untuk menyetarakan kelas minoritas
smt = SMOTE(random_state=42, k_neighbors=5)

x_train, y_train = smt.fit_sample(
    x_data, y_data
)

sns.catplot(
    x='Tertarik',
    data=pd.DataFrame(y_train),
    kind='count'
)
```




Setelah keseluruhan dataset dipersiapkan maka langkah terakhir yang akan dilakukan adalah melakukan latihan terhadap dataset tersebut. Dalam tugas ini akan digunakan 5 macam algoritma pembelajaran mesin yang digunakan diantaranya adalah naive bayes, stochastic gradient boosting dengan parameter class_weight balanced, AdaBoost, Decision Tree class_weight balanced, dan Random Forest class_weight balanced. Adapun proses pemodelan tersebut dilakukan sebagai berikut.

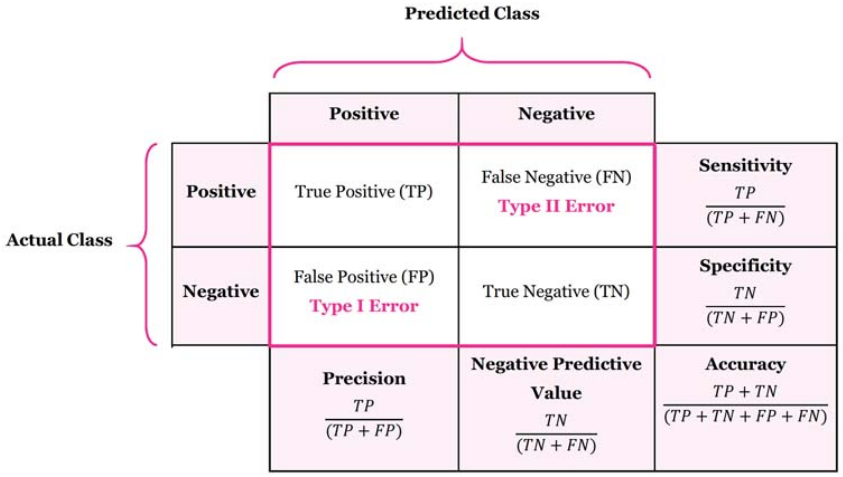
```
# Mendefinisikan Array Untuk Algoritma Machine Learning Yang Digunakan
classifier = [
    DecisionTreeClassifier(class_weight='balanced'),
    SGDClassifier(class_weight='balanced'),
    GaussianNB(),
    AdaBoostClassifier(),
    RandomForestClassifier(class_weight='balanced'),
]

name = [
    'Decision Tree',
    'SGDClassifier',
    'Naive Bayes',
    'AdaBoost',
    'Random Forest',
]

# Melakukan pelatihan terhadap model yang telah didefinisikan sebelumnya
for models in classifier:
    models.fit(x_train, y_train)
```

## 4. Evaluasi

### 4.1 Metrik Yang Digunakan

- **Confusion Matrix**

  Confusion matriks merupakan matriks pembanding hasil klasifikasi yang dilakukan dengan hasil klasifikasi sebenarnya.




- **Precision**

  Precision merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. Metrik ini dipilih jika pengambil keputusan lebih menginginkan terjadinya True Positive dan sangat tidak menginginkan terjadinya False Positive.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

- **Recall**

  Recall merupakan merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Metrik ini dipilih jika pengambil keputusan lebih menginginkan terjadinya lebih memilih False Positive terjadi daripada terjadinya False Negative.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- **F1 Score**

  F1 Score merupakan perbandingan rata-rata presisi dan recall. Skor F1 juga dapat digambarkan sebagai rata-rata harmonis atau rata-rata tertimbang dari precision dan recall. F1 Score dipilih Jika lebih mementingkan recall dan precision yang tinggi atau menginginkan nilai False Positive kecil dan False Negative kecil.

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

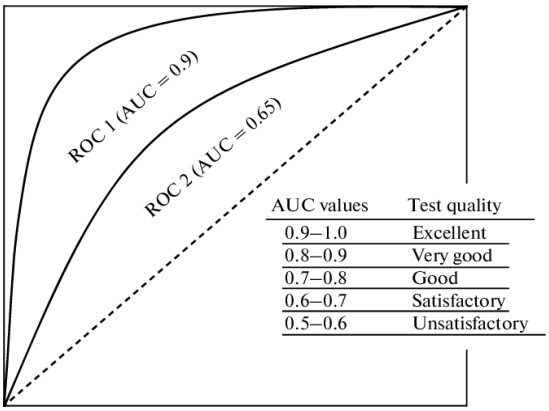
- **Accuracy**

  Accuracy merupakan rasio prediksi Benar (positif dan negatif) dengan pada keseluruhan data. Jika yang dipentingkan adalah seberapa akurat sistem mengklasifikasi dengan benar, maka accuracy merupakan pilihan yang tepat.

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

- **Kurva ROC-AUC**

  Kurva ROC dibuat berdasarkan nilai yang telah didapatkan dari perhitungan dengan confusion matrix, yaitu antara False Positive Rate dengan True Positive Rate. Untuk membandingkan nilai kinerja algoritma dapat dilakukan dengan membandingkan luas di bawah kurva atau AUC (Area Under Curve).




Kelebihan dari penggunaan kurva ROC untuk mengevaluasi klasifikasi adalah ROC bukan sekedar untuk mencari rata-rata akurasi tetapi ROC memvisualisasikan semua threshold klasifikasi yang mungkin, sedangkan error rate classifier hanya mewakili tingkat kesalahan akurasi untuk satu threshold saja.

## 4.1 Hasil Test Model

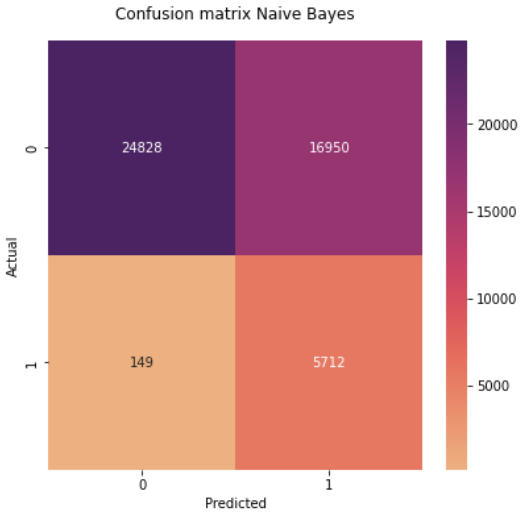
Dari hasil pemodelan yang telah dilakukan, didapatkan data dengan menggunakan beberapa metrik sebagai alat ukur dari model yang digunakan. Adapun hasil evaluasi model dijabarkan pada tabel sebagai berikut.

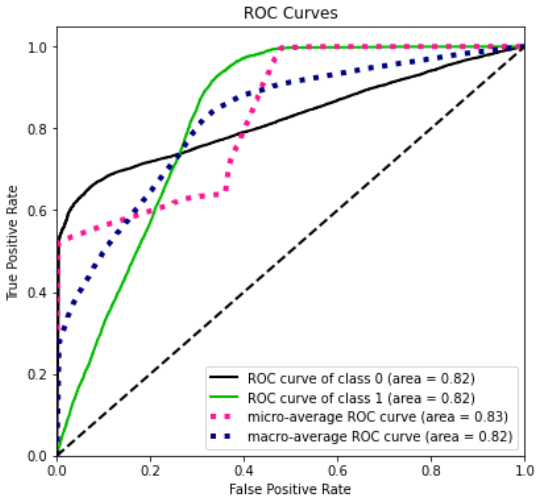
- **Tanpa Over Sampling**

| No | Model | F1-Score | Recall | Precision | ROC-AUC | Accuracy |
|---|---|---|---|---|---|---|
| 1 | SGDClassifier | 0.400 | 0.975 | 0.251 | 0.784 | 0.639 |
| 2 | Naive Bayes | 0.401 | 0.975 | 0.252 | 0.784 | 0.641 |
| 3 | Decision Tree | 0.289 | 0.277 | 0.202 | 0.593 | 0.832 |
| 4 | Random Forest | 0.154 | 0.097 | 0.377 | 0.537 | 0.869 |
| 5 | AdaBoost | 0.000 | 0.000 | 0.000 | 0.500 | 0.877 |

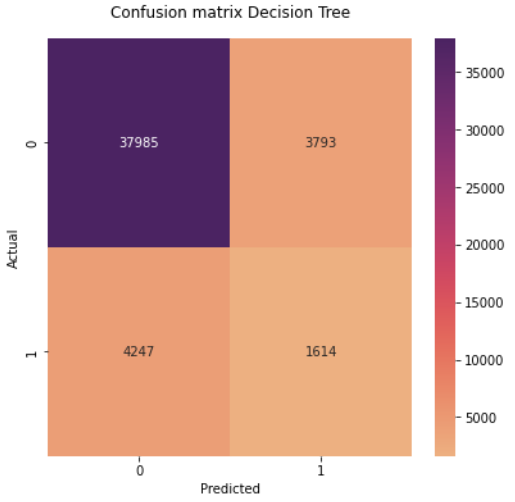
Berikut merupakan hasil dari visualisasi Confusion Matrix dan Kurva ROC dari kelima model pada tabel diatas.

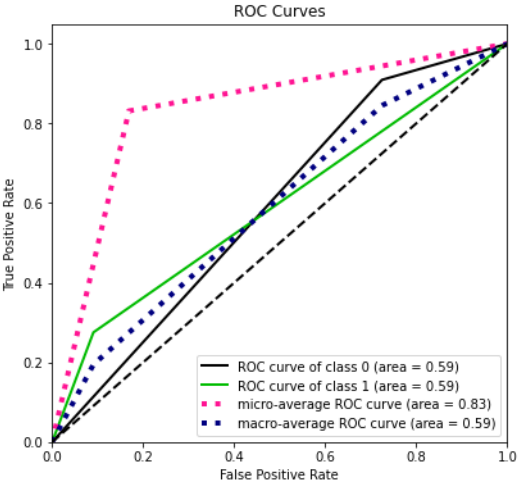
- Naive Bayes



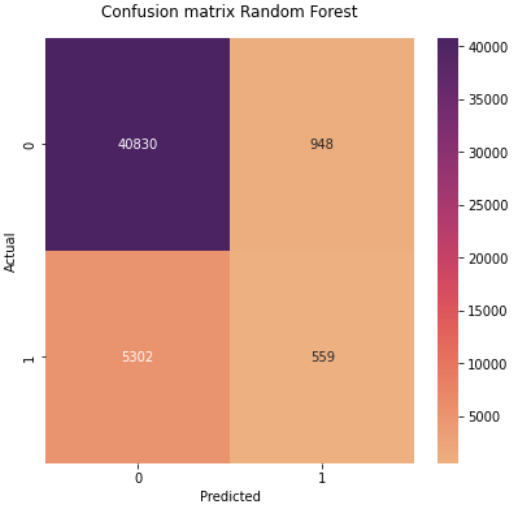



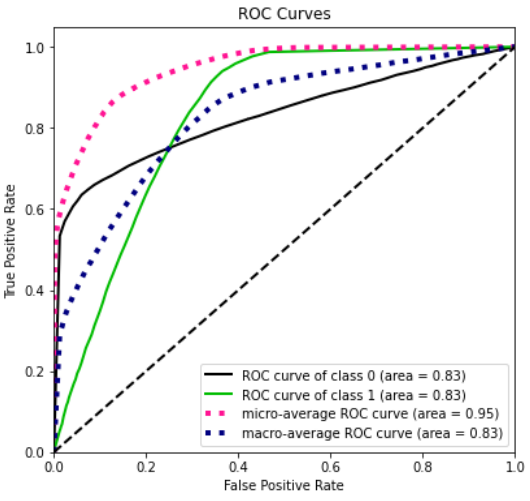

- Decision Tree



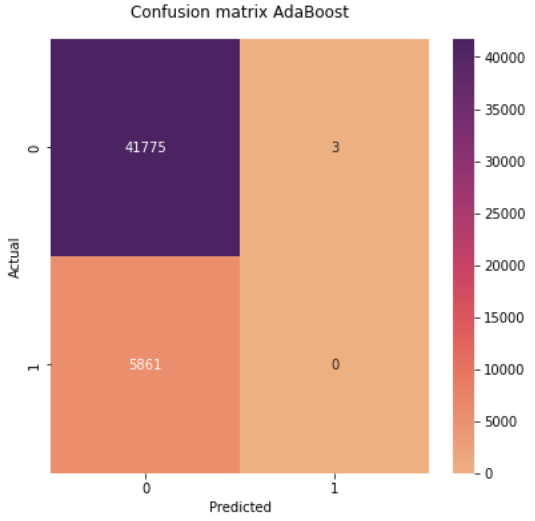



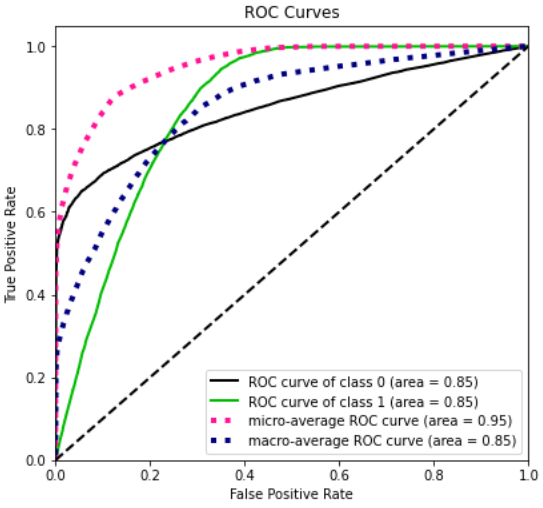

- Random Forest



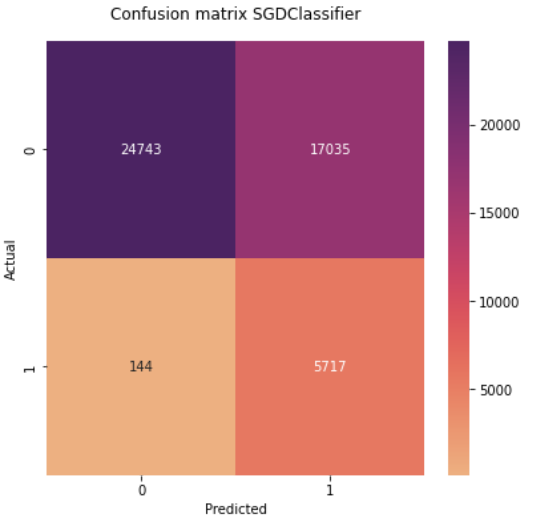



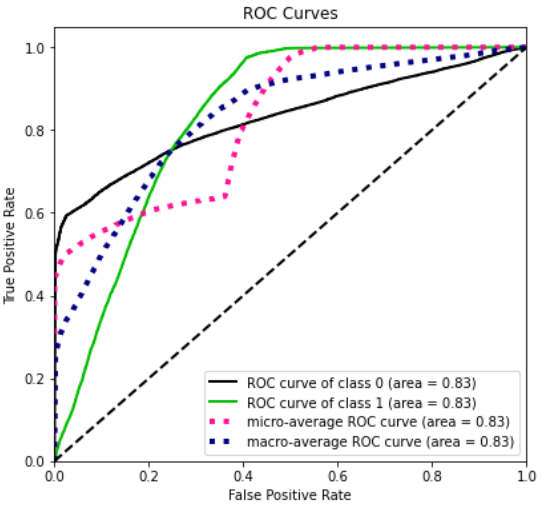
- AdaBoost




- SGDClassifier



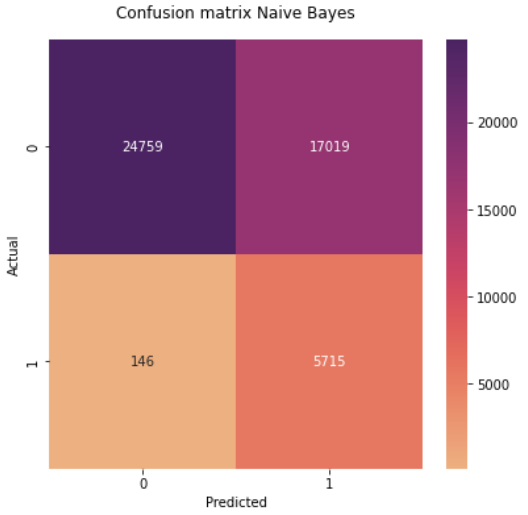

- **Menggunakan Metode SMOTE Oversampling**

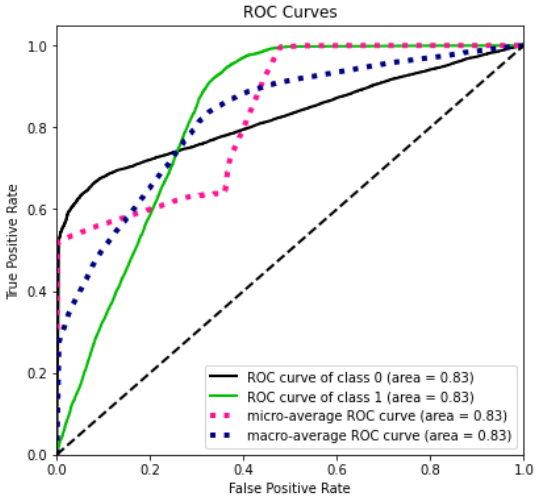
| No | Model | F1-Score | Recall | Precision | ROC-AUC | Accuracy |
|---|---|---|---|---|---|---|
| 1 | SGDClassifier | 0.435 | 0.975 | 0.251 | 0.784 | 0.639 |
| 2 | Naive Bayes | 0.400 | 0.975 | 0.251 | 0.784 | 0.640 |
| 5 | AdaBoost | 0.468 | 0.872 | 0.290 | 0.786 | 0.721 |
| 4 | Random Forest | 0.364 | 0.404 | 0.331 | 0.645 | 0.826 |
| 5 | Decision Tree | 0.310 | 0.337 | 0.287 | 0.610 | 0.815 |

Dari hasil pemodelan dengan menggunakan oversampling nilai dari ROC-AUC AdaBoost memiliki hasil tertinggi karena mayoritas kelas sama maka algoritma AdaBoost dapat mempelajari kelas lain lebih baik dibandingkan model lain. Adapun jika user lebih condong ke nilai true positif maka recall menjadi pilihan utama yaitu dengan model SDG dan Naive Bayes

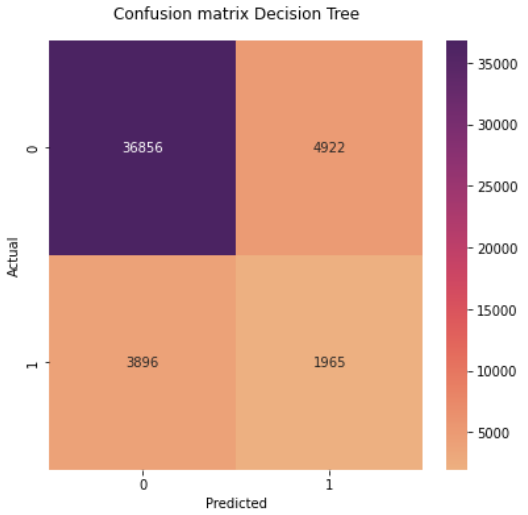
Berikut merupakan hasil dari visualisasi Confusion Matrix dan Kurva ROC dari kelima model pada tabel diatas.

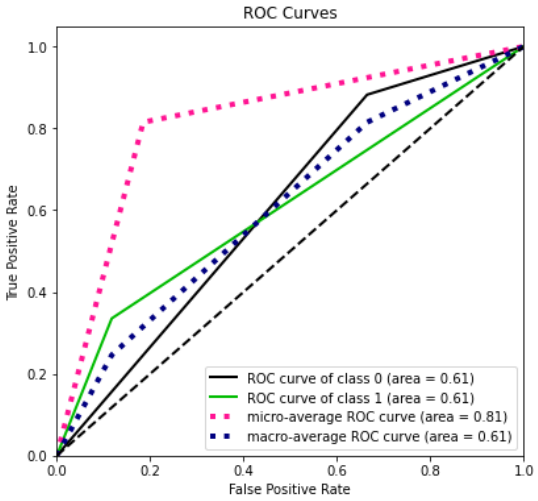
- Naive Bayes



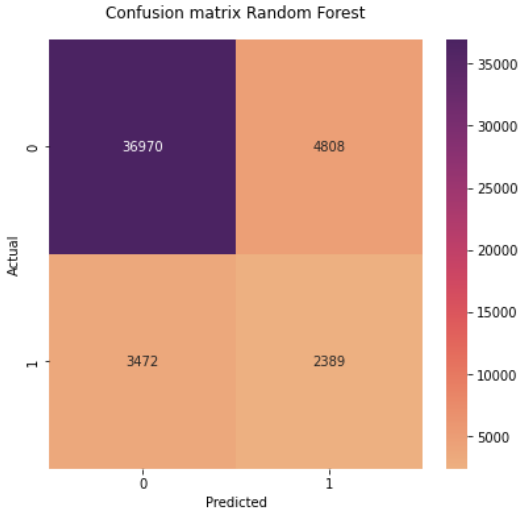



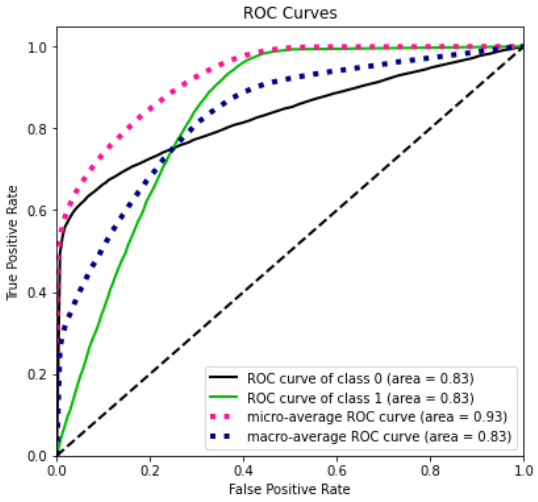

- Decision Tree



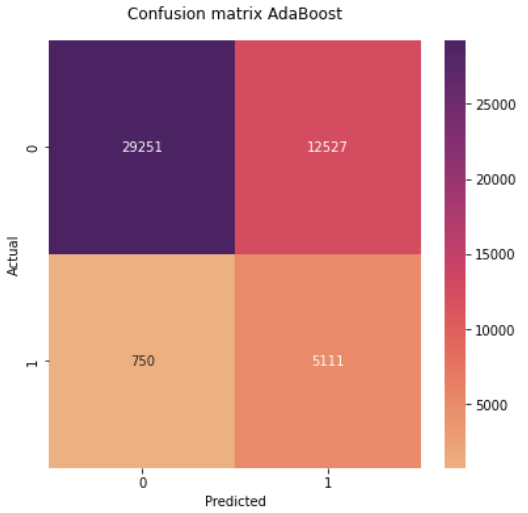



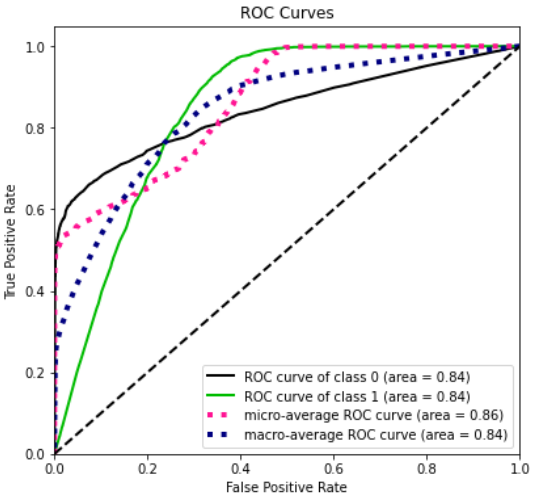
- Random Forest



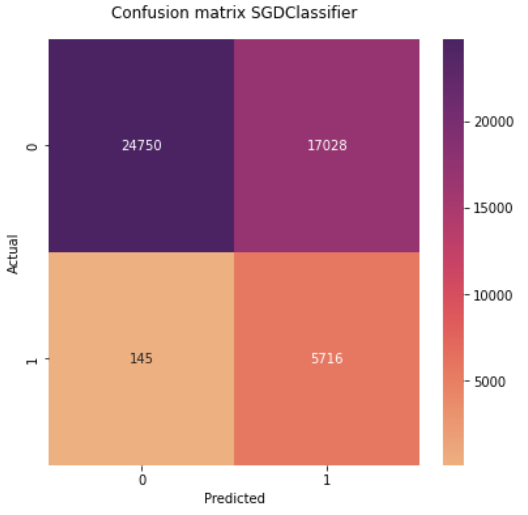



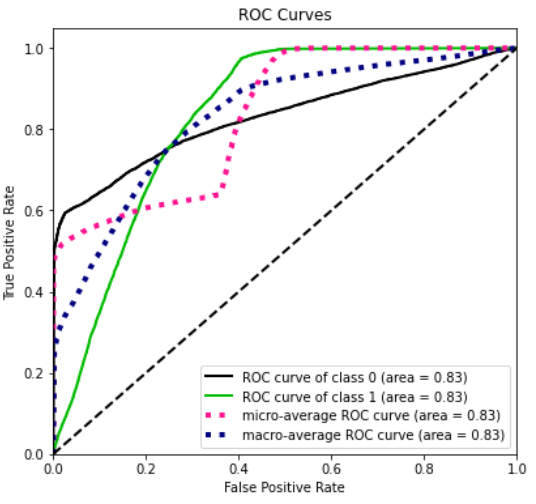

- AdaBoost




- SGDClassifier

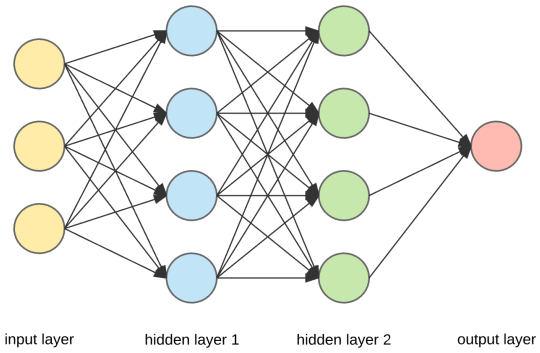


## 5. Eksperimen

### 5.1 Model Deep Learning

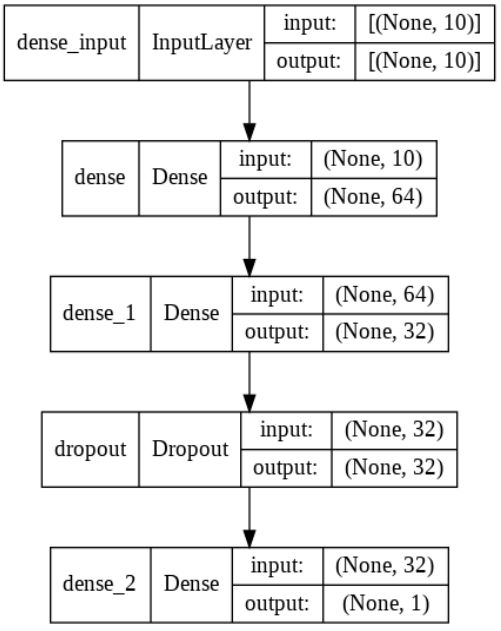
Deep Learning merupakan pengembangan dari Artificial Neural Network atau Jaringan Syaraf Tiruan yang menirukan sistem dasar otak manusia bekerja. Sistem dasar otak manusia bekerja ini disebut neural networks yang dimana selanjutnya akan diimplementasikan dalam konsep deep learning dengan berbagai macam algoritma dan layer berlapis seperti *single layer*, *hidden layer*, dan *output laye*r.




Dalam pengimplementasiannya pada tugas ini akan digunakan library Keras dan Tensorflow sebagai bahan utama untuk melakukan pemodelan deep learning untuk melakukan percobaan sehingga hasilnya dapat dianalisis dan digunakan untuk mengevaluasi maupun dibandingkan dengan model-model *machine learning* sebelumnya.

### 5.2 Arsitektur Model

Model yang digunakan adalah sequential yang artinya dalam membangun model, tiap layer akan disusun layer per layer. Membangun model dengan menggunakan metode sequential hanya terbatas pada permasalahan yang mudah seperti kasus klasifikasi saat ini dan tidak memiliki layer yang terlalu banyak.




| | |
|---|---|
| Dense Input Layer | Layer neural network ini terdapat shape inputan yang masuk berupa (None, 10). Merepresentasikan 10 Kolom feature variable |
| Dense Layer (64), (32) | Merupakan layer tradisional biasa untuk neural network yang akan mengaktifkan fungsi aktivasi ReLU yang menentukan neural network mana saja yang aktif. |
| Dropout Layer | Layer yang berfungsi untuk menonaktifkan beberapa neural network neuron secara random yang tidak diperlukan. |
| Dense Output | Layer yang menyimpulkan hasil dari seluruh inputan terakhir yang pada kasus ini mengoutputkan 1 variabel hasil prediksi. Pada layer ini juga mengaktifkan fungsi aktivasi sigmoid. |

Dalam pembuatan model di atas disebutkan beberapa macam parameter yang digunakan dan beberapa yang belum disebutkan adapun keseluruhan parameter yang digunakan antara lain.

- **Fungsi Aktivasi ReLU**
  ReLu merupakan fungsi aktivasi yang dapat digunakan di mana saja, walaupun pada prosesnya mengubah nilai negatif menjadi 0. fungsi ini merupakan fungsi yang paling populer dan efektif pada proses hidden layer.

- **Fungsi Aktivasi Sigmoid**
  Sigmoid merupakan fungsi aktivasi sigmoid berguna untuk mengubah nilai aktivasi menjadi nilai antara 0 sampai 1. Fungsi ini berguna untuk masalah Klasifikasi Biner dan sebagian besar digunakan sebagai output layer

- **Optimizer Adam**
  Optimizer Adam adalah algoritma optimasi yang dapat digunakan sebagai ganti dari prosedur classical stochastic gradient descent untuk memperbarui bobot secara iteratif yang didasarkan pada data training.

- **Loss Binary Crossentropy**
  Loss Binary Crossentropy merupakan fungsi aktivasi yang digunakan untuk klasifikasi binary, yang mana hanya memiliki 2 pilihan seperti 1 dan 0, ya dan tidak, kanan dan kiri.

Adapun berikut merupakan hasil pengkodean dari arsitektur neural network diatas beserta fungsi aktivasi, metrik evaluasi yang digunakan serta parameternya.

```
def define_model():
    metrics = [
        keras.metrics.BinaryAccuracy(name='accuracy'),
        keras.metrics.Precision(name='precision'),
        keras.metrics.Recall(name='recall'),
        keras.metrics.AUC(name='auc'),
        keras.metrics.AUC(name='prc', curve='PR'),
    ]

    model = Sequential()
    model.add(Dense(64, activation="relu"))
    model.add(Dense(32, activation="relu"))
    model.add(Dropout(0.5)),
    model.add(Dense(1, activation='sigmoid'))

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=1e-3),
        loss=BinaryCrossentropy(),
        metrics=metrics
    )

    return model
```

Selain beberapa parameter yang telah disebutkan, digunakan juga beberapa parameter lainnya saat akan melakukan training data yaitu dengan melakukan train test split sebanyak 20% untuk data validasi dan sisanya 80% untuk training.

```
def transform_data(df):
  X = df.drop('Tertarik', axis=1)
  y = df['Tertarik']

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
  return X_train, X_test, y_train, y_test
```

```
def calculate_class_weight(df):
    total = len(df)

    neg = len(df[df['Tertarik'] == 0])
    pos = len(df[df['Tertarik'] == 1])

    weight_for_0 = (1 / neg) * (total / 2.0)
    weight_for_1 = (1 / pos) * (total / 2.0)
    weights = {0: weight_for_0, 1: weight_for_1}

    return weights


EarlyStop = EarlyStopping(
    monitor='val_roc_auc',
    patience=6,
    mode='max',
    restore_best_weights=True,
    verbose=0
)
```

Proses pembagian dataset train tersebut dilakukan untuk memonitoring hasil validasi skor dari metrik roc_auc pada model yang sedang di train dengan menggunakan sistem callback berupa EarlyStopping.

Sistem Early Stopping ini akan menunggu peningkatan nilai metrik roc_auc maksimum setiap epoch sebanyak 6 kali. Disamping penggunaan callback digunakan juga parameter class weight sebagai bentuk pembobotan karena kelas train yang tidak seimbang.

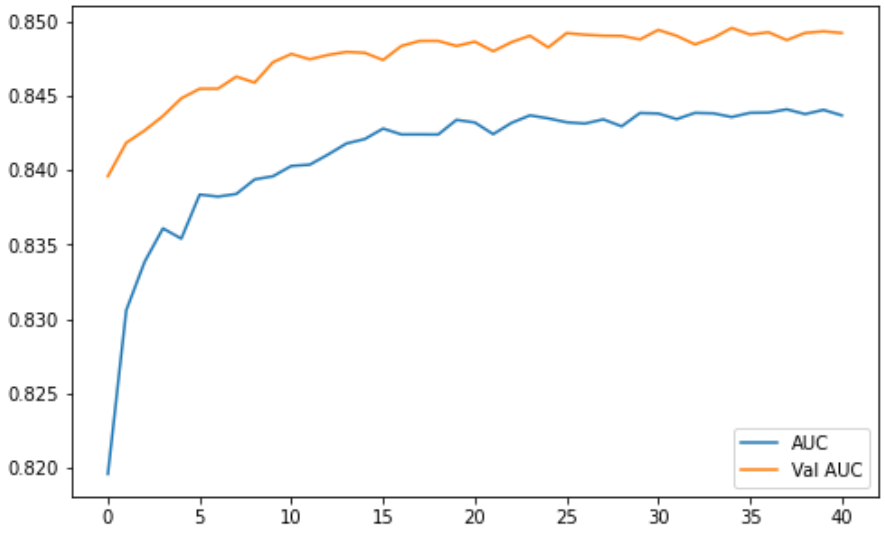
### 5.3 Proses Pemodelan

Setelah seluruh parameter siap maka langkah selanjutnya yang dilakukan adalah melatih model dengan data train dan validation yang telah dipersiapkan dan mengatur nilai epoch sebesar 100 dan batch_size sebesar 256. Adapun proses penempatan parameter dan proses latih dilakukan dan dijabarkan sebagai berikut.

```
# Train Test Split Data
trainX, testX, trainY, testY = transform_data(df_train)

# Mendefinisikan Model Sesuai Parameter Yang Ditentukan
model = define_model()

# Melatih Model
history = model.fit(
    trainX, trainY,
    epochs=100,
    class_weight=calculate_class_weight(df_train),
    verbose=1,
    callbacks=EarlyStop,
    batch_size=256,
    validation_data=(testX, testY)
)
```

```
Epoch 29/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4369 - accuracy: 0.6803 - precision: 0.2682 - recall: 0.9375 - roc_auc: 0.8430 - prc: 0.3329
Epoch 30/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4366 - accuracy: 0.6806 - precision: 0.2683 - recall: 0.9365 - roc_auc: 0.8439 - prc: 0.3366
Epoch 31/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4362 - accuracy: 0.6814 - precision: 0.2687 - recall: 0.9357 - roc_auc: 0.8438 - prc: 0.3358
Epoch 32/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4376 - accuracy: 0.6790 - precision: 0.2674 - recall: 0.9374 - roc_auc: 0.8434 - prc: 0.3344
Epoch 33/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4365 - accuracy: 0.6806 - precision: 0.2683 - recall: 0.9367 - roc_auc: 0.8439 - prc: 0.3345
Epoch 34/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4371 - accuracy: 0.6792 - precision: 0.2676 - recall: 0.9381 - roc_auc: 0.8438 - prc: 0.3353
Epoch 35/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4368 - accuracy: 0.6797 - precision: 0.2679 - recall: 0.9373 - roc_auc: 0.8436 - prc: 0.3352
Epoch 36/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4370 - accuracy: 0.6811 - precision: 0.2685 - recall: 0.9358 - roc_auc: 0.8439 - prc: 0.3348
Epoch 37/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4376 - accuracy: 0.6790 - precision: 0.2673 - recall: 0.9362 - roc_auc: 0.8439 - prc: 0.3363
Epoch 38/100
893/893 [==============================] - 3s 4ms/step - loss: 0.4366 - accuracy: 0.6813 - precision: 0.2687 - recall: 0.9361 - roc_auc: 0.8441 - prc: 0.3370
Epoch 39/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4366 - accuracy: 0.6803 - precision: 0.2681 - recall: 0.9366 - roc_auc: 0.8438 - prc: 0.3353
Epoch 40/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4367 - accuracy: 0.6803 - precision: 0.2680 - recall: 0.9361 - roc_auc: 0.8441 - prc: 0.3361
Epoch 41/100
893/893 [==============================] - 4s 4ms/step - loss: 0.4371 - accuracy: 0.6800 - precision: 0.2680 - recall: 0.9369 - roc_auc: 0.8437 - prc: 0.3345
```

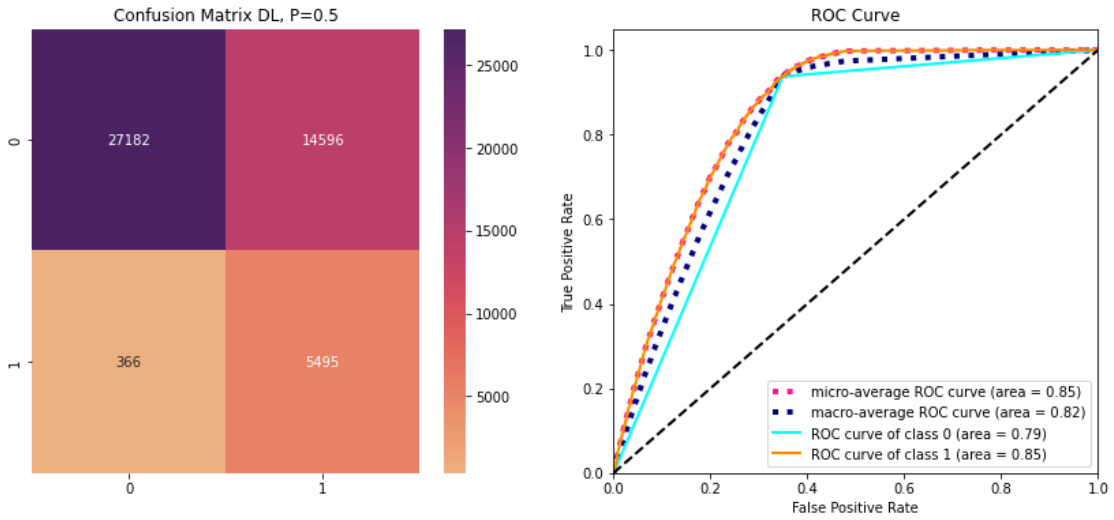
Proses melatih model berhenti pada epoch ke 41 dengan nilai loss sebesar 0.44, accuracy sebesar 0.68, precision sebesar 0.27, recall sebesar 0.94, roc_auc sebesar 0.84 dan roc_precision sebesar 0.33. Sedangkan untuk validasi didapatkan val_loss sebesar 0.47, val_accuracy sebesar 0.69, val_precision sebesar 0.28, val_recall sebesar 0.93, val_roc_auc sebesar 0.85 dan val_roc_precision sebesar 0.35. Adapun hasil evaluasi latih model pada metrik yang utama digunakan pada model ini sebagai acuan terbaik yaitu ROC_AUC dijabarkan melalui visualisasi berikut.



## 5.2 Hasil Test Model

| Model Deep Learning (ANN) | |
|---|---|
| **Metrik** | **Nilai** |
| F1-Score | 0.60 |
| Recall | 0.94 |
| Precision | 0.27 |
| ROC-AUC | 0.85 |
| Accuracy | 0.69 |

- **Confusion Matrix dan Kurva ROC**




Pada hasil test model diatas dapat kita lihat bahwasannya model neural network yang dibuat memiliki akurasi ROC AUC yang paling tinggi jika dibandingkan dengan model pembelajaran mesin saja sebelumnya. Hal ini disebabkan pada model ini kita melakukan proses pembobotan class weight sehingga kedua kelas dapat diprediksi secara seimbang. Begitu juga pada Kurva ROC yang divisualisasikan dapat kita lihat kedua prediksi kelas hampir seimbang yang menandakan seimbangnya prediksi antara kelas 0 dan 1 dalam hasil prediksi test ini

## 6. Kesimpulan

Dalam projek tugas besar tahap kedua ini digunakan beberapa macam metode untuk melakukan klasifikasi data yang terdiri atas 5 model machine learning dan 1 model deep learning. Adapun berdasarkan hasil analisis pada projek didapatkan hasil:

- Terdapat 2 macam metode evaluasi yang dapat digunakan untuk memecahkan masalah evaluasi ini dikarenakan data yang imbalance yaitu dengan lebih condong ke nilai recall untuk mendapatkan true positive lebih banyak dan membiarkan yang tidak tertarik tetap dikategorikan ke tertarik atau menggunakan metrik ROC-AUC untuk mendapatkan keseimbanan antara true positive dan true negative sehingga tidak salah sasaran pelanggan.

- Metode Oversampling dapat meningkatkan nilai metrik evaluasi seperti contoh pada model AdaBoost yang sebelumnya memiliki skor recall dari 0 menjadi 0.87 dan ROC-AUC dari 0.5 menjadi 0.79.

- Pada model machine learning metode AdaBoost mendapatkan skor ROC-AUC terbesar yaitu sebesar 0.79 disusul oleh metode Naive Bayes dan Stochastic Gradient Descent sebesar 0.78.

- Pada model machine learning metode Naive Bayes dan Stochastic Gradient Descent mendapatkan skor recall terbesar yaitu 0.98 sedangkan yang kedua adalah AdaBoost sebesar 0.87

- Model deep learning menjadi model yang paling cocok untuk mengatasi masalah pada kasus tugas besar ini. Hal ini disebabkan, pada data test model ini mendapatkan skor ROC-AUC sebesar 0.85 dan skor recall sebesar 0.94 sehingga sangat seimbang untuk prediksi kedua kelas.

## 7. Sumber Referensi


https://repository.its.ac.id/60454/

https://scikit-learn.org/stable/modules/classes.html

https://vitalflux.com/micro-average-macro-average-scoring-metrics-multi-class-classification-python/

https://towardsdatascience.com/deep-learning-unbalanced-training-data-solve-it-like-this-6c528e9efea6

https://resources.experfy.com/ai-ml/imbalanced-datasets-guide-classification/

https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/

https://medium.com/@opam22/menilik-activation-functions-7710177a54c9

https://keras.io/api/metrics/

https://keras.io/api/layers/

https://arifinrio95.medium.com/memahami-roc-dan-auc-2e0e4f3638bf

https://peltarion.com/knowledge-center/documentation/evaluation-view/measure-performance-when-working-with-imbalanced-data

https://hackernoon.com/simple-guide-on-how-to-generate-roc-plot-for-keras-classifier-2ecc6c73115a

https://repository.uinjkt.ac.id/dspace/bitstream/123456789/55347/1/REZKY%20FIRMANSYAH-FST.pdf