

# **Laporan Final Project**

## **Kecerdasan Komputasional A**

### **Klasifikasi Pengaruh Adanya Transaksi Pembelian Tiket Pesawat pada Terjadinya Booking Hotel**



Dibuat oleh :

Rafid Ferdianto	05111840000032
M. Afif Fadhlurrahman	05111840000093
Alberto Sanjaya	05111840000150

Dosen Pengampu:  
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

Departemen Teknik Informatika  
Fakultas Teknologi Informasi dan Komunikasi  
Institut Teknologi Sepuluh Nopember Surabaya  
Tahun Pelajaran 2020/2021

# Daftar Isi

1. Latar Belakang Masalah
2. Deskripsi Dataset
  - a. Dataset Flight
  - b. Dataset Hotel
  - c. Dataset Test
3. Metode yang Digunakan
  - a. K-Nearest Neighbours
  - b. Decision Tree
  - c. Random Forest
  - d. Gradient Boosting
  - e. MLP Classifier
  - f. Logistic Regression
  - g. Support Vector Machine
4. Skenarion Uji Coba
5. Uji Coba
  - a. Sebelum SMOTE
  - b. Sesudah SMOTE
6. Kesimpulan dan Saran
7. Referensi

# I Latar Belakang Masalah

Di Musim-musim liburan pada umumnya, ada banyak kemungkinan yang tentunya dapat menentukan adanya pemesanan kamar pada pihak penyedia jasa pemesanan kamar hotel. Dimana salah satu yang dapat dijadikan sebagai acuan adalah jika dapat memprediksikan jumlah pemesanan tiket pesawat yang terjadi sehingga dapat dijadikan arahan dalam mengeluarkan suatu kebijakan yang tentunya harus dapat menguntungkan baik pihak perhotelan maupun penyedia jasa tersebut, salah satu contohnya yaitu dengan memprediksikan apakah akan terjadi *cross selling* pada transaksi pemesanan tiket pesawat seorang pelanggan. *Cross selling* disini dapat didefinisikan sebagai pelanggan membuat booking hotel bersamaan juga dengan membeli tiket pesawat terbang.

## II Deskripsi Dataset

### a. Dataset Flight

Dataset Flight berisi data dari transaksi pemesanan tiket pesawat mulai dari tanggal 1 Januari 2018 sampai dengan 31 Desember 2018, data tersebut terletak pada file flight.csv dimana isinya terdiri dari :

Field	Deskripsi
account_id	Unique key dari pengguna tiket.com
order_id	Order id
member_duration_days	Lama sejak pengguna mendaftar
gender	Gender
trip	Jenis perjalanan
service_class	Kelas di penerbangan
price	Harga tiket
is_tx_promo	Indikasi penggunaan promo
no_of_seats	Jumlah kursi yang dipesan
airlines_name	Nama perusahaan penerbangan
route	Rute Penerbangan
payment_day_of_year	Tanggal pembelian tiket
hotel_id	Unique key dari hotel yang dipesan
visited_city	Daftar kota yang pernah didatangi konsumen
log_transaction	Daftar transaksi yang pernah dilakukan konsumen
is_cross_sell	Mengecek apakah konsumen memesan hotel bersamaan dengan tiket pesawat atau tidak

## b. Dataset Hotel

Dataset Hotel berisi data dari hotel yang terdaftar, data tersebut terletak pada file `hotel_master.csv` dimana isinya terdiri dari :

Field	Deskripsi
hotel_id	Unique key dari hotel
starRating	Jumlah bintang sebuah hotel
city	Kota tempat hotel berada
free_wifi	Indikasi fasilitas Wifi gratis
pool_access	Indikasi fasilitas kolam renang
free_breakfast	Indikasi fasilitas sarapan gratis

## c. Dataset Test

Dataset Test berisi data dari test transaksi pemesanan tiket pesawat, data tersebut terletak pada file `test.csv` dimana isinya terdiri dari :

Field	Deskripsi
account_id	Unique key dari pengguna tiket.com
order_id	Order id
member_duration_days	Lama sejak pengguna mendaftar
gender	Gender
trip	Jenis perjalanan
service_class	Kelas di penerbangan
price	Harga tiket
is_tx_promo	Indikasi penggunaan promo
no_of_seats	Jumlah kursi yang dipesan
airlines_name	Nama perusahaan penerbangan
route	Rute Penerbangan
payment_day_of_year	Tanggal pembelian tiket
hotel_id	Unique key dari hotel yang dipesan

visited_city	Daftar kota yang pernah didatangi konsumen
log_transaction	Daftar transaksi yang pernah dilakukan konsumen
is_cross_sell	Mengecek apakah konsumen memesan hotel bersamaan dengan tiket pesawat atau tidak

### III Metode dan Skenario Uji Coba

Dalam final project ini digunakan beberapa jenis metode klasifikasi untuk membuat model, antara lain:

#### a. K-Nearest Neighbours

Algoritma K-Nearest Neighbor (K-NN) adalah sebuah metode klasifikasi terhadap sekumpulan data berdasarkan pembelajaran data yang sudah terklasifikasi sebelumnya. Termasuk dalam supervised learning, dimana hasil query instance yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam K-NN. Digunakan dengan cara, mengimport library sklearn, dan memanggil fungsi `KNeighborsClassifier(n_neighbors=K)` dengan  $K > 0$ .

#### b. Decision Tree

Decision tree adalah salah satu model prediksi menggunakan struktur pohon atau struktur berhirarki. Konsep dari pohon keputusan adalah mengubah data menjadi decision tree dan aturan-aturan keputusan. Digunakan dengan cara mengimport library sklearn, dan memanggil fungsi `DecisionTreeClassifier()`

#### c. Random Forest

Algoritma Random Forest adalah algoritma klasifikasi supervised. Random Forest pengukur meta yang cocok dengan sejumlah pengklasifikasi pohon keputusan pada berbagai sub-sampel dari kumpulan data dan menggunakan rata-rata untuk meningkatkan akurasi prediksi dan kontrol over-fitting. Digunakan dengan cara mengimport library sklearn, dan memanggil fungsi `RandomForestClassifier()`

#### d. Gradient Boosting

Merupakan teknik machine learning untuk masalah regresi dan klasifikasi, yang menghasilkan model prediksi berupa ensemble model prediksi lemah, biasanya Decision Tree. Digunakan dengan cara mengimport library sklearn, dan memanggil fungsi `GradientBoostingClassifier()`

#### e. MLP Classifier

`MLPClassifier` adalah singkatan dari Multi-layer Perceptron classifier yang dalam namanya terhubung ke Neural Network. Tidak seperti algoritme klasifikasi lain seperti Support Vectors atau Naive Bayes Classifier, `MLPClassifier` mengandalkan Neural

Network yang mendasari untuk melakukan tugas klasifikasi. Digunakan dengan cara mengimport library sklearn, dan memanggil fungsi `MLPClassifier()`

## f. Logistic Regression

Merupakan algoritma yang dapat digunakan untuk regresi atau klasifikasi. Regresi Logistik digunakan untuk memprediksi variabel kategori dengan bantuan variabel dependen. Pertimbangan ada dua kelas dan titik data baru harus diperiksa kelas mana yang akan dimilikinya. Kemudian algoritma menghitung nilai probabilitas yang berkisar dari 0 dan 1. Digunakan dengan cara mengimport library sklearn, dan memanggil fungsi `LogisticRegression()`

## g. Support Vector Machine

Support Vector Machine (SVM) merupakan salah satu metode dalam supervised learning yang biasanya digunakan untuk klasifikasi (seperti Support Vector Classification) dan regresi (Support Vector Regression). Dalam pemodelan klasifikasi, SVM memiliki konsep yang lebih matang dan lebih jelas secara matematis dibandingkan dengan teknik-teknik klasifikasi lainnya. SVM juga dapat mengatasi masalah klasifikasi dan regresi dengan linier maupun non-linear. Digunakan dengan cara mengimport library sklearn, dan memanggil fungsi `SVC()`.

Selain metode diatas, terdapat Metode lain yang kami gunakan, adapun metode yang digunakan sebagai berikut :

## h. SMOTE

**Synthetic Minority Oversampling Technique (SMOTE)** Merupakan teknik statistik untuk meningkatkan jumlah kasus dalam kumpulan data Anda secara seimbang. Modul ini bekerja dengan membuat instance baru dari kasus minoritas yang ada yang Anda berikan sebagai input. Penerapan SMOTE ini tidak mengubah jumlah kasus mayoritas.

Cara penggunaan dari K Fold adalah dengan meng-*import* library **Sklearn** dan memanggil fungsi `KFOLD()`

## i. Confussion Matrix

**Confusion Matrix** digunakan untuk membandingkan hasil klasifikasi dari tiap model yang dibuat. Confusion matrix yang juga sering disebut error matrix. Pada dasarnya confusion matrix memberikan informasi perbandingan hasil klasifikasi yang dilakukan oleh sistem (model) dengan hasil klasifikasi sebenarnya. Confusion matrix berbentuk tabel matriks yang menggambarkan kinerja model klasifikasi pada serangkaian data uji yang nilai sebenarnya diketahui. Gambar dibawah ini merupakan



confusion matrix dengan 4 kombinasi nilai prediksi dan nilai aktual yang berbeda. Perhatikan gambar dibawah ini:

		Actual Values	
		1 (Positive)	0 (Negative)
Predicted Values	1 (Positive)	<b>TP</b> (True Positive)	<b>FP</b> (False Positive) <i>Type I Error</i>
	0 (Negative)	<b>FN</b> (False Negative) <i>Type II Error</i>	<b>TN</b> (True Negative)

- True Positive (TP), Merupakan data positif yang diprediksi benar. Contohnya, pasien menderita kanker (class 1) dan dari model yang dibuat memprediksi pasien tersebut menderita kanker (class 1).
- True Negative (TN), Merupakan data negatif yang diprediksi benar. Contohnya, pasien tidak menderita kanker (class 2) dan dari model yang dibuat memprediksi pasien tersebut tidak menderita kanker (class 2).
- False Postive (FP) — Type I Error, Merupakan data negatif namun diprediksi sebagai data positif. Contohnya, pasien tidak menderita kanker (class 2) tetapi dari model yang telah memprediksi pasien tersebut menderita kanker (class 1).
- False Negative (FN) — Type II Error, Merupakan data positif namun diprediksi sebagai data negatif. Contohnya, pasien menderita kanker (class 1) tetapi dari model yang dibuat memprediksi pasien tersebut tidak menderita kanker (class 2)

Cara penggunaan dari Confussion Matriks adalah dengan meng-*import* library **Sklearn** dan memanggil fungsi `confussion_matrix()`

## j. K Fold

Serta Kami juga menggunakan **K Fold** dimana K fold merupakan metode statistik yang digunakan untuk memperkirakan keterampilan model pembelajaran mesin. Kfold bekerja dengan membagi data menjadi beberapa lipatan dan memastikan bahwa setiap lipatan digunakan sebagai set pengujian di beberapa titik.

Cara penggunaan dari K Fold adalah dengan meng-*import* library **Sklearn** dan memanggil fungsi `KFOLD()`

## IV Eksplorasi Data

### a. Link Colab yang digunakan

[https://colab.research.google.com/drive/17Z\\_bGePYrzga6rebYWotoalFpP3am2Cg?usp=sharing](https://colab.research.google.com/drive/17Z_bGePYrzga6rebYWotoalFpP3am2Cg?usp=sharing)

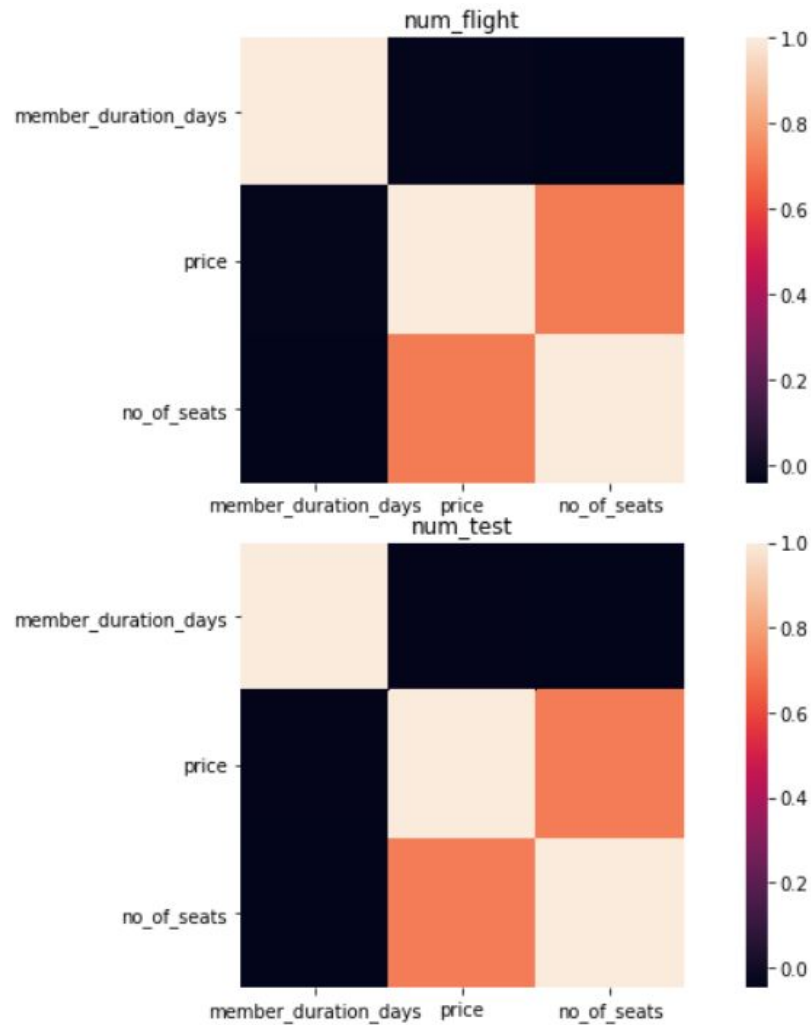
### b. Preprocessing dan Informasi dari setiap dataset

- Dataset Flight memiliki total 117946 record yang terbagi menjadi 14 kolom
- Dataset Hotel memiliki total 2962 record yang terbagi menjadi 6 kolom
- Dataset Test memiliki total 10000 record yang terbagi menjadi 13 kolom
- Pada setiap dataset juga tidak memiliki record yang terduplikasi dan juga *missing value*
- Pada tiap dataset mengalami pengurangan jumlah kolom yang tidak digunakan dengan cara di drop dengan rincian sebagai berikut :
  - Kolom Dataset Flight : 'account\_id' & 'route'
  - Kolom Dataset Hotel : 'city'
  - Kolom Dataset Test : 'account\_id' & 'route'
- Pemberian index berdasarkan unique value yang terdapat pada setiap dataset dengan rincian sebagai berikut :
  - Index dataset Flight : 'order\_id'
  - Index dataset Hotel : 'hotel\_id'
  - Index dataset Test : 'order\_id'

### c. Analisis Data

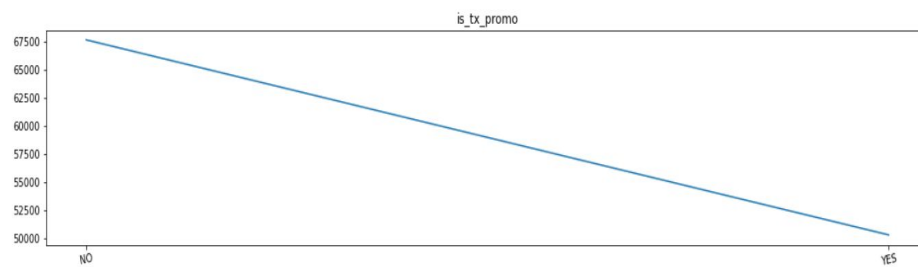
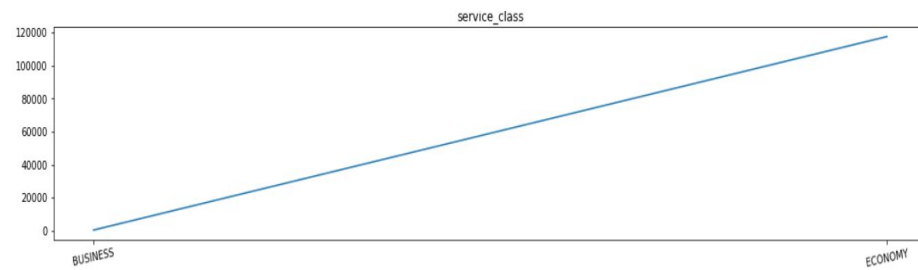
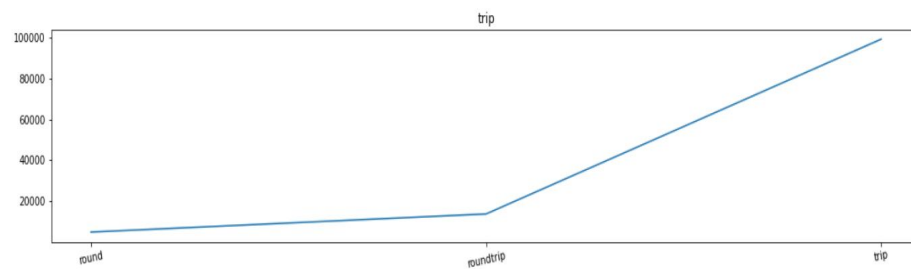
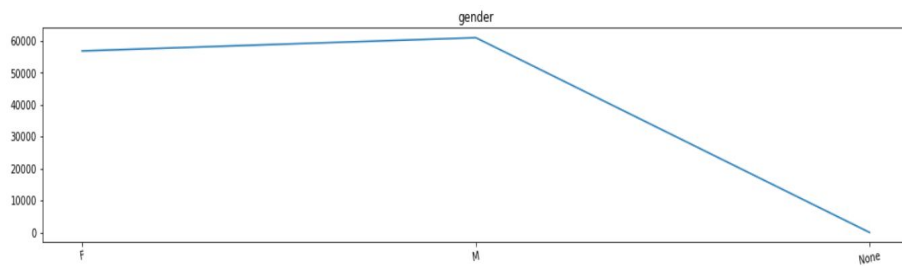
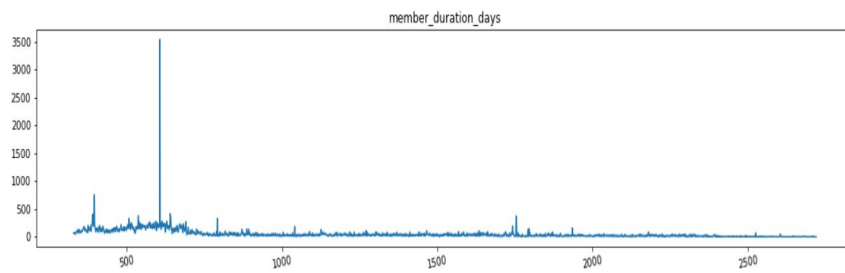
- Mengecek data dari setiap dataset sehingga didapatkan asumsi perubahan berdasarkan informasi dari setiap dataset sebagai berikut
  - Flight :
    - 3 tipe data String yang merepresentasikan hash value yang terdiri dari : 'account\_id', 'order\_id', 'airlines\_name'
    - 6 tipe data String biasa yang terdiri dari : 'gender', 'trip', 'service\_class', 'is\_tx\_promo', 'hotel\_id', 'route'
    - 2 tipe data String yang merepresentasikan daftar yang terdiri dari : 'visited\_city', 'log\_transaction'
    - 3 tipe data Integer yang terdiri dari : 'member\_duration\_days', 'price', 'no\_of\_seats'
  - Hotel :
    - 4 String yang terdiri dari : 'free\_wifi', 'pool\_access', 'free\_breakfast'
    - 1 Integer yang terdiri dari : 'starRating'
    - 1 tipe data String yang merepresentasikan hash value yang terdiri dari : 'hotel\_id'
  - Test :

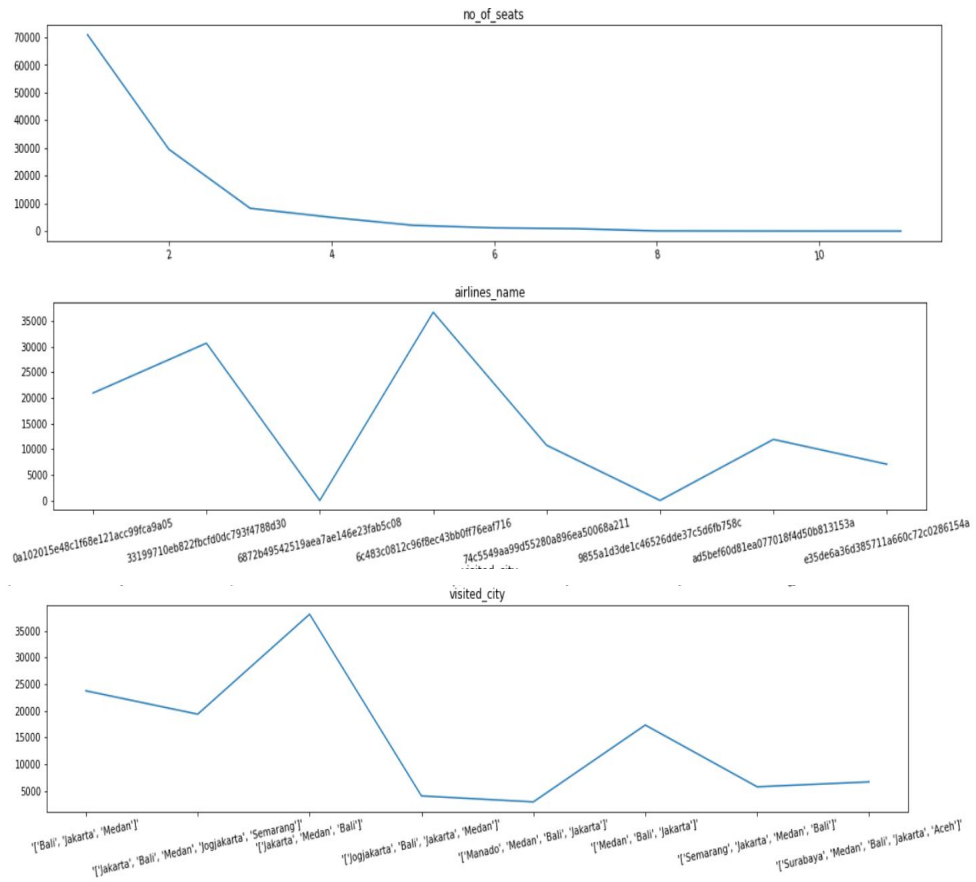
- 3 String yang merepresentasikan hash value yang terdiri dari :  
'airlines\_name', 'account\_id', 'order\_id'
- 5 String biasa yang terdiri dari : 'gender', 'trip', 'service\_class',  
'is\_tx\_promo', 'route'
- 2 String yang merepresentasikan daftar yang terdiri dari :  
'visited\_city', 'log\_transaction'
- 3 Integer yang terdiri dari : 'member\_duration\_days', 'no\_of\_seats',  
'price'
- Mengecek korelasi antar fitur dengan menggunakan analisis *heatmap*



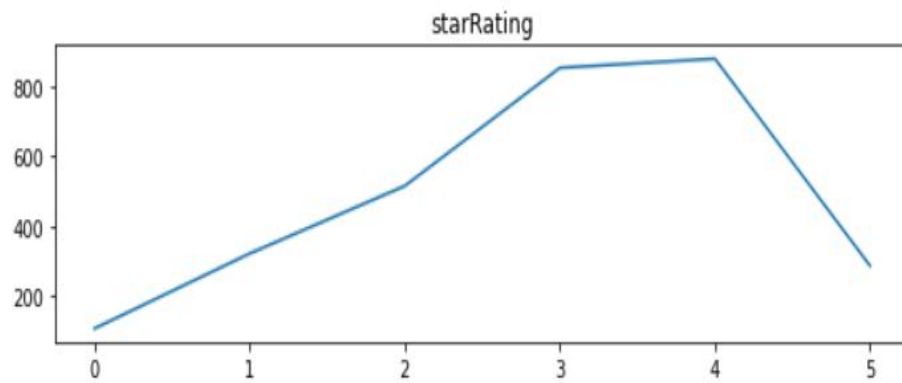
- Mengecek frekuensi dari *unique values* dari setiap fitur per dataset

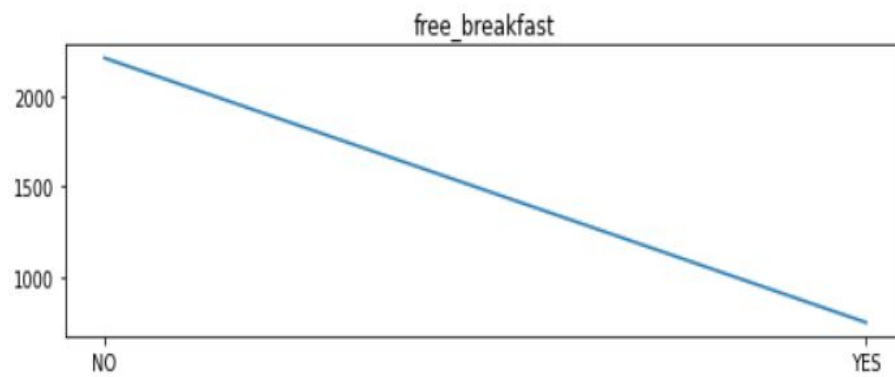
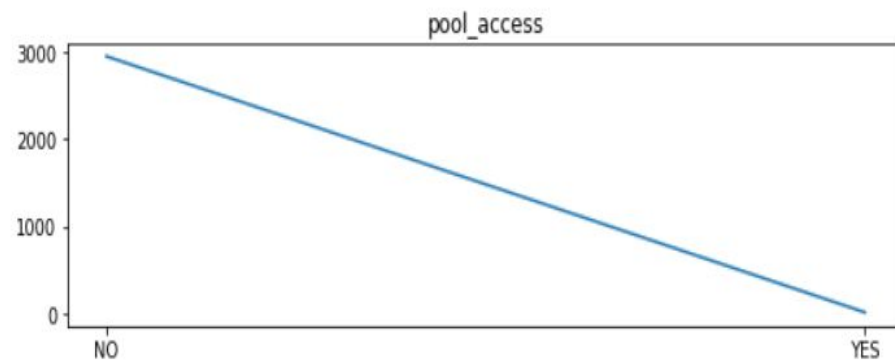
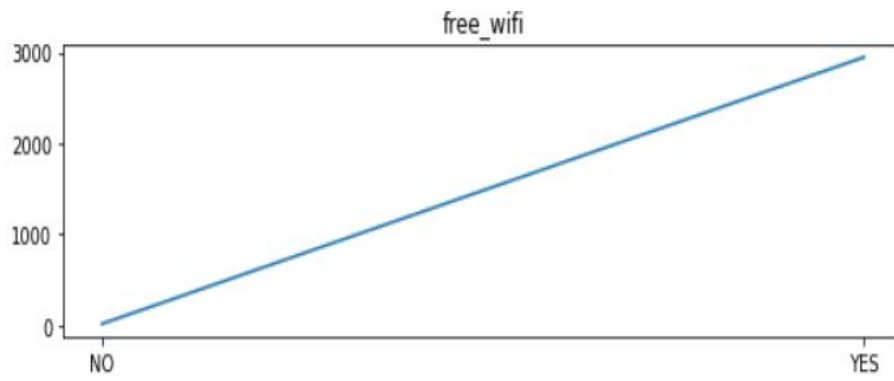
- Dataset Flight



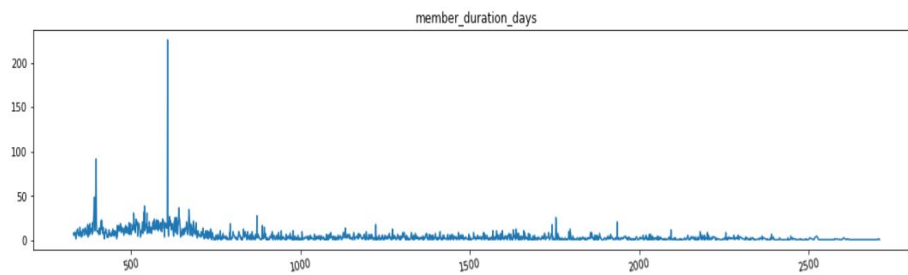


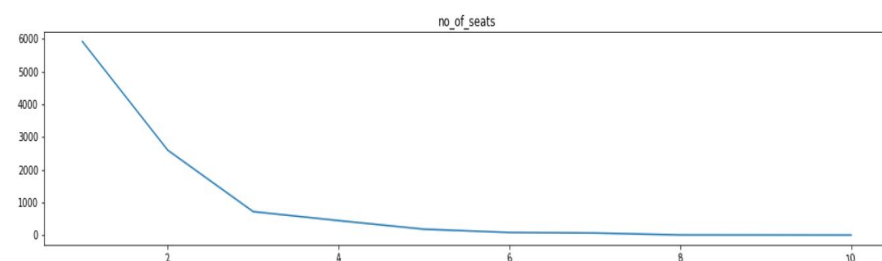
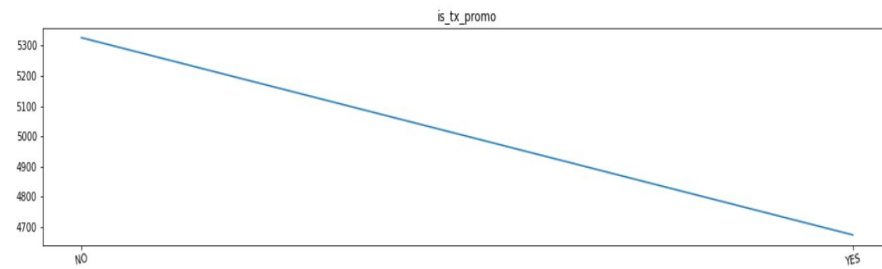
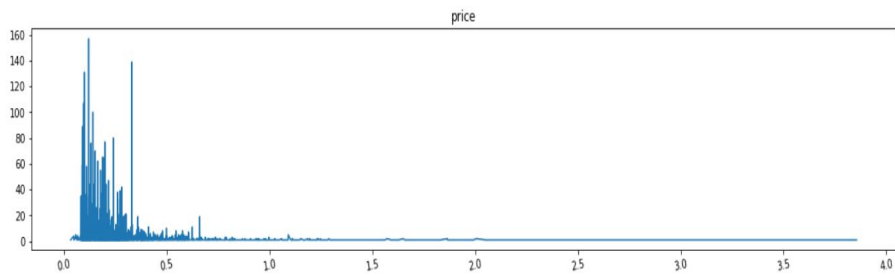
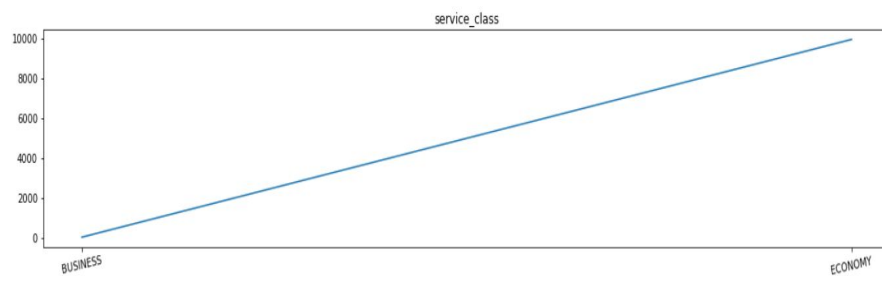
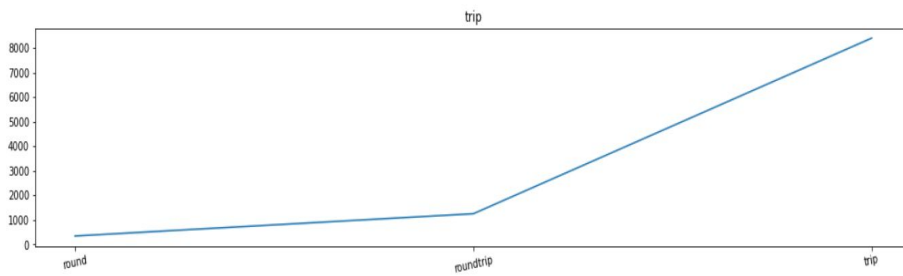
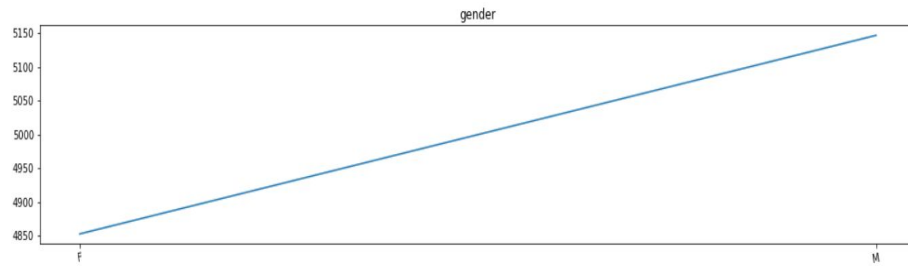
# ○ Dataset Hotel

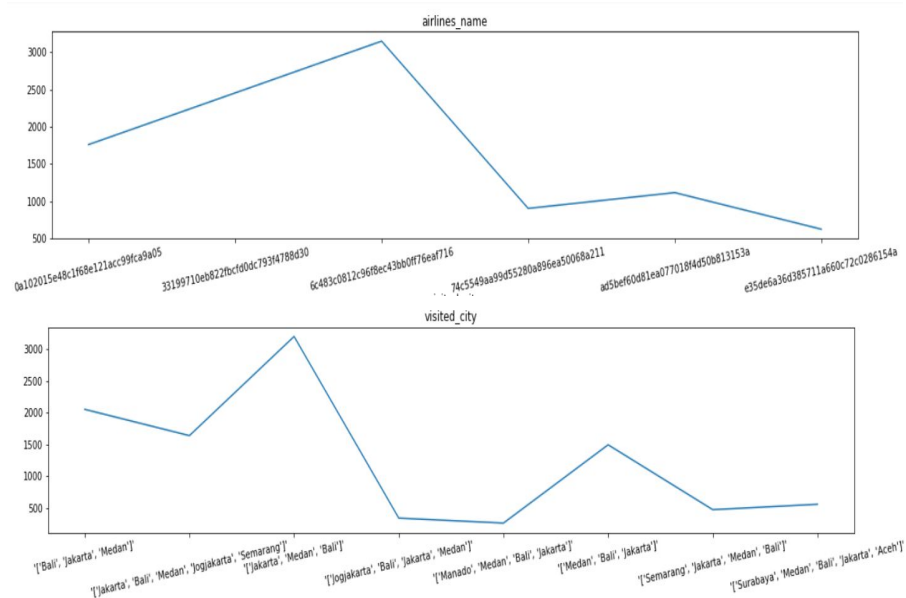




- Dataset Test







Dari hasil mengecek frekuensi dari setiap *unique value* didapatkan hal-hal berikut :

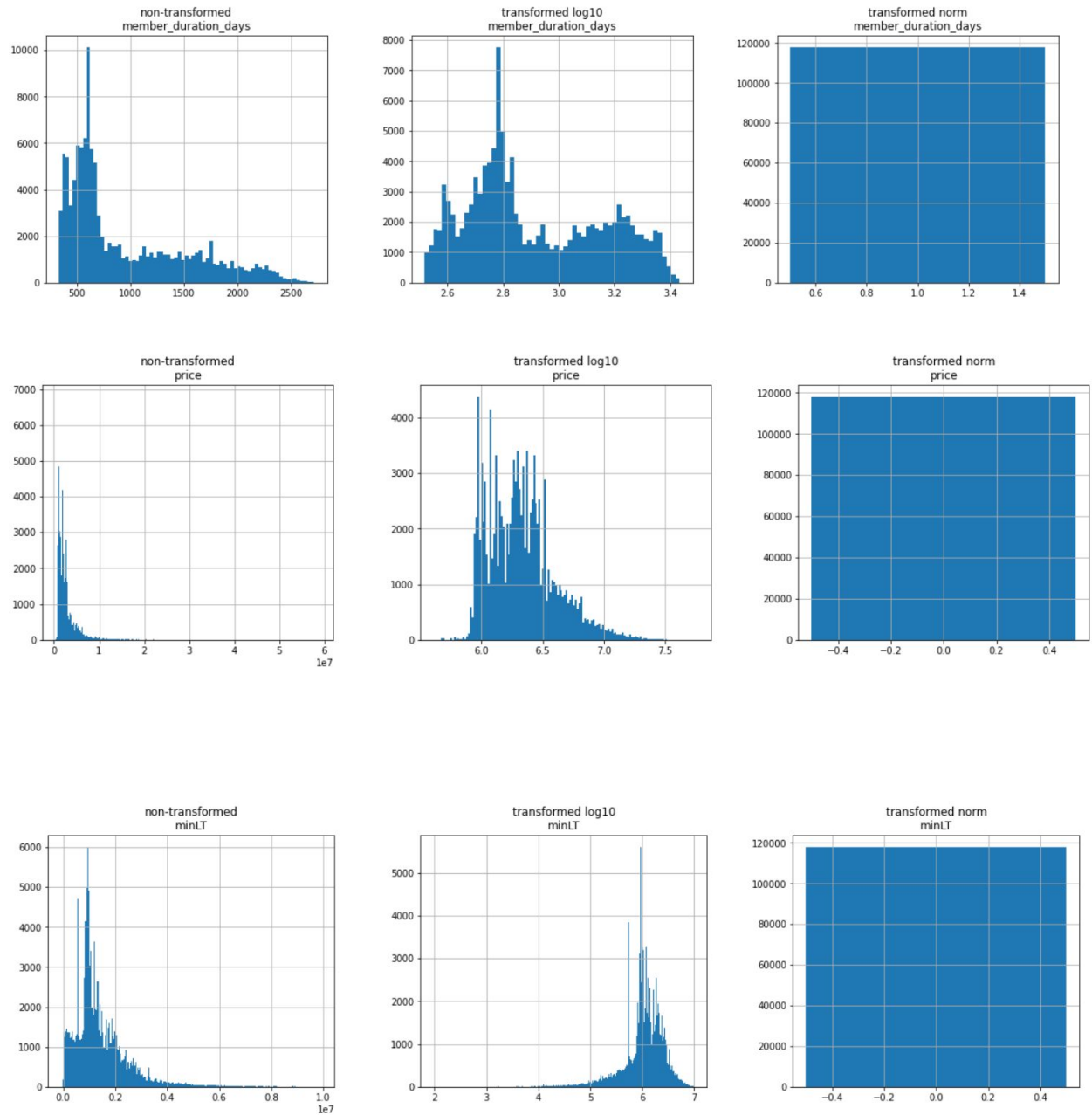
- Tidak ada 'gender' pada dataset Flight yang perlu digantikan
- Baik 'round' maupun 'roundtrip' pada dataset Flight dan Test perlu disamakan menjadi 1 nilai
- Pada bagian 'visited\_city', 'Jakarta','Bali','Medan' selalu muncul di semua variasi yang ada

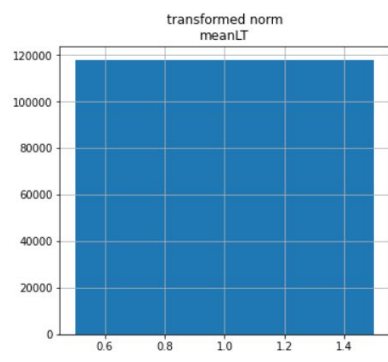
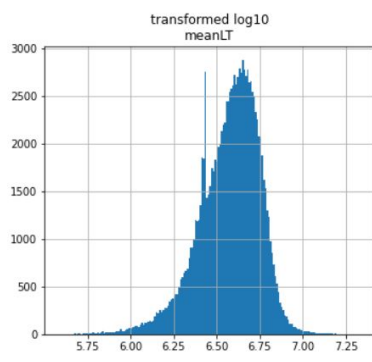
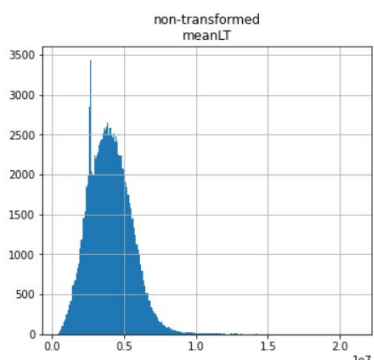
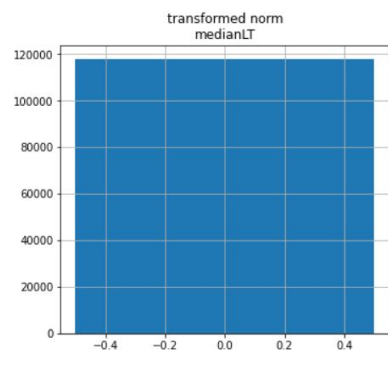
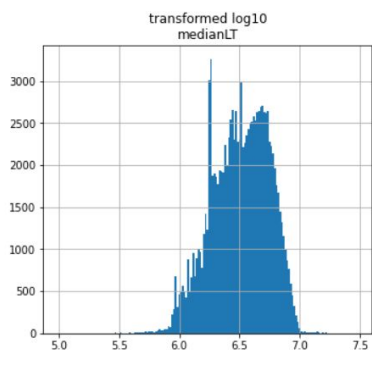
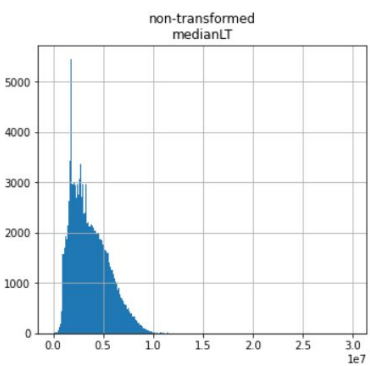
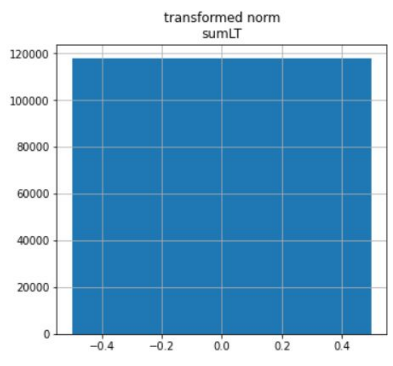
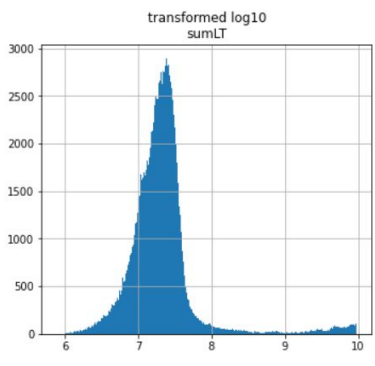
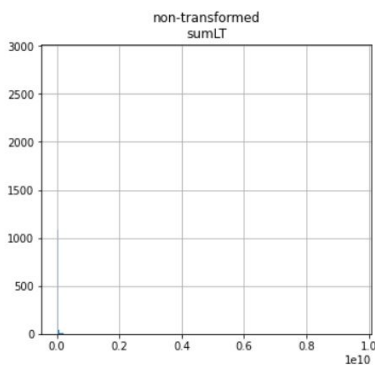
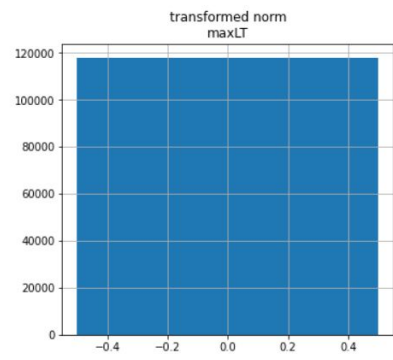
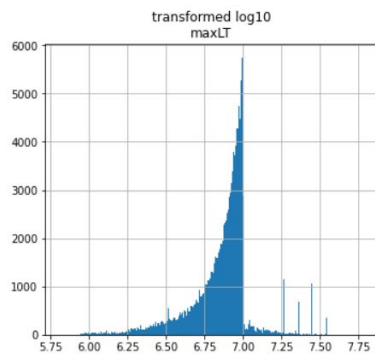
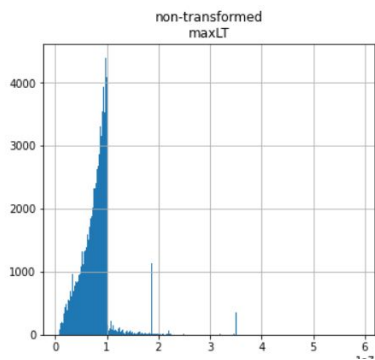
#### d. Feature Engineering

- Membuat min, max, sum, median dan mean dari float dalam 'log\_transaction'
- Memisahkan nilai 'visited\_city' menjadi fitur yang berbeda dengan *one-hot encoding*  
Teknik encoding seperti ini sangat penting untuk categorical data sebelum data diproses dalam machine learning.  
Hal ini dikarenakan machine learning tidak bisa menerima bentuk inputan string dan hanya bisa dalam bentuk data numerik.
- Mengatasi nilai "None" di Flight pada kolom 'gender'  
Hal ini diatasi dengan mengganti nilai gender "None" dengan nilai gender yang paling banyak, yaitu "M".
- Mengatasi nilai 'round' dan 'roundtrip' di dataset Flight and Test  
Hal ini diatasi dengan mengganti nilai "roundtrip" dengan nilai "round".
- Menggabungkan dataset Hotel pada Dataset Flight dan Dataset Test  
Hal ini dilakukan dengan meng-*concat* dataset hotel ke penerbangan, agar datanya dapat diolah sesuai keinginan.

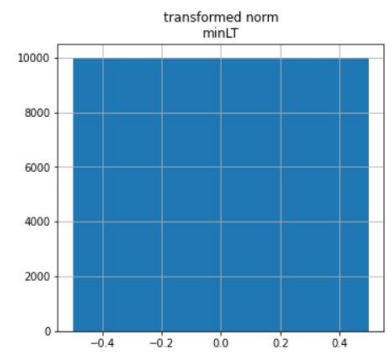
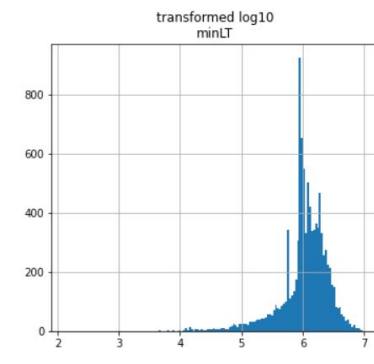
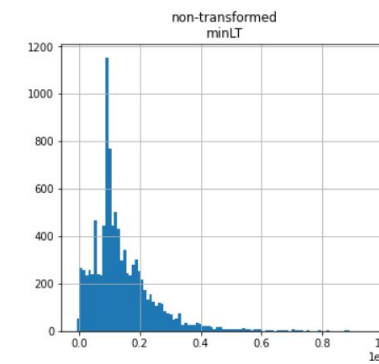
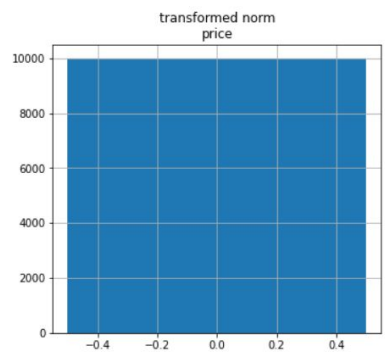
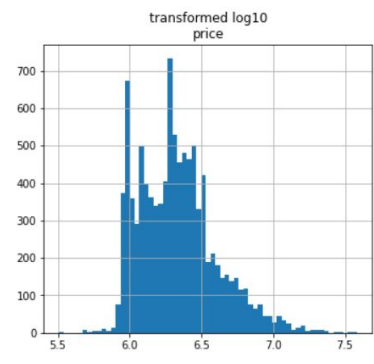
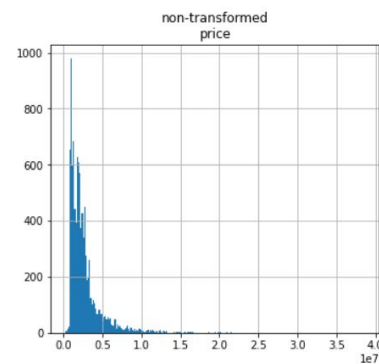
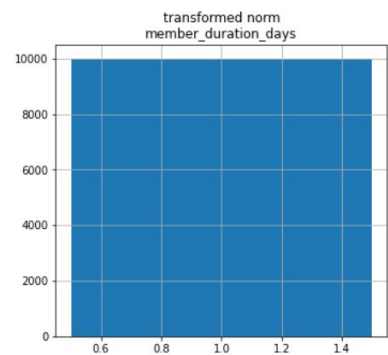
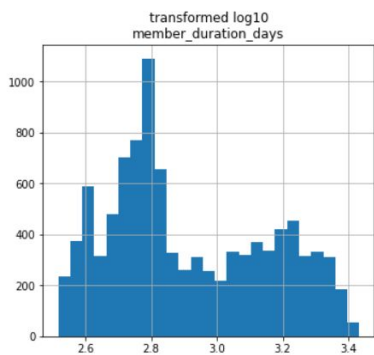
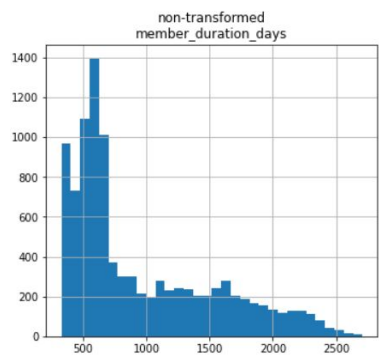


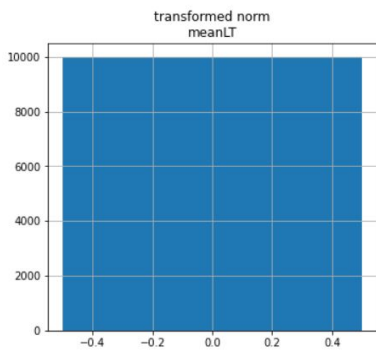
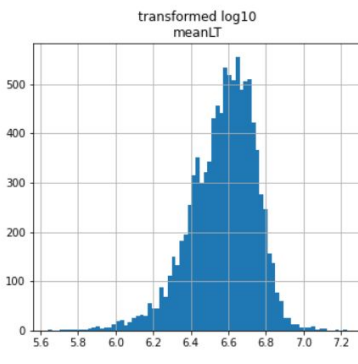
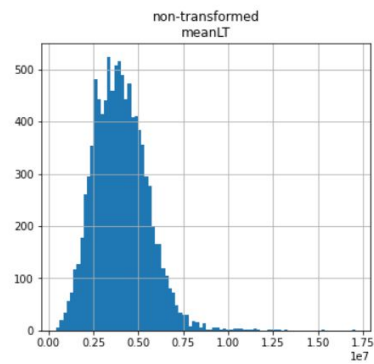
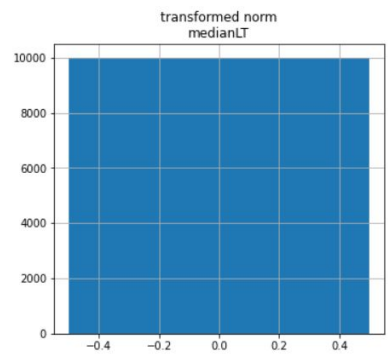
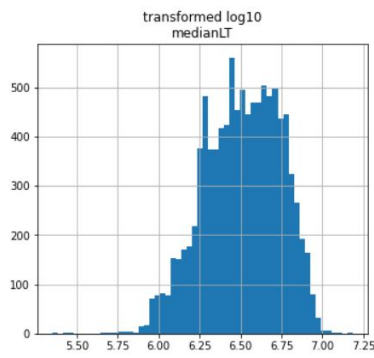
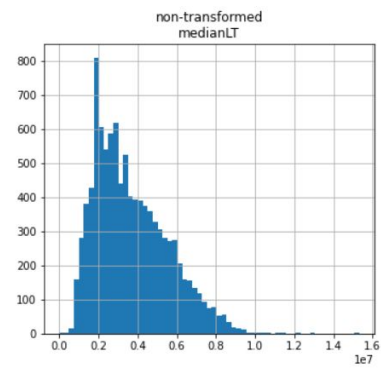
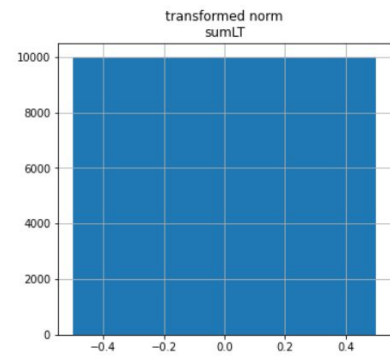
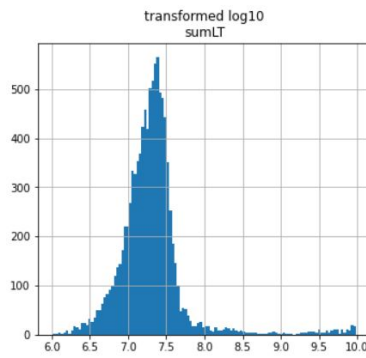
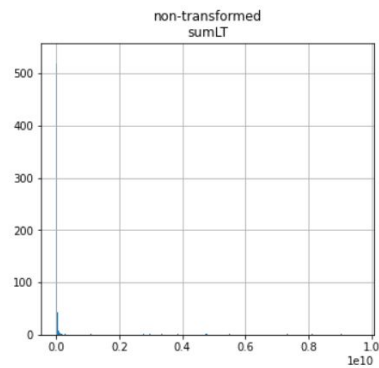
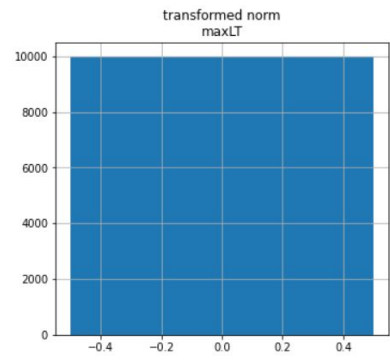
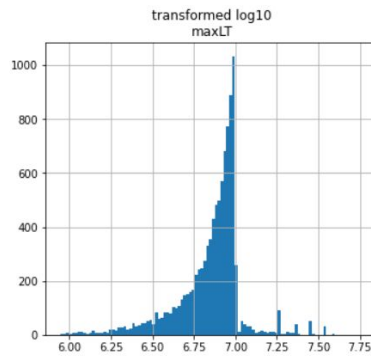
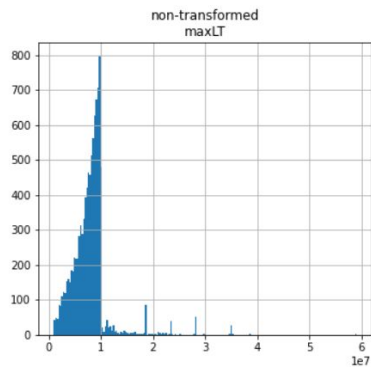
- Transform skewed data using log10  
Kemudian dengan melakukan transformasi log10 pada dataset Flight dan Test hingga didapat hasil sebagai berikut :  
Dataset Flight



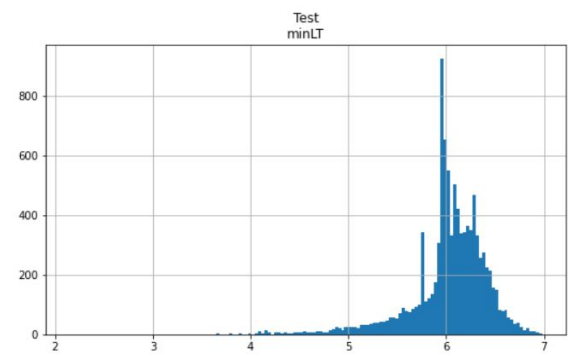
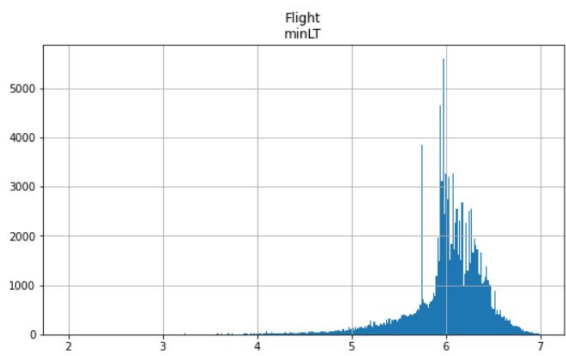
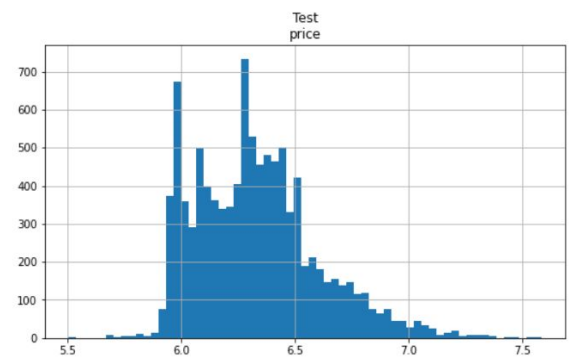
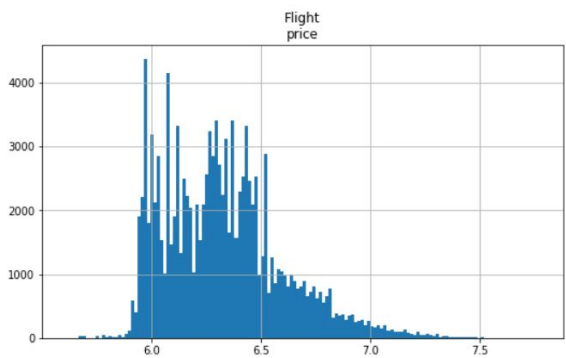
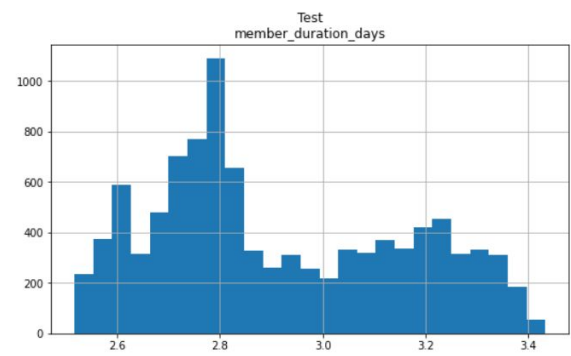
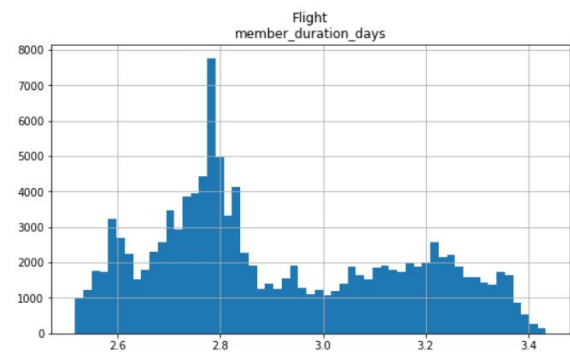


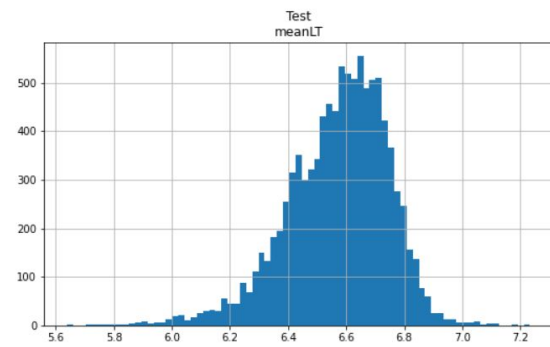
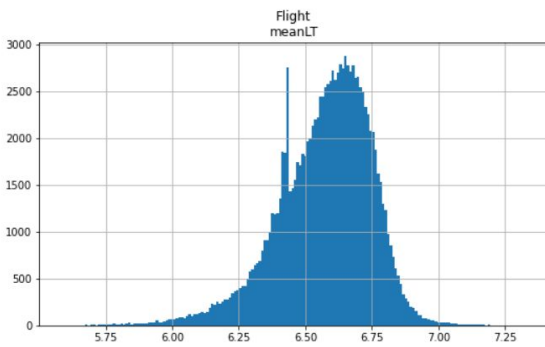
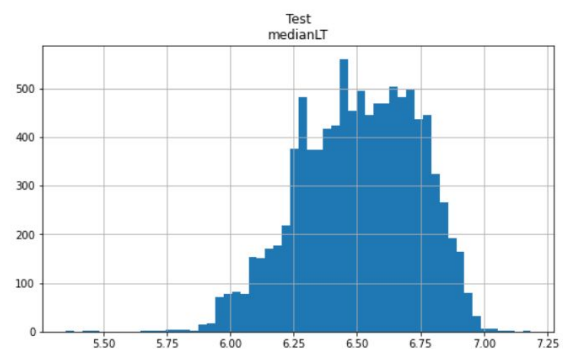
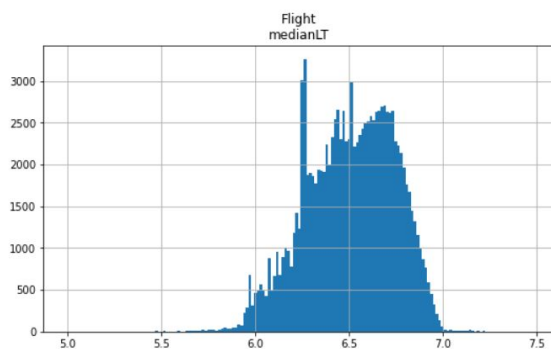
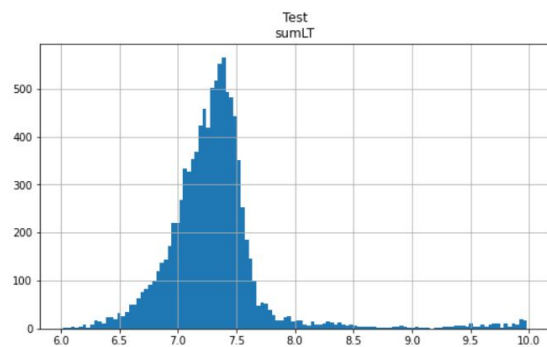
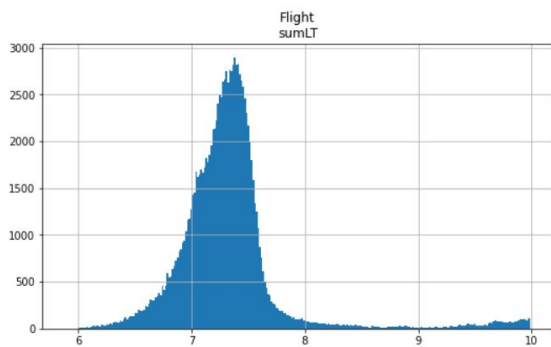
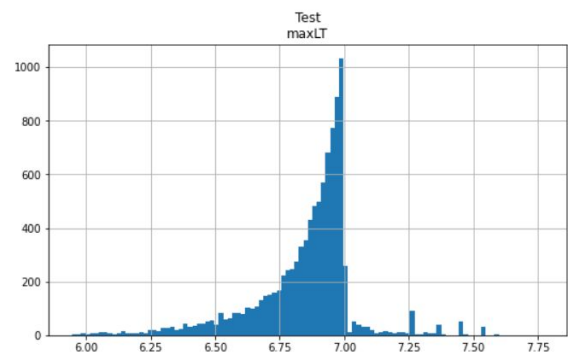
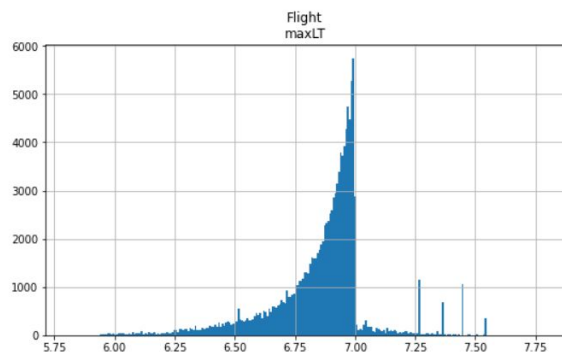
Dataset Test

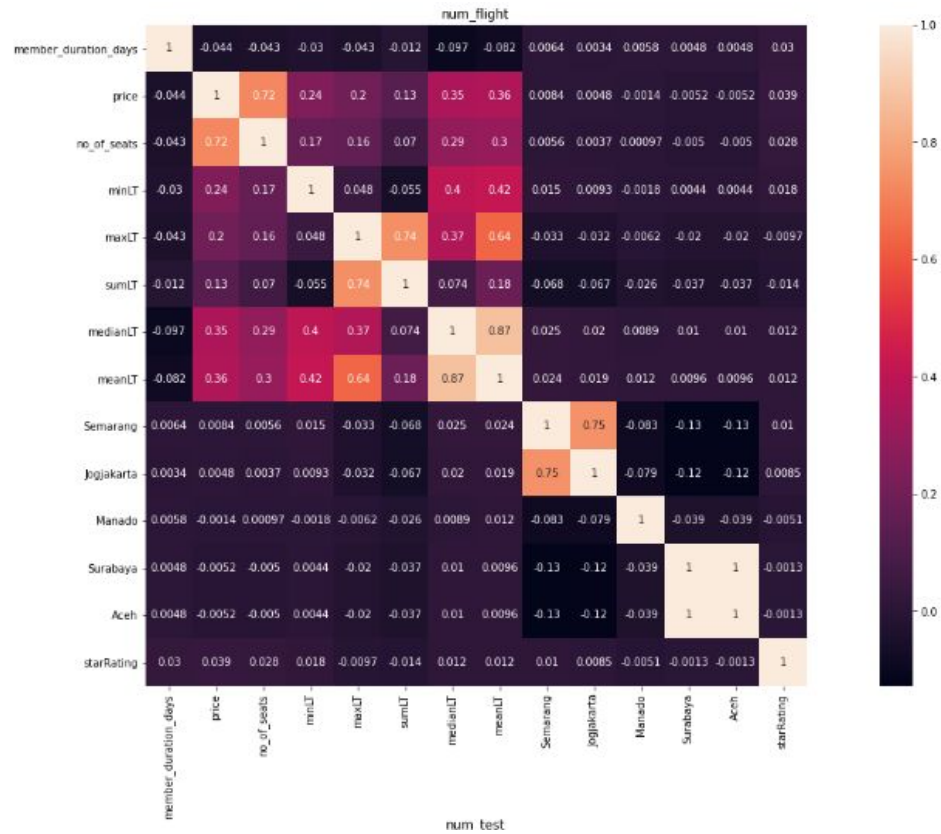


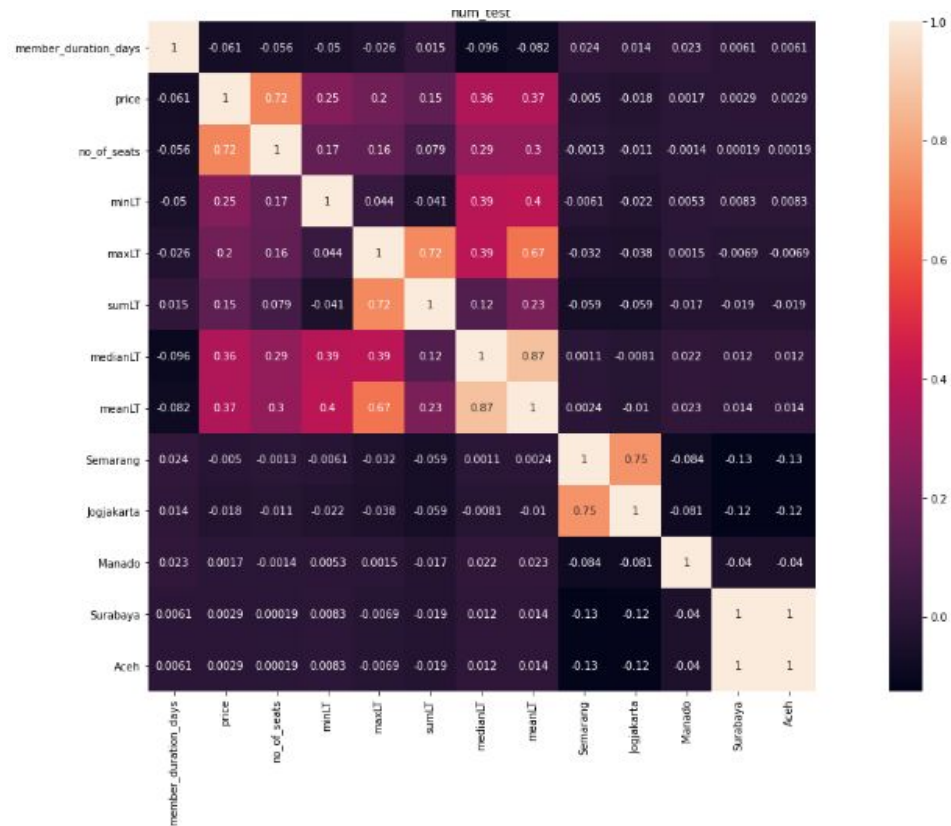


Dataset Flight & Test









- Mengenerate *one-hit encoding* menjadi feature kategorikal
- Menormalisasi features  
Hal ini dilakukan agar tiap fitur memiliki pengaruh yang sama.



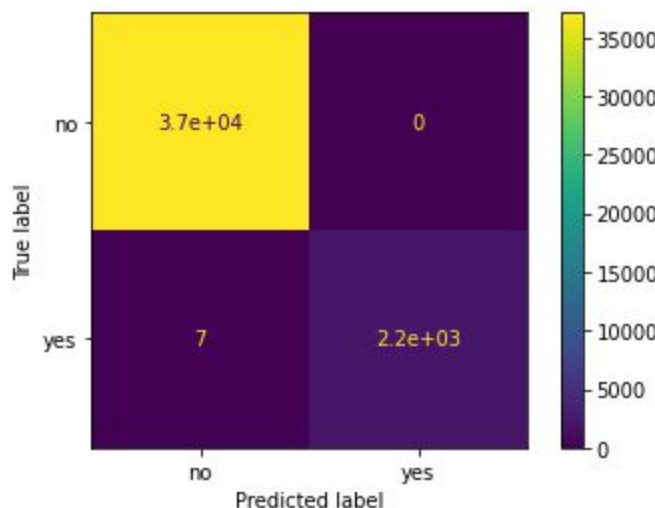
## V Uji Coba

Kami menggunakan `precision_recall_fscore_support` untuk mengukur nilai *micro* dan *macro*. *Macro* digunakan untuk menghitung metrik dari setiap label serta mencari bobot rata-rata yang digunakan. Sedangkan *Micro* digunakan untuk menghitung metrik global dengan menghitung total dari nilai benar positif, salah negatif, serta salah positif.

Source code yang kami gunakan untuk mengecek nilai *macro* dan *micro*, hasil yang didapat, serta confusion matrix, untuk setiap model adalah sebagai berikut:

### a. Sebelum SMOTE

#### i. K-Nearest Neighbours

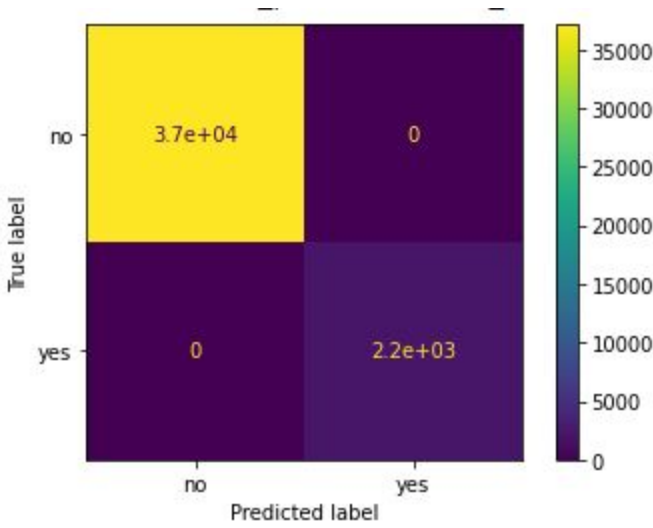
Source code	<pre>micro[counter] = precision_recall_fscore_support(testY, prediksi_knn, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_knn, average='macro')</pre>									
Hasil yang didapat	<div>Micro</div> <div>mean prediction: 0.9998982583094222</div> <div>mean recall: 0.9998982583094222</div> <div>mean f1-score: 0.9998982583094222</div> <div>Macro</div> <div>mean prediction: 0.9999460537895226</div> <div>mean recall: 0.9991090829747197</div> <div>mean f1-score: 0.9995271513339851</div>									
Hasil Confusion Matrix yang dihasilkan	 <table><tr><th></th><th>no</th><th>yes</th></tr><tr><th>no</th><td>3.7e+04</td><td>0</td></tr><tr><th>yes</th><td>7</td><td>2.2e+03</td></tr></table>		no	yes	no	3.7e+04	0	yes	7	2.2e+03
	no	yes								
no	3.7e+04	0								
yes	7	2.2e+03								

## ii. Decision Tree

Source code	<pre>micro[counter] = precision_recall_fscore_support(testY, prediksi_dt, average='micro')     macro[counter] = precision_recall_fscore_support(testY, prediksi_dt, average='macro')</pre>									
Hasil yang didapat	<div>Micro</div> <div>mean prediction: 1.0</div> <div>mean recall: 1.0</div> <div>mean f1-score: 1.0</div> <div>Macro</div> <div>mean prediction: 1.0</div> <div>mean recall: 1.0</div> <div>mean f1-score: 1.0</div>									
Hasil Confusion Matrix yang dihasilkan	<table><tr><th></th><th>no</th><th>yes</th></tr><tr><th>no</th><td>3.7e+04</td><td>0</td></tr><tr><th>yes</th><td>0</td><td>2.2e+03</td></tr></table>		no	yes	no	3.7e+04	0	yes	0	2.2e+03
	no	yes								
no	3.7e+04	0								
yes	0	2.2e+03								

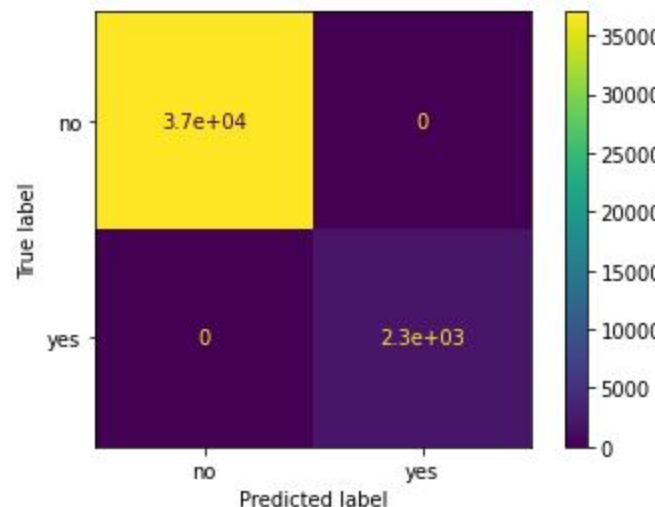
## iii. Random Forest

Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_rf, average='micro') macro[counter] = precision_recall_fscore_support(testY, </pre>
-------------	--

	prediksi_rf, average='macro')									
Hasil yang didapat	<div>Micro</div> <div>mean prediction: 1.0</div> <div>mean recall: 1.0</div> <div>mean f1-score: 1.0</div> <div>Macro</div> <div>mean prediction: 1.0</div> <div>mean recall: 1.0</div> <div>mean f1-score: 1.0</div>									
Hasil Confusion Matrix yang dihasilkan	 <table><tr><th></th><th>no</th><th>yes</th></tr><tr><th>no</th><td>3.7e+04</td><td>0</td></tr><tr><th>yes</th><td>0</td><td>2.2e+03</td></tr></table>		no	yes	no	3.7e+04	0	yes	0	2.2e+03
	no	yes								
no	3.7e+04	0								
yes	0	2.2e+03								

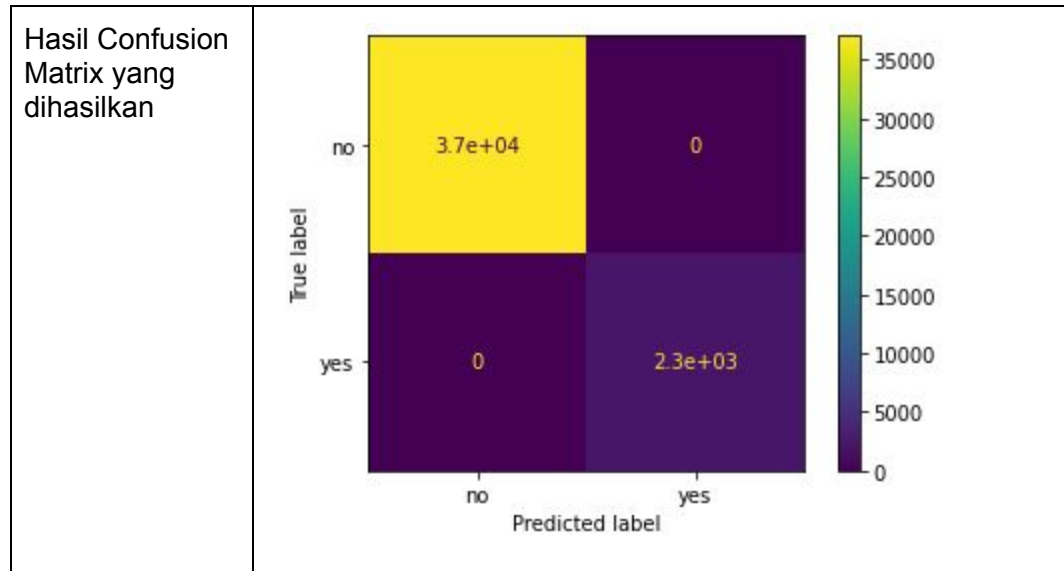
#### iv. Gradient Boosting

Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_gb, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_gb, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p> <p>Macro</p>

	mean prediction: 1.0 mean recall: 1.0 mean f1-score: 1.0									
Hasil Confusion Matrix yang dihasilkan	 <table><tr><th>True label \ Predicted label</th><th>no</th><th>yes</th></tr><tr><th>no</th><td>3.7e+04</td><td>0</td></tr><tr><th>yes</th><td>0</td><td>2.3e+03</td></tr></table>	True label \ Predicted label	no	yes	no	3.7e+04	0	yes	0	2.3e+03
True label \ Predicted label	no	yes								
no	3.7e+04	0								
yes	0	2.3e+03								

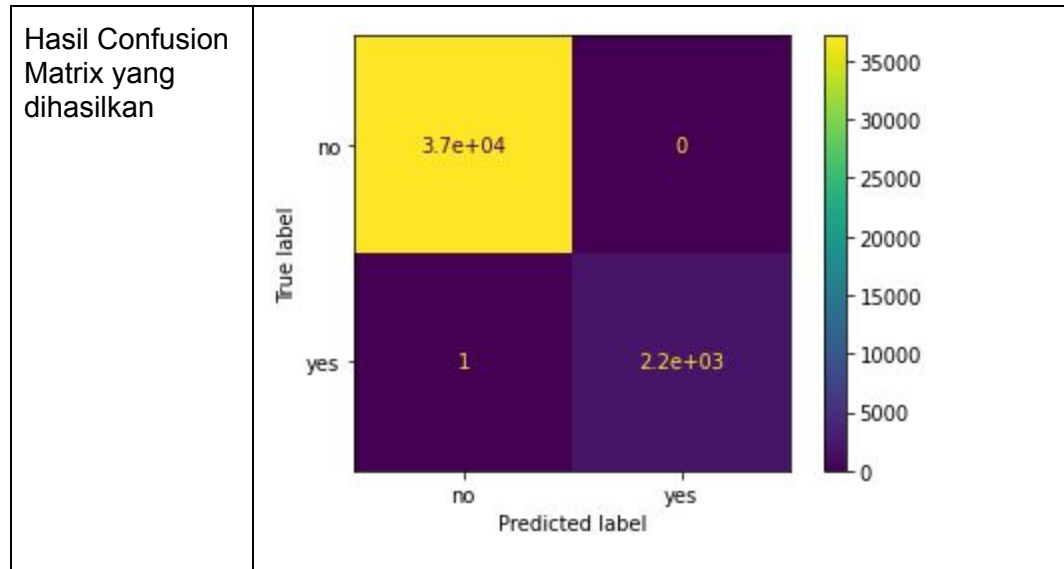
#### v. MLP Classifier

Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_MLP, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_MLP, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> mean prediction: 0.9999915214718724 mean recall: 0.9999915214718724 mean f1-score: 0.9999915214718724  <p>Macro</p> mean prediction: 0.9999955044865224 mean recall: 0.9999256616116563 mean f1-score: 0.9999605747276314



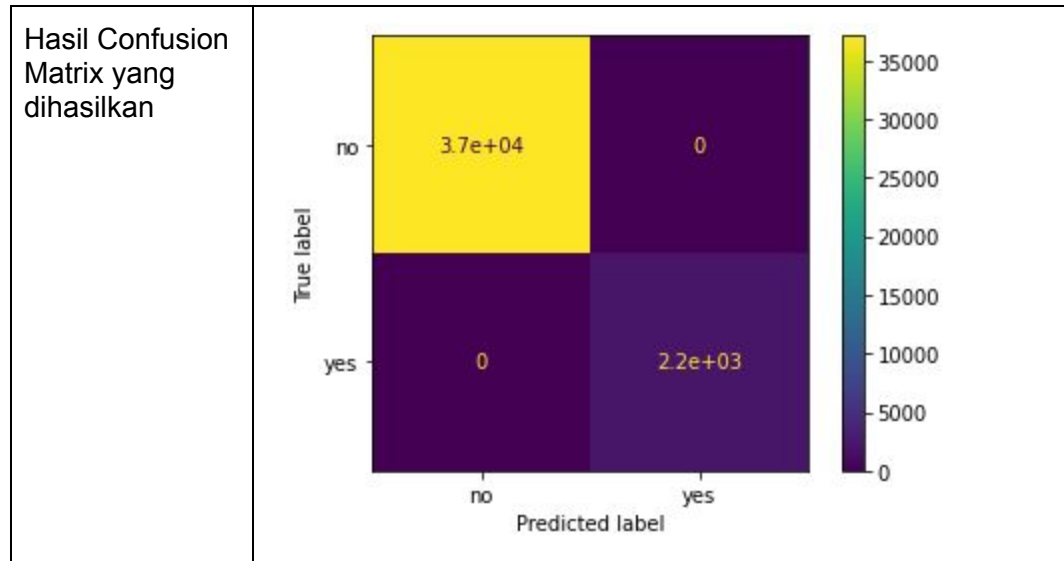
## vi. Logistic Regression

Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_lr, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_lr, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> <p>mean prediction: 0.9999321726375833</p> <p>mean recall: 0.9999321726375833</p> <p>mean f1-score: 0.9999321726375833</p> <p>Macro</p> <p>mean prediction: 0.999964007448512</p> <p>mean recall: 0.9994134435910484</p> <p>mean f1-score: 0.9996885166285022</p>



## vii. Support Vector Machine

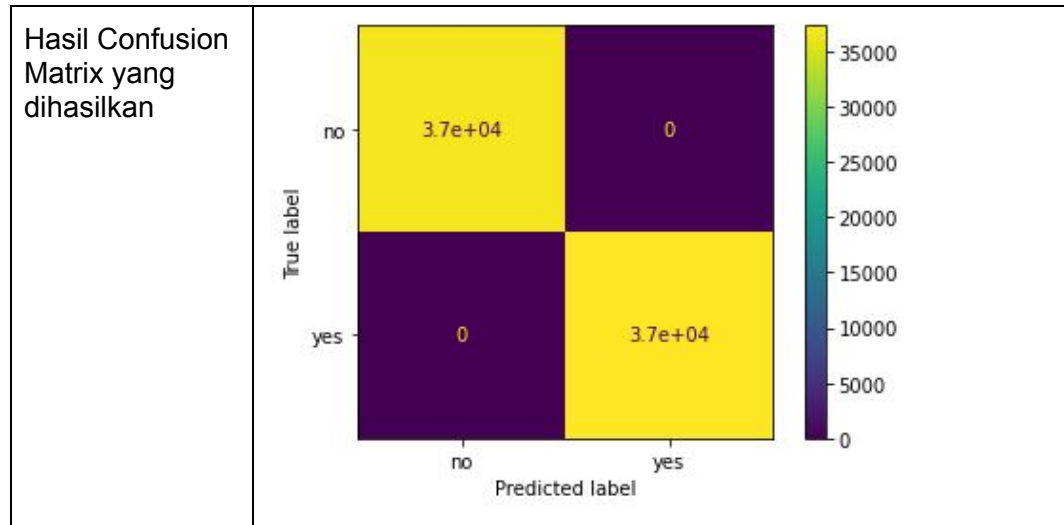
Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_svm, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_svm, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p> <p>Macro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p>



## b. Sesudah SMOTE

### i. K-Nearest Neighbours

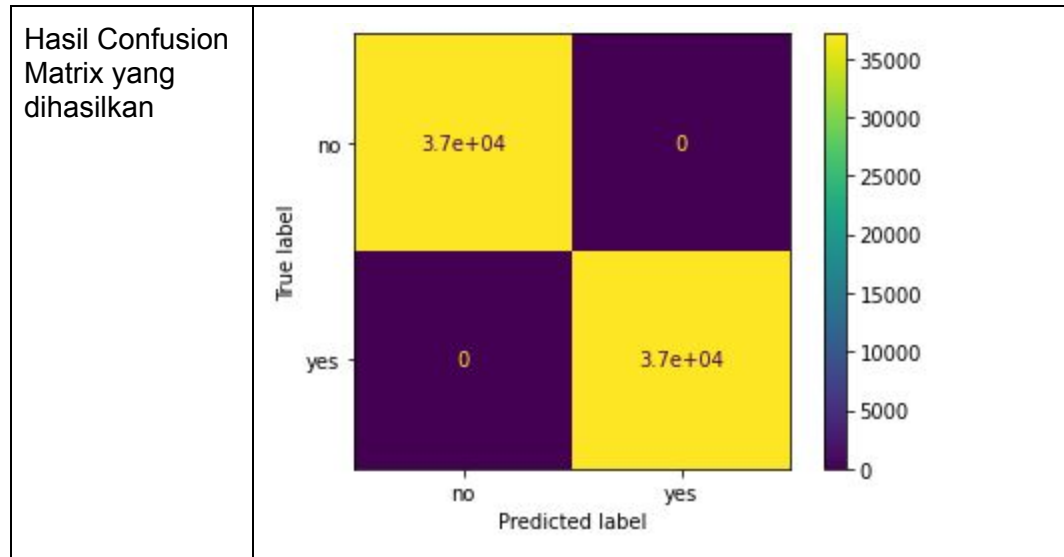
Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_knn, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_knn, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p> <p>Macro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p>



## ii. Decision Tree

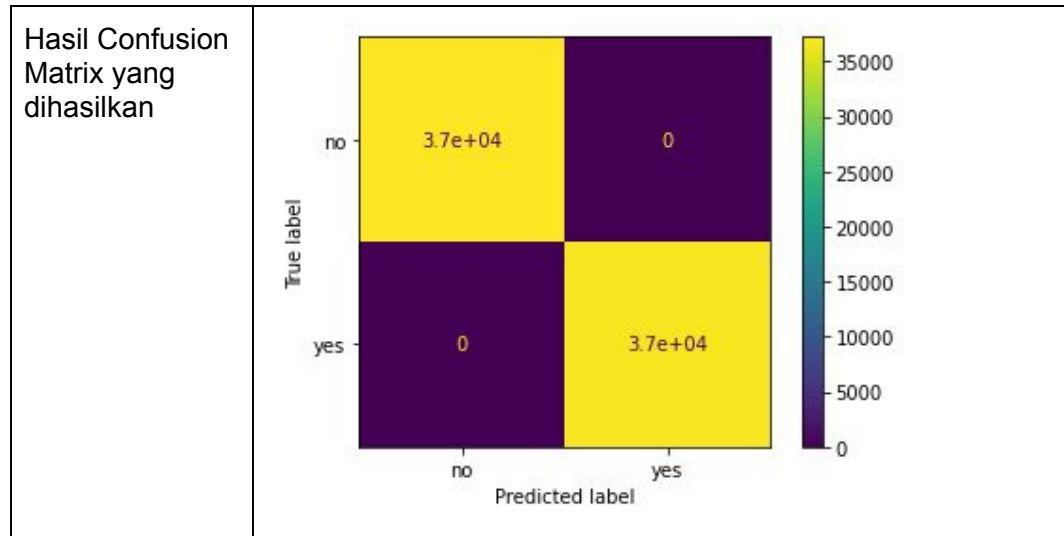
Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_dt, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_dt, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p> <p>Macro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p>





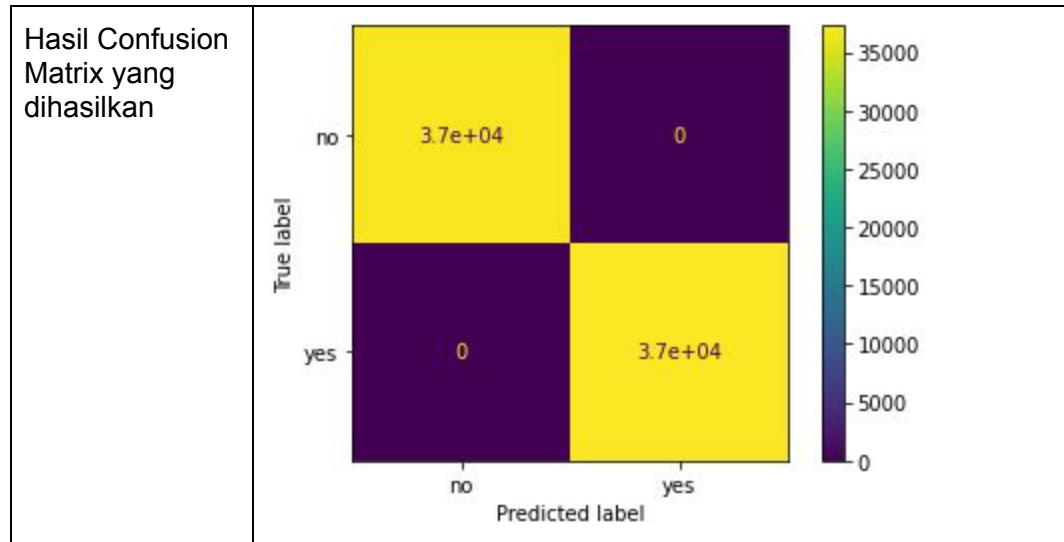
### iii. Random Forest

Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_rf, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_rf, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p> <p>Macro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p>



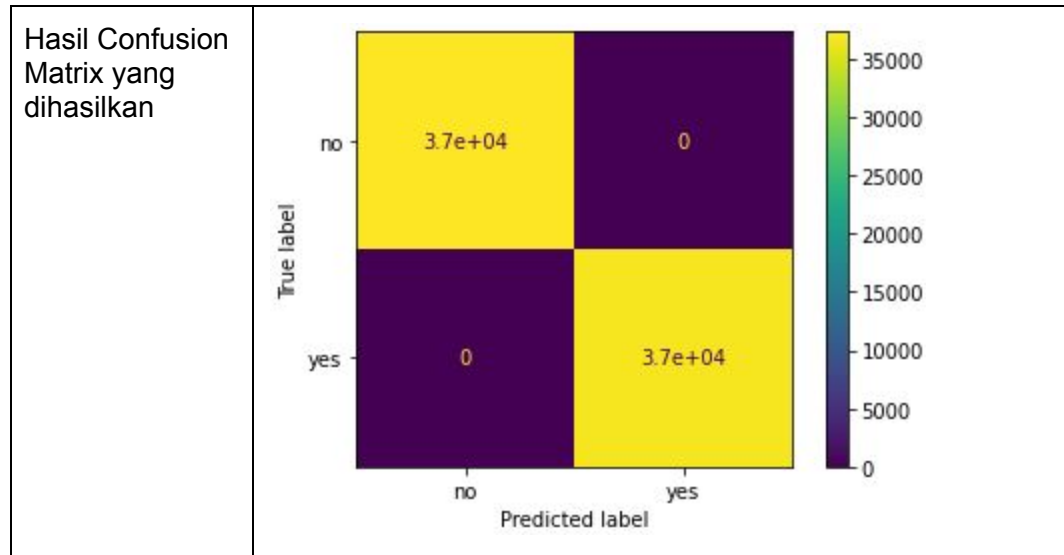
#### iv. Gradient Boosting

Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_gb, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_gb, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p> <p>Macro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p>



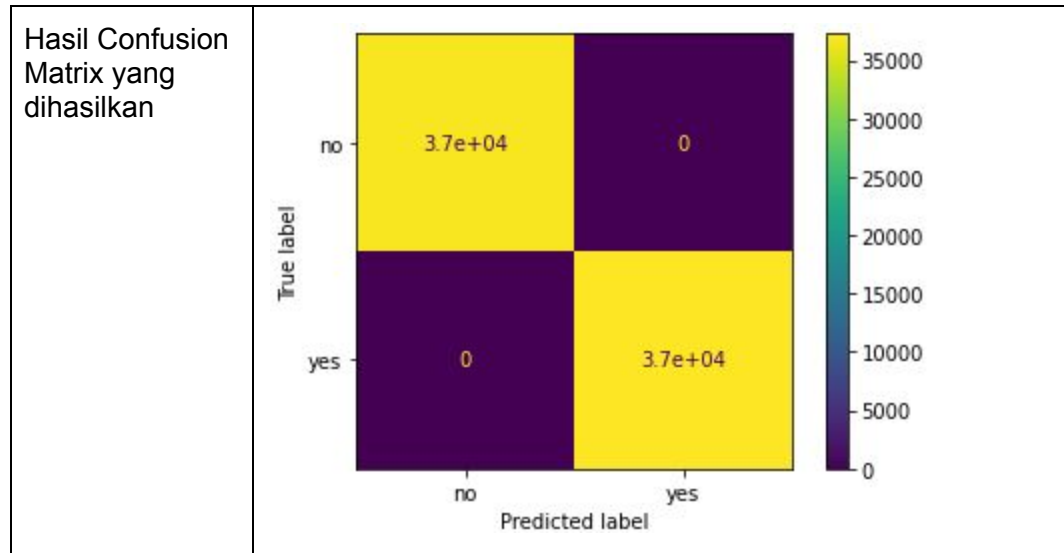
# v. MLP Classifier

Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_MLP, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_MLP, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p> <p>Macro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p>



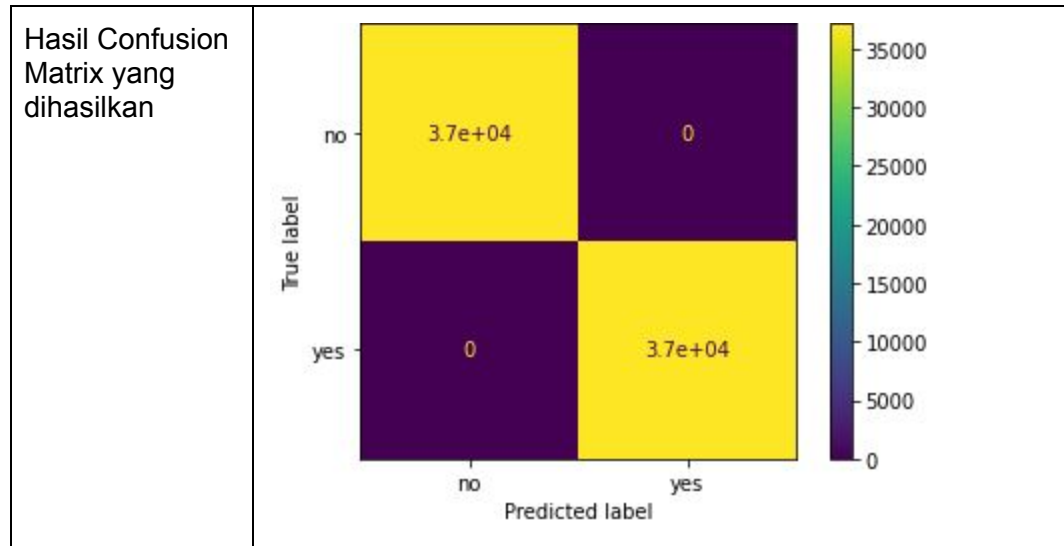
## vi. Logistic Regression

Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_lr, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_lr, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p> <p>Macro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p>



## vii. Support Vector Machine

Source code	<pre> micro[counter] = precision_recall_fscore_support(testY, prediksi_svm, average='micro') macro[counter] = precision_recall_fscore_support(testY, prediksi_svm, average='macro') </pre>
Hasil yang didapat	<p>Micro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p> <p>Macro</p> <p>mean prediction: 1.0</p> <p>mean recall: 1.0</p> <p>mean f1-score: 1.0</p>



### c. Perbandingan Confusion Matriks

Nama Algoritma	Sebelum SMOTE	Sesudah SMOTE
K-Nearest Neighbor	<p>True label</p> <p>no</p> <p>yes</p> <p>no</p> <p>yes</p> <p>Predicted label</p> <p>3.7e+04</p> <p>0</p> <p>7</p> <p>2.2e+03</p> <p>Color scale: 0 to 35000</p>	<p>True label</p> <p>no</p> <p>yes</p> <p>no</p> <p>yes</p> <p>Predicted label</p> <p>3.7e+04</p> <p>0</p> <p>0</p> <p>3.7e+04</p> <p>Color scale: 0 to 35000</p>
Decision Tree	<p>True label</p> <p>no</p> <p>yes</p> <p>no</p> <p>yes</p> <p>Predicted label</p> <p>3.7e+04</p> <p>0</p> <p>0</p> <p>2.2e+03</p> <p>Color scale: 0 to 35000</p>	<p>True label</p> <p>no</p> <p>yes</p> <p>no</p> <p>yes</p> <p>Predicted label</p> <p>3.7e+04</p> <p>0</p> <p>0</p> <p>3.7e+04</p> <p>Color scale: 0 to 35000</p>

Random Forest	<table border="1"> <thead> <tr> <th></th><th>no</th><th>yes</th></tr> </thead> <tbody> <tr> <th>no</th><td>3.7e+04</td><td>0</td></tr> <tr> <th>yes</th><td>0</td><td>2.2e+03</td></tr> </tbody> </table>		no	yes	no	3.7e+04	0	yes	0	2.2e+03	<table border="1"> <thead> <tr> <th></th><th>no</th><th>yes</th></tr> </thead> <tbody> <tr> <th>no</th><td>3.7e+04</td><td>0</td></tr> <tr> <th>yes</th><td>0</td><td>3.7e+04</td></tr> </tbody> </table>		no	yes	no	3.7e+04	0	yes	0	3.7e+04
	no	yes																		
no	3.7e+04	0																		
yes	0	2.2e+03																		
	no	yes																		
no	3.7e+04	0																		
yes	0	3.7e+04																		
Gradien Boosting	<table border="1"> <thead> <tr> <th></th><th>no</th><th>yes</th></tr> </thead> <tbody> <tr> <th>no</th><td>3.7e+04</td><td>0</td></tr> <tr> <th>yes</th><td>0</td><td>2.3e+03</td></tr> </tbody> </table>		no	yes	no	3.7e+04	0	yes	0	2.3e+03	<table border="1"> <thead> <tr> <th></th><th>no</th><th>yes</th></tr> </thead> <tbody> <tr> <th>no</th><td>3.7e+04</td><td>0</td></tr> <tr> <th>yes</th><td>0</td><td>3.7e+04</td></tr> </tbody> </table>		no	yes	no	3.7e+04	0	yes	0	3.7e+04
	no	yes																		
no	3.7e+04	0																		
yes	0	2.3e+03																		
	no	yes																		
no	3.7e+04	0																		
yes	0	3.7e+04																		
MLPClassifier	<table border="1"> <thead> <tr> <th></th><th>no</th><th>yes</th></tr> </thead> <tbody> <tr> <th>no</th><td>3.7e+04</td><td>0</td></tr> <tr> <th>yes</th><td>0</td><td>2.3e+03</td></tr> </tbody> </table>		no	yes	no	3.7e+04	0	yes	0	2.3e+03	<table border="1"> <thead> <tr> <th></th><th>no</th><th>yes</th></tr> </thead> <tbody> <tr> <th>no</th><td>3.7e+04</td><td>0</td></tr> <tr> <th>yes</th><td>0</td><td>3.7e+04</td></tr> </tbody> </table>		no	yes	no	3.7e+04	0	yes	0	3.7e+04
	no	yes																		
no	3.7e+04	0																		
yes	0	2.3e+03																		
	no	yes																		
no	3.7e+04	0																		
yes	0	3.7e+04																		
Logistic Regression	<table border="1"> <thead> <tr> <th></th><th>no</th><th>yes</th></tr> </thead> <tbody> <tr> <th>no</th><td>3.7e+04</td><td>0</td></tr> <tr> <th>yes</th><td>1</td><td>2.2e+03</td></tr> </tbody> </table>		no	yes	no	3.7e+04	0	yes	1	2.2e+03	<table border="1"> <thead> <tr> <th></th><th>no</th><th>yes</th></tr> </thead> <tbody> <tr> <th>no</th><td>3.7e+04</td><td>0</td></tr> <tr> <th>yes</th><td>0</td><td>3.7e+04</td></tr> </tbody> </table>		no	yes	no	3.7e+04	0	yes	0	3.7e+04
	no	yes																		
no	3.7e+04	0																		
yes	1	2.2e+03																		
	no	yes																		
no	3.7e+04	0																		
yes	0	3.7e+04																		
Support Vector Machine	<table border="1"> <thead> <tr> <th></th><th>no</th><th>yes</th></tr> </thead> <tbody> <tr> <th>no</th><td>3.7e+04</td><td>0</td></tr> <tr> <th>yes</th><td>0</td><td>2.2e+03</td></tr> </tbody> </table>		no	yes	no	3.7e+04	0	yes	0	2.2e+03	<table border="1"> <thead> <tr> <th></th><th>no</th><th>yes</th></tr> </thead> <tbody> <tr> <th>no</th><td>3.7e+04</td><td>0</td></tr> <tr> <th>yes</th><td>0</td><td>3.7e+04</td></tr> </tbody> </table>		no	yes	no	3.7e+04	0	yes	0	3.7e+04
	no	yes																		
no	3.7e+04	0																		
yes	0	2.2e+03																		
	no	yes																		
no	3.7e+04	0																		
yes	0	3.7e+04																		

#### d. Tabel Perbandingan

	Algoritma	Sebelum SMOTE						Setelah SMOTE					
		micro			macro			micro			macro		
		predic tion	recall	f1-sco re	predic tion	recall	f1-sco re	predic tion	recall	f1-sco re	predic tion	recall	f1-sco re
1	KNN	0.999 898	0.999 898	0.999 898	0.999 946	0.999 109	0.999 527	1.0	1.0	1.0	1.0	1.0	1.0
2	Decision Tree	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	Random Forest	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	Gradient Boosting	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	MLP Classifier	0.999 992	0.999 992	0.999 992	0.999 996	0.999 926	0.999 961	1.0	1.0	1.0	1.0	1.0	1.0
6	Logistic Regressi on	0.999 932	0.999 932	0.999 932	0.999 964	0.999 413	0.999 689	1.0	1.0	1.0	1.0	1.0	1.0
7	Support Vector Machine	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

## VI Kesimpulan dan Saran

Kesimpulan yang didapat dari uji coba yang dilakukan adalah sebelum menggunakan SMOTE pada model terbaik didapat dengan menggunakan metode klasifikasi Decision Tree, Random Forest, Gradient Boosting, serta Support Vector Machine, dengan nilai yang sama, yaitu 1. Sedangkan setelah menggunakan SMOTE seluruh model mendapat nilai yang sama, yaitu 1. Selain itu, hal ini juga membuktikan bahwa menggunakan SMOTE dapat membuat model yang dibuat menjadi lebih baik atau lebih akurat.

Saran ketika akan menggunakan metode klasifikasi K-Nearest Neighbours, MLP Classifier maupun Logistic Regression, agar dapat menghasilkan hasil yang lebih akurat dapat dilakukan dengan menerapkan metode SMOTE pada ketiga algoritma tersebut. Atau untuk cara lainnya adalah menggunakan metode klasifikasi selain dari ketiga metode klasifikasi tersebut seperti Support Vector Machine, Gradient Boosting, Random Forest ataupun Decision Tree.





## VII Referensi

<https://medium.com/@ksnugroho/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f>

<https://medium.com/analytics-vidhya/balance-your-data-using-smote-98e4d79fcddb>

<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/smote#:~:text=SMOTE%20stands%20for%20Synthetic%20Minority,that%20you%20supply%20as%20input.>

<https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833>

[https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.KFold.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html)

<https://machinelearningmastery.com/k-fold-cross-validation/>

<https://informatikalogi.com/algorithm-k-nn-k-nearest-neighbor/>

<https://medium.com/iykra/mengenal-decision-tree-dan-manfaatnya-b98cf3cf6a8d>

<https://chiragsehra42.medium.com/decision-trees-explained-easily-28f23241248>

<https://medium.com/@Synced/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674>

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

<https://medium.com/analytics-vidhya/introduction-to-the-gradient-boosting-algorithm-c25c653f826b>

<https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/>

<https://medium.com/codex/machine-learning-logistic-regression-with-python-5ed4ded9d146>

<https://medium.com/@samsudiney/penjelasan-sederhana-tentang-apa-itu-svm-149fec72bd02>