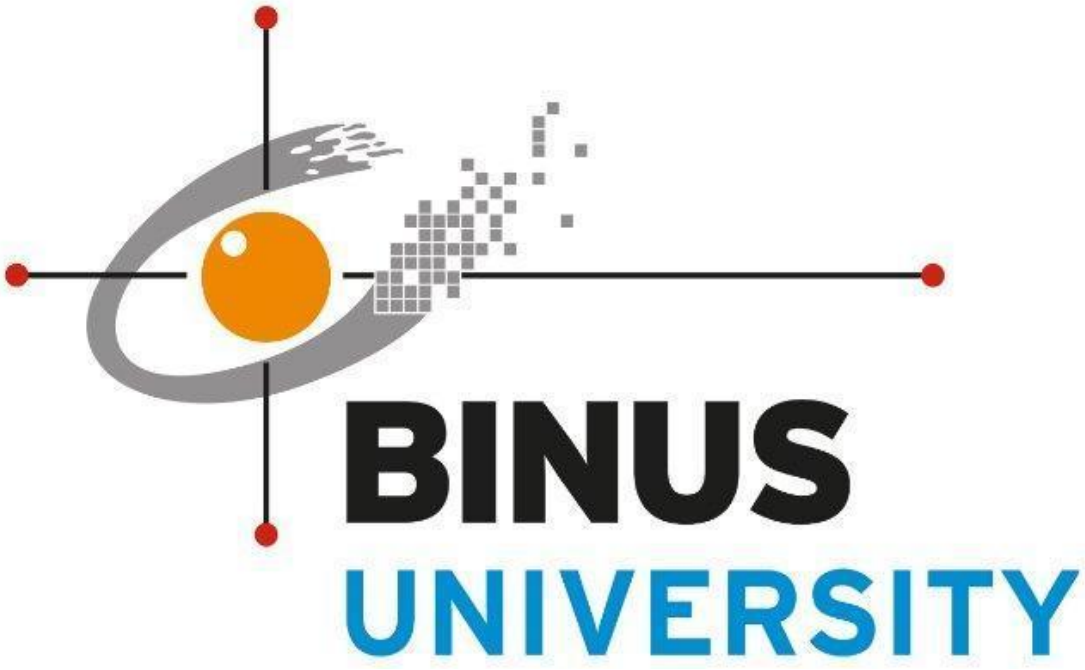
LAPORAN AOL CHURN DATASET

Supervised Learning - Machine Learning



Disusun Oleh:

Belinda Mutiara - 2540119596 - belinda.mutiara@binus.ac.id

Evelyn Zefanya Rahardjo - 2540118914 - evelyn.rahardjo@binus.ac.id

Jasmine Mutia Alifa - 2502026873 - jasmine.alifa@binus.ac.id

Program Data Science

Daftar Isi

BAB I

PENDAHULUAN

1.1 Latar Belakang

Retensi pelanggan merupakan salah satu kunci performa bagi perusahaan berbasis pelanggan. Persaingan ketat bisa saja terjadi antar perusahaan sejenis sebab pelanggan bebas memilih satu dari sekian banyak penyedia. Satu hal saja pengalaman buruk yang dialami pelanggan memungkinkan pelanggan untuk berpindah kepada pesaing. Hal inilah yang dinamakan dengan *customer churn*.

Prediksi *customer churn* dapat membuat perusahaan (dalam hal ini perusahaan bank) mengetahui pelanggan mana yang cenderung meninggalkan atau berhenti berlangganan dari perusahaan. Bagi banyak perusahaan tentu prediksi retensi merupakan hal yang penting, sebab untuk mendapatkan nasabah baru tentunya membutuhkan biaya yang lebih tinggi ketimbang mempertahankan nasabah yang sudah ada. Setiap nasabah tentunya memiliki perilaku dan preferensi yang berbeda-beda, oleh karena itu, penting bagi perusahaan bank untuk melakukan komunikasi secara aktif dengan masing-masing nasabah agar nama mereka tetap tercatat dalam daftar nama nasabah perusahaan.

1.1.1 Customer Churn

Customer churn adalah persentase pelanggan yang berhenti menggunakan produk atau layanan perusahaan selama jangka waktu tertentu. Memprediksi Customer churn merupakan hal yang penting terutama bagi perusahaan industri yang bergerak di bidang telekomunikasi, keuangan, dan sebagainya. Kemampuan memprediksi Customer churn merupakan hal yang penting bagi perusahaan, sebab dengan memprediksi hal tersebut, perusahaan masih memiliki waktu untuk melakukan sesuatu agar pelanggan tidak memberhentikan langganan

Retensi pelanggan dapat dicapai dengan menambahkan beberapa layanan dan produk yang baik, akan tetapi cara yang paling efektif adalah dengan benar-benar mengenal

pelanggan. Jumlah data yang besar dikumpulkan menjadi sebuah dataset untuk membuat pemodelan tentang prediksi customer churn. Dengan mengetahui siapa yang paling mungkin untuk meninggalkan perusahaan, perusahaan dapat mengambil tindakan untuk mencegah pelanggan pergi

Perusahaan dengan churn rate tinggi (kehilangan banyak nasabah) cenderung untuk memiliki tingkat pertumbuhan yang lebih rendah

1.1.2 Pentingnya Mencegah Retensi Nasabah Bagi Perusahaan Bank

Berikut merupakan beberapa alasan mengapa pemodelan *customer churn* penting bagi pihak perusahaan bank:

- Biaya untuk menarik pelanggan baru bisa mencapai lima sampai enam kali lipat lebih mahal ketimbang mempertahankan pelanggan yang sudah ada. Mempertahankan nasabah setia merupakan salah satu cara untuk mempermudah pertumbuhan dan mengatasi keuangan perusahaan.
- Pelanggan yang sudah bertahan lama menjadi lebih mudah dilayani, menghasilkan keuntungan yang lebih tinggi, serta dapat memberikan rekomendasi kepada calon pelanggan baru.
- Kehilangan pelanggan dalam hal ini nasabah dapat menyebabkan turunnya keuntungan bagi bank.

1.2 Problem Statement

Kehilangan pelanggan karena persaingan merupakan masalah tersendiri bagi perusaahaan keuangan karena biaya untuk mendapatkan pelanggan baru sangat mahal dan oleh karena itu perusahaan ingin mempertahankan pelanggan mereka yang sudah ada.

1.3 Tujuan

Dengan adanya projek ini, tujuan kami ialah:

a. Membuat pemodelan klasifikasi agar dapat memberikan gambaran tentang peluang retensi seorang nasabah menggunakan metode Logistic Regression, Gaussian Naive Bayes, Random Forest, dan LGBM.

b. Mengetahui tiga fitur yang paling berpengaruh terhadap retensi nasabah sebuah bank.

1.4 Deskripsi Dataset

Dataset customer churn memiliki beberapa variable yang memiliki fungsi berbeda. Variable yang ada dalam dataset meliputi :

- RowNumber : berisi nomor data (baris) dari dataset.
- CustomerId : nilai unik ID pelanggan yang ada pada setiap pelanggan.
- Surname : nama keluarga pelanggan, variabel kategorikal (rangkaian huruf)
- CreditScore : skor kredit pelanggan, variabel numerik kontinu. pelanggan dengan skor kredit yang lebih tinggi cenderung meninggalkan bank.
- Geography : negara yang terdaftar di akun pelanggan, variabel kategorikal.
- Gender : jenis kelamin pelanggan, variabel kategori dalam dua kata 'Pria' atau 'Wanita'.
- Age : umur pelanggan, variabel numerik diskrit.
- Tenure : jumlah tahun selama menjadi nasabah bank, variabel numerik diskrit.
- Balance : saldo keseluruhan di seluruh rekening yang dipegang oleh pelanggan, variabel numerik kontinu.
- NumOfProducts: jumlah produk yang dibeli nasabah melalui bank, variabel numerik diskrit (lebih besar dari 0).
- HasCrCard: mengacu pada apakah pelanggan memiliki kartu kredit atau tidak, variabel kategori yang mengambil nilai 1 = Ya dan 0 = Tidak.
- IsActiveMember : indikasi pelanggan aktif atau tidak, variabel kategori yang mengambil nilai 1 = Ya dan 0 = Tidak.
- EstimatedSalary : estimasi gaji tahunan pelanggan, variabel numerik kontinu.
- Exited : indikasi pelanggan sudah churn (keluar) atau belum, variabel kategori yang mengambil nilai 1 = Ya dan 0 = Tidak.

BAB II

Landasan Teori

2.1 Klasifikasi

Klasifikasi adalah proses pengelompokan data ke dalam beberapa kategori. Dalam machine learning, klasifikasi dibagi menjadi yaitu *supervised learning* dan *unsupervised learning*. Supervised learning merupakan algoritma yang mempelajari model, dimana setiap data tersebut memiliki label. Dari data tersebut, model akan memprediksi sebuah label. Biasanya, dalam supervised learning akan ada data training berisi semua label ada dan data testing dimana salah satu label akan dihilangkan. Unsupervised learning merupakan algoritma yang datanya tidak mempunyai label. Unsupervised learning digunakan untuk mencari pola atau trend dari data yang diberikan.

Klasifikasi termasuk dalam algoritma supervised learning. Algoritma yang dapat digunakan dalam klasifikasi sendiri sangat beragam. Beberapa algoritma klasifikasi meliputi logistic regression, random forest, gaussian naive bayes, Light Gradient Boosting Machine Classifier (LGBM Classifier).

2.2 Logistic Regression

Logistic Regression merupakan algoritma klasifikasi *(Supervised Learning)* yang fungsinya untuk mencari hubungan antara fitur (input) diskrit/kontinu dengan probabilitas hasil output diskrit tertentu. Pengaplikasiannya sendiri di *machine learning* misalnya untuk menentukan apakah seseorang mempunyai kemungkinan terinfeksi COVID-19 atau tidak. *Logistic Regression* ini banyak digunakan dalam pemecahan masalah *classification*, dan kasus penggunaan paling umum dalam *binary logistic regression* dimana hasilnya *binary* (ya / tidak). Terdapat 3 tipe *logistic regression* yaitu *binary logistic regression, multinomial logistic regression,* dan *ordinal logistic regression.*

Logistic regression cukup populer, dan banyak digunakan di *machine learning. Logistic regression* sendiri sebenarnya mirip dengan *linear regression* tetapi yang membedahkan adalah

fungsinya dimana *logistic regression* digunakan untuk memecahkan masalah klasifikasi sedangkan *linear regression* digunakan untuk memecahkan masalah regresi. Ada 5 step dalam *logistic regression* yaitu data *pre-processing, fitting logistic regression* pada *training set*, prediksi hasil test, test akurasi dari hasilnya *(confusion matrix)*, dan visualisasi hasil test nya.

Beberapa asumsi yang menyatakan bahwa *training data* cocok / *good fit* untuk *logistic regression* adalah *Independent variable* yang mempengaruhi hasilnya itu *independent* satu sama lain, hasil prediksinya sangat biner, *independent variable* nya dapat dihubungan secara *linear* dengan *log oods*, dan ukuran sampel yang cukup besar. Apabila *training data* tidak sesuai dengan asumsi artinya *logistic regression* tidak dapat digunakan.

2.3 Random Forest

Random Forest merupakan salah satu algoritma *machine learning* terbaik, dan terpopuler. *Random Forest* merupakan algoritma klasifikasi yang terdiri dari banyak *decision tree. Random Forest* bisa digunakan dalam kasus klasifikasi, dan regresi dengan kumpulan data berukuran besar di *machine learning.* Konsepnya sendiri didasarkan pada konsep *ensemble learning* yang merupakan proses menggambungkan *multiple classifiers* untuk memecahkan masalah yang kompleks, dan untuk meningkatkan performa dari modelnya. Random forest pengamplikasiannya misalnya untuk identifikasi pasar potensial saham, dan identifikasi resiko peminjaman di bank.

Random Forest adalah *classifier* yang berisi sejumlah *decision tree* pada berbagai *subset* dari kumpulan data yang diberikan, dan mengambil rata-rata untuk meningkatkan akurasi prediksi dari kumpulan data. *Random Forest* tidak hanya mengandalkan satu pohon saja tetapi *random forest* mengambil prediksi dari setiap pohon berdasarkan suara mayoritas prediksi, dan memprediksi hasil akhirnya. Jadi *random forest* ini seperti *voting.* Semakin besar jumlah pohon di hutan berarti akurasinya lebih tinggi, dan bisa mencegah masalah overfitting.

Random forest penggunaannya harus dihindari apabila ekstrapolasi, dan ketika datanya jarang digunakan. Di *random forest* sendiri ada 5 step yaitu data *pre-processing, fitting random forest algorithm* pada *training set*, prediksi hasil test, test akurasi dari hasilnya *(confusion matrix)*, dan visualisasi hasil test nya.

2.4 Gaussian Naive Bayes

Gaussian Naive Bayes merupakan algoritma klasifikasi *(supervised learning algorithm)* yang didasarkan pada teorema probabilitas *Bayes*, dan digunakan untuk memecahkan *classification task.* Salah satu kelebihan *Gaussian Naive Bayes Algorithm* ini adalah sangat sederhana tetapi juga sangat efektif serta cepat dalam membuat prediksi. *Gaussian Naive Bayes* mempunyai asumsi bahwa data dari masing-masing label diambil dari distribusi *Gaussian* sederhana. *Gaussian Naive Bayes* ini digunakan ketika kita ingin mengasumsikan semua variable kontinu yang terkait dengan setiap fitur didistribusikan menurut *Gaussian Distribution*. Contoh penggunaannya adalah untuk memprediksi pendapatan.

2.5 Light Gradient Boosting Machine Classifier (LGBM Classifier)

Light Gradient Boosting Machine Classifier merupakan salah satu algoritma klasifikasi yang bekerja dengan baik pada data dengan ukuran besar sebab algoritma ini rentan mengalami overfitting apabila diterapkan pada data dengan ukuran sampel yang lebih kecil. Cara kerja algoritma LGBM didasarkan pada cara kerja algoritma *Gradient Boosting Decision Tree* (GBDT) dan ditambah dengan dua teknik baru yakni *Gradients Based One Side Sampling* (GOSS) dan *Exclusive Feature Bundling* (EFB). *Gradients Based One Side Sampling* merupakan metode pengambilan sampel baru dengan mengambil contoh sampel berdasarkan gradien. Sedangkan *Exclusive Feature Bundling* merupakan metode mengurangi jumlah fitur dengan hampir tanpa kerugian, salah satu contoh EFB ini adalah one-hot-encoded. Kedua teknik ini baik GOSS dan EFB dirancang untuk meningkatkan efisiensi dan performa algoritma GBDT membentuk algoritma baru yang disebut *Light Gradient Boosting Machine.*

Algoritma LGBM merupakan salah satu algoritma yang cukup populer karena memiliki kinerja yang baik serta memiliki ketahanan terhadap berbagai jenis data, korelasi, distribusi, serta keragaman hyperparameter. Selain itu beberapa keunggulan dari algoritma ini adalah kecepatan yang baik dalam menangani training set, membutuhkan memori yang lebih kecil, memiliki akurasi yang lebih baik, dan mampu menangani data dengan skala yang besar. LGBM dapat digunakan untuk membuat model regresi, klasifikasi biner, klasifikasi multiclass, ataupun menangani permasalahan peringkat (*ranking*).

2.6 Evaluation Metrics

Evaluation Metrics merupakan cara untuk mengevaluasi model *machine learning* yang telah dibuat. Terdapat banyak evaluation metrics yang dapat digunakan untuk mengukur performa model yang telah dibuat. Hal ini disesuaikan dengan task/ tugas dari machine learning yang digunakan. Tentunya setiap task machine learning memiliki evaluation metrics yang berbeda-beda. Dalam kesempatan kali ini kita akan membahas evaluation metrics untuk task klasifikasi.

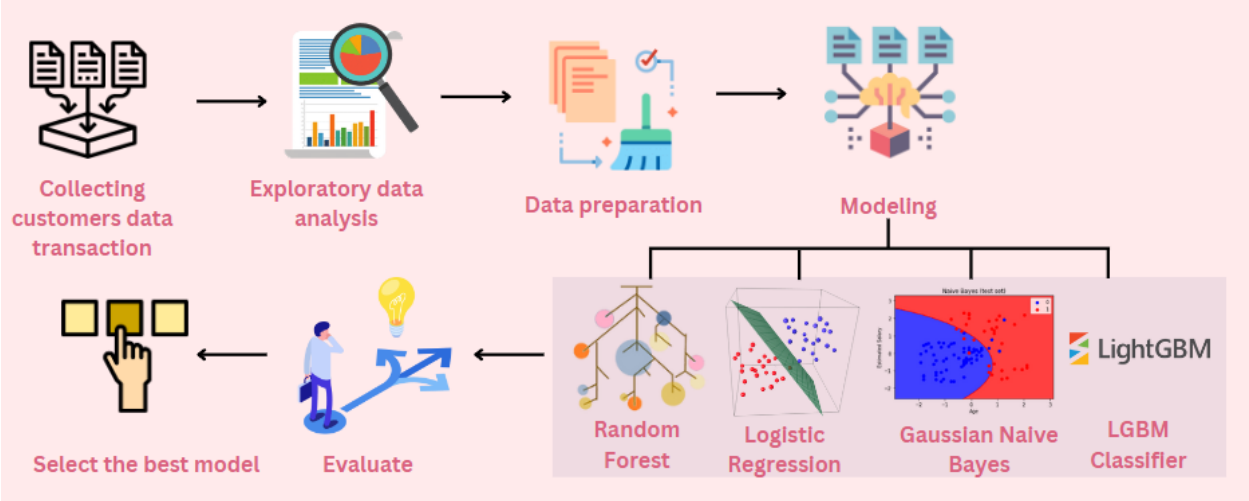
Seperti yang sudah dibahas sebelumnya, klasifikasi merupakan task dalam machine learning yang bertujuan untuk memprediksi label kelas dari sebuah observasi. Terdapat beberapa metrics yang sering digunakan dalam tugas klasifikasi yakni akurasi, recall, presisi, F1 Score, dan ROC-AUC. Pertama yaitu akurasi, akurasi mendeskripsikan tentang seberapa sering model memprediksi test set dengan tepat. Nilai akurasi didapatkan dari jumlah TP (True Positive) dan TN (True Negative) dibagi dengan jumlah TP, TN, FP (False Positive), dan FN (False Negative). Selanjutnya terdapat recall, recall mendeskripsikan tentang true positive rate, ketika keadaan sebenarnya adalah benar (True), berapa kali model memprediksi benar (True). Untuk mendapatkan nilai recall didapatkan dari rumus jumlah TP dibagi dengan jumlah TP dan FN. Metrics ketiga ialah presisi, presisi mendeskripsikan positive predicted value, ketika model memprediksi benad (True), berapa kali hal tersebut benar. Nilai presisi didapat dari jumlah TP dibagi dengan jumlah TP dan FP. Kemudian terdapat metrics F1-Score, F1-Score mendeskripsikan nilai harmonik rata-rata dari presisi dan recall. Nilai ini didapatkan dari dua kali presisi dikali recall dibagi dengan jumlah presisi dan recall. Metrics terakhir ialah ROC-AUC, metrics ini merupakan luas area dibawah kurva ROC, ROC-AUC mendeskripsikan tentang seberapa baik model dapat membedakan antar label antar kelas.

Lalu kapan sebaiknya menggunakan metrics akurasi, recall, presisi, F1-Score, dan ROC-AUC? Pertama, kita menggunakan metrics akurasi ketika data bersifat cukup imbang (balanced) dan kita ingin mengetahui seberapa sering model kita memprediksi dengan tepat. Recall dan Presisi digunakan ketika label dari variabel target bersifat tidak terlalu imbang (imbalanced data) serta ketika kelas positive lebih jarang muncul, serta ketika kita lebih mempedulikan FP ketimbang FN. Recall dipakai ketika kita ingin mengetahui saat keadaan sebenarnya adalah benar (True), berapa kali model memprediksi benar (True). Sedangkan Presisi dipakai ketika kita ingin mengetahui saat model memprediksi benar (True), berapa kali hal

tersebut tepat. F1-Score digunakan ketika jumlah FN dan FP seimbang, ketika menambah data tidak menambah performa data, dan ketika jumlah TN tinggi. ROC-AUC digunakan ketika data

BAB III

Metode



Dalam melakukan analisa mengenai churn customers, tahap pertama kami melakukan pengumpulan data customer dari bank, data kami dapatkan melalui Kaggle. Tahap kedua kami melakukan *exploratory data analysis* (EDA). Proses EDA yang kami lakukan yaitu load data dan data overview. Kami mengimpor library yang dibutuhkan serta membaca data yang sudah kami download melalui Kaggle, selanjutnya kami juga melihat isi dari data, dimensi data, dan variable-variable yang ada.

Tahap ketiga kami melakukan *data preparation.* Pada tahap ini kami melihat korelasi yang ada antar variabel, mengecek data yang kosong, mengecek duplikasi data, mengecek distribusi data, dan melakukan *feature engineering*. Feature engineering terdiri *feature selection* dan *feature scaling* menggunakan *standard scaler*. Kami melakukan *feature selection* untuk menentukan variabel yang kemungkinan besar berpengaruh pada modelling dan menghapus variabel yang tidak relevan.

Kemudian kami melakukan tahap modeling, kami mencoba empat macam algoritma pengklasifikasian yaitu logistic regression, random forest, gaussian naive bayes, serta light gradient boosting machine. Tahap ini dimulai dari pembagian jumlah data ke dalam kelompok train dan test dengan proporsi 80:20. Selanjutnya keempat algoritma diaplikasikan terhadap data customer churn. Untuk mengukur performa digunakan metrics recall, presisi, dan akurasi. Dari

tahap tersebut model akan di bandingkan berdasarkan nilai metricsnya. Selanjutnya akan dipilih satu model terbaik.

3.1 Load Libraries and Data Overview

3.1.1 Load Libraries

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

import statsmodels.api as sm
from scipy.stats import chi2_contingency
from sklearn.preprocessing import LabelEncoder, StandardScaler, OrdinalEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
```

```
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from lightgbm import LGBMClassifier, plot_importance
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, VotingClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from imblearn.over_sampling import SMOTE
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, recall_score, precision_score, auc, roc_auc_score, roc_curve

!pip install scikit-plot
import scikitplot as skplt
import math
```

Code diatas dibuat dengan tujuan untuk meng-import beberapa library yang akan digunakan sebagai alat bantu dalam menganalisis data.

3.1.2 Data Overview

```
#Mendifinikan variabel df (data frame) yang membaca data csv berjudul "CustomerChrun.csv".

df = pd.read_csv("https://raw.githubusercontent.com/belindamutiaraaaaa/data/main/CustomerChurn.csv")
```

```
print(type(df))
row,column=df.shape

print("Jumlah baris data Customer Churn adlah ", row)
print("Jumlah kolom data Customer Churn adalah ", column)
```

```
<class 'pandas.core.frame.DataFrame'>
Jumlah baris data Customer Churn adlah  10000
Jumlah kolom data Customer Churn adalah  14
```

Data Customer Churn mempunyai 100000 baris, dan 14 kolom.

```
['RowNumber' 'CustomerId' 'Surname' 'CreditScore' 'Geography' 'Gender'
 'Age' 'Tenure' 'Balance' 'NumOfProducts' 'HasCrCard' 'IsActiveMember'
 'EstimatedSalary' 'Exited']
```

Terdapat 14 kolom yaitu 'RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance','NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exited

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Disini bisa dilihat bawa setiap kolom mempunyai 10000 data. Tipe data yang ada adalah float, integer, dan object.

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Dari 5 data pertama kita dapat melihat bahwa :

- Tiap customer punya CustomerID yang berbeda satu sama lain.
- Customer yang punya credit score paling banyak dari 5 data di atas ada pada rentang 600an.
- Umur customer rata-rata 35-45 an.
- Balance customer ada yang 0.
- Kebanyakan customer active member, dan punya kartu kredit.
- Lebih banyak customer yang not churn dari pada churn.

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

Dari 5 data terakhir kita dapat melihat bahwa :

- Gendernya kebanyakan male dari pada female.
- Number of product yang rata - rata dibeli 1, dan 2.
- Kebanyakan mempunyai kartu kredit.

3.1.3 Checking Duplicate Data

```
print(df.shape)
print(df[df.duplicated()].shape)
```

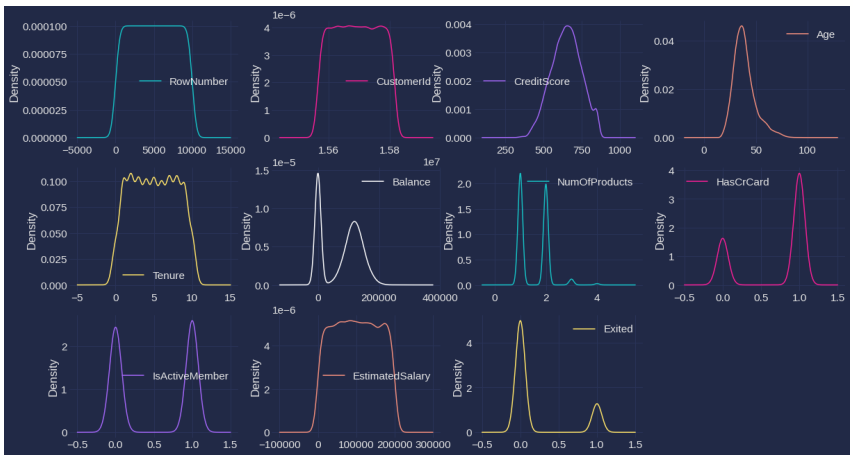
```
(10000, 14)
(0, 14)
```

```
df[df.duplicated(keep=False)]
```

```
RowNumber  CustomerId  Surname  CreditScore  Geography  Gender  Age  Tenure  Balance  NumOfProducts  HasCrCard  IsActiveMember  EstimatedSalary  Exited
```

- DF mempunyai 10000 baris, dan 14 columns of data
- Setelah dicheck dengan duplicated() method ternyata tidak ada data yang terduplikasi

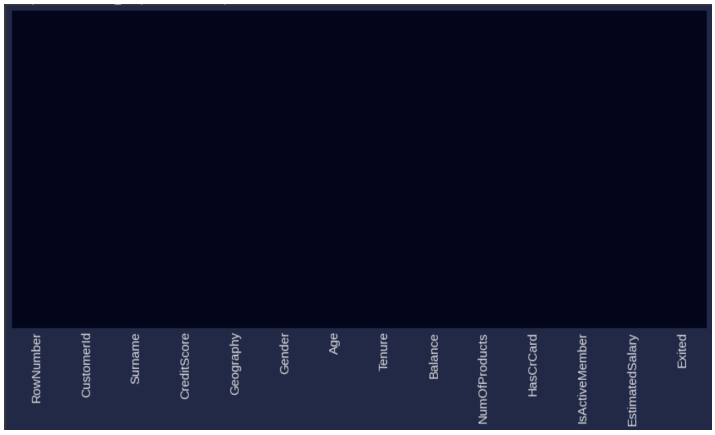
3.1.4 Data Distribution



3.1.5 Checking Missing Value

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
(10000, 14)
```

Dari pengecekan missing value dalam dataset, dapat dilihat ouput 0 menunjukkan bahwa tidak ada missing value di semua variable.



Dari pengecekan missing value dalam dataset, dapat dilihat ouput 0 menunjukkan bahwa tidak ada missing value di semua variable.

Check for imbalance for categorical data

```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

Output diatas menunjukkan banyaknya unique value dari variable Geography dengan 3 jenis Geography yang berbeda yaitu France, Germany, Spain.

```
Male       5457
Female     4543
Name: Gender, dtype: int64
```

Output diatas menunjukkan banyaknya unique value dari variable Gender dengan 2 macam Gender yang berbeda yaitu Male dan Female.

```
1    7055
0    2945
Name: HasCrCard, dtype: int64
```

Output diatas menunjukkan banyaknya unique value dari variable HasCrCard dengan 2 macam nilai yang berbeda yaitu, 1 berarti memiliki CrCard dan 0 tidak memiliki CrCard.

```
1    5151
0    4849
Name: IsActiveMember, dtype: int64
```

Output diatas menunjukkan banyaknya unique value dari variable HasCrCard dengan 2 macam nilai yang berbeda yaitu, 1 berarti memiliki CrCard dan 0 tidak memiliki CrCard.

3.2 Data Description

	count	mean	std	min	25%	50%	75%	max
RowNumber	10000.0	5.000500e+03	2886.895680	1.00	2500.75	5.000500e+03	7.500250e+03	10000.00
CustomerId	10000.0	1.569094e+07	71936.186123	15565701.00	15628528.25	1.569074e+07	1.575323e+07	15815690.00
CreditScore	10000.0	6.505288e+02	96.653299	350.00	584.00	6.520000e+02	7.180000e+02	850.00
Age	10000.0	3.892180e+01	10.487806	18.00	32.00	3.700000e+01	4.400000e+01	92.00
Tenure	10000.0	5.012800e+00	2.892174	0.00	3.00	5.000000e+00	7.000000e+00	10.00
Balance	10000.0	7.648589e+04	62397.405202	0.00	0.00	9.719854e+04	1.276442e+05	250898.09
NumOfProducts	10000.0	1.530200e+00	0.581654	1.00	1.00	1.000000e+00	2.000000e+00	4.00
HasCrCard	10000.0	7.055000e-01	0.455840	0.00	0.00	1.000000e+00	1.000000e+00	1.00
IsActiveMember	10000.0	5.151000e-01	0.499797	0.00	0.00	1.000000e+00	1.000000e+00	1.00
EstimatedSalary	10000.0	1.000902e+05	57510.492818	11.58	51002.11	1.001939e+05	1.493882e+05	199992.48
Exited	10000.0	2.037000e-01	0.402769	0.00	0.00	0.000000e+00	0.000000e+00	1.00

Dari output diatas, dapat dilihat jumlah dari data yang ada, nilai rata-rata, standar deviasi, nilai minimal,
nilai q1 (kuartil bawah), median (q2), nilai q3 (kuartil atas), nilai maksimum dari tiap variable dalam dataset.

- Terdapat 10000 baris pada setiap kolom.
- Maksimum EstimatedSalary nya 199992.48, dan minimum EstimatedSalary nya 51002.11.
- Customer paling muda adalah 18 tahun, dan paling tua 92 tahun.
- Maksimum Balance dari tiap customer adalah 250898.09 dan miminum Balance nya 0.
- Pada kolom exited -> 0 artinya customer tidak meninggalkan bank (not churn), 1 artinya meninggalkan bank (churn).

Exited	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	5024.694964	1.569117e+07	651.853196	37.408389	5.033279	72745.296779	1.544267	0.707146	0.554565	99738.391772
1	4905.917526	1.569005e+07	645.351497	44.837997	4.932744	91108.539337	1.475209	0.699067	0.360825	101465.677531

- Notes : 0 artinya not churn customer, dan 1 artinya churn customer.
- Rata-rata not churn customer berusia 37 tahun, dan churn customer berusia 44 tahun.
- Rata rata not churn customer credit score nya 652 sedangkan churn customer credit score nya 645.
- Rata-rata not churn customer, dan churn customer menjadi client bank selama 5 tahun.
- Rata-rata balance not churn customer 72745 dan balance churn customer 91108

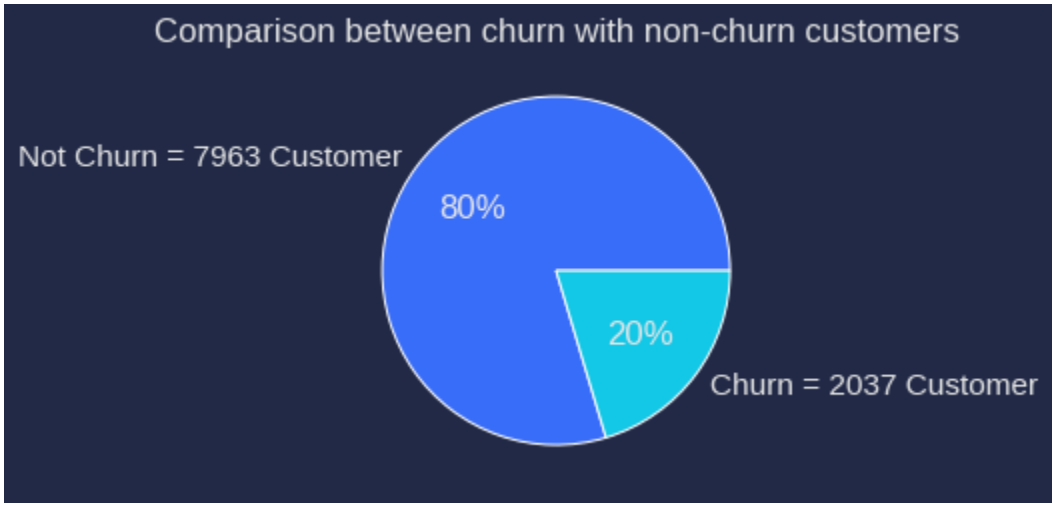
```
Continuous:  Age, CreditScore, Balance, EstimatedSalary
Categorical:  Geography, Gender, Tenure, NumOfProducts, HasCrCard, IsActiveMember
```

Di atas merupakan continous, dan categorical variable.

3.2.1 Exited

```
Not Churn Customer: 7963
Churn Customer: 2037
```

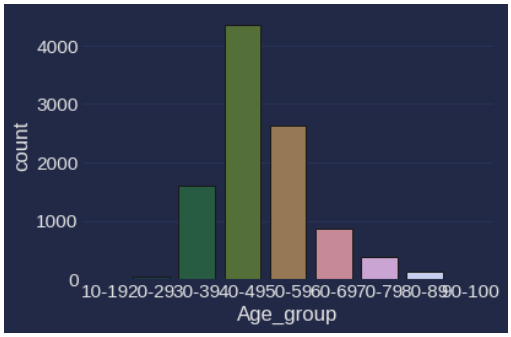
Terdapat 7963 not churn customer, dan 2037 churn customer



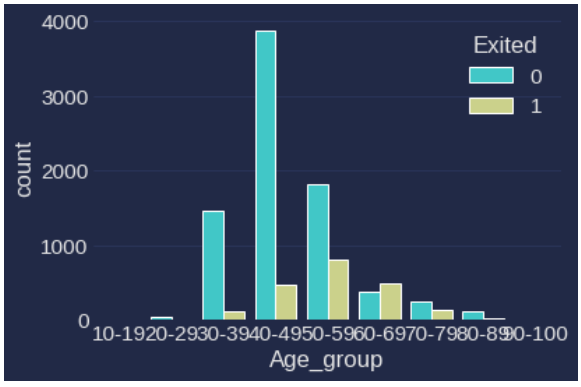
Dari 100 persen customer, customer yang churn adalah 20%, dan customer not churn adalah 80%.

3.2.2 Continuous Variable : Age, Balance, CreditScore, EstimatedSalary

A. Age

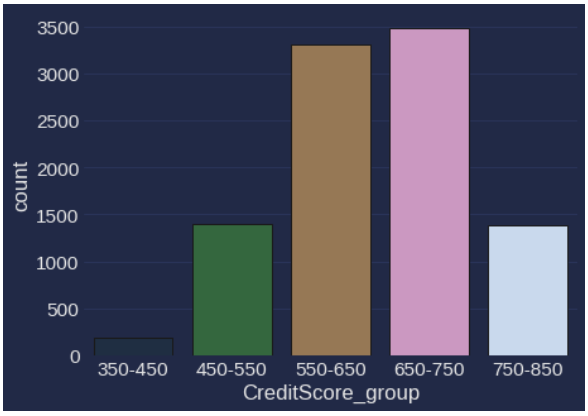


- Customer paling banyak berada di usia 40-49 dimana terdapat sekitar 4000 lebih customer.
- Lalu yang kedua paling banyak berada di usia 50-59 tahun dimana terdapat sekitar lebih dari 2500 customer.
- Paling sedikit customer berada di usia 90 -100 tahun, dan 20-29 tahun.

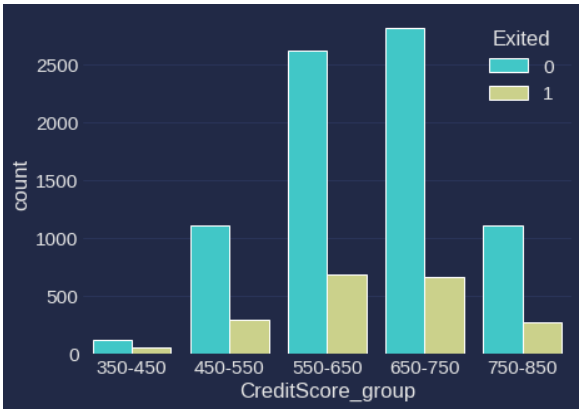


- Not churn customer paling banyak di usia 40-49 tahun, dan churn customer paling banyak di usia 50-59 tahun.
- Pada usia 60-69 tahun churn customer lebih banyak dari pada not churn customer.

B. Credit Score

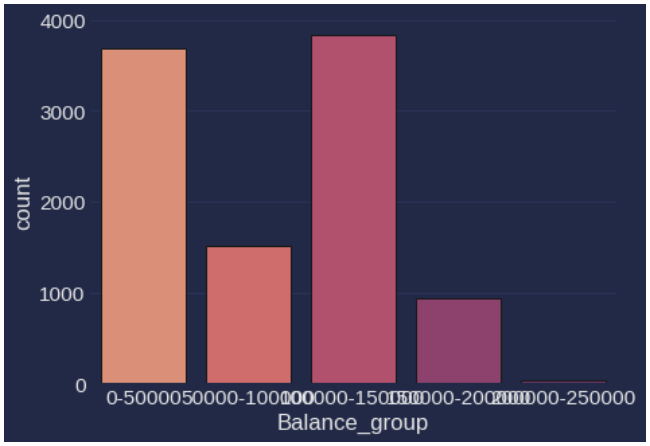


- Angka credit score tertinggi adalah 850, dan customer yang mempunyai 750-850 credit score adalah sekitar 1400 customer.
- Customer yang mempunyai 350-450 credit score ada sekitar 150 customer (paling sedikit).
- Customer yang mempunyai 650-750 credit score ada sekitar 3400 customer (paling banyak).

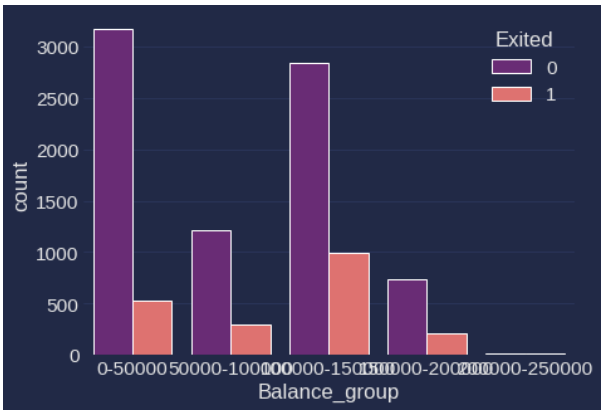


- Churn customer paling tinggi terdapat pada customer yang mempunyai credit score 550-650 tetapi not churn customernya juga kedua tertinggi dimana ada sekitar 2500 lebih yang mempunyai credit score 550-650.
- Not churn customer paling tinggi terdapat pada customer yang mempunyai credit score 650-750 dimana terdapat 2650 customer lebih.

C. Balance



- Terdapat sekitar 3750 customer yang mempunyai 100000-150000 balance (paling tinggi).
- Terdapat sekitar 10 customer yang mempunyai 200000-250000 balance (paling rendah).



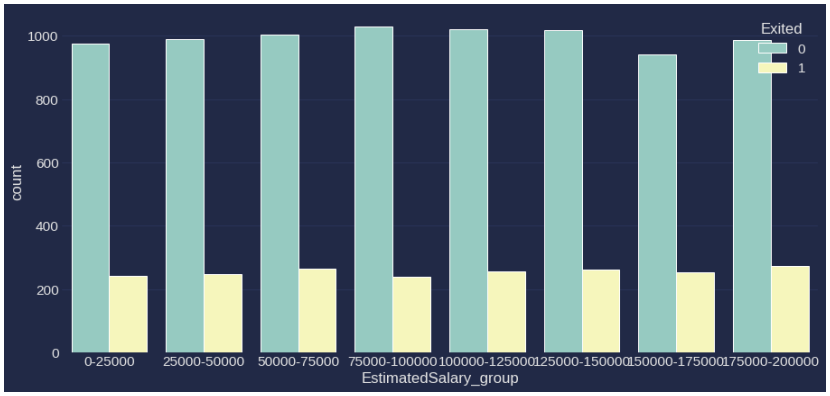
Not Churn Customer

- Terdapat 3000 lebih customer not churn yang mempunyai balance 50000 (paling banyak).
- Terdapat sekitar 10 customer not churn yang mempunyai balance 200000-250000 (paling sedikit).

Churn Customer

- Terdapat 900 lebih customer churn yang mempunyai balance 100000-150000 (paling banyak).
- Terdapat sekitar 10 customer churn yang mempunyai balance 200000-250000 (paling sedikit).

D. Estimated Salary



Not Churn Customer

- Banyak nya customer dengan estimated salary tertentu hampir sama. Tetapi yang terbanyak adalah 75000 - 100000.

3.2.3 Categorical Variable : Geography, Gender, Tenure, NumOfProducts, HasCrCard, IsActiveMember

- Customer terbanyak berasal dari France, dan paling sedikit berasal dari Germany.
- Lebih banyak customer bergender "Male" dari pada "Female".
- Produk yang dibeli customer paling banyak adalah 1, dan paling sedikit adalah 4.
- Customer kebanyakan mempunyai credit card.

- Jumlah customer yang bukan active member, dan active member hampir mirip.

A. Geography

Geography	Exited
France	0.161548
Spain	0.166734
Germany	0.324432

France memiliki rata-rata terendah yaitu 0.161548 dan Germany paling tinggi yaitu 0.324432.
Berarti customer yang berasal dari Germany paling cepat keluar.

B. Tenure

Tenure	Exited
7	0.172179
2	0.191794
8	0.192195
6	0.202689
4	0.205258
10	0.206122
5	0.206522
3	0.211100
9	0.216463
1	0.224155
0	0.230024

Tenure adalah berapa tahun client menjadi customer di bank.
Disini rata-rata tertinggi adalah 0 tahun, dan terendah 7 tahun.
Berarti yang 0 tahun paling cepat keluar.

C. Gender

Gender	Exited
Male	0.164559
Female	0.250715

Rata-rata gender terendah adalah male, dan tertinggi adalah Female. Ini berarti gender female lebih cepat keluar dari male.

D. NumOfProducts

NumOfProducts	Exited
2	0.075817
1	0.277144
3	0.827068
4	1.000000

Customer yang membeli 4 product lebih cepat keluar.

E. HasCrCard

HasCrCard	Exited
1	0.201843
0	0.208149

Customer yang tidak punya credit card lebih cepat keluar dimana rata-ratanya 0.208149.

F. IsActiveMember

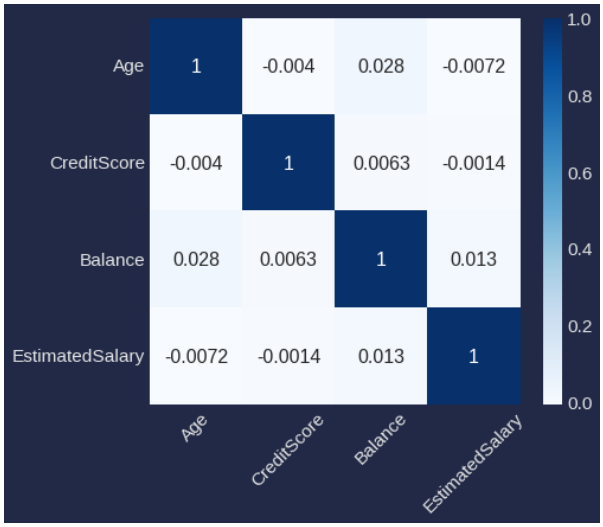
IsActiveMember	Exited
1	0.142691
0	0.268509

Customer yang bukan active member lebih mempunyai kemungkinan lebih cepat keluar dari pada yang aktif.

3.3 Checking Corelation

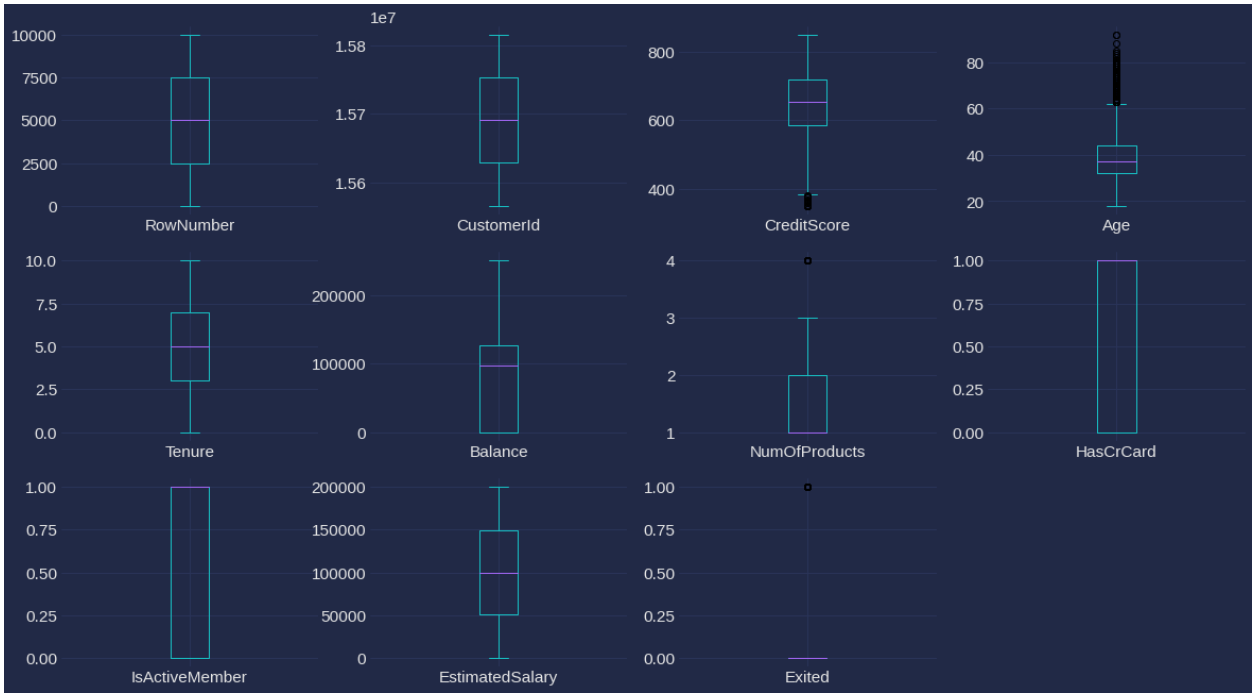
	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495	-0.009067	0.007246	0.000599	0.012044	-0.005988	-0.016571
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883	-0.012419	0.016972	-0.014025	0.001665	0.015271	-0.006248
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	0.000599	-0.014025	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.012044	0.001665	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.005988	0.015271	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.016571	-0.006248	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

Hasil diatas menunjukkan nilai korelasi dari tiap variable yang mewakili hubungan dengan variable lainnya. Nilai 0 berarti korelasi antar variable nya rendah, sedangkan 1 berarti korelasi antar variable nya tinggi.



Visualisasi plot diatas menunjukkan nilai korelasi tertinggi pada variable Balance dengan CreditScore. Hal ini berarti korelasi antar kedua variable saling mempengaruhi satu dan lainnya. Sedangkan, korelasi terendah ada pada variable EstimatedSalary dengan CreditScore.

3.4 Checking Outliers



Terdapat outliers pada CreditScore, Age, NumOfProducts tetapi ini represent natural variations di populasi. Jadi tidak perlu dihapus.

3.5 Data Preprocessing

3.5.1 Feature Selection

```
df.drop(["RowNumber", "CustomerId", "Surname", "Age_group", "EstimatedSalary_group", "CreditScore_group", "Balance_group", "EstimatedSalary"],
```

Setelah melakukan EDA, kolom RowNumber, CustomerId, Surname, Age_group, EstimatedSalary_group, CreditScore_group,Balance_group, dan EstimatedSalary dihapus karena pengaruh nya kecil terhadap peningkatan hasil akurasi dalam prediksi.

Are the variables statistically significant?

	Variable	Chi-square	p-value
3	NumOfProducts	1503.629362	0.000000e+00
0	Geography	301.255337	3.830318e-66
5	IsActiveMember	242.985342	8.785858e-55
1	Gender	112.918571	2.248210e-26
2	Tenure	13.900373	1.775846e-01
4	HasCrCard	0.471338	4.923724e-01

Cek variable yang signifikan terhadap model yang ingin dibuat, nilai p-value < 0.05 berarti variable signifikan dan p-value > 0.05 berarti variable tidak signifikan.

```
df.drop(['Tenure', "HasCrCard"], axis=1, inplace=True)
```

Berdasarkan pengecekan variable signifikan, kolom Tenure dan HasCrCard dihapus karena kurang signifikan dilihat dari nilai p-value variable nya.

3.5.2 Encoding Categorical Feature

```
df = pd.get_dummies(df, columns=["Gender"], drop_first = True)

df = pd.get_dummies(df, columns=["Geography"], drop_first = True)

df
```

	CreditScore	Age	Balance	NumOfProducts	IsActiveMember	Exited	Gender_Male	Geography_Germany	Geography_Spain
0	619	42	0.00	1	1	1	0	0	0
1	608	41	83807.86	1	1	0	0	0	1
2	502	42	159660.80	3	0	1	0	0	0
3	699	39	0.00	2	0	0	0	0	0
4	850	43	125510.82	1	1	0	0	0	1
...	...	...	...	...	...	...	...	...	...
9995	771	39	0.00	2	0	0	1	0	0
9996	516	35	57369.61	1	1	0	1	0	0
9997	709	36	0.00	1	1	1	0	0	0
9998	772	42	75075.31	2	0	1	1	1	0
9999	792	28	130142.79	1	0	0	0	0	0

10000 rows × 9 columns

- Encoding Categorical Feature -> mengganti categorical data into binary vector representation

- Ini digunakan saat working dengan machine learning algorithms seperti decision trees and support vector machines yang hanya menerima numeric inputs.
- Nilai dalam kolom direpresentasikan sebagai 1 dan 0 seperti di atas.

3.5.3 Feature Scaling

```
#scaler = StandardScaler()
scaler = MinMaxScaler(feature_range=(0,1))

select_columns = ['CreditScore', 'Age', 'Balance']
df[select_columns] = scaler.fit_transform(df[select_columns])

print('Features Scaled!')
```

```
Features Scaled!
```

	CreditScore	Age	Balance	NumOfProducts	IsActiveMember	Exited	Gender_Male	Geography_Germany	Geography_Spain
0	0.538	0.324324	0.000000	1	1	1	0	0	0
1	0.516	0.310811	0.334031	1	1	0	0	0	1
2	0.304	0.324324	0.636357	3	0	1	0	0	0
3	0.698	0.283784	0.000000	2	0	0	0	0	0
4	1.000	0.337838	0.500246	1	1	0	0	0	1

3.6 Creating a Train and Test Set

```
X = df.drop(["Exited"], axis=1)
y = df["Exited"]
```

Target variable (Y) pada modelling adalah `Exited`. Hapus kolom variable target untuk independent variable (X) agar datanya tidak mengandung kolom dari nilai yang mau diprediksi.

```
train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)
```

Split data modelling menjadi training dan test set dengan proporsi test set 20% dan training set 80% dari jumlah data.

```
print('Train set: {} rows x {} columns'.format(train_df.shape[0],
                                               train_df.shape[1]))
print(' Test set: {} rows x {} columns'.format(test_df.shape[0],
                                               test_df.shape[1]))
```

```
Train set: 8000 rows x 9 columns
 Test set: 2000 rows x 9 columns
```

Lihat dimensi dari training set dan test set yang sudah dibuat. Ada 8000 baris dan 9 kolom training set. Untuk test set, berisi 2000 baris dan 9 kolom data.

```
X_train, X_test, y_train, y_test = train_test_split(df.drop('Exited',axis=1),
                                                    df['Exited'], test_size=0.2,
                                                    random_state=42)
```

Split data menjadi 4 bagian. Ada training dan test set pada depedent variable (Y). Ada training dan test set pada independent variable (X). Hapus kolom target variable, yaitu `Exited` pada training dan test set yang berada di independent variable (X).

```
print('Train set: {} rows x {} columns'.format(X_train.shape,
                                               y_train.shape))
```

```
Train set: (8000, 8) rows x (8000,) columns
```

Terdapat 8000 baris dan 8 kolom pada training set independent variable (X). Sedangkan, ada 8000 baris dan 1 kolom pada training set pada dependent variable (Y).

```
over = SMOTE(sampling_strategy='auto', random_state=42)
X_train, y_train = over.fit_resample(X_train, y_train)

y_train.value_counts()
```

```
0    6356
1    6356
Name: Exited, dtype: int64
```

Ada 6356 baris data pada target variable. Terdapat 2 nilai pada target variable (Y) yaitu 0 berarti tidak meninggalkan bank dan 1 berarti meninggalkan bank.

3.7 Building Machine Learning Models

```
### Membuat fungsi untuk memvisualisasikan confusion matrix
def confmatrix(y_pred, title):
    cm = metrics.confusion_matrix(y_test, y_pred)
    df_cm = pd.DataFrame(cm, columns=np.unique(y_test), index = np.unique(y_test))
    df_cm.index.name = 'Actual'
    df_cm.columns.name = 'Predicted'

    plt.figure(figsize = (10,7))
    plt.title(title)

    sns.set(font_scale=1.4) # For label size
    sns.heatmap(df_cm, cmap="Blues", annot=True,annot_kws={"size": 16}) # Font size
```

Membuat fungsi untuk menampilkan confusion matrix dari hasil model machine learning menggunakan beberapa algoritma yang berbeda. Mengatur ukuran plot, judul, label x dan y, dan ukuran font dari plot.

3.8 Logistic Regression

```
### Instantiate the Algorithm
logreg = LogisticRegression()

### Train/Fit the model
logreg.fit(X_train, y_train)
```

```
LogisticRegression()
```

Algoritma pertama yaitu logistic regression. Untuk menggunakan algoritmanya, inisialisasi algoritma terlebih dahulu lalu fit dengan data training X dan Y.

```
### Predict on the test set
logreg_pred = logreg.predict(X_test)
```

Membuat prediksi menggunakan logistic regression dengan inisialisasi nilai prediksi yang digunakan yaitu data testing X (independent variable).

```
### Get performance metrics
logreg_score = metrics.accuracy_score(y_test, logreg_pred) * 100

### Print classification report
print("Classification report for {}:\n{}".format(logreg, metrics.classification_report(y_test, logreg_pred)))
print("Accuracy score:", logreg_score)
```

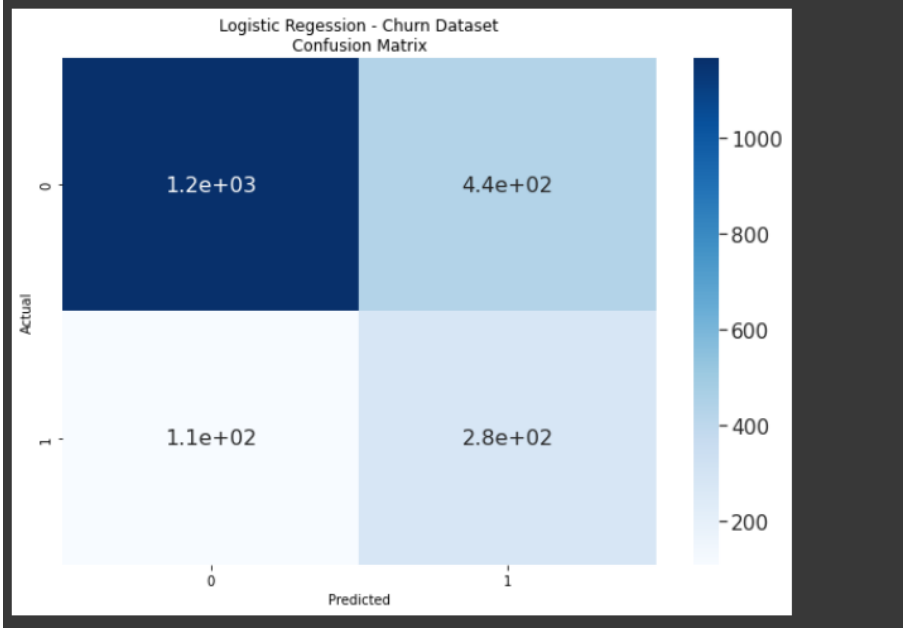
```
Classification report for LogisticRegression():
              precision    recall  f1-score   support

           0       0.91      0.73      0.81      1607
           1       0.39      0.72      0.51       393

    accuracy                           0.73      2000
   macro avg       0.65      0.72      0.66      2000
weighted avg       0.81      0.72      0.75      2000

Accuracy score: 72.5
```

Cek akurasi dari model logistic regression menggunakan performance metrics. Hasil output menunjukkan nilai akurasi menggunakan logistic regression sebesar 72.5% atau dibulatkan menjadi 73%. Output menunjukkan nilai recall sebesar 73% dan precision sebesar 81%. Nilai F1-score pada logistic regression sebesar 75%. Nilai akurasi didapatkan menggunakan 2000 sampel data yang diprediksi.

```
logreg_score_recall = metrics.recall_score(y_test, logreg_pred, average='weighted') * 100
logreg_score_precision = metrics.precision_score(y_test, logreg_pred, average='weighted') * 100
```

Untuk menghitung nilai recall dan precision dapat dilakukan menggunakan function mectrics.recall_score serta metrics.precison_score pada y_test yaitu prediksi dari logistic regression dengan menggunakan nilai rata-rata weighted lalu dikali dengan 100.

```
### Plot the confusion matrix
confmatrix(logreg_pred, "Logistic Regession - Churn Dataset\nConfusion Matrix")
```



Confusion matrix diatas menunjukkan nilai recall (false negative) sebesar 73%. Nilai recall pada confusion matrix dapat dilihat di kotak dengan label X bernilai 0 dan Y bernilai 1 berwarna putih. Sedangkan nilai precision menggunakan logistic regression sebesar 81%. Nilai precision (false positive) dapat dilihat melalui kotak dengan label X bernilai 1 dan Y bernilai 0 yang berwarna biru muda.

3.9 Random Forest

```
### Instantiate the Algorithm
ranfor = RandomForestClassifier()

### Train/Fit the model
ranfor.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

Algoritma kedua yaitu random forest. Untuk menggunakan random forest, inisialisasi algoritmanya terlebih dahulu lalu fit dengan data training X dan Y. Setelah itu, output akan menunjukkan function algoritma random forest, berarti algoritma sudah bisa digunakan.

```
ranfor_pred = ranfor.predict(X_test)
```

Membuat prediksi menggunakan random forest dengan inisialisasi nilai prediksi yang digunakan yaitu data testing X (independent variable).

```
### Get performance metrics
ranfor_score = metrics.accuracy_score(y_test, ranfor_pred) * 100

### Print classification report
print("Classification report for {}:\n{}".format(ranfor, metrics.classification_report(y_test, ranfor_pred)))
print("Accuracy score:", ranfor_score)
```

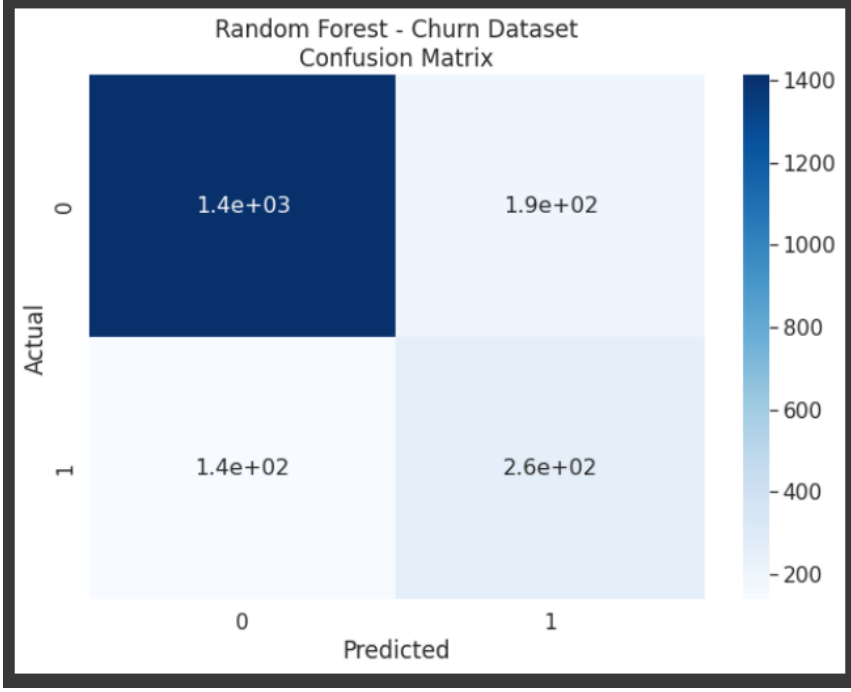
```
Classification report for RandomForestClassifier():
              precision    recall  f1-score   support

           0       0.91      0.88      0.90      1607
           1       0.57      0.65      0.61       393

    accuracy                           0.83      2000
   macro avg       0.74      0.76      0.75      2000
weighted avg       0.84      0.83      0.84      2000

Accuracy score: 83.45
```

Cek akurasi dari model yang telah dibuat menggunakan performance metrics. Hasil output menunjukkan nilai akurasi menggunakan random forest sebesar 83.45% atau dibulatkan menjadi 83%. Output menunjukkan nilai recall sebesar 83% dan precision sebesar 84%. Nilai F1-score pada random forest sebesar 84%, berarti cukup baik karena mendekati 1. Nilai akurasi didapatkan menggunakan 2000 sampel data yang diprediksi.

```
ranfor_score_recall = metrics.recall_score(y_test, ranfor_pred, average='weighted') * 100
ranfor_score_precision = metrics.precision_score(y_test, ranfor_pred, average='weighted') * 100
```

Untuk menghitung nilai recall dan precision dapat dilakukan menggunakan function mectrics.recall_score serta metrics.precison_score pada y_test yaitu prediksi dari random forest dengan menggunakan nilai rata-rata weighted lalu dikali dengan 100.

```
### Plot the confusion matrix
confmatrix(ranfor_pred, "Random Forest - Churn Dataset\nConfusion Matrix")
```



Confusion matrix diatas menunjukkan nilai recall (false negative) sebesar 83%. Nilai recall pada confusion matrix dapat dilihat di kotak dengan label X bernilai 0 dan Y bernilai 1 berwarna putih. Sedangkan nilai precision menggunakan logistic regression sebesar 84%. Nilai precision (false positive) dapat dilihat melalui kotak dengan label X bernilai 1 dan Y bernilai 0 yang berwarna kombinasi biru muda dan putih.

3.10 Gaussian Naive Bayes

```
### Instantiate the Algorithm
gnb = GaussianNB()

### Train the model
gnb.fit(X_train, y_train)
```

```
GaussianNB()
```

Algoritma ketiga yaitu gaussian naive bayes. Untuk menggunakan gaussian naive bayes, inisialisasi algoritmanya terlebih dahulu lalu fit dengan data training X dan Y. Setelah itu, output akan menunjukkan function algoritma gaussian naive bayes, berarti algoritma sudah bisa digunakan.

```
### Predict on the Test Set
gnb_pred = gnb.predict(X_test)
```

Membuat prediksi menggunakan gaussian naive bayes dengan inisialisasi nilai prediksi yang digunakan yaitu data testing X (independent variable).

```
### Get performance metrics
gnb_score = metrics.accuracy_score(y_test, gnb_pred) * 100

### Print classification report
print("Classification report for {}:\n{}".format(gnb, metrics.classification_report(y_test, gnb_pred)))
print("Accuracy score:", gnb_score)
```

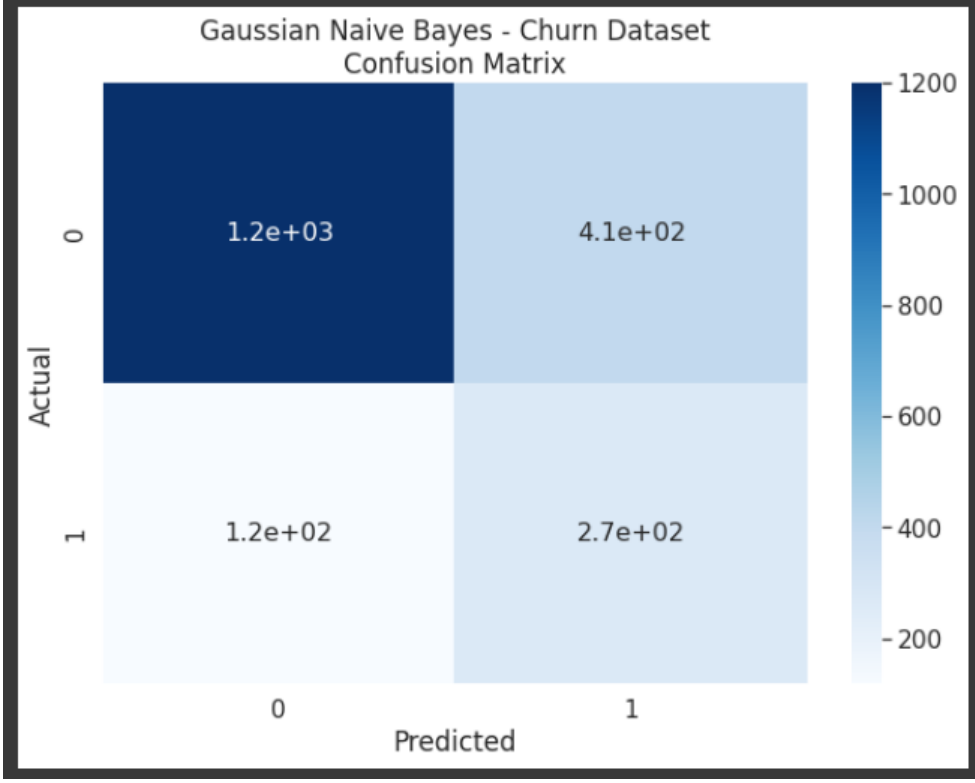
```
Classification report for GaussianNB():
              precision    recall  f1-score   support

           0       0.91      0.75      0.82      1607
           1       0.40      0.70      0.51       393

    accuracy                           0.74      2000
   macro avg       0.66      0.72      0.67      2000
weighted avg       0.81      0.74      0.76      2000

Accuracy score: 73.75
```

Hasil output menunjukkan nilai akurasi menggunakan gaussian naive bayes sebesar 73.75% atau dibulatkan menjadi 74%. Output menunjukkan nilai recall sebesar 74% dan precision sebesar 81%. Nilai F1-score pada gaussian naive bayes sebesar 76%. Nilai akurasi diatas didapatkan menggunakan 2000 sampel data yang diprediksi.

```
gnb_score_recall = metrics.recall_score(y_test, gnb_pred, average='weighted') * 100
gnb_score_precision = metrics.precision_score(y_test, gnb_pred, average='weighted') * 100
```

Untuk menghitung nilai recall dan precision dapat dilakukan menggunakan function mectrics.recall_score serta metrics.precison_score pada y_test yaitu prediksi dari gaussian naive bayes dengan menggunakan nilai rata-rata weighted lalu dikali dengan 100.

```
### Plot the confusion matrix
confmatrix(gnb_pred, "Gaussian Naive Bayes - Churn Dataset\nConfusion Matrix")
```



Confusion matrix diatas menunjukkan nilai recall (false negative) sebesar 74%. Nilai recall pada confusion matrix dapat dilihat di kotak dengan label X bernilai 0 dan Y bernilai 1 berwarna putih. Sedangkan nilai precision menggunakan logistic regression sebesar 81%. Nilai precision (false positive) dapat dilihat melalui kotak dengan label X bernilai 1 dan Y bernilai 0 yang berwarna biru muda.

3.11 LGBM Classifier

```
### Instantiate the Algorithm
lgbm = LGBMClassifier()

### Train the model
lgbm.fit(X_train, y_train)
```

```
LGBMClassifier()
```

Algoritma keempat yaitu LGBM classifier. Untuk menggunakan LGBM classifier, inisialisasi algoritmanya terlebih dahulu lalu fit dengan data training X dan Y. Setelah itu, output akan menunjukkan function algoritma LGBM classifier, berarti algoritma sudah bisa digunakan.

```
### Predict on the Test Set
lgbm_pred = lgbm.predict(X_test)
```

Membuat prediksi menggunakan LGBM classifier dengan inisialisasi nilai prediksi yang digunakan yaitu data testing X (independent variable).

```
### Get performance metrics
lgbm_score = metrics.accuracy_score(y_test, lgbm_pred) * 100

### Print classification report
print("Classification report for {}:\n{}".format(lgbm, metrics.classification_report(y_test, lgbm_pred)))
print("Accuracy score:", lgbm_score)
```

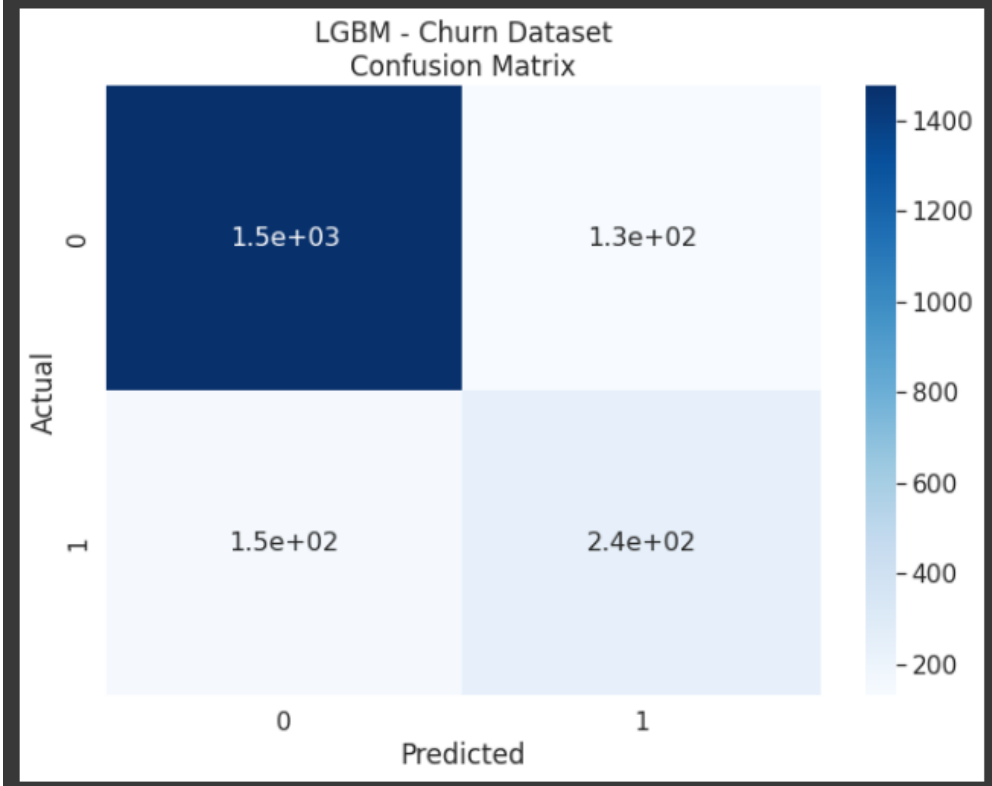
```
Classification report for LGBMClassifier():
              precision    recall  f1-score   support

           0       0.91      0.92      0.91      1607
           1       0.65      0.62      0.63       393

    accuracy                           0.86      2000
   macro avg       0.78      0.77      0.77      2000
weighted avg       0.86      0.86      0.86      2000

Accuracy score: 86.0
```

Cek akurasi dari model dengan LGBM classifier menggunakan performance metrics. Hasil output menunjukkan nilai akurasi menggunakan LGBM classifier sebesar 86%. Output menunjukkan nilai recall sebesar 86% dan precision sebesar 86%. Nilai F1-score pada gaussian naive bayes sebesar 86%. Nilai akurasi diatas didapatkan menggunakan 2000 sampel data yang diprediksi. Akurasi menggunakan LGBM classifier merupakan yang terbaik dibandingkan 3 algoritma lainnya.

```
lgbm_score_recall = metrics.recall_score(y_test, lgbm_pred, average='weighted') * 100
lgbm_score_precision = metrics.precision_score(y_test, lgbm_pred, average='weighted') * 100
```

Untuk menghitung nilai recall dan precision dapat dilakukan menggunakan function mectrics.recall_score serta metrics.precison_score pada y_test yaitu prediksi dari light gradient boosting machine dengan menggunakan nilai rata-rata weighted lalu dikali dengan 100.

```
### Plot the confusion matrix
confmatrix(lgbm_pred, "LGBM - Churn Dataset\nConfusion Matrix")
```



Confusion matrix diatas menunjukkan nilai recall (false negative) sebesar 86%. Nilai recall pada confusion matrix dapat dilihat di kotak dengan label X bernilai 0 dan Y bernilai 1 berwarna putih. Sedangkan nilai precision menggunakan logistic regression sebesar 86%. Nilai precision (false positive) dapat dilihat melalui kotak dengan label X bernilai 1 dan Y bernilai 0 yang berwarna putih.

BAB IV

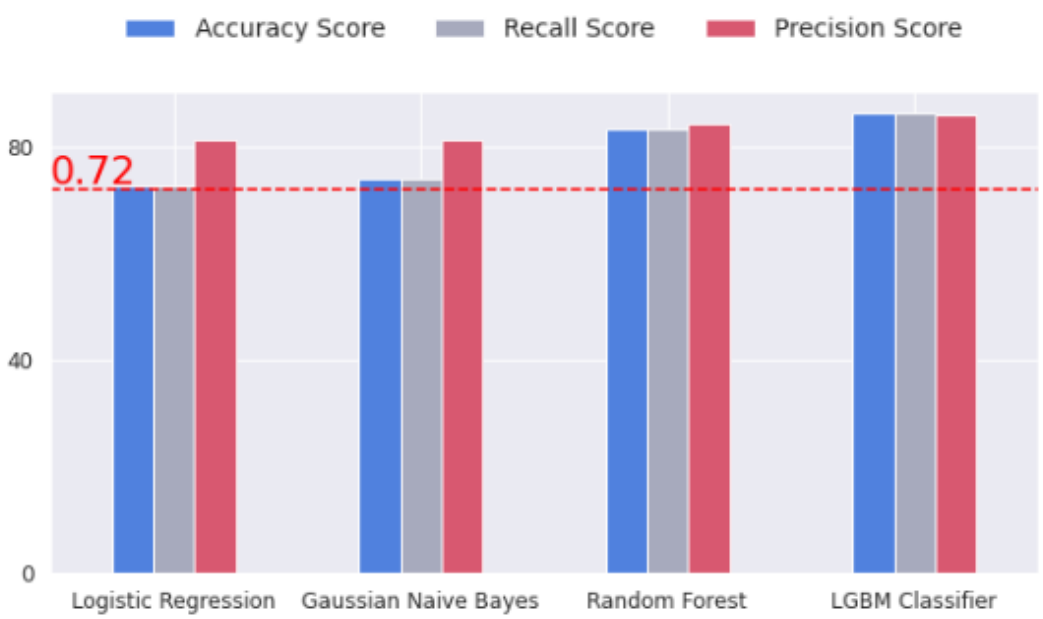
Pembahasan

4.1 Analisa

Pada mulanya kami membandingkan kinerja/performance model klasifikasi yang telah kami buat kedalam lima metrik individu yakni akurasi,recall, dan presisi. Perlu diingat bahwa dataset ini memiliki proporsi target yang kurang imbang. Nilai 0 pada target "Exited" jauh lebih banyak ketimbang 1 pada kolom yang sama dan oleh karena itu, kita tidak akan terlalu memperhatikan metrics akurasi untuk mengukur performance model yang telah dibuat. Kita akan melihat metrics lain yang cocok digunakan untuk data imbalance mengukur performance model dari presisi serta akurasi.

Pada kesempatan kali ini kami akan menggunakan metrics Recall dan Precision karena data kami bersifat cukup imbalanced. Metrics yang kami utamakan ialah Recall sebab tujuan kami adalah tidak melewatkan satu nasabah pun yang terindikasi *churn*. Recall menggambarkan berapa banyak kasus aktual positif yang berhasil diprediksi oleh model secara tepat, tentu kami menginginkan nilai recall yang semakin tinggi. Kami lebih mentoleransi nilai false negatif ketimbang nilai false positive. Tidak masalah jika kami salah mengatakan seorang nasabah tergolong *churn* ketimbang mengabaikan nasabah yang kenyataanya tergolong churn namun terprediksi tidak *churn*.

	Accuracy Score	Recall Score	Precision Score
Logistic Regression	72.500000	72.500000	81.120194
Gaussian Naive Bayes	73.750000	73.750000	81.024120
Random Forest	83.100000	83.100000	84.185744
LGBM Classifier	86.000000	86.000000	85.743669



Semua model pengklasifikasi yang kami buat memiliki nilai Recall lebih dari 72%. LGBM Classifier merupakan model yang memiliki nilai Recall paling tinggi diantara keempat model (86%). Secara keseluruhan LGBM Classifier memiliki performance yang paling baik dalam hal akurasi, recall, maupun presisi.

4.2 Feature Importance

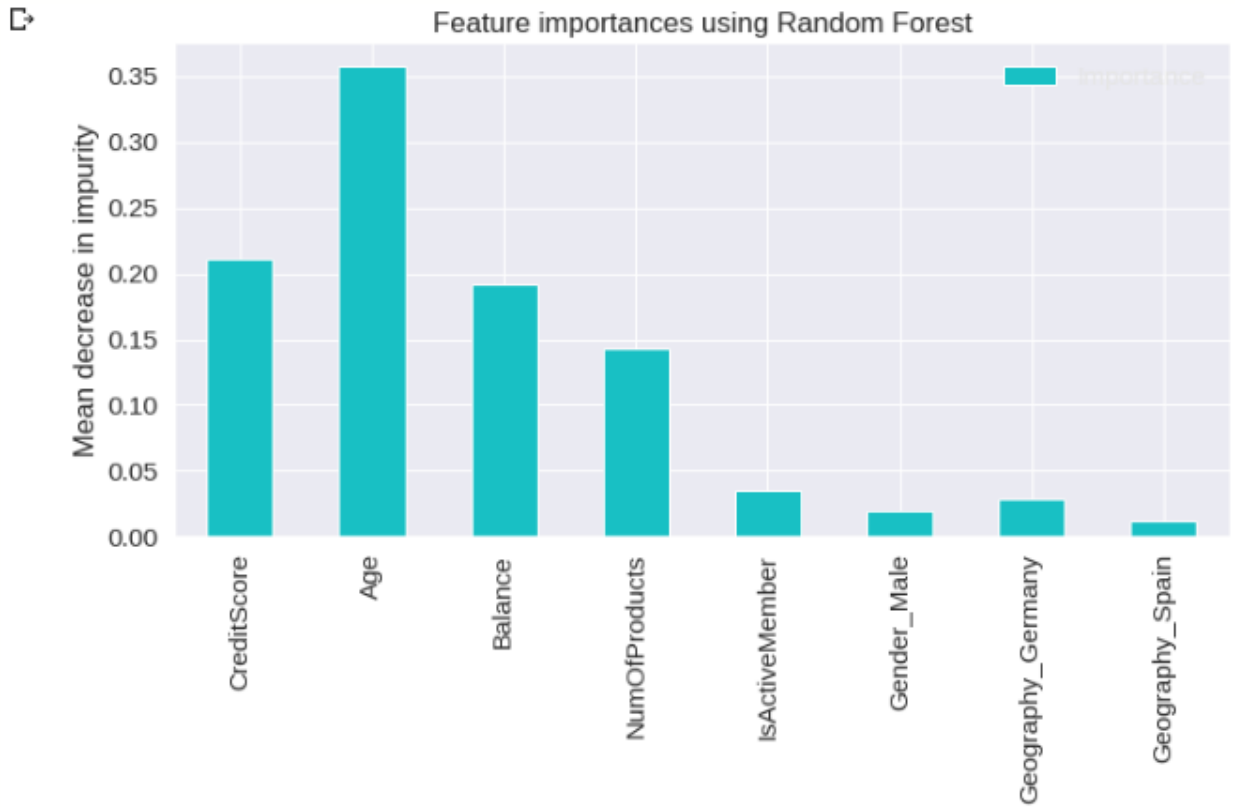
Beberapa algoritma dapat memberikan list atribut yang dianggap penting dalam pemodelan. *Feature Importance* mewakili skor setiap fitur, dimana skor yang lebih tinggi berarti bahwa fitur tersebut akan memiliki efek yang lebih besar pada model yang digunakan untuk memprediksi variabel tertentu (dalam hal ini target variabel kita yakni "Exited"). Berikut merupakan beberapa feature importance dari masing-masing algoritma:

4.2.1 Feature Importance Logistic Regression

	feature	importance
1	Age	414.061165
6	Geography_Germany	2.171957
2	Balance	1.910331
7	Geography_Spain	1.143665
3	NumOfProducts	0.902688
0	CreditScore	0.672564
5	Gender_Male	0.569309
4	IsActiveMember	0.401723

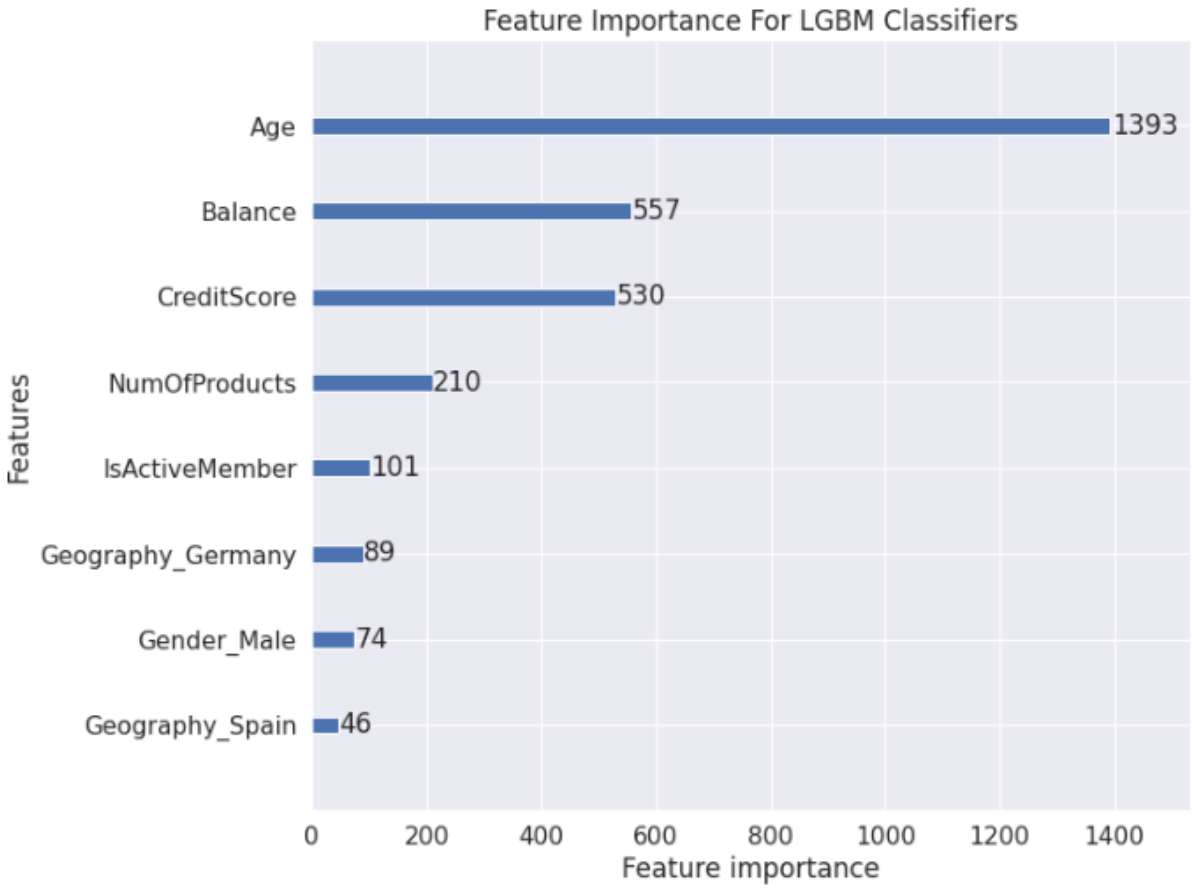
Pada algoritma logistic regression, tiga atribut teratas yang dianggap penting ialah usia, geografi(German), dan Balance. Sedangkan fitur yang tidak terlalu mempengaruhi ialah isActiveMember, Gender (Male), serta CreditScore.

4.2.2 Feature Importance Random Forest



Mirip dengan algoritma *logistic regression*, algoritma random forest memiliki 3 atribut terpenting yakni age, balance, dan credit score. Sedangkan fitur yang tidak terlalu mempengaruhi ialah Geography(Spain), Geography(Germany), serta Gender (Male).

4.2.3 Feature Importance Light Gradient Boosting Machine Classifier



Algoritma LGBM Classifier memiliki kemiripan dengan Random Forest, dimana 3 atribut yang dianggap paling berpengaruh ialah age, balance, dan creditscore. Sedangkan fitur yang tidak terlalu mempengaruhi ialah Geography(Spain), Geography(Germany), serta Gender (Male).

BAB IV

Kesimpulan

Dari serangkaian langkah yang telah kami lakukan dapat diketahui bahwa model klasifikasi terbaik didapatkan dari algoritma Light Gradient Boost Machine (LGBM). Algoritma tersebut memberikan nilai recall, presisi, dan akurasi tertinggi berturut-turut yaitu 86%, 86%, 86%. Hal ini disebabkan karena LGBM Classifier mampu bekerja baik pada dataset berukuran besar dengan waktu pelatihan yang jauh lebih signifikan dibandingkan dengan XGBoost. Selain itu algoritma LGBM mampu menghasilkan pohon yang lebih kompleks dengan akurasi yang lebih tinggi dibandingkan algoritma boosting lainnya.

Dalam pemodelan ini dapat diketahui juga tiga fitur yang paling berpengaruh terhadap model pengklasifikasian LGBM ini yaitu Age, Balance, dan Credit Score.

Source

Teori Classification

- https://algorit.ma/blog/classification-adalah-2022/

-https://student-activity.binus.ac.id/himmat/2021/03/machine-learning-supervised-and-unsupervised-learning/

Teori Logistic Regression

- https://vincentmichael089.medium.com/machine-learning-2-logistic-regression-96b3d4e7b603
- https://www.capitalone.com/tech/machine-learning/what-is-logistic-regression/
- https://www.javatpoint.com/logistic-regression-in-machine-learning

Teori Random Forest

- https://algorit.ma/blog/cara-kerja-algoritma-random-forest-2022/
- https://www.javatpoint.com/machine-learning-random-forest-algorithm

Teori Gaussian Naive Bayes

- https://www.javatpoint.com/machine-learning-naive-bayes-classifier

Teori LGBM Classifier

- https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf
- https://jurnal.tau.ac.id/index.php/siskom-kb/article/view/328/278