

# Data Goddesses

---

Dokumen  
Laporan Final  
Project

Customer Bank Churn



 **Nama Anggota :**

1. Aleisya Zahari Salam
2. Alicia Gofina
3. Devi Nur Aisyah
4. Deva Khofifah Jauharotun Naqiyah
5. Fiorent Arie Hernanti
6. Najmi Laily Fahira
7. Tisa Safina Alchalista

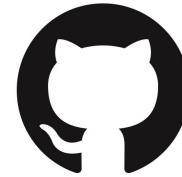
**Data Goddesses**  
**Kelompok 9A**



# GDrive & Github



[Link GDrive - Data Goddesses](#)



[Link Github - Data Goddesses](#)

# Stage 0

---

*Preparation*



# Latar Belakang Masalah

Problem Statement	Roles
<p>Bank CBA adalah perusahaan yang bergerak dibidang keuangan. Saat ini bank CBA <b>mengalami churn sebanyak 20.4%</b>, yang berarti sebanyak 20.4% nasabah berhenti menggunakan layanan bank. Churn dapat memiliki dampak serius, diantaranya:</p> <ul style="list-style-type: none"> <li>- Kehilangan nasabah dan pendapatan</li> <li>- Profitabilitas menurun</li> <li>- Keaktifan nasabah menurun</li> <li>- Penurunan kinerja</li> <li>- Pengaruh buruk pada citra merek</li> </ul>	<p>Untuk memahami dan mengurangi permasalahan ini, team business/manager meminta bantuan tim data untuk menganalisis sebab dan memberikan insight yang dapat diaplikasikan oleh perusahaan.</p> <p>Peran kami dalam dataset ini yaitu sebagai tim <b>data science</b> dari perusahaan bank CBA. Tugas yang kami lakukan yaitu <b>menganalisis dataset</b> bank untuk memahami pola perilaku nasabah bank, sehingga bisa <b>membangun model machine learning</b> yang mampu untuk <b>memprediksi nasabah churn atau tidak</b>.</p>

# Latar Belakang Masalah

Goals	Objectives	Business Metrics
Bank ingin mengurangi tingkat churn nasabah sehingga bank dapat mempertahankan nasabah yang ada dan memaksimalkan profit.	<ul style="list-style-type: none"><li>• Melakukan deep analytics atau EDA untuk mengidentifikasi faktor-faktor utama yang berpengaruh kepada keputusan nasabah churn.</li><li>• Membangun model machine learning klasifikasi untuk memprediksi apakah nasabah akan churn atau tidak</li></ul>	<b>Churn Rate</b> (persentase nasabah yang berhenti menggunakan layanan atau produk).

# Stage 1

---

*EDA, Insights & Visualization*



# Descriptive Analysis



## Customer Bank Churn Prediction

*Predicting Churn for Bank Customers*

**Link Dataset:**

<https://www.kaggle.com/datasets/adammaus/predicting-churn-for-bank-customers/data>

# Descriptive Analysis

## Basic Datasets Information

Dataset ini berisi data nasabah sebuah bank dan beberapa feature yang dapat menggambarkan value dari nasabah tersebut.

### Feature Description:

- **RowNumber**: Nomor baris dalam dataset
- **CustomerId**: ID unik untuk setiap nasabah
- **Surname**: Nama nasabah
- **CreditScore**: Skor kredit nasabah
- **Geography**: Lokasi geografis nasabah
- **Gender**: Jenis kelamin nasabah (male or female)
- **Age**: Usia nasabah
- **Tenure**: Jangka waktu nasabah telah menjadi nasabah bank
- **Balance**: Saldo rekening nasabah
- **NumOfProducts**: Jumlah produk yang dimiliki oleh nasabah
- **HasCrCard**: Indikator apakah nasabah memiliki kartu kredit (0: Tidak, 1: Ya)
- **IsActiveMember**: Indikator apakah nasabah aktif sebagai anggota (0: Tidak, 1: Ya)
- **EstimatedSalary**: Estimasi gaji nasabah
- **Exited**: Variabel target, indikator apakah nasabah churn atau tidak (0: Tidak, 1: Ya)



# Descriptive Analysis

## Basic Datasets Information

```
Shape of data : (10000, 14)
Number of rows : 10000
Number of columns : 14
```

```
<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
```

## Checking Missing Values

```
# jumlah entry NULL di setiap kolom
df.isna().sum()

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
```

### Observations:

- Dataset memiliki **14 columns** dan **10000 rows** data
- Tipe data masing-masing kolom sudah tepat, dan terdapat **3 jenis** data yaitu : **int64, object, float64**
- Antara nama kolom dan isi masing-masingnya sudah sesuai (tidak terdapat invalid data)

### Observations:

- Hasil output yang kita sertakan menunjukkan bahwa **tidak ada kolom yang memiliki nilai kosong** dalam dataset.
- Semua kolom memiliki jumlah entri yang sama dengan jumlah total baris data, ini berarti **tidak terdapat missing value**
- Dengan demikian, kita **tidak perlu melakukan imputasi untuk mengisi nilai kosong** dalam dataset ini.



# Descriptive Analysis



## Descriptive Numerical & Categorical

```
# title Default title text  
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
<b>count</b>	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
<b>mean</b>	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
<b>std</b>	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
<b>min</b>	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
<b>25%</b>	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
<b>50%</b>	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
<b>75%</b>	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
<b>max</b>	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

```
df.describe(include='O')
```

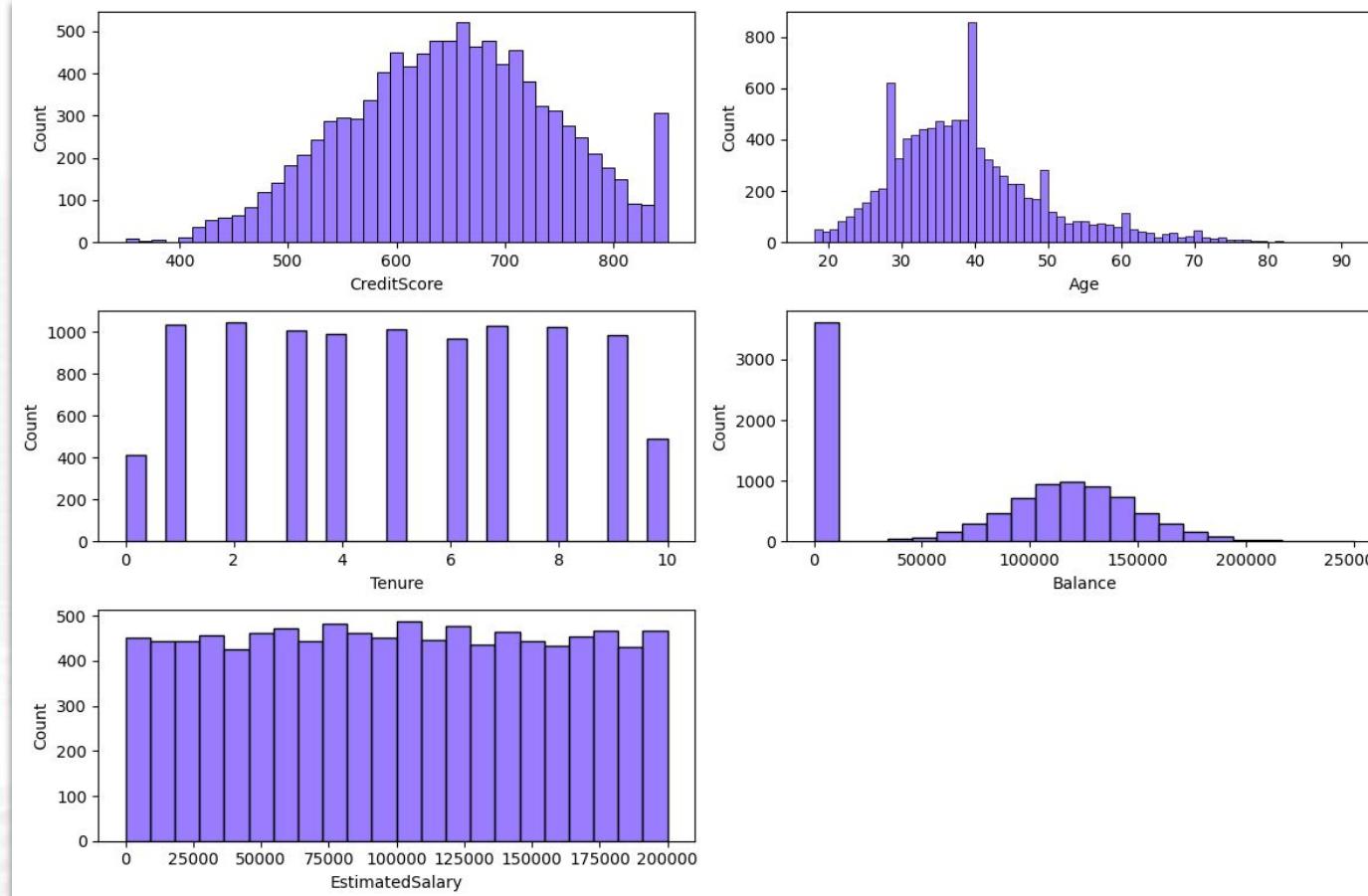
	Surname	Geography	Gender
<b>count</b>	10000	10000	10000
<b>unique</b>	2932	3	2
<b>top</b>	Smith	France	Male
<b>freq</b>	32	5014	5457

# Descriptive Analysis

## Observations:

- Terdapat data dengan jarak **mean** dan **median** yang **cukup kecil**. Dapat ditemukan pada kolom **CreditScore**, **Tenure**, **Age**, dan **EstimatedSalary**
- Terdapat data dengan nilai **median lebih besar dari mean**. Dapat ditemukan pada kolom **IsActiveMember**, **hasccard**, dan **balance**, pada kolom ini memiliki distribusi yang cenderung **negative skew**.
- Terdapat data dengan nilai **median yang lebih kecil dari mean**. Dapat ditemukan pada kolom **NumOfProducts**, pada kolom ini memiliki distribusi yang cenderung **positive skew**.
- Untuk kolom **RowNumber** dan **CustomerID** tidak akan dilakukan analisis lebih lanjut. Sebab kolom **RowNumber** merupakan **indeks** dari data dan **CostumerID** merupakan **data identifier** masing-masing nasabah.
- Pada data dengan tipe **kategorik**, **Geography** dan **Gender** sudah memiliki nilai **unique** yang sesuai, sedangkan **Surname** memiliki nilai **unique** yang sangat banyak sehingga tidak diperlukan untuk pembuatan model machine learning.

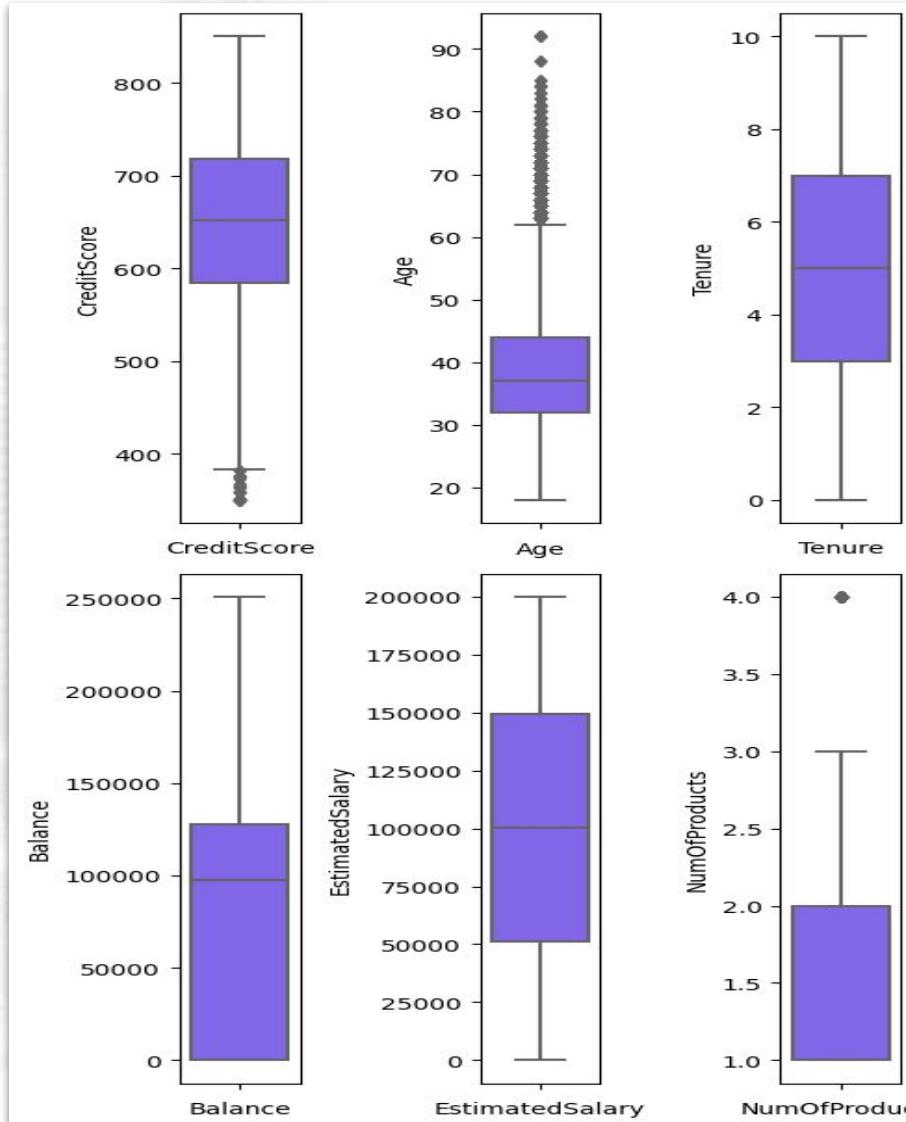
# Univariate Analysis



## Histplot Analysis

- Kolom **Credit Score** memiliki distribusi **negative skew** mendekati **normal**. Mayoritas customer memiliki jumlah credit score di rentang **600 sampai 700**, dan ditemukan **outliers** pada kolom ini dengan nilai **400 kebawah**.
- Kolom **Age** memiliki distribusi **Positive skew**. mayoritas customer berusia di rentang **30 tahun sampai 45 tahun**. Ditemukan pula **outliers** pada kolom ini dengan nilai paling ekstrem sekitar **90 tahun keatas**.
- Kolom **Tenure** memiliki distribusi merata.
- Kolom **Balance** cenderung memiliki distribusi yang normal, namun pada kolom ini terdapat satu nilai yang **mendominasi** yaitu **0**
- Kolom **EstimatedSalary** memiliki distribusi merata. Dimana jumlah customer pada tiap estimated salary itu **hampir sama rata**, artinya tidak ada kelompok gaji yang paling mendominasi.

# Univariate Analysis



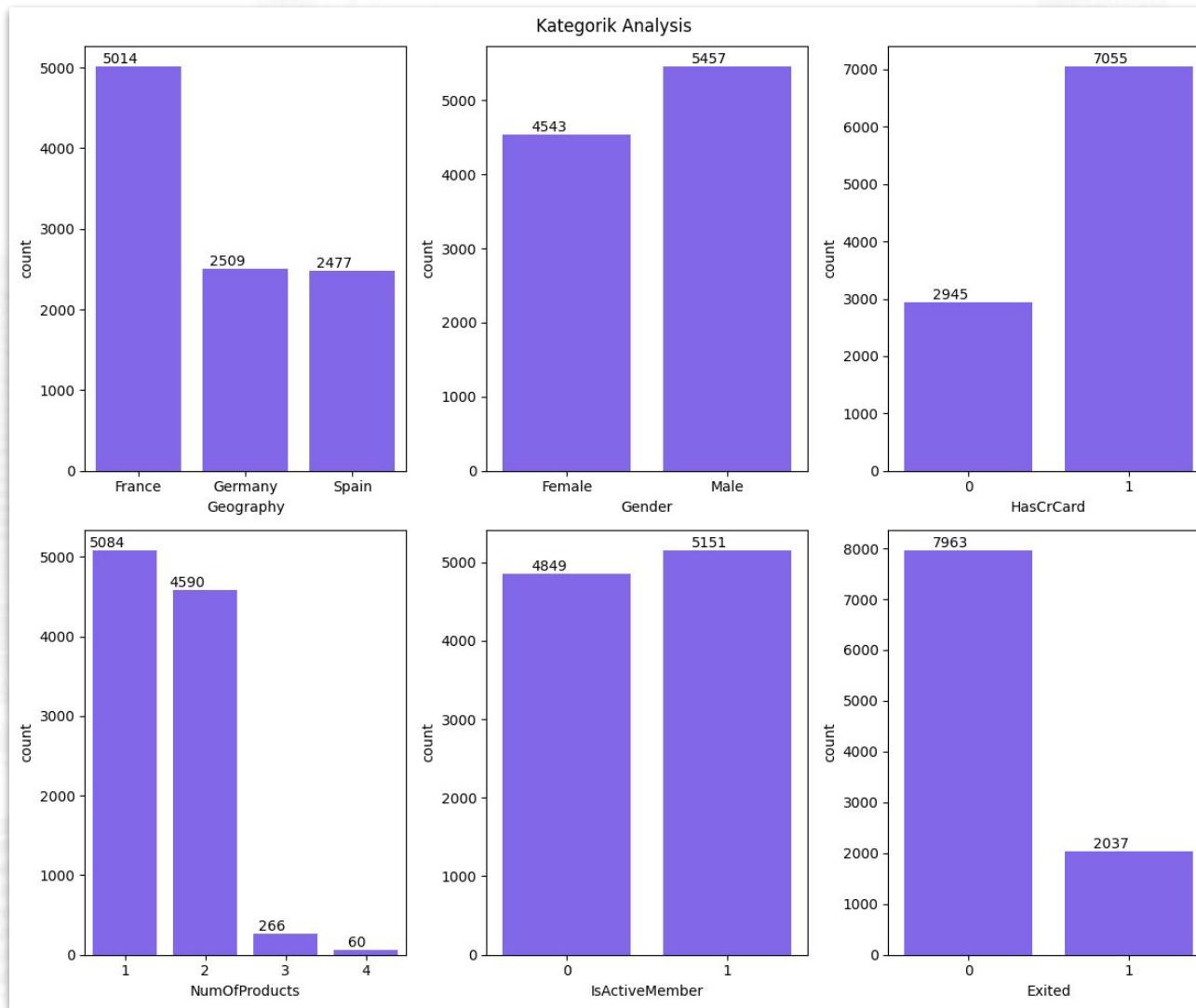
## Boxplot Analysis

- Kolom **CreditScore** ditemukan data **outlier** yang sebagian besar terletak di bawah **batas bawah IQR**.
- Kolom **Age** ditemukan data **outlier** yang sebagian besar terletak di atas **batas atas IQR**.
- Kolom **Tenure** tidak ditemukan adanya data outlier.
- Kolom **Balance** tidak ditemukan adanya data outlier.
- Kolom **EstimatedSalary** tidak ditemukan adanya data outlier.
- Kolom **NumOfProducts** ditemukan data **outlier** yang sebagian besar terletak di atas **batas atas IQR**.

## Tindak lanjut

- Untuk data yang mengandung outlier akan dihandle menggunakan metode IQR.

# Univariate Analysis



## Categorical Data Analysis

- Dalam kolom **Geography** terdapat nilai yang paling banyak yaitu **France**
- Dalam kolom **Gender** frekuensi tertinggi terdapat pada nilai '**1**' dan tidak terlalu terlihat perbandingan jumlah yang tinggi diantara nilai.
- Dalam kolom **HasCrCard** didapatkan bahwa **majoritas** nasabah **memiliki Credit Card**
- Dalam kolom **NumOfProducts**, dua nilai yang paling sering muncul adalah '**1**' dan '**2**'.
- Dalam kolom **IsActiveMember** tidak terlihat ada data dengan perbandingan yang cukup tinggi.
- Dalam kolom **Exited** terdapat nilai '**0**' yang paling umum, yaitu nasabah yang tidak churn.

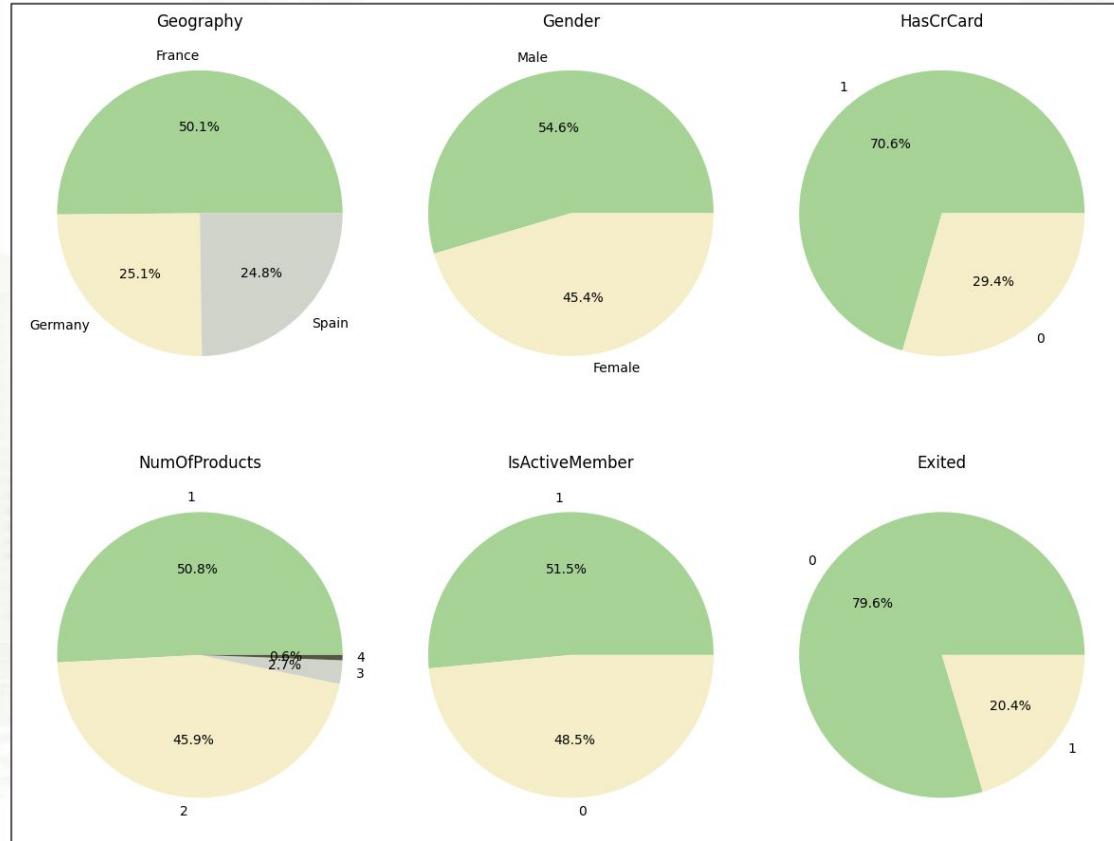
## Tindak Lanjut

- Berdasarkan visualisasi tersebut, ditemukan beberapa data yang **imbalance**. Oleh karena itu perlu dianalisis lebih lanjut untuk mengetahui sejauh mana ketidakseimbangan data tersebut.

## Catatan Tambahan

- Kolom **RowNumber** termasuk **index kolom** dan **CustomerID** merupakan **identifer unik** masing-masing nasabah, sehingga keduanya tidak dimasukkan kedalam tipe data kategorik.

# Univariate Statistics



Degree of imbalance

Mild

Moderate

Extreme

Proportion of Minority Class

20-40% of the data set

1-20% of the data set

<1% of the data set

## Imbalanced Data Analysis

source : [Google developer machine learning 'Imbalanced Data'](#)

### Hasil observasi :

- Dalam kolom **Geography** terdapat ketidakseimbangan data, kolom **Germany** dan **Spain** termasuk kedalam kategori **mild** dengan rasio masing-masing **25.1%** dan **24.8%**
- Dalam kolom **Gender** tidak terdapat ketidakseimbangan data.
- Dalam kolom **HasCrCard** terdapat ketidakseimbangan data pada nilai '1', termasuk kedalam kategori **mild** dengan rasio **29.4%**
- Dalam kolom **NumOfProducts** terdapat ketidakseimbangan data yang masuk kedalam kategori **extreme** untuk kolom nilai **4** sebanyak **0.6%**, dan kolom nilai **3** termasuk **moderate** dengan **ratio 2.7 %**
- Dalam kolom **IsActiveMember** tidak terdapat ketidakseimbangan data.
- Dalam kolom **Exited** terdapat ketidakseimbangan data. kolom dengan nilai '1' memiliki rasio **20.4%** yang termasuk kedalam kategori **mild**.

# Univariate Statistics

	Column Name	is Outlier	Lower Limit	Upper Limit	Outlier	No Outlier
0	CreditScore	True	383.00000	919.00000	15	9985
1	Age	True	14.00000	62.00000	359	9641
2	Tenure	False	-3.00000	13.00000	0	10000
3	Balance	False	-191466.36000	319110.60000	0	10000
4	EstimatedSalary	False	-96577.09625	296967.45375	0	10000
5	NumOfProducts	True	-0.50000	3.50000	60	9940

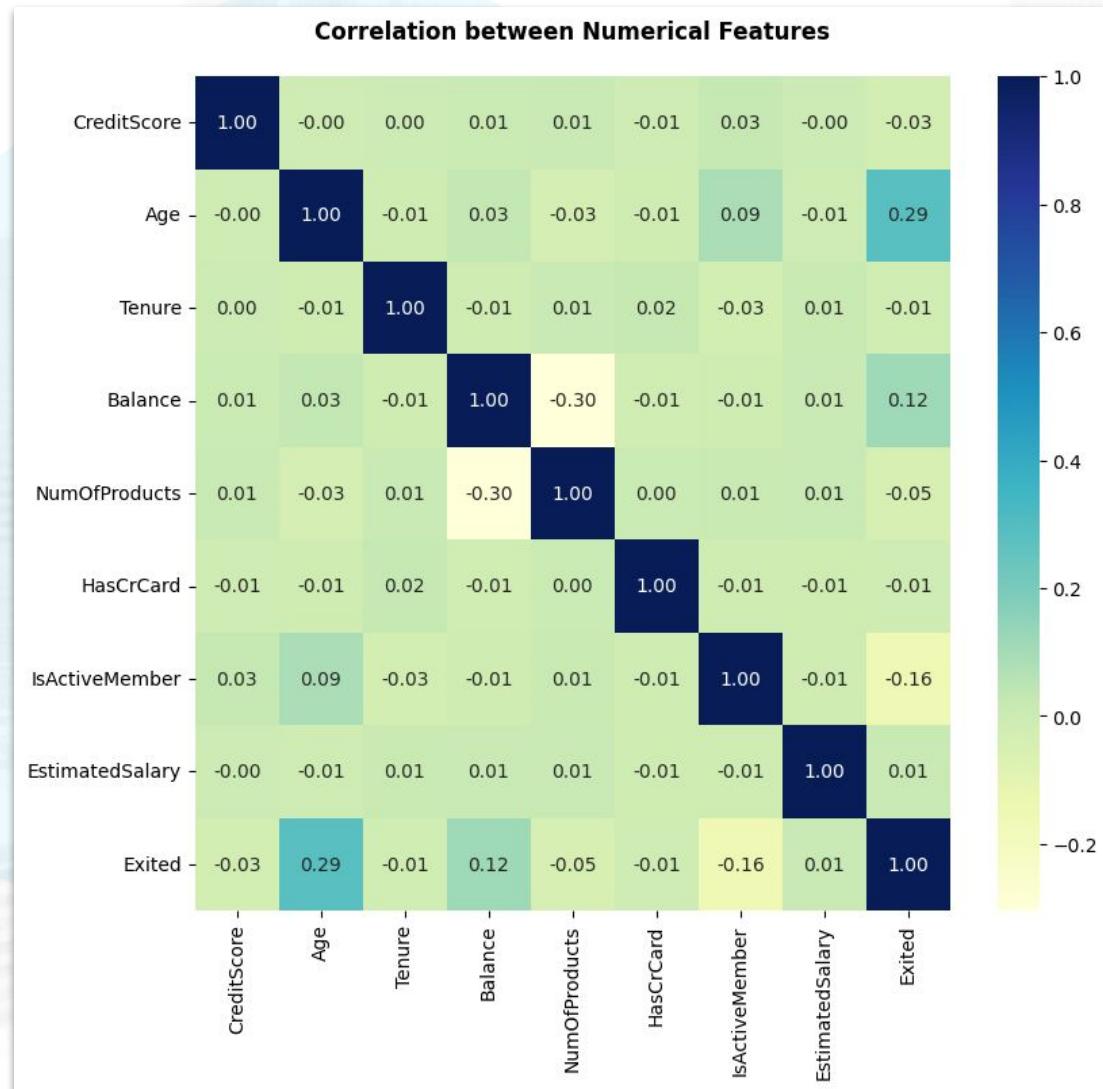
Jumlah baris: 10000  
 Outlier All Data : 432  
 Not Outlier All Data : 9568

## Observations :

Pada data tipe numerik terdapat outlier pada kolom **CreditScore, Age, dan NumOfProducts**.

- Pada kolom **CreditScore** terdapat **15** data **outlier**, dengan nilai terjauh terdapat **di bawah IQR** yaitu **350000**
- Pada kolom **Age** terdapat **359** data **outlier**, dengan nilai terjauh terdapat **di atas IQR** yaitu **92**.
- Pada kolom **NumOfProducts** terdapat **60** data **outlier**, dengan nilai terjauh terdapat **di atas IQR** yaitu **4**

# Multivariate Analysis

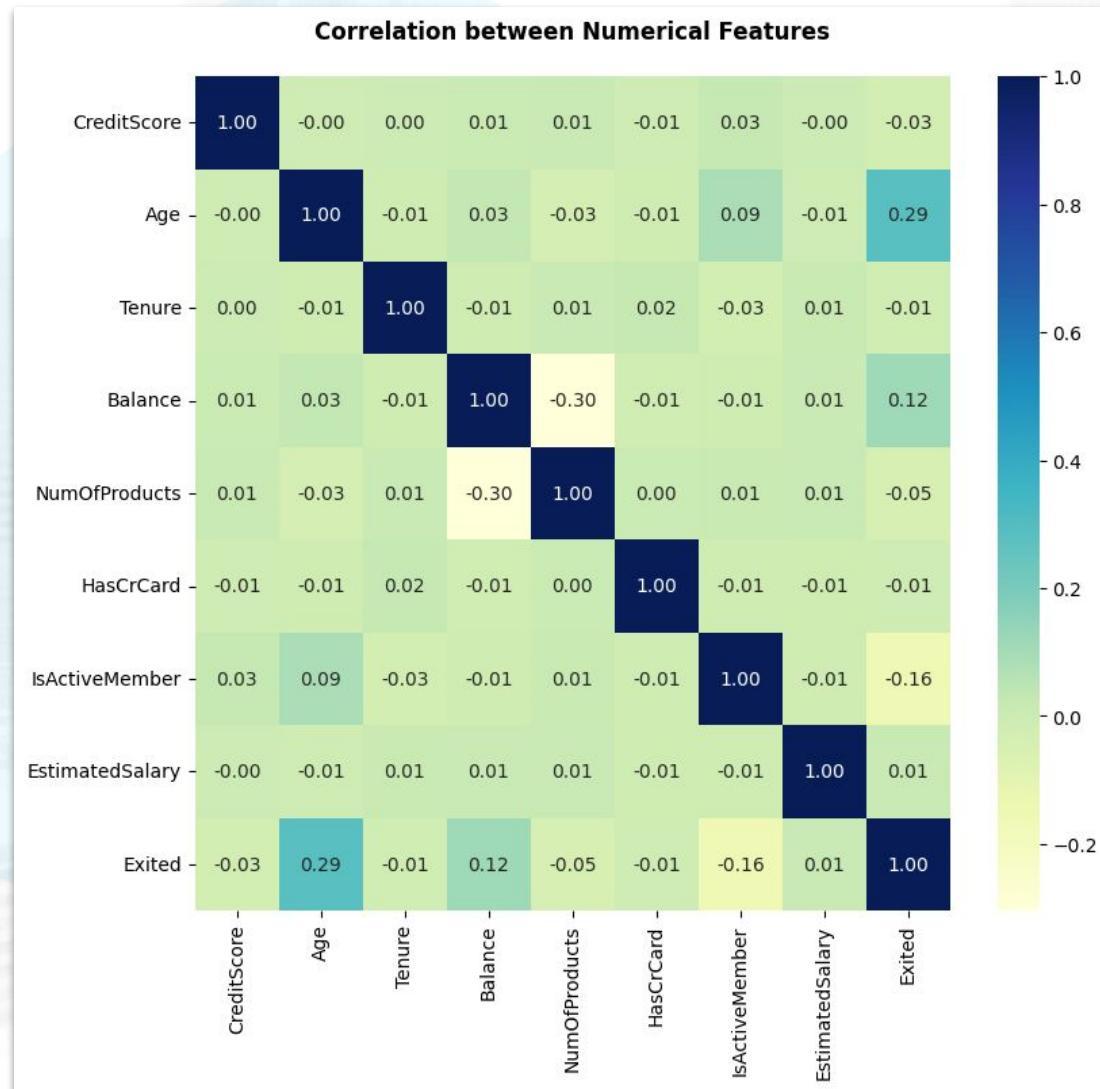


## 1. Korelasi feature ke target (Exited)

Beberapa feature yang kemungkinan besar relevan dan perlu dipertahankan :

- **Age** memiliki korelasi **positif 0.29**. Dapat dikatakan bahwa semakin tua nasabah, semakin tinggi kecenderungan mereka akan churn.
- **IsActiveMember** memiliki korelasi **negatif -0.16**. Dapat dikatakan bahwa semakin tinggi balance yang dimiliki nasabah, semakin tinggi juga kecenderungan mereka untuk churn.
- **Balance** memiliki korelasi **positif 0.12**. Dapat dikatakan bahwa nasabah yang tidak aktif memiliki kemungkinan keluar yang lebih tinggi (hubungan berbalik)
- **NumOfProducts** memiliki korelasi **negatif -0.05**. Dapat dikatakan bahwa nasabah dengan lebih banyak produk (NumOfProduct) cenderung memiliki kemungkinan churn yang lebih rendah.

# Multivariate Analysis



## 2. Korelasi antar feature

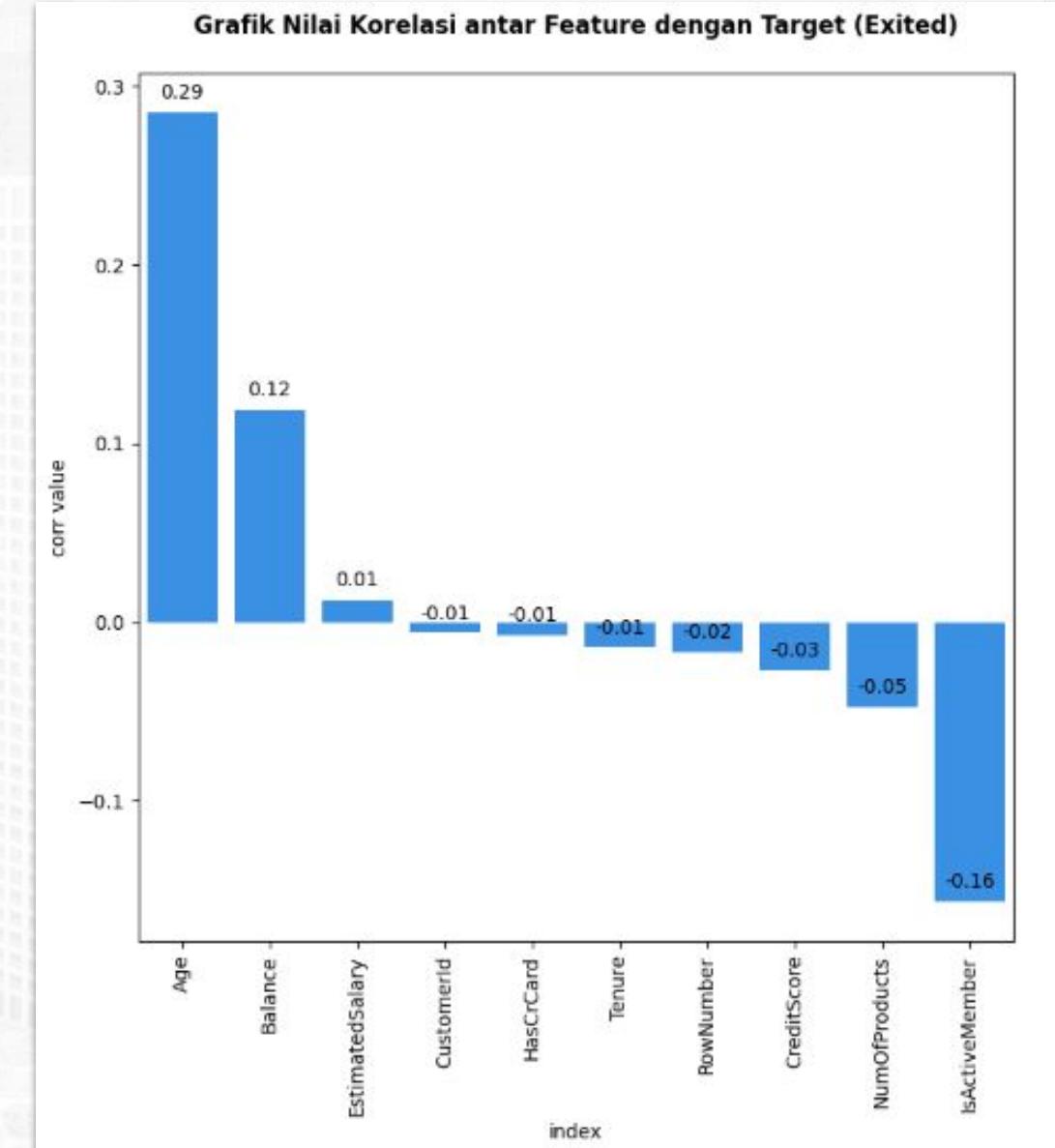
- **Age** dan **IsActiveMember** memiliki korelasi **positif 0.09**. Dapat dikatakan bahwa nasabah yang lebih tua, sedikit lebih cenderung menjadi anggota yang aktif.
- **Balance** dan **NumOfProduct** memiliki korelasi negatif **-0.30**. Dapat dikatakan bahwa nasabah yang dengan balance atau saldo rendah cenderung memiliki lebih banyak produk.

Karena korelasi antar kolom pada dataset tersebut cenderung memiliki nilai yang rendah, yaitu berada dalam range 0.00 hingga 0.29. Kami memutuskan untuk membuat nilai threshold pada angka **0.05** sehingga feature-feature dengan nilai korelasi  $\geq 0.05$  kemungkinan akan kami pertahankan.

# Multivariate Analysis

**Nilai korelasi** antar feature-feature **dengan target**

		index	corr value	Corr type
0	Exited	Exited	1.000000	Positif
1	Age	Age	0.285323	Positif
2	IsActiveMember	IsActiveMember	0.156128	Negatif
3	Balance	Balance	0.118533	Positif
4	NumOfProducts	NumOfProducts	0.047820	Negatif
5	CreditScore	CreditScore	0.027094	Negatif
6	RowNumber	RowNumber	0.016571	Negatif
7	Tenure	Tenure	0.014001	Negatif
8	EstimatedSalary	EstimatedSalary	0.012097	Positif
9	HasCrCard	HasCrCard	0.007138	Negatif
10	CustomerId	CustomerId	0.006248	Negatif

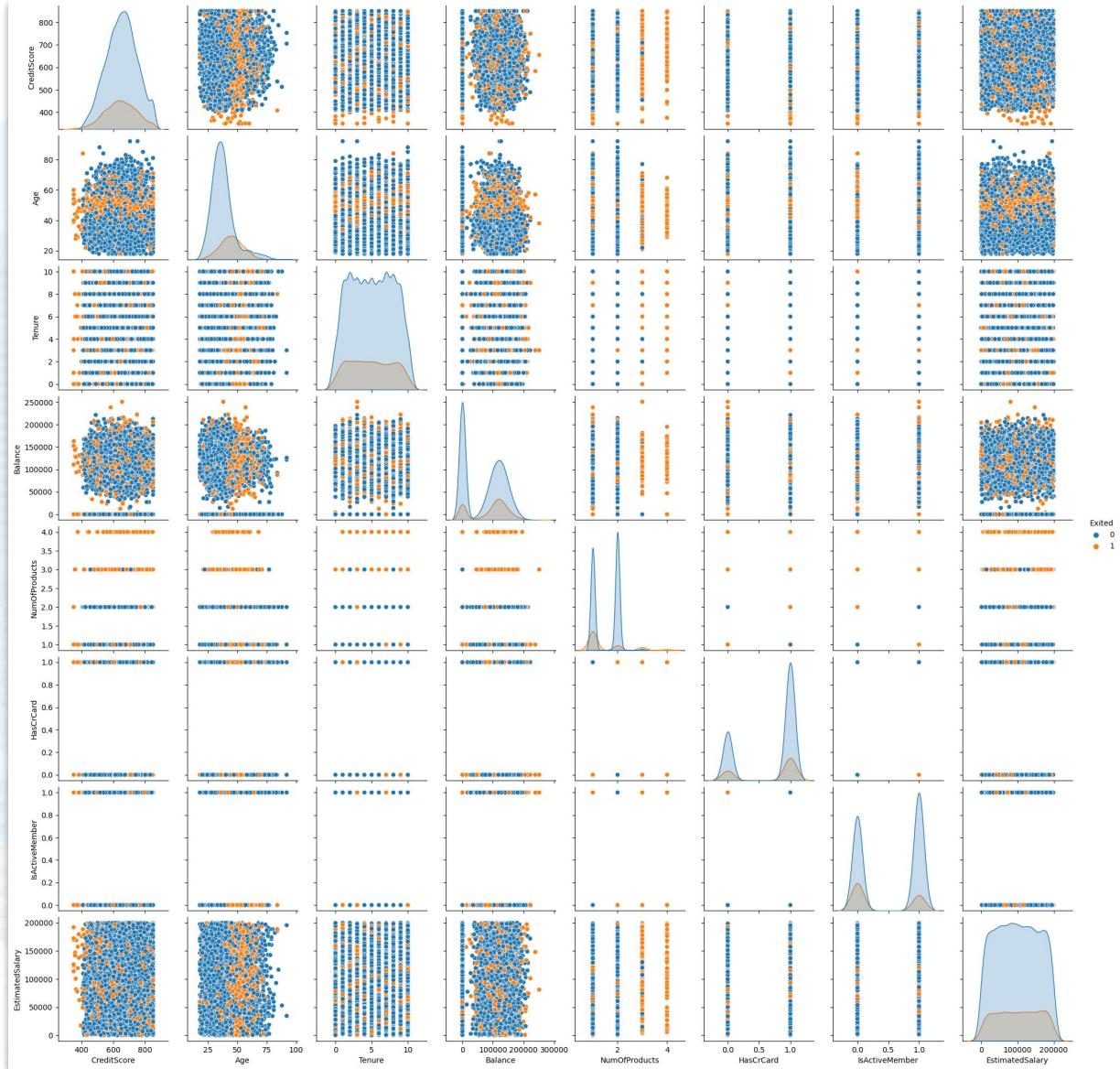


# Multivariate Analysis

Dari hasil nilai korelasi berdasarkan nilai korelasi terbesar terhadap variabel target "Exited", kita dapat menarik beberapa insight:

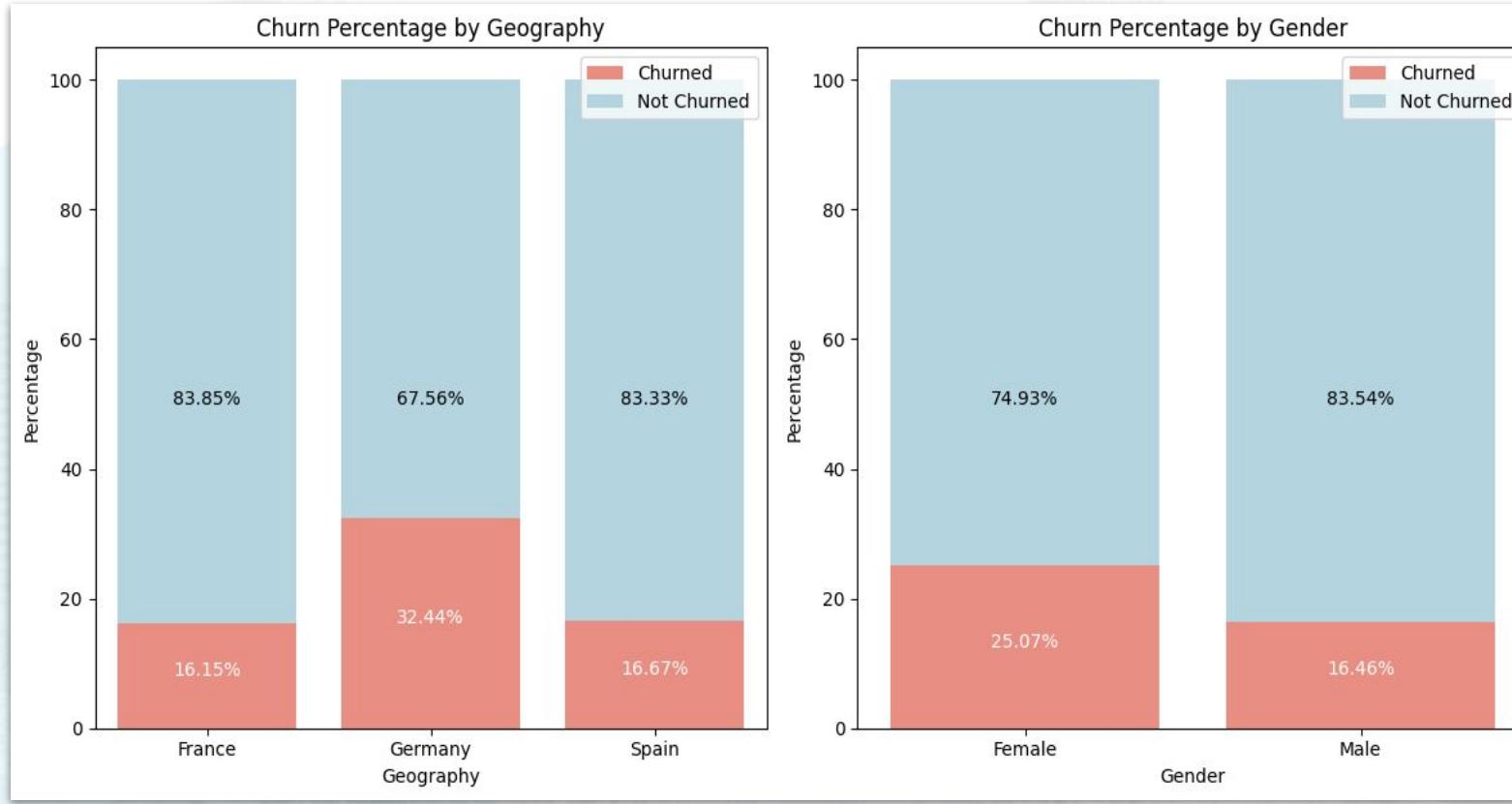
1. **Age** (Usia) memiliki korelasi positif yang cukup signifikan dengan keputusan nasabah untuk keluar. Semakin tua usia nasabah, semakin tinggi kemungkinan mereka untuk keluar dari layanan bank.
2. **IsActiveMember** (Status Keanggotaan Aktif) memiliki korelasi negatif yang cukup signifikan dengan keputusan nasabah untuk keluar. Nasabah yang tidak aktif cenderung memiliki tingkat churn yang lebih tinggi.
3. **Balance** (Saldo) juga memiliki korelasi positif yang cukup signifikan. Ini menunjukkan bahwa nasabah dengan saldo yang lebih tinggi cenderung memiliki kecenderungan untuk tetap sebagai nasabah.
4. **NumOfProducts** (Jumlah Produk) memiliki korelasi negatif yang cukup signifikan. Nasabah yang hanya menggunakan satu produk cenderung memiliki kecenderungan untuk keluar.
5. **CreditScore** (Skor Kredit), **Tenure** (Lama menjadi nasabah), **EstimatedSalary** (Perkiraan Gaji), dan variabel lainnya memiliki korelasi yang lebih rendah, menunjukkan dampaknya yang kurang signifikan terhadap keputusan nasabah untuk keluar.

# Multivariate Analysis



Ditemukan **sedikit kecenderungan** bahwa nasabah dengan **CreditScore** dibawah 400 dan **EstimatedSalary** sedang ke tinggi cenderung melakukan churn. Meskipun terdapat pola ini, kedua feature tersebut tidak cukup dikatakan berkorelasi dengan target, sehingga feature ini tidak relevan dan tidak perlu dipertahankan.

# Multivariate Analysis



Berdasarkan hasil keseluruhan analisis Multivariate tersebut, dapat diambil kesimpulan bahwa **feature yang paling relevan** dan harus dipertahankan adalah sebagai berikut :

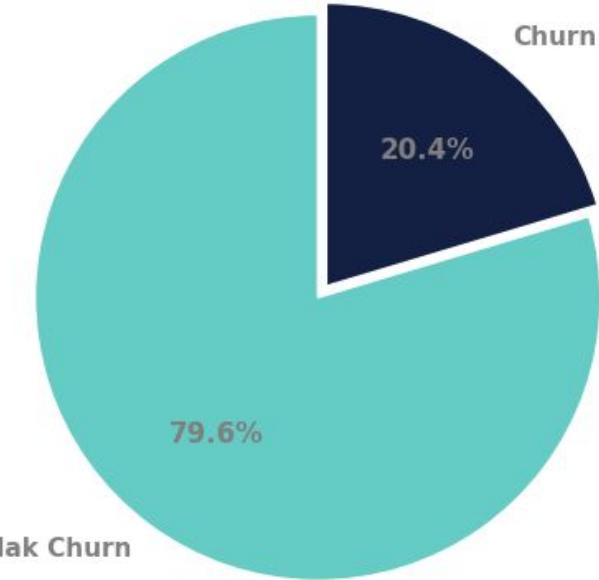
- **Age**
- **IsActiveMember**
- **Balance**
- **NumOfProduct**
- **Geography**

Korelasi antara fitur-fitur dan label Exited:

1. Pada **Geography**, terdapat **perbedaan yang signifikan** antara negara Germany, dengan France dan Spain yaitu dalam **proporsi nasabah yang churn**.
2. Sementara itu, **Gender tidak menunjukkan perbedaan yang signifikan** dalam visualisasi diatas, fitur mungkin kurang relevan (diperlukan analisis lebih lanjut)

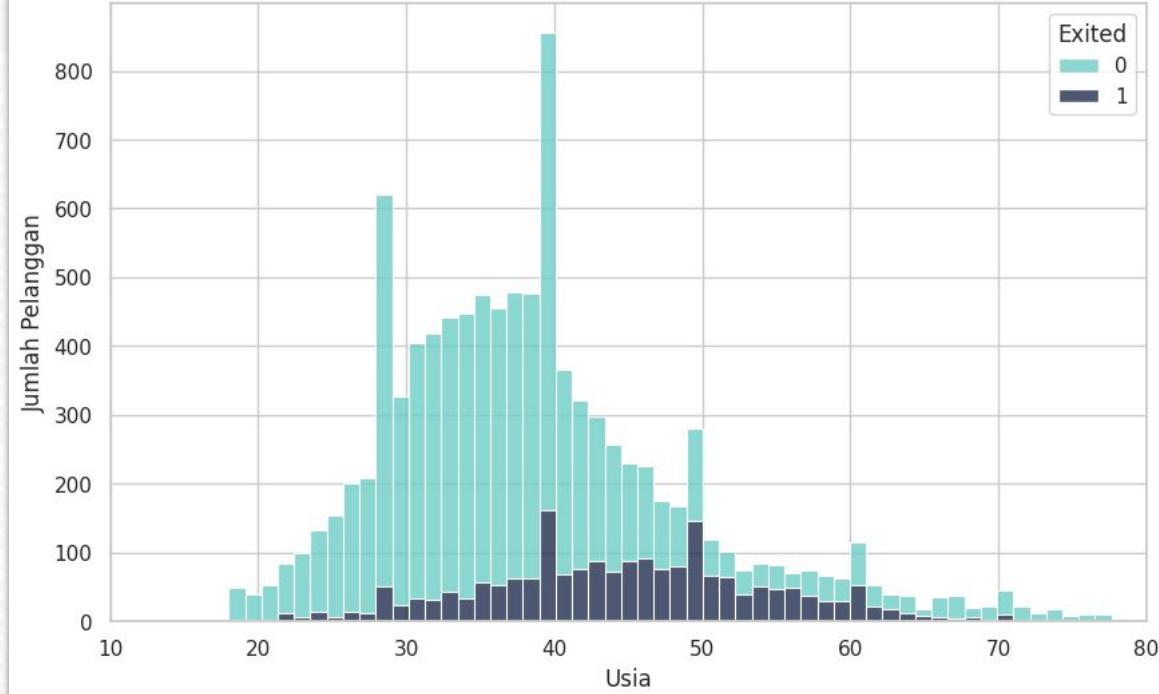
# Business Insight

Perbandingan Pelanggan yang Churn dan Tidak Churn



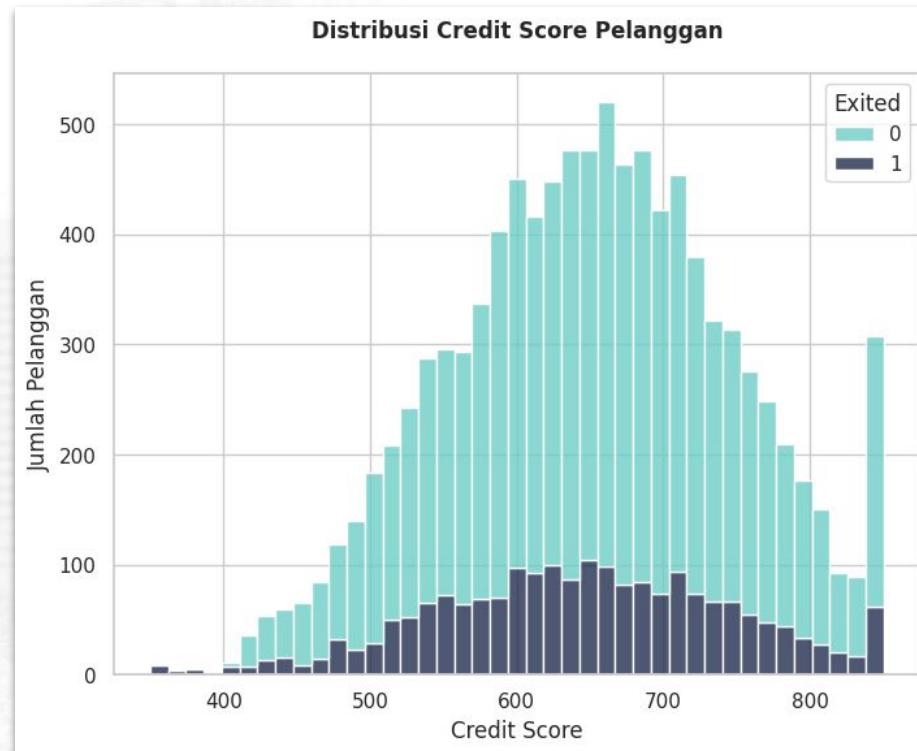
Grafik ini tersebut menunjukkan bahwa **tingkat churn** pada Bank CBA sebesar **20.4%**. Melampaui indikator tingkat churn yang baik pada umumnya, yaitu 5-7%.

Distribusi Usia Pelanggan

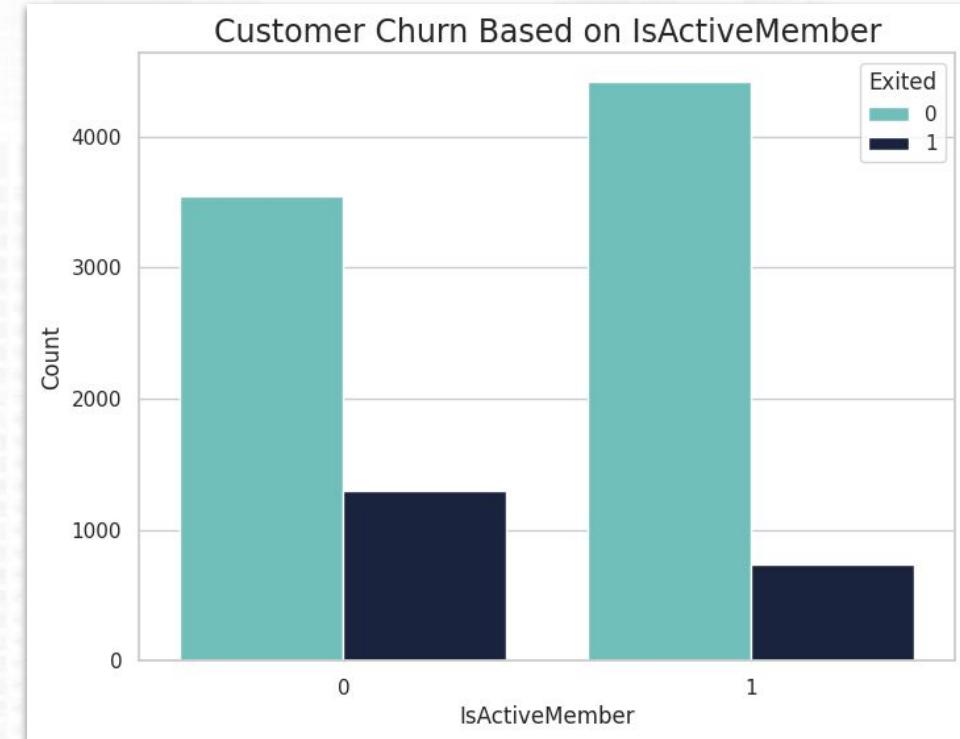


Terdapat kecenderungan bahwa **yang tua lebih banyak churn**, dimana customer **churn paling tinggi** terjadi di rentang **usia 40-60 tahun**. Ini mengindikasi bahwa **usia dapat menjadi faktor** dalam penentuan customer churn atau tidak.

# Business Insight

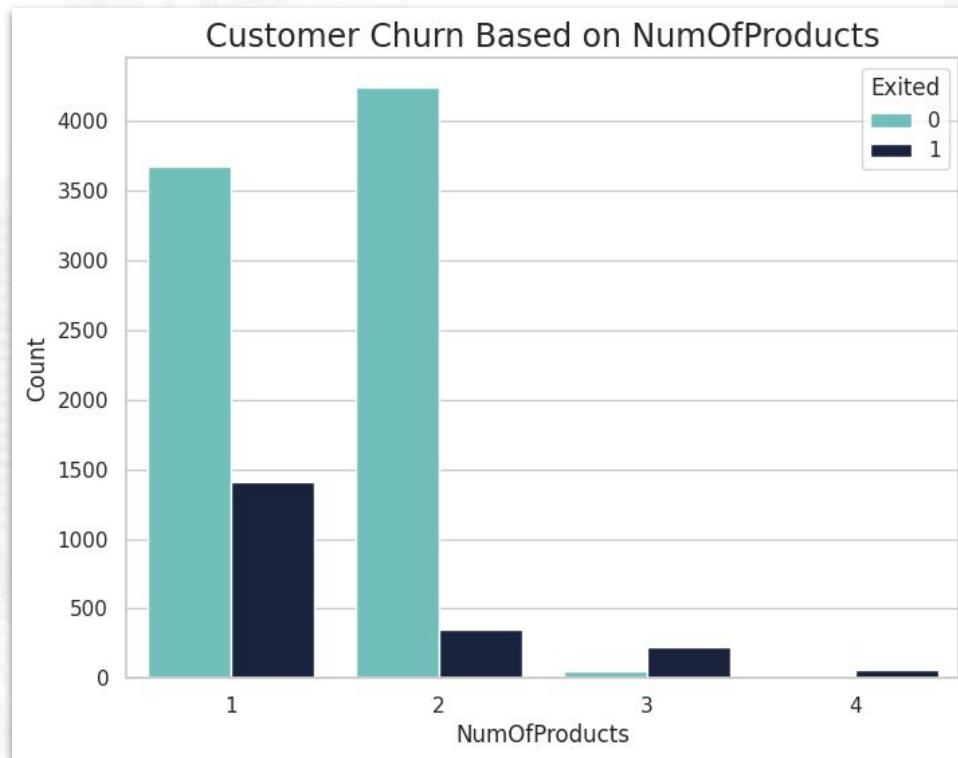


Terdapat kecenderungan **churn yang tinggi** pada nasabah dengan **CreditScore** rentang **600 - 700**.

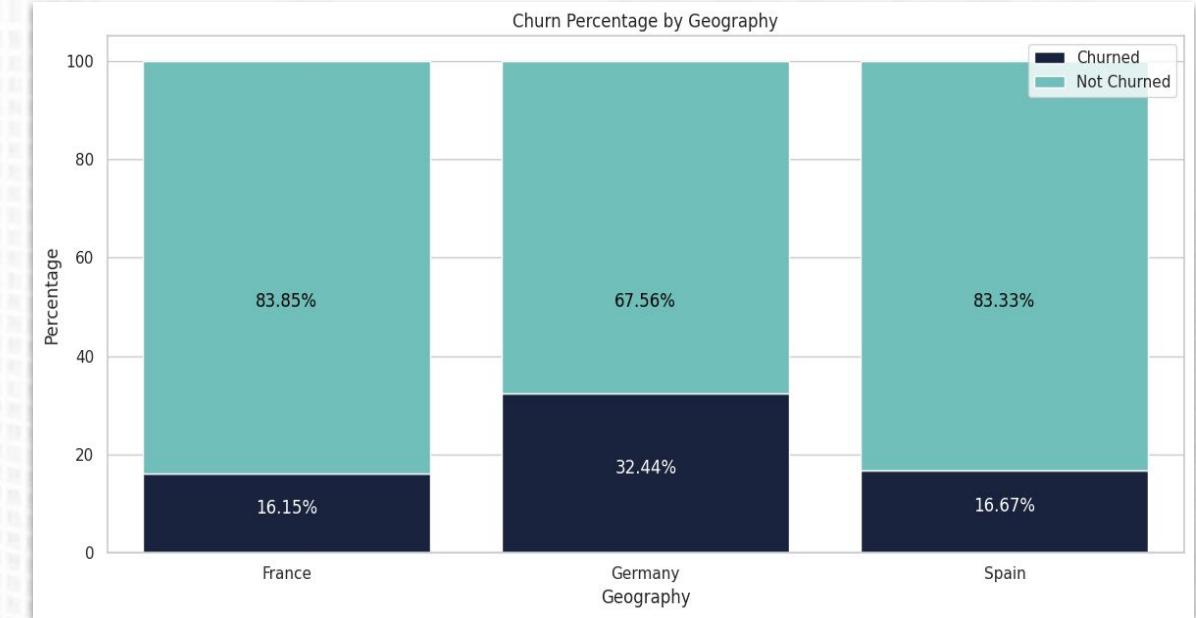


**Churn lebih tinggi** pada **customer** yang **tidak aktif**. Berdasarkan insight tersebut, tidak mengherankan jika churn yang tinggi pada customer yang tidak aktif. Namun, cukup mengkhawatirkan mengingat proporsi keseluruhan nasabah yang tidak aktif cukup tinggi.

# Business Insight



**Churn lebih tinggi** pada customer yang hanya membeli satu produk.



**Customer Germany** memiliki **tingkat churn yang lebih tinggi** dibanding france dan spain.

# Insight Recommendation

- **Tingkat Churn Bank CBA:** Bisnis perlu meningkatkan **strategi retensi nasabah** seperti contohnya peningkatan dalam layanan nasabah, memberikan insentif kepada nasabah yang mungkin berisiko keluar, atau meninjau ulang kebijakan atau produk yang mungkin menjadi faktor pengambilan keputusan nasabah untuk meninggalkan perusahaan.
- **Usia Nasabah :** Bank perlu meninjau dan **menyesuaikan strategi retensi mereka di antara kelompok usia yang berbeda**, terutama kelompok usia 40-60 tahun. Seperti memberikan penawaran yang dapat membuat customer dalam kelompok usia ini lebih puas atau terikat dengan layanan bank.
- **CreditScore Nasabah:** Mengembangkan penawaran khusus atau paket layanan yang ditujukan untuk memenuhi kebutuhan dan ekspektasi nasabah dalam rentang CreditScore tersebut.
- **Member Aktif:** Bank perlu membuat suatu program khusus yang dapat **mendorong customer ini menjadi lebih aktif**, seperti memberikan penawaran khusus atau meningkatkan komunikasi dengan customer.
- **Num Of Product:** Memberikan penawaran seperti bundling produk untuk **mendorong customer menggunakan atau membeli lebih dari satu produk** atau memberikan informasi tentang produk lainnya yang tersedia.

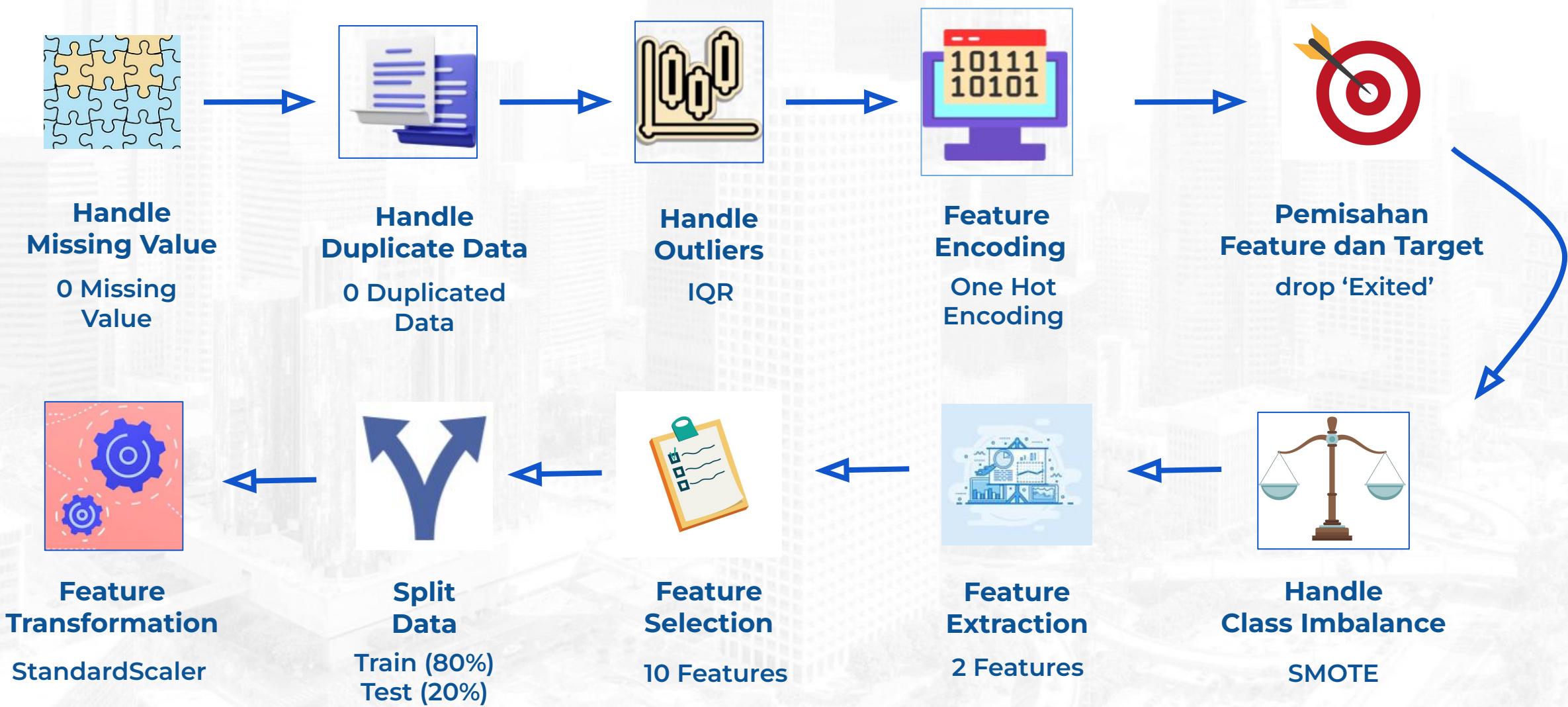
## Stage 2

---

*Data Pre-processing*



# DATA PRE-PROCESSING



# Handle Missing Values & Duplicated

## A. Missing Value

```
df.isna().sum()
```

```
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
```

## B. Duplicated Data

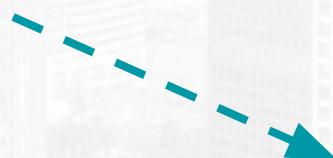
```
df.duplicated().sum()
```

```
0
```

Tidak ditemukan adanya missing value dan duplicated data, sehingga **tidak melakukan handling missing value dan duplicated data**

# Handle Outliers

	Column Name	is Outlier	Lower Limit	Upper Limit	Outlier	No Outlier
0	CreditScore	True	383.00000	919.00000	15	9985
1	Age	True	14.00000	62.00000	359	9641
2	Tenure	False	-3.00000	13.00000	0	10000
3	Balance	False	-191466.36000	319110.60000	0	10000
4	EstimatedSalary	False	-96577.09625	296967.45375	0	10000
5	NumOfProducts	True	-0.50000	3.50000	60	9940



Dilakukan handling outlier menggunakan metode **Interquartile Range (IQR)**

IQR dipilih karena **tidak dipengaruhi oleh nilai ekstrem** dan tidak memerlukan asumsi distribusi data tertentu.

Ditemukan outlier pada kolom **Creditscore, Age, dan NumOfProducts**.

Namun, pada kolom **NumOfProducts** tidak dihandle karena nilainya benar=benar ada (special case)

```
# Kolom yang akan di-handle outlier
columns_to_handle_outlier = ['CreditScore', 'Age']

# Iterasi melalui setiap kolom untuk menangani outlier
for column in columns_to_handle_outlier:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1

    # Menghitung batas bawah dan atas outlier
    lower_limit = Q1 - (1.5 * IQR)
    upper_limit = Q3 + (1.5 * IQR)

    # Menghapus outlier dari DataFrame
    df = df[(df[column] >= lower_limit) & (df[column] <= upper_limit)]

print('Jumlah baris setelah handle outlier :', df.shape[0])
Jumlah baris setelah handle outlier : 9626
```

# Feature Encoding

```
[ ] # OHE pada feature Geography dan Gender
df = pd.get_dummies(df, columns=['Geography', 'Gender'], prefix=['Geo', 'Is'])
df.head()
```

**One Hot Encoding (OHE)** pada kolom kategorik **Geography** dan **Gender**.

- One Hot Encoding dilakukan pada Kolom **Gender** karena merupakan data kategorikal **bukan ordinal**, dalam hal ini tidak ada nilai yang lebih tinggi atau lebih rendah antara female dan male
- Kolom **Geography** di encode karena merupakan data kategorikal yang **memiliki lebih dari dua kategori**, yaitu France, Germany, dan Spain.

Geo_France	Geo_Germany	Geo_Spain	Is_Female	Is_Male
1	0	0	1	0
0	0	1	1	0
1	0	0	1	0
1	0	0	1	0
0	0	1	1	0

One hot encoding mengubah setiap kategori **menjadi** kolom yang **bernilai 0 atau 1**. Kolom baru yang bernilai 1 menunjukkan bahwa data tersebut termasuk ke dalam kategori tersebut.

# Pemisahan Feature dan Target

Pemisahan features dan target (pada kasus ini, **Exited**) dilakukan untuk mengakomodasi peran unik keduanya, memastikan model dapat belajar secara efektif, dan menghasilkan prediksi yang akurat

```
df = df.drop(columns = ['RowNumber', 'CustomerId', 'Surname'])

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

X.columns

Index(['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Geo_France', 'Geo_Germany',
       'Geo_Spain', 'Is_Female', 'Is_Male'],
      dtype='object')
```

**X menyimpan feature-feature** yang digunakan untuk memprediksi hasil, **y merupakan target** atau yang menyimpan informasi labelnya



# Handle Class Imbalance

```
smote = SMOTE(random_state=0)

# Resample the dataset
X_resampled, y_resampled = smote.fit_resample(X, y)

# Print the class distribution before and after resampling
print("Original Class Distribution:")
print(y.value_counts())

print("\nClass Distribution After Resampling:")
print(pd.Series(y_resampled).value_counts())
```

```
Original Class Distribution:
0    7677
1    1949
Name: Exited, dtype: int64

Class Distribution After Resampling:
1    7677
0    7677
Name: Exited, dtype: int64
```

Melakukan handling class imbalance menggunakan metode **SMOTE**

**SMOTE** adalah sebuah teknik yang digunakan untuk mengatasi masalah ketidakseimbangan. Tujuan penggunaan SMOTE:

- Meningkatkan Kinerja Model
- Mengurangi Overfitting
- Mengurangi Resiko Bias
- Meningkatkan Akurasi Prediksi

Handling imbalance pada data target **Exited** (label 1) yang sebelumnya **1949 menjadi 7677**

# Feature Extraction

**Feature Extraction** dilakukan dengan **tujuan** melakukan **eksplorasi** hubungan antar kolom baru dengan churn.

## 1. Kolom **Salary\_Category**

Membagi **EstimatedSalary** menjadi 3 kategori low, mid, high berdasarkan nilai persentile

```
percentile_25e = df['EstimatedSalary'].quantile(0.25)
percentile_75e = df['EstimatedSalary'].quantile(0.75)

df['Salary_Category'] = df['EstimatedSalary'].apply(lambda x: 'low' if x <= percentile_25e else ('mid' if percentile_25e < x <= percentile_75e else 'high'))
```

## 2. Kolom **Tenure\_Category**

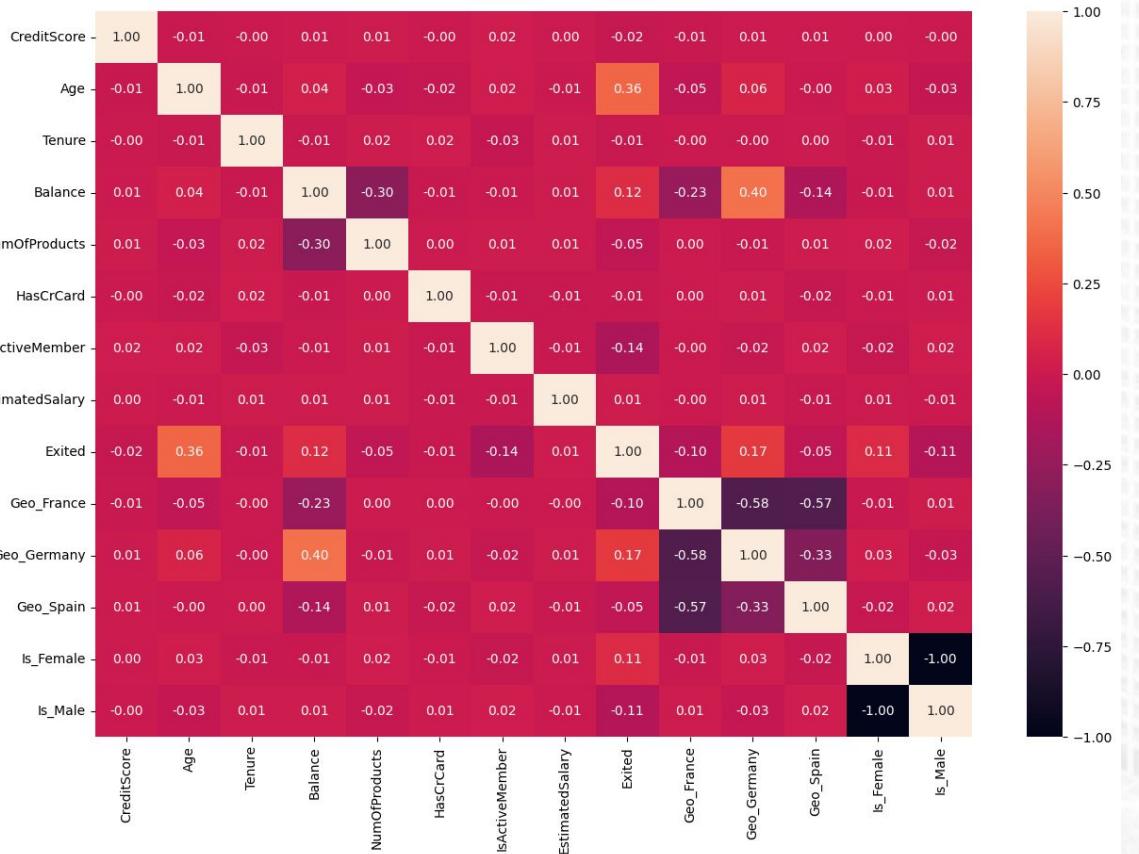
Membagi **Tenure** menjadi 3 kategori low, mid, high berdasarkan nilai persentile

```
percentile_25t = df['Tenure'].quantile(0.25)
percentile_75t = df['Tenure'].quantile(0.75)

df['Tenure_Category'] = df['Tenure'].apply(lambda x: 'low' if x <= percentile_25t else ('mid' if percentile_25t < x <= percentile_75t else 'high'))
```

# Feature Selection

Pemilihan feature dilakukan dengan hasil **pengamatan EDA, analisis korelasi** dan Uji **Chi Square**



Pertimbangan berdasarkan EDA yang dikuatkan dengan analisis korelasi yang telah dilakukan :

Kolom yang tidak dihilangkan :

- **Age**
- **IsActiveMember**
- **Balance**
- **NumOfProducts**
- **Geography**
- **CreditScore**

# Feature Selection

**Uji chi-square** pada kolom-kolom berikut

```
categorical = ['Is_Female','Is_Male', 'Geo_Germany', 'Geo_Spain', 'Geo_France', 'IsActiveMember', 'HasCrCard','Salary_Category','Tenure_Category']
```

```
from scipy.stats import chi2_contingency

chi2_array, p_array = [], []
for column in categorical:

    crosstab = pd.crosstab(df[column], df['Exited'])
    chi2, p, dof, expected = chi2_contingency(crosstab)
    chi2_array.append(chi2)
    p_array.append(p)

df_chi = pd.DataFrame({
    'Variable': categorical,
    'Chi-square': chi2_array,
    'p-value': p_array
})
df_chi.sort_values(by='Chi-square', ascending=False)
```

	Variable	Chi-square	p-value
2	Geo_Germany	288.032718	1.334179e-64
5	IsActiveMember	197.296810	8.123591e-45
0	Is_Female	109.185095	1.478175e-25
1	Is_Male	109.185095	1.478175e-25
4	Geo_France	102.557298	4.190698e-24
3	Geo_Spain	28.120386	1.139987e-07
7	Salary_Category	3.095103	2.127682e-01
8	Tenure_Category	2.430056	2.967016e-01
6	HasCrCard	0.648381	4.206922e-01

**Uji chi - square** digunakan untuk menentukan independensi pada setiap variabel terhadap variabel target atau variabel kategorikal lainnya

Terlihat bahwa **Salary\_Category**, **Tenure\_Category**, dan **HasCrCard** tidak ada hubungan yang signifikan antara feature dan variabel target (gagal menolak H0)

# Feature Selection

Berdasarkan dari EDA dan uji chi-square yang telah dilakukan, berikut adalah feature yang **tidak akan digunakan** dalam machine learning :

- **RowNumber**, hal ini karena kolom tersebut merupakan **index kolom** dan tidak akan memberikan informasi yang berguna pada machine learning.
- **CustomerID**, kolom ini berisikan **identifier unik** dari masing-masing nasabah dan biasanya hanya digunakan untuk referensi atau identifikasi. Oleh karena itu, kolom ini akan dibuang dan dibuktikan hasil EDA korelasi kolom ini dengan target yaitu 0.006248.
- **Surname**, sama halnya dengan CustomerID kolom ini lebih baik tidak digunakan karena **hanya sebagai informasi customer** dan tidak memiliki info prediktif hanya identifikasi unik.
- **EstimatedSalary** dan **Tenure**, berdasarkan EDA sebelumnya terlihat bahwa kolom ini kurang terlihat dalam memberikan insight apakah customer churn atau tidak. Oleh karena itu dilakukan uji chi-square dimana kolom keduanya akan dibagi menjadi 3 kelompok (low(<25%), mid(>25% dan <75%), dan high (>75%)) Hasil uji didapatkan bahwa **korelasi keduanya tidak signifikan**.
- **HasCrCard**, berdasarkan uji chi-square yang telah dilakukan terlihat bahwa **tidak terdapat hubungan yang signifikan** terhadap variabel target.

# Feature Selection

Feature yang **digunakan** adalah

- **CreditScore**
- **Age**
- **Balance**
- **NumOfProducts**
- **IsActiveMember**
- **Geo\_France**
- **Geo\_Germany**
- **Geo\_Spain**
- **Is\_Male**
- **Is\_Female**

```
# Melakukan drop kolom yang sudah tidak digunakan setelah feature selection
X_resampled = X_resampled.drop(['HasCrCard', 'EstimatedSalary', 'Tenure'], axis=1)

X_resampled.columns

Index(['CreditScore', 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember',
       'Geo_France', 'Geo_Germany', 'Geo_Spain', 'Is_Female', 'Is_Male'],
      dtype='object')
```

Berdasarkan **EDA, analisis korelasi, dan test Chi Square** yang telah dilakukan **kolom-kolom** ini terlihat bahwa masing-masing nilai **mempengaruhi customer churn**, untuk itu akan dimasukkan sebagai fitur dalam model machine learning.

# Split Data

Untuk **menghindari data leakage (kebocoran data)**, proses penyiapan data (**fit**) hanya menggunakan dataset pelatihan. Koefisien atau model yang disiapkan harus didasarkan pada informasi dari dataset pelatihan saja, agar evaluasi model bersifat objektif pada data yang belum pernah dilihat sebelumnya.

**Split Training Set dan Testing Set**    - - - - - ➤ **80 : 20**

```
x_train, x_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size = 0.2, random_state = 0)
```

# Feature Transformation

```
scaller = StandardScaler()

x_train = scaller.fit_transform(x_train)
x_test = scaller.transform(x_test)

print("Mean setelah standardisasi :")
print(x_train.mean())

print("\nDeviasi Standar setelah standardisasi :")
print(x_train.std())

Mean setelah standardisasi :
-4.0088424316677475e-17

Deviasi Standar setelah standardisasi :
1.0
```

Melakukan **scaling** menggunakan **StandardScaler**

Sehingga data yang terstandarisasi memiliki **rata-rata nol dan deviasi standar satu**. Scaling dilakukan untuk memastikan bahwa setiap fitur memiliki dampak yang setara pada model.

## Stage 3

---

*Machine Learning Modelling  
dan Evaluation*



# Model Algoritma

Berdasarkan referensi yang disediakan oleh **Scikit-Learn** dan sumber-sumber lainnya, kami akan membandingkan **enam jenis model klasifikasi** yang berbeda. Keputusan untuk mengikutsertakan model-model ini dalam uji coba kami didasarkan pada pertimbangan berikut:

1. **Linear SVC** dan **K-Nearest Neighbors Classifier** cocok digunakan untuk dataset dengan ukuran sampel kurang dari 100 ribu. Kedua model ini efisien untuk dataset yang relatif kecil.
2. **Decision Tree** dipilih ketika kecepatan pemrosesan diperlukan dan explainable.
3. **Logistic Regression** dipilih karena kemampuannya untuk memberikan interpretasi yang jelas dan cocok untuk model klasifikasi dengan **variabel dependen biner** (dua kategori, dalam hal ini churn atau tidak churn)
4. **Random Forest** dan **Gradient Boosting Classifier** digunakan karena kedua model ini dikenal memiliki tingkat akurasi yang tinggi dibandingkan dengan beberapa model lainnya.

Terlepas dari alasan-alasan di atas, kami juga ingin menjelajahi model-model lain yang tidak dijelaskan oleh Rakamin Academy untuk **memahami perbedaan dan dampak dari masing-masing model** ini.

# Evaluasi Model

Parameter evaluasi model yang dipilih untuk menentukan model terbaik berfokus pada :

- **Recall** : primary metrics evaluation
  - Mereduksi false negative (nasabah yang diprediksi tidak churn, namun actualnya churn)
  - Meningkatkan revenue dan profit karena tingkat churn dapat diturunkan
- **Precision** : secondary metrics evaluation
  - Mereduksi false positive (nasabah yang diprediksi churn, namun actualnya tidak churn)
  - Mengurangi cost strategi pencegahan customer churn
- **Recall** : combined metrics evaluation
  - Digunakan untuk mengukur keseimbangan antara recall dan precision.

# Fungsi Untuk Metrics Evaluation

```
▶ # fungsi untuk metrics evalution

def eval_classification(model):
    y_pred = model.predict(x_test)
    y_pred_train = model.predict(x_train)

    print("Accuracy (Test Set): %.2f" % accuracy_score(y_test, y_pred))
    print("Accuracy (Train set): %.2f" % accuracy_score(y_train, y_pred_train))
    print("\nPrecision (Test Set): %.2f" % precision_score(y_test, y_pred))
    print("Precision (Train Set): %.2f" % precision_score(y_train, y_pred_train))
    print("\nRecall (Test Set): %.2f" % recall_score(y_test, y_pred))
    print("Recall (Train Set): %.2f" % recall_score(y_train, y_pred_train))
    print("\nF1-Score (Test Set): %.2f" % f1_score(y_test, y_pred))
    print("F1-Score (Train Set): %.2f" % f1_score(y_train, y_pred_train))

def roc_auc_eval(model):
    y_pred_proba = model.predict_proba(x_test)
    y_pred_proba_train = model.predict_proba(x_train)

    print("\nroc_auc (test-proba): %.2f" % roc_auc_score(y_test, y_pred_proba[:, 1]))
    print("roc_auc (train-proba): %.2f" % roc_auc_score(y_train, y_pred_proba_train[:, 1]))
```

Membuat fungsi perhitungan metrics **untuk mempermudah melihat hasil performa model** percobaan.

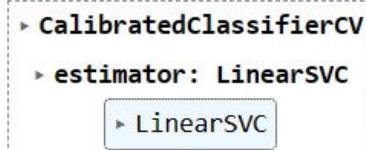
# Modeling

## 1. LinearSVC

```

▶ svc = LinearSVC()
svc.fit(x_train, y_train)
calibrated_svc = CalibratedClassifierCV(svc, method='sigmoid', cv='prefit')
calibrated_svc.fit(x_train, y_train)

```



```

eval_classification(svc)

svc_probs = calibrated_svc.predict_proba(x_test)[:, 1]
svc_probs_train = calibrated_svc.predict_proba(x_train)[:, 1]
print("\nroc_auc (test-proba): %.2f" % roc_auc_score(y_test, svc_probs))
print("roc_auc (train-proba): %.2f" % roc_auc_score(y_train, svc_probs_train))

```

Accuracy (Test Set): 0.84  
 Accuracy (Train set): 0.84

Precision (Test Set): 0.89  
 Precision (Train Set): 0.89

Recall (Test Set): 0.78  
 Recall (Train Set): 0.78

F1-Score (Test Set): 0.83  
 F1-Score (Train Set): 0.83

roc\_auc (test-proba): 0.92  
 roc\_auc (train-proba): 0.92

### Observasi :

Didapatkan hasil **recall** yang **lebih rendah** dibandingkan yang lain baik untuk data train dan test.  
**Tidak sesuai** dengan metric model yang diinginkan.

# Modeling

## 2. K-Nearest Neighbors (KNN)

```
▶ knn = KNeighborsClassifier()  
      knn.fit(x_train, y_train)  
  
👤 ▾ KNeighborsClassifier  
      KNeighborsClassifier()
```

```
▶ eval_classification(knn)  
      roc_auc_eval(knn)  
  
👤 Accuracy (Test Set): 0.86  
      Accuracy (Train set): 0.91  
  
Precision (Test Set): 0.87  
      Precision (Train Set): 0.92  
  
Recall (Test Set): 0.86  
      Recall (Train Set): 0.89  
  
F1-Score (Test Set): 0.87  
      F1-Score (Train Set): 0.90  
  
roc_auc (test-proba): 0.93  
      roc_auc (train-proba): 0.97
```

### Observasi :

- Terdapat **jarak** pada masing-masing **metriks** yang **cukup besar** (sekitar 0.4 - 0.5) antara data train dan data test, dimana **metriks data train lebih tinggi** dibandingkan dengan test.
- Hal ini mengindikasikan terdapat kecenderungan model **Overfit**.

# Modeling

## 3. Logistic Regression

```
▶ logit = LogisticRegression()  
logit.fit(x_train, y_train)  
  
▼ LogisticRegression  
LogisticRegression()
```

```
▶ eval_classification(logit)  
roc_auc_eval(logit)  
  
👤 Accuracy (Test Set): 0.84  
Accuracy (Train set): 0.84  
  
Precision (Test Set): 0.88  
Precision (Train Set): 0.89  
  
Recall (Test Set): 0.79  
Recall (Train Set): 0.78  
  
F1-Score (Test Set): 0.83  
F1-Score (Train Set): 0.83  
  
roc_auc (test-proba): 0.92  
roc_auc (train-proba): 0.92
```

### Observasi :

- Terdapat **jarak** pada masing-masing **metriks** yang **cukup besar** (sekitar 0.4 - 0.5) antara data train dan data test, dimana **metriks data train lebih tinggi** dibandingkan dengan test.
- Hal ini mengindikasikan terdapat kecenderungan model **Overfit**.

# Modeling

## 4. Decision Tree

```
▶ dt = DecisionTreeClassifier()  
    dt.fit(x_train, y_train)
```

```
▼ DecisionTreeClassifier  
  DecisionTreeClassifier()
```

```
eval_classification(dt)  
roc_auc_eval(dt)  
  
Accuracy (Test Set): 0.83  
Accuracy (Train set): 1.00  
  
Precision (Test Set): 0.82  
Precision (Train Set): 1.00  
  
Recall (Test Set): 0.86  
Recall (Train Set): 1.00  
  
F1-Score (Test Set): 0.84  
F1-Score (Train Set): 1.00  
  
roc_auc (test-proba): 0.83  
roc_auc (train-proba): 1.00
```

### Observasi :

- Secara keseluruhan model memiliki **kinerja yang baik** jika dilihat dari **metriks di data test**, namun terjadi **Overfit pada data train**

# Modeling

## 5. Random Forest

```
rf = RandomForestClassifier()  
rf.fit(x_train, y_train)
```

```
RandomForestClassifier  
RandomForestClassifier()
```

```
eval_classification(rf)  
roc_auc_eval(rf)  
  
Accuracy (Test Set): 0.88  
Accuracy (Train set): 1.00  
  
Precision (Test Set): 0.89  
Precision (Train Set): 1.00  
  
Recall (Test Set): 0.88  
Recall (Train Set): 1.00  
  
F1-Score (Test Set): 0.88  
F1-Score (Train Set): 1.00  
  
roc_auc (test-proba): 0.95  
roc_auc (train-proba): 1.00
```

### Observasi :

- Secara keseluruhan model memiliki **kinerja yang baik** jika dilihat dari **metriks di data test** dan **lebih baik dari Decision Tree**, namun terjadi **overfit pada data train**.

# Modeling



## 6. Gradien Boosting Classifier (GBC)

```
gbc = GradientBoostingClassifier()  
gbc.fit(x_train, y_train)  
  
GradientBoostingClassifier  
GradientBoostingClassifier()
```

```
eval_classification(gbc)  
roc_auc_eval(gbc)  
  
Accuracy (Test Set): 0.88  
Accuracy (Train set): 0.88  
  
Precision (Test Set): 0.90  
Precision (Train Set): 0.91  
  
Recall (Test Set): 0.86  
Recall (Train Set): 0.85  
  
F1-Score (Test Set): 0.88  
F1-Score (Train Set): 0.88  
  
roc_auc (test-proba): 0.95  
roc_auc (train-proba): 0.95
```

### Observasi :

- Secara keseluruhan model memiliki **kinerja yang baik** jika dilihat dari **metriks di data test** maupun **data train**. Diperlukan pertimbangan GBC sebagai salah satu pilihan model untuk digunakan.

# Model Evaluation

## (Pemilihan dan perhitungan metriks model)

### 1. Membuat function evaluasi setiap model

```
▶ # Create a dictionary to store results
results = {'Model': [], 'Accuracy': [], 'Precision': [], 'Recall': [], 'F1 Score': []}

# Function to evaluate a model and add results to the dictionary
def evaluate_model(model, model_name, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    results['Model'].append(model_name)
    results['Accuracy'].append(accuracy)
    results['Precision'].append(precision)
    results['Recall'].append(recall)
    results['F1 Score'].append(f1)

# Print confusion matrix
print(f"Confusion Matrix for {model_name}:\n")
print(confusion_matrix(y_test, y_pred))
print("\n" + "="*40 + "\n")
```



# Model Evaluation

## (Pemilihan dan perhitungan metriks model)

### 2. Define masing-masing model untuk dijalankan

```
# Evaluate each model
models = {
    'LinearSVC': svc,
    'KNN': knn,
    'Logistic Regression': logit,
    'Decision Tree': dt,
    'Random Forest': rf,
    'Gradient Boosting': gbc
}

for model_name, model in models.items():
    evaluate_model(model, model_name, x_train, x_test, y_train, y_test)
```

### 3. Visualisasi untuk evaluasi setiap model

```
# Convert results to a DataFrame for visualization
results_df = pd.DataFrame(results)

# Sort the DataFrame by Recall in descending order
results_df = results_df.sort_values(by='Recall', ascending=False)

# Create a heatmap with x-axis labels above the chart
fig, ax = plt.subplots(figsize=(10, 6))
sns.heatmap(results_df.set_index('Model'),
            annot=True,
            cmap="YlGnBu",
            fmt=".3f",
            linewidths=.5,
            ax=ax,
            cbar_kws={'label': 'Metrics'})
ax.xaxis.tick_top() # Move x-axis ticks to the top
plt.title('Model Performance Metrics\n')
plt.show()
```



# Model Evaluation

## (Pemilihan dan perhitungan metriks model)



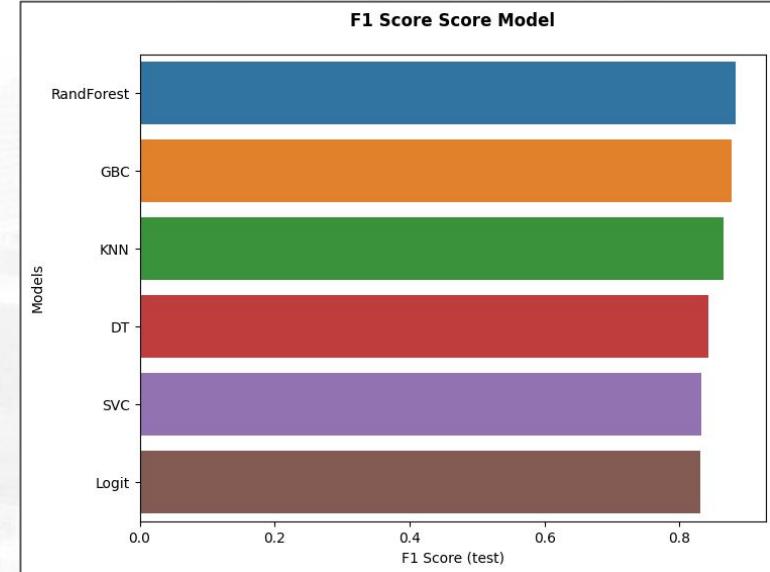
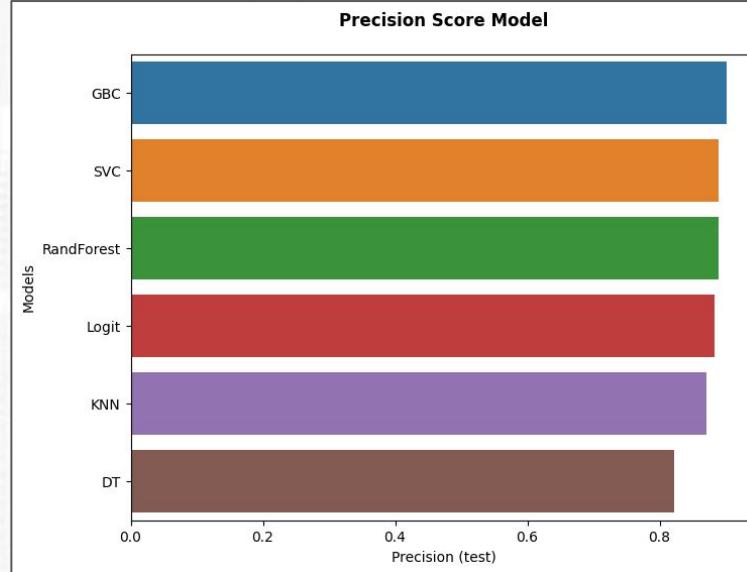
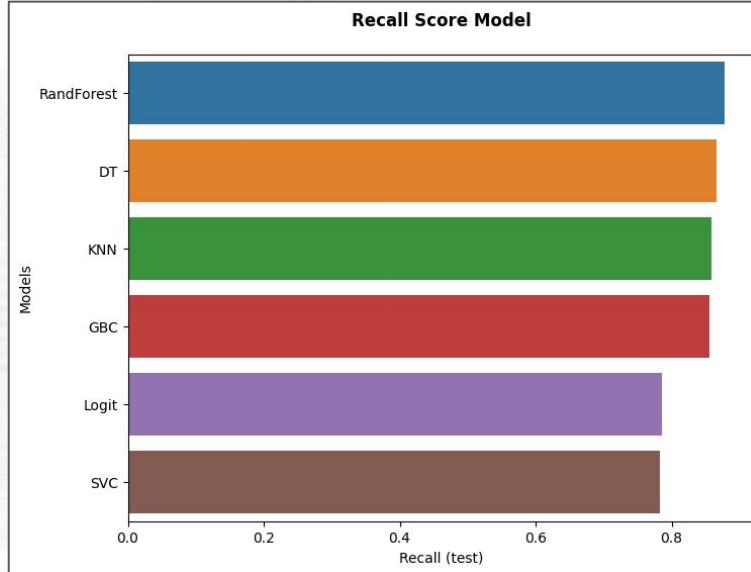
Nilai **Recall, Precision, dan F1 Score** mendapatkan evaluasi yang baik pada :

1. **Random Forest,**
2. **Decision Tree**
3. **KNN**

**Recall tertinggi** (sesuai dengan **primary metrics**) didapatkan pada model **Random Forest**.

# Model Evaluation

## (Visualisasi metrics model)



Dari visualisasi tersebut terlihat bahwa model **RandomForest** memiliki nilai **Recall** dan **F1 Score tertinggi** diantara model-model lainnya. Walau **precision tertinggi pada model GBC** (Gradient Boosting Classifier), namun nilai precision pada RandomForest juga masih yang tergolong tinggi. Sehingga RandomForest sesuai dengan metrics evaluation yang dipilih.

# Model Evaluation

## (Menentukan model best fit)

```
# Daftar model yang ingin dibandingkan
models = [knn, svc, logit, dt, rf, gbc]

# Daftar nama model untuk tampilan
model_names = ["K-Nearest Neighbors", "LinearSVC", "Logistic Regression", "Decision Tree", "Random Forest", "Gradient Boosting"]

# Inisialisasi daftar untuk menyimpan hasil evaluasi
results = []

# Fungsi untuk mengevaluasi model dan mencetak hasilnya
def evaluate_model(model, x_train, y_train, x_test, y_test):
    model.fit(x_train, y_train)
    train_pred = model.predict(x_train)
    train_recall = recall_score(y_train, train_pred)
    test_pred = model.predict(x_test)
    test_recall = recall_score(y_test, test_pred)
    return train_recall, test_recall

# Lakukan loop untuk model dan tampilkan hasilnya
for i, model in enumerate(models):
    train_recall, test_recall = evaluate_model(model, x_train, y_train, x_test, y_test)
    if train_recall > test_recall:
        result = f"{model_names[i]}: Model overfitting"
    elif train_recall < test_recall:
        result = f"{model_names[i]}: Model underfitting"
    else:
        result = f"{model_names[i]}: Model best-fit"
    results.append((model_names[i], result, train_recall, test_recall))

# Tampilkan hasil evaluasi
for model_name, result, train_recall, test_recall in results:
    print(f"{result} (Train recall: {train_recall:.2f}, Test recall: {test_recall:.2f})")
```

K-Nearest Neighbors: Model overfitting (Train recall: 0.89, Test recall: 0.86)  
LinearSVC: Model underfitting (Train recall: 0.78, Test recall: 0.78)  
Logistic Regression: Model underfitting (Train recall: 0.78, Test recall: 0.79)  
Decision Tree: Model overfitting (Train recall: 1.00, Test recall: 0.86)  
Random Forest: Model overfitting (Train recall: 1.00, Test recall: 0.87)  
Gradient Boosting: Model underfitting (Train recall: 0.85, Test recall: 0.86)

### Observasi:

- K-Nearest Neighbors (KNN):** Model ini cenderung overfitting. Hal ini terlihat dari tingkat recall pada data pelatihan yang tinggi (89%) dibandingkan dengan tingkat recall pada data uji (86%).
- LinearSVC dan Logistic Regression:** Model ini menunjukkan underfitting. Kedua tingkat recall pada data pelatihan dan data uji relatif rendah. Model ini mungkin terlalu sederhana untuk dataset yang digunakan.
- Decision Tree:** Model ini adalah contoh overfitting yang tinggi. Tingkat recall pada data pelatihan mencapai 100%, sementara pada data uji hanya sekitar 86%.
- Random Forest:** Model ini juga overfitting, tetapi tidak separah Decision Tree. Tingkat recall pada data pelatihan mencapai 100%, dan pada data uji sekitar 87%. Ini menunjukkan bahwa model ini lebih baik daripada Decision Tree, tetapi masih mengalami masalah overfitting.
- Gradient Boosting:** Model ini juga mengalami overfitting, meskipun tingkat recall pada data uji (85%) lebih baik daripada sebagian besar model lainnya. Namun, tingkat recall pada data pelatihan masih kurang.

Pemilihan **Random Forest** sebagai model pilihan adalah karena meskipun model ini juga overfitting, tingkat recall pada data uji adalah yang tertinggi di antara model-model yang diuji (sekitar 87%). Meskipun overfitting, model ini memberikan hasil yang relatif baik dalam memprediksi data baru dibandingkan dengan model lainnya. Namun, perlu dilakukan lebih banyak eksperimen dan fine-tuning untuk mengatasi masalah overfitting.

# Model Evaluation

## (Cross Validation)

### Cross Validation dan menentukan model best-fit (semua model)

```
# Create a list of models
models = [
    ("Logistic Regression", logit),
    ("K-Nearest Neighbors", knn),
    ("LinearSVC", svc),
    ("Decision Tree", dt),
    ("Random Forest", rf),
    ("Gradient Boosting", gbc)
]

# Perform 5-fold cross-validation for each model
for model_name, model in models:
    cv_scores = cross_val_score(model, x_train, y_train, cv=5, scoring='recall')

    # calculate the mean recall of the model using cross-validation
    mean_recall = cv_scores.mean()

    # Set a threshold to determine if the model is the best fit or not
    threshold = 0.85

    if mean_recall >= threshold:
        print(f"{model_name} is a best fit model with Mean recall: {100 * mean_recall:.2f}%")
    else:
        print(f"{model_name} is not the best fit model with Mean recall: {100 * mean_recall:.2f}%")

Logistic Regression is not the best fit model with Mean recall: 78.11%
K-Nearest Neighbors is not the best fit model with Mean recall: 83.98%
LinearSVC is not the best fit model with Mean recall: 77.72%
Decision Tree is not the best fit model with Mean recall: 83.42%
Random Forest is a best fit model with Mean recall: 85.65%
Gradient Boosting is not the best fit model with Mean recall: 84.06%
```

### Cross Validation RandomForest

```
▶ # Inisialisasi model Random Forest
rf_classifier = RandomForestClassifier(random_state=0)

# Melakukan cross-validation dan menghitung skor
score = cross_val_score(rf_classifier, x_train, y_train, cv=5, scoring='recall')

print(f'scores for each fold are: {score}')
print(f'Average score: {:.2f}'.format(score.mean()))

scores for each fold are: [0.86158886 0.84930385 0.85913186 0.85737705 0.85831286]
Average score: 0.86
```

# Model Evaluation

## (Kesimpulan Pengambilan Model)

Berdasarkan percobaan pemodelan tersebut, dapat disimpulkan bahwa model yang akan digunakan untuk memprediksi churn nasabah adalah Random Forest Classifier. Ini disarankan atas dasar beberapa alasan:

- **Tujuan Model Metriks:** Untuk memprediksi churn nasabah diperlukan tingkat recall tinggi. Model ini memberikan tingkat recall tertinggi pada data uji dibandingkan dengan model lainnya (sekitar 88%). Ini menandakan kemampuan model untuk mengidentifikasi dengan baik kelas positif pada data yang tidak terlihat.
- **Waktu Komputasi:** Dibandingkan dengan Gradien Boosting Classifier yang memiliki recall yang baik, pengaturan parameter (hyperparameter tuning) untuk Random Forest Classifier lebih cepat dan efisien. Ini dapat menghemat waktu dan sumber daya
- **Non-Asumsional:** Random Forest Classifier tidak bergantung pada asumsi bahwa data harus berdistribusi normal, yang sesuai dengan karakteristik dataset yang tidak mengikuti distribusi normal. Dengan demikian, Random Forest Classifier adalah pilihan yang baik untuk memprediksi churn nasabah dalam konteks ini.

# Hyperparameter Tuning

## GridSearchCV

```
[ ] param_grid = {  
    'n_estimators': [50, 100],  
    'max_depth': [6, 8, 10],  
    'min_samples_split': [2, 5, 10],  
    'min_samples_leaf': [1, 2, 4],  
    'max_features': ['auto', 'sqrt', 'log2']  
}  
  
rf = RandomForestClassifier(random_state=0)  
  
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, scoring='recall', cv=5)  
grid_search.fit(x_train, y_train)  
  
best_rf = grid_search.best_estimator_  
best_params = grid_search.best_params_  
best_score = grid_search.best_score_  
  
print("Best Random Forest Model:")  
print(best_rf)  
print("\nBest Parameters:")  
print(best_params)  
print("\nBest Recall Score:")  
print(best_score)  
  
  
Best Random Forest Model:  
RandomForestClassifier(max_depth=10, max_features='auto', min_samples_leaf=2,  
                      min_samples_split=10, n_estimators=50, random_state=0)  
  
Best Parameters:  
{'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 50}  
  
Best Recall Score:  
0.8535399632120944
```

Melakukan hyperparameter tuning pada model **Random Forest**.

# Hyperparameter Tuning

## Menggunakan model Random Forest terbaik

```
rf_model2 = best_rf.fit(x_train, y_train)

# Evaluasi model
eval_classification(rf_model2)

Accuracy (Test Set): 0.88
Accuracy (Train set): 0.90

Precision (Test Set): 0.90
Precision (Train Set): 0.93

Recall (Test Set): 0.87
Recall (Train Set): 0.88

F1-Score (Test Set): 0.88
F1-Score (Train Set): 0.90
```

Berdasarkan hasil tuning parameter menggunakan GridSearchCV tersebut, didapatkan bahwa :

- **Best model** atau model terbaik random forest **menggunakan parameter max\_depth** (kedalaman maksimum), **max\_feature** (jumlah fitur maksimum), **min\_sample\_leaf** (jumlah daun minimum), **min\_samples\_split** (pembagian minimum), dan **n\_estimators** (jumlah pohon keputusan) masing-masingnya adalah **10, auto, 2, 10, dan 50** untuk mendapatkan score recall terbaik.
- Model tersebut **memiliki kinerja yang baik dengan nilai recall yang tinggi**, hal tersebut menunjukkan bahwa model mampu mengidentifikasi sebagian besar positif sejati, dalam hal ini adalah masalah prediksi customer churn di bank.
- Pada model tersebut, **perbedaan antara skor data pelatihan dan data uji tidaklah terlalu besar**, sehingga mengindikasikan bahwa model tersebut tidak terlalu disesuaikan dengan data pelatihan atau dengan kata lain memiliki kemampuan untuk memprediksi data yang baru.

# Feature Importance (Random Forest)

```
feature_importances = rf_model2.feature_importances_

# Pair feature names with their importances
feature_names = X_resampled.columns.tolist()
feature_importance_dict = dict(zip(feature_names, feature_importances))

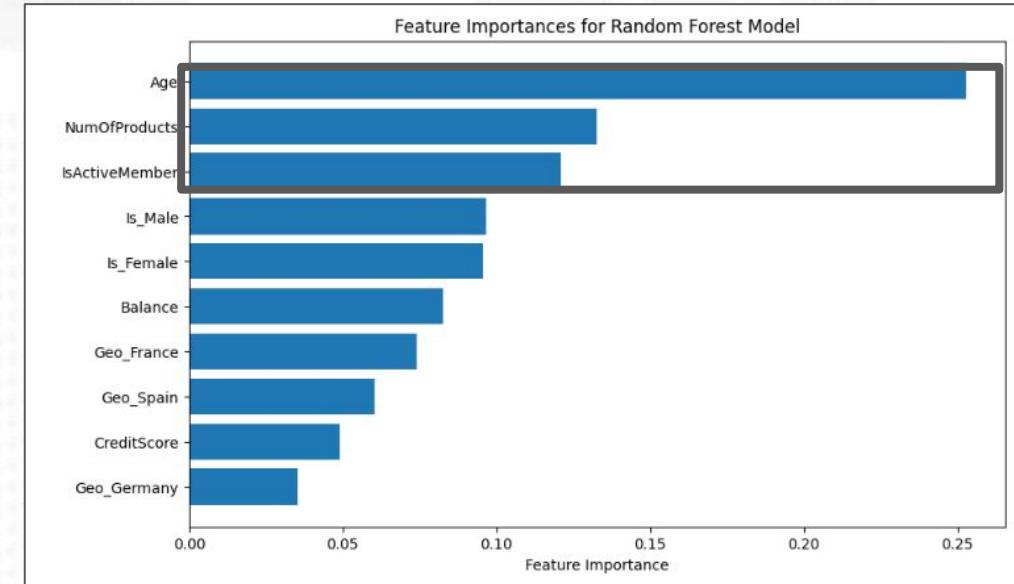
# Sort feature importances in descending order
sorted_feature_importance = sorted(feature_importance_dict.items(), key=lambda x: x[1], reverse=True)
```

```
# Print the feature importances
for feature, importance in sorted_feature_importance:
    print(f'{feature}: {importance:.4f}')
```

```
Age: 0.2529
NumOfProducts: 0.1326
IsActiveMember: 0.1211
Is_Male: 0.0967
Is_Female: 0.0955
Balance: 0.0827
Geo_France: 0.0740
Geo_Spain: 0.0601
CreditScore: 0.0491
Geo_Germany: 0.0352
```

# Feature Importance (Random Forest)

```
# Extract feature names and their importances  
feature_names = [feature for feature, importance in sorted_feature_importance]  
feature_importances = [importance for feature, importance in sorted_feature_importance]  
  
# Reverse the lists to have the highest importance at the top  
feature_names = feature_names[::-1]  
feature_importances = feature_importances[::-1]  
  
# Create a bar plot to visualize feature importances  
plt.figure(figsize=(10, 6))  
plt.barh(feature_names, feature_importances)  
plt.xlabel('Feature Importance')  
plt.title('Feature Importances for Random Forest Model')  
plt.show()
```



Feature Importance **tiga** tertinggi yaitu **Age**, **NumOfProducts**, dan **IsActiveMember** dengan masing-masing nilai secara berurutan adalah 0.2529, 0.1326, dan 0.1211.

**Tiga fitur** ini yang paling teratas dalam **mempengaruhi** hasil apakah customer akan **churn atau tidak**, dan akan dijadikan sebagai acuan business recommendation.

# Feature Importance

## Evaluasi Feature Importance Random Forest

### Top 3 Feature Importance

Berdasarkan Feature Importance Random Forest, didapatkan bahwa :

1. Feature paling penting adalah **Age** yakni sekitar **0.2529**. Hal ini menunjukkan bahwa **usia customer memiliki pengaruh besar** terhadap keputusan mereka untuk tetap atau keluar.
2. **NumOfProducts** memiliki feature importance sekitar **0.1326**. Hal ini menunjukkan bahwa customer yang menggunakan **lebih banyak produk** mungkin lebih cenderung untuk tetap menjadi nasabah.
3. **IsActiveMember** memiliki feature importance **0.1211**. Hal ini menunjukkan bahwa bagi member atau **customer yang aktif memiliki pengaruh yang besar** bagi bank.

# Business Recommendation & Simulation

---



# Business Recommendation



## Pertahankan Fokus pada Segment Usia

ditujukan untuk seluruh **customer** dari berbagai usia

Sesuaikan strategi retensi berdasarkan segmentasi **usia** atau **life stages financial**. **early life** (15-30), **mid life** (31-50), **pra pensiun** (51-60), **pensiun** (60+). Menawarkan produk atau program yang sesuai kebutuhan usia, seperti menawarkan program pensiun untuk customer yang lebih tua.



## Optimalkan Penawaran Produk

Diutamakan pada **customer** yang hanya menggunakan satu jenis produk

Memberikan **diskon** atau **cashback** untuk customer yang melakukan transaksi beberapa produk bank sekaligus.

**Goals:** Mengoptimalkan penawaran produk untuk meningkatkan keterlibatan customer



## Stimulasi Keanggotaan Aktif

Diutamakan untuk **customer** yang kurang aktif/tidak aktif

Membuat **level keanggotaan** (**Silver**, **Gold**, **Platinum**) dengan intensif tambahan, akses eksklusif, atau layanan prioritas.

**Goals:** Agar customer menjadi lebih aktif dalam menggunakan layanan bank

# Business Recommendation

## Early Life (Usia 15-30 tahun)

Individu pada usia ini mungkin memiliki mobilitas yang tinggi dalam berkarir seperti perpindahan pekerjaan atau pergeseran prioritas keuangan. Dan mungkin belum memiliki banyak modal finansial karena baru saja memulai meniti karir.

**Implikasi perbankan:** Bank dapat fokus **menyediakan layanan yang fleksibel** dan **nyaman** untuk memenuhi gaya hidup dinamis individu pada tahap ini. **menarik pelanggan dengan produk yang kompetitif** dan **beradaptasi dengan perkembangan kebutuhan** dapat membantu **mengurangi churn**

## Pra-Pensiun (Usia 50-60 tahun)

Saat mendekati masa pensiun, individu mungkin akan mempersiapkan rencana untuk menghadapi masa tersebut. Individu mungkin akan mencari lembaga keuangan yang ahli pada perencanaan pensiun dan pengelolaan kekayaan.

**Implikasi perbankan:** Bank dapat **menyediakan layanan** yang berfokus pada **perencanaan pensiun, saran investasi, dan perencanaan keuangan** yang disesuaikan. Dengan memenuhi kebutuhan pelanggan pada usia ini **diharapkan** dapat **mempertahankan pelanggan** dalam penggunaan layanan bank

1

## SEGMENTASI USIA

(Life Stage Financial)

2

## Mid Life (Usia 30-50 tahun)

Di masa ini individu mulai memantapkan landasan keuangan dengan langkah-langkah strategis membeli rumah atau aset lainnya dan mendukung kesejahteraan keluarga.

**Implikasi perbankan:** Bank dapat **menawarkan solusi keuangan** untuk mengurangi churn seperti dengan **menawarkan layanan hipotek, tabungan pendidikan, dan opsi investasi**. Dengan solusi ini **diharapkan pelanggan** merasa **mendapatkan dukungan personal** dalam setiap peristiwa penting dalam hidup sehingga dapat **meningkatkan loyalitas** pelanggan terhadap bank.

3

4

## Pensiun (Usia 60 tahun keatas)

Saat pensiun, prioritas keuangan beralih dari akumulasi kekayaan menjadi distribusi pendapatan dan perencanaan warisan. Individu mungkin menilai ulang hubungan perbankan mereka untuk memastikan lembaga keuangan mereka sejalan dengan kebutuhan mereka saat pensiun

**Implikasi perbankan:** Bank dapat **menawarkan layanan** yang sesuai untuk **pensiunan**, seperti rekening yang disesuaikan, bantuan perencanaan warisan, produk keuangan yang berkaitan dengan kesehatan, dan produk lainnya yang sesuai untuk dapat mempertahankan pelanggan.

# Business Simulation

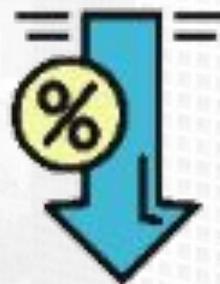
## Churn Rate

Before model

**20.4%**

Menurun hingga

**15.56%**



After model

**4.84%**

$$\text{Churn rate} = (\text{FN} / \text{Total Customer}) \times 100$$

$$= (484 / 10000) \times 100$$

Berdasarkan hasil perhitungan tersebut, terbukti bahwa **dengan menerapkan machine learning klasifikasi** menggunakan model **Random Forest** dapat berpotensi **menurunkan churn dari 20.4% menjadi 4.84%**

# Pembagian Tugas

---

Stage 0, 1, 2, 3, 4



# Pembagian Tugas

NAMA	STAGE 0	STAGE 1	STAGE 2	STAGE 3	STAGE 4
<b>Fiorent Arie Hernanti</b>	Team Leader, Objectives	Multivariate Analysis, Github	Feature tambahan, Github	Modelling: (split data, modelling, model evaluation: pemilihan metrics)	Team Leader, Source code, Notulen Mentoring, Laporan Final Project Stage 2
<b>Aleisya Zahari Salam</b>	Roles	Univariate Analysis, Github	Feature selection, Github	Modelling: (split data, modelling, model evaluation: pemilihan metrics)	PPT Final Presentation, Laporan Final Project Stage 3
<b>Alicia Gofina</b>	Business Metrics	Business Insight	Handle missing value, handle class imbalance	Feature Importance	Laporan Final Project: Stage 0, 2
<b>Devi Nur Aisyah</b>	Goal	Multivariate Analysis	Feature extraction	Modelling : (model evaluation: penentuan bestfit/overfit/underfit dan cross val; Hyperparameter Tuning)	Laporan Final Project: Stage 1
<b>Tisa Safina Alchalista</b>	Problem Statement	Descriptive Statistics	Handle duplicate data, handle outliers	Feature Importance	Laporan Final Project: Stage 3
<b>Najmi Laily Fahira</b>	Objectives	Business Insight	Feature encoding, handle class imbalance	Modelling : (model evaluation: penentuan bestfit/overfit/underfit dan cross val; Hyperparameter Tuning)	PPT Final Presentation
<b>Deva Khofifah Jauharotun Naqiyah</b>	Business Metrics	Univariate Analysis	Feature transformation, handle outliers	Modelling : (model evaluation: penentuan bestfit/overfit/underfit dan cross val; Hyperparameter Tuning)	Laporan Final Project:Stage 3

# Terima Kasih