STAGE 2 - HOMEWORK - DATA PREPROCESSING

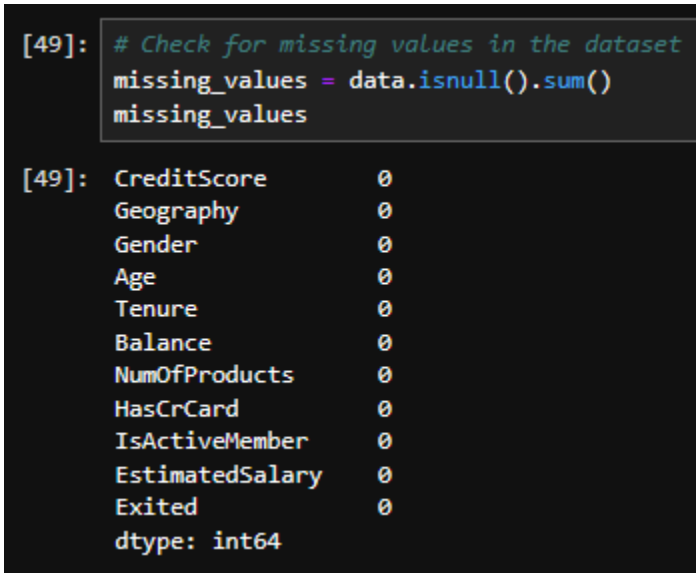
1.) Data Cleansing

Berdasarkan eksplorasi awal, dataset ini berisi beberapa fitur tentang pelanggan sebuah bank, dan kolom target "Exited" yang menunjukkan apakah pelanggan tersebut churn (bernilai 1) atau tidak (bernilai 0).

Fitur-fitur yang tersedia di dataset ini meliputi:

1. RowNumber: Nomor baris
2. CustomerId: ID pelanggan
3. Surname: Nama belakang pelanggan
4. CreditScore: Skor kredit pelanggan
5. Geography: Negara asal pelanggan
6. Gender: Jenis kelamin pelanggan
7. Age: Umur pelanggan
8. Tenure: Lamanya pelanggan menjadi nasabah bank
9. Balance: Saldo rekening pelanggan
10. NumOfProducts: Jumlah produk yang dimiliki pelanggan di bank
11. HasCrCard: Apakah pelanggan memiliki kartu kredit (1 = Ya, 0 = Tidak)
12. IsActiveMember: Apakah pelanggan aktif (1 = Ya, 0 = Tidak)
13. EstimatedSalary: Gaji estimasi pelanggan
14. Exited: Apakah pelanggan churn (1 = Ya, 0 = Tidak)

A. Handle Missing Values



```
[49]: # Check for missing values in the dataset
      missing_values = data.isnull().sum()
      missing_values

[49]: CreditScore        0
      Geography          0
      Gender             0
      Age                0
      Tenure             0
      Balance            0
      NumOfProducts      0
      HasCrCard          0
      IsActiveMember     0
      EstimatedSalary    0
      Exited             0
      dtype: int64
```

Gambar diatas menjawab pertanyaan bahwa tidak ada nilai yang hilang di setiap kolom. Oleh karena itu, tidak perlu mengambil tindakan apapun terkait dengan nilai yang hilang.

B. Handle Duplicated Data

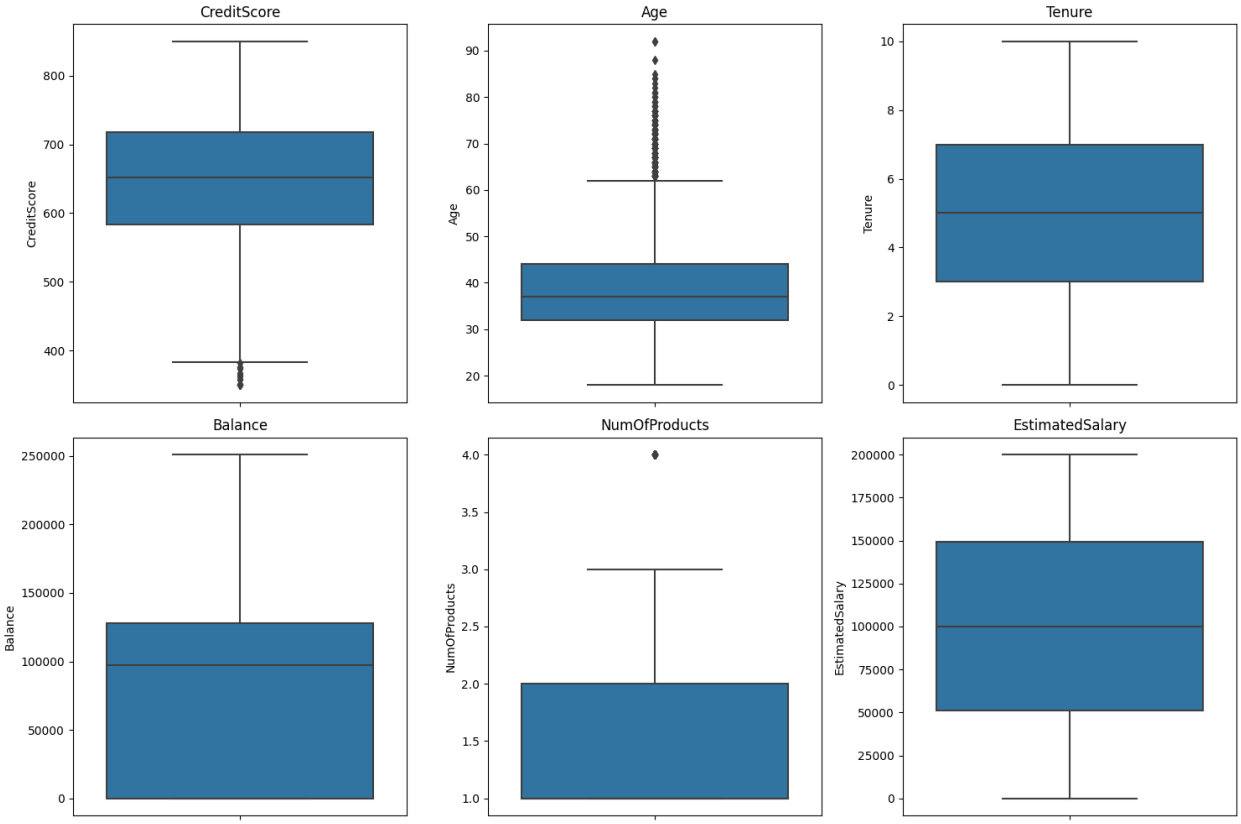
```
# Check for duplicated rows in the dataset
duplicated_rows = data.duplicated().sum()
duplicated_rows
```

```
0
```

Tidak ada baris yang duplikat dalam dataset ini. Oleh karena itu, kita tidak perlu mengambil tindakan apapun terkait dengan data duplikat.

C. Handle Outliers.

Kita akan fokus pada fitur numerik untuk memeriksa adanya outlier. Untuk memudahkan identifikasi, kita akan menggunakan visualisasi berupa boxplot.



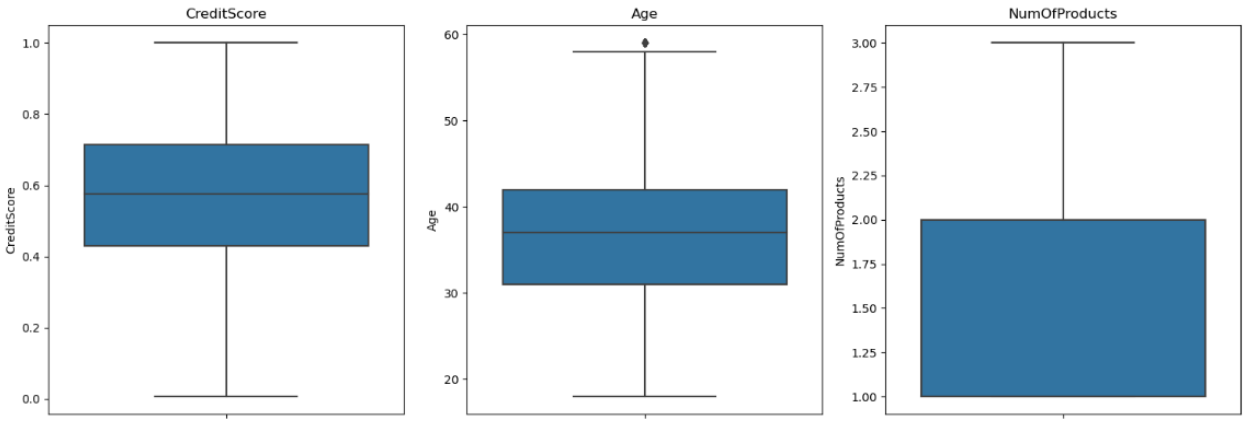
Dari boxplot di atas, kita dapat mengamati beberapa hal:

1. **CreditScore:** Terdapat beberapa nilai yang lebih rendah daripada whisker bawah, yang dapat dianggap sebagai outlier.
2. **Age:** Terdapat beberapa nilai yang lebih tinggi daripada whisker atas, yang dapat dianggap sebagai outlier.
3. **Tenure:** Tidak tampak adanya outlier.
4. **Balance:** Tidak tampak adanya outlier.
5. **NumOfProducts:** Terdapat beberapa nilai yang lebih tinggi daripada whisker atas, yang dapat dianggap sebagai outlier.
6. **EstimatedSalary:** Tidak tampak adanya outlier.

Meskipun kita dapat mengidentifikasi beberapa outlier, keputusan untuk menanganinya tergantung pada konteks bisnis dan tujuan analisis. Dalam banyak kasus, outlier mungkin mengandung informasi yang penting.

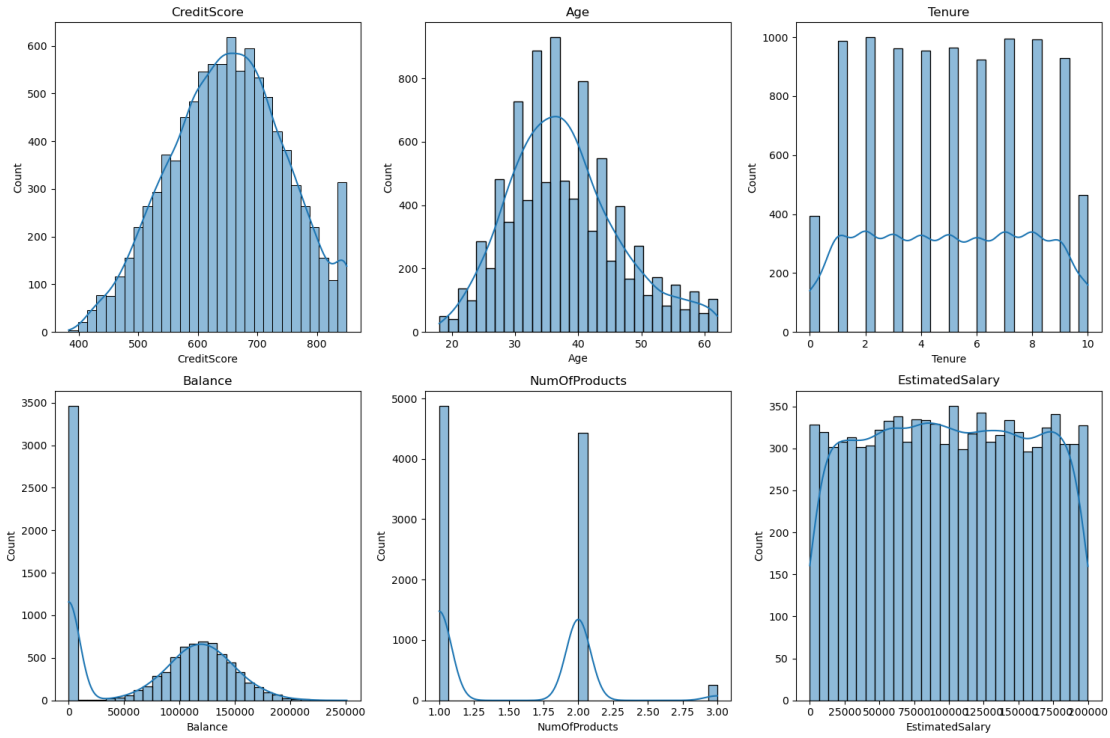
Sebagai contoh, dalam analisis churn, pelanggan dengan perilaku yang "tidak biasa" (mis. skor kredit yang sangat rendah atau usia yang sangat tinggi) mungkin justru adalah segmen yang penting untuk dipahami. Tetapi, outlier bisa saja mengakibatkan model yang akan dibuat menghasilkan hasil yang kurang memuaskan.

Untuk saat ini, kami akan mengatasi outlier tersebut dengan memakai metode interquartile range (IQR) untuk mengurangi outlier yang ada. Hasil dari penghapusan outlier pada kolom-kolom yang teridentifikasi memiliki outlier sebagai berikut:



D. Feature Transformation

Transformasi fitur dapat meningkatkan performa model dengan mengubah distribusi atau skala data. Beberapa metode transformasi populer meliputi normalisasi, standarisasi, dan transformasi logaritma. Pertama, kita lihat distribusi dari fitur numerik untuk memutuskan apakah transformasi diperlukan.

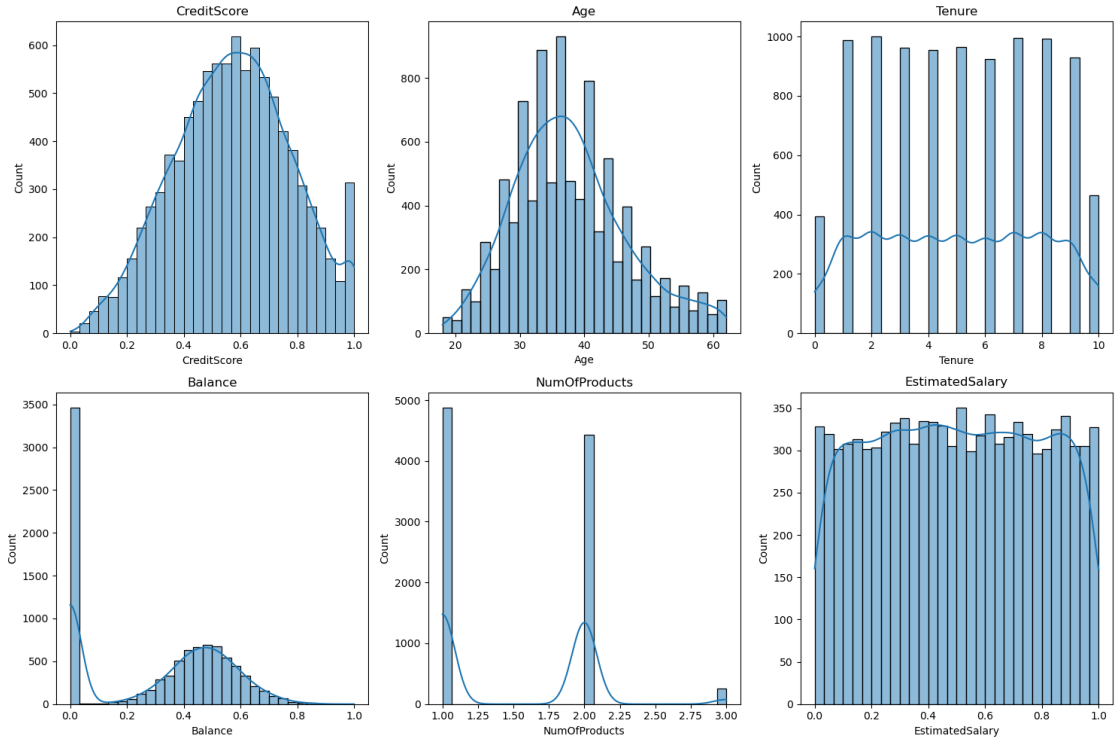


Berdasarkan histogram di atas:

1. CreditScore: Distribusi tampaknya mendekati normal.
2. Age: Distribusi sedikit condong ke kanan.
3. Tenure: Distribusi tampak multi-modal, dengan beberapa puncak.
4. Balance: Terdapat dua puncak, salah satunya di nol yang menunjukkan banyak pelanggan dengan saldo nol.
5. NumOfProducts: Distribusi adalah kategorikal dengan beberapa nilai yang dominan.
6. EstimatedSalary: Distribusi tampaknya seragam.

Pada Feature-Feature yang memiliki rentang yang jauh kami melakukan transformasi feature menggunakan normalisasi (MinMaxScaler) untuk mengubah rentang datanya menjadi 0-1 yang bertujuan agar model yang nantinya kami buat dapat menghasilkan nilai yang optimal.

Plot hasil normalisasi sebagai berikut :



E. Feature Encoding

Encoding adalah proses konversi fitur kategorikal menjadi format yang dapat dimengerti oleh algoritma machine learning. Kita akan memeriksa tipe data dari setiap kolom dan menentukan apakah ada fitur kategorikal yang perlu di-encode.

Berdasarkan tipe data, kita memiliki beberapa fitur kategorikal:

1. Geography
2. Gender
3. Exited

Untuk fitur Geography, Gender dan Exited, kita perlu melakukan encoding. Ada berbagai metode encoding seperti One-Hot Encoding, Label Encoding, dan lainnya. Untuk tujuan ini, kita akan menggunakan Label Encoding.

Gambar dibawah merupakan contoh code untuk melakukan proses **Label Encoding**

```
## Menggunakan Label Encoder
from sklearn.preprocessing import LabelEncoder
# Buat objek LabelEncoder
label_encoder = LabelEncoder()

# Lakukan label encoding pada kolom "Geography" dan "Gender"
data['Geography'] = label_encoder.fit_transform(data['Geography'])
data['Gender'] = label_encoder.fit_transform(data['Gender'])
data['Exited'] = label_encoder.fit_transform(data['Exited'])

# Hasil label encoding
display(data[['Geography','Gender', 'Exited']])
```

Setelah melakukan label encoding terhadap 3 kolom categorik yang penting, kami mendapatkan hasil dibawah ini :

Pada kolom Geography: 0 = Germany, 1 = France, 2 = Spain
Pada kolom Gender: 0 = Female, 1 = Male
Pada kolom Exited : 0 = No, 1 = Yes

F. Handle Class Imbalance

Kita akan memeriksa distribusi kelas target (Exited) untuk melihat apakah ada ketidakseimbangan kelas yang perlu diatasi.

Distribusi kelas target (Exited) adalah sebagai berikut:

```
[103]: # Check the distribution of the target class 'Exited'
       class_distribution = data['Exited'].value_counts(normalize=True)
       class_distribution

[103]: 0    0.802362
       1    0.197638
       Name: Exited, dtype: float64
```

1. Kelas 0 (Tidak Churn): 80.23%
2. Kelas 1 (Churn): 19.76%

Terdapat ketidakseimbangan kelas di mana kelas 0 memiliki representasi yang jauh lebih tinggi dibandingkan dengan kelas 1. Ketidakseimbangan ini dapat mempengaruhi performa model, terutama dalam menilai kelas minoritas.

Ada beberapa metode untuk menangani ketidakseimbangan kelas, seperti:

1. **Resampling:** Teknik ini melibatkan penambahan atau pengurangan sampel dari kelas tertentu untuk mencapai distribusi yang lebih seimbang.
2. **Menggunakan metriks evaluasi yang tepat:** Akurasi mungkin bukan metrik yang ideal dalam kasus ketidakseimbangan kelas. Metrik lain seperti F1-score, AUC-ROC, atau precision dan recall mungkin lebih informatif.
3. **Penggunaan algoritma yang mendukung penimbangan kelas:** Beberapa algoritma memungkinkan penimbangan kelas saat pelatihan, yang memberikan penalti lebih tinggi untuk kesalahan pada kelas minoritas.
4. Penggunaan teknik ensemble seperti **Random OverSampling Boost (ROSB)** atau **Synthetic Minority Over-sampling Technique (SMOTE).**

Keputusan tentang bagaimana menangani ketidakseimbangan kelas tergantung pada tujuan analisis dan model yang akan digunakan. Jika kita ingin fokus pada identifikasi pelanggan yang mungkin churn (kelas 1), maka mungkin perlu mempertimbangkan resampling atau teknik lain untuk meningkatkan sensitivitas model terhadap kelas tersebut.

Untuk saat ini, asumsi kita akan menggunakan algoritma yang mendukung penimbangan kelas dan metrik evaluasi yang sesuai saat pelatihan model. Ketidakseimbangan kelas akan di-handle dengan menggunakan algoritma yang mendukung penimbangan kelas dan metrik evaluasi yang tepat. Imbalancing dilakukan dengan menetapkan threshold 0.5 untuk memperbanyak label yes.

Imbalancing terhadap data target dengan menetapkan threshold 50%

```
[104]: data['Exited'] = data['Exited'] > 0.8
       print(data['Exited'].value_counts())

       False    7677
       True     1891
       Name: Exited, dtype: int64

[105]: X = data[[col for col in data.columns if col not in['Exited']]].values
       y = data['Exited'].values
       print(X.shape)
       print(y.shape)

       (9568, 10)
       (9568,)

[106]: from imblearn import under_sampling, over_sampling
       X_over_SMOTE, y_over_SMOTE = over_sampling.SMOTE(sampling_strategy=0.5).fit_resample(X,y)

[107]: print(pd.Series(y_over_SMOTE).value_counts())

       False    7677
       True     3838
       dtype: int64
```

Gambar diatas merupakan hasil before dan after imbalancing data menggunakan teknik **Synthetic Minority Over-sampling Technique (SMOTE).** Kelas 1 (yes) yang sebelumnya berjumlah 1891 menjadi 3838 setelah dilakukannya oversampling terhadap class minoritas tersebut

2.) A.) Feature selection (membuang feature yang kurang relevan atau redundan)

Baca dataset :

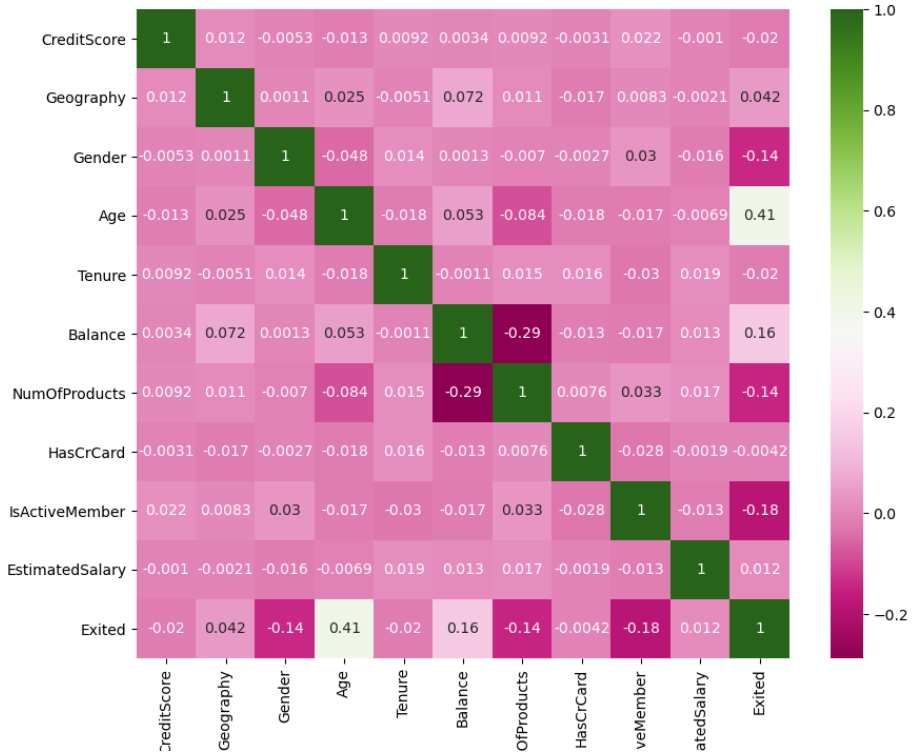
```
[19]: data.head()
```

[19]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	0.505353	0.0	0.0	42.0	2.0	0.000000	1.0	1.0	1.0	0.506735	True
1	0.481799	2.0	0.0	41.0	1.0	0.334031	1.0	0.0	1.0	0.562709	False
2	0.254818	0.0	0.0	42.0	8.0	0.636357	3.0	1.0	0.0	0.569654	True
3	0.676660	0.0	0.0	39.0	1.0	0.000000	2.0	0.0	0.0	0.469120	False
4	1.000000	2.0	0.0	43.0	2.0	0.500246	1.0	1.0	1.0	0.395400	False

Pada data diatas, sebelumnya kami sudah membuang beberapa kolom yang tidak terlalu dibutuhkan untuk pemodelan seperti (Surname, CustomerId, RowNumber) sehingga total kolom nya sekarang hanya ada 11.

Sebelum kita memasuki seleksi fitur ada baiknya kita melihat korelasi antar fitur terlebih dahulu dengan menggunakan heatmap agar lebih jelas.



Insight Feature Selection:
Dapat dilihat pada visualisasi diatas, tidak ditemukannya fitur yang redundan. Artinya kita tidak perlu membuang suatu fitur, fitur-fitur di dalam dataset ini bisa dipakai dan tinggal menambahkan feature baru agar bisa mengambil insight lebih banyak serta membantu meningkatkan performa model.

B.) Feature extraction (membuat feature baru dari feature yang sudah ada)

```
data['Balance_to_EstimatedSalary'] = data['Balance'] / data['EstimatedSalary']
data['Age_to_Tenure'] = data['Age'] / data['Tenure']
data['CreditScore_to_Age'] = data['CreditScore'] / data['Age']
data['CreditScore_to_Balance'] = data['CreditScore'] / (data['Balance'] + 1)
data['NumOfProducts_to_Age'] = data['NumOfProducts'] / data['Age']
```

Penjelasan fitur yang dibuat :
1. **Balance_to_EstimatedSalary**: Fitur ini menghitung rasio antara saldo rekening pelanggan dan gaji estimasi pelanggan. Ini dapat memberikan indikasi seberapa besar persentase dari gaji pelanggan yang disimpan di rekening bank. Fitur ini mungkin berguna untuk mengidentifikasi pola-pola yang berkaitan dengan besarnya saldo rekening relatif terhadap gaji.

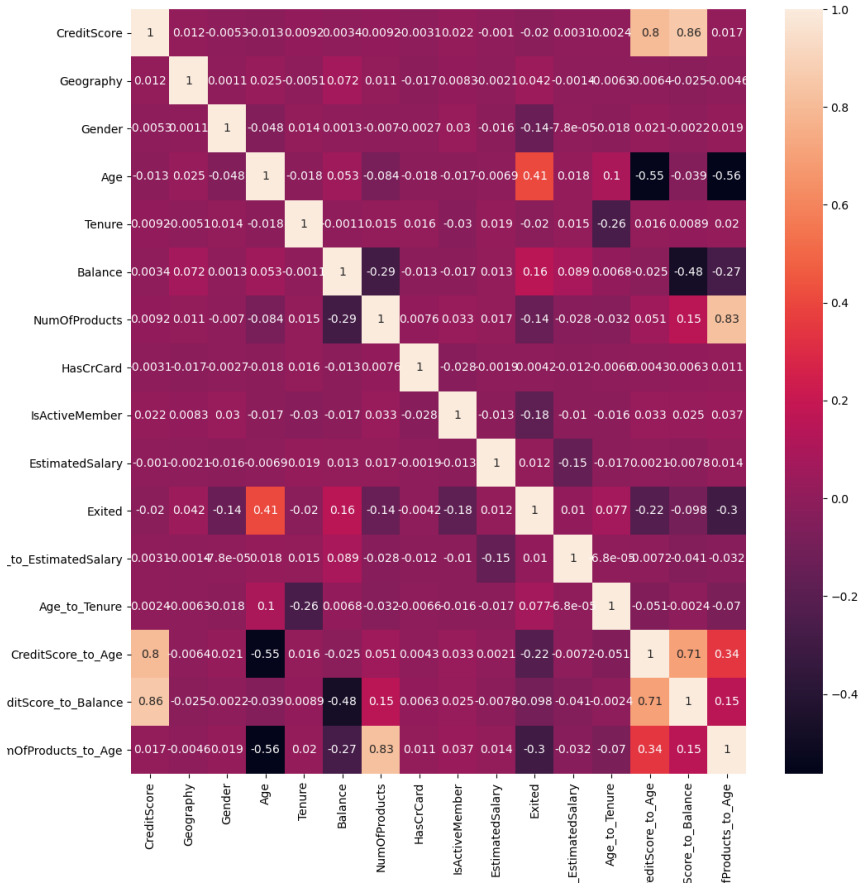
2. **Age_to_Tenure**: Fitur ini menghitung rasio antara usia pelanggan dan lamanya pelanggan menjadi nasabah bank. Ini mencoba mengukur seberapa lama pelanggan telah menjadi nasabah dalam konteks usianya. Hal ini bisa membantu dalam memahami apakah pelanggan yang lebih muda atau lebih tua cenderung menjadi pelanggan baru atau setia.

3. **CreditScore_to_Age**: Fitur ini menghitung rasio antara skor kredit pelanggan dan usia pelanggan. Ini mencoba melihat hubungan antara skor kredit dan usia, apakah skor kredit cenderung berbeda antara kelompok usia yang berbeda.

4. **CreditScore_to_Balance**: Fitur ini menghitung rasio antara skor kredit pelanggan dan saldo rekening pelanggan. Ini dapat memberikan gambaran tentang apakah ada korelasi antara skor kredit dan seberapa banyak uang yang disimpan di rekening.

5. **NumOfProducts_to_Age**: Fitur ini menghitung rasio antara jumlah produk yang dimiliki pelanggan dan usia pelanggan. Ini bisa membantu dalam memahami sejauh mana pelanggan yang lebih muda atau lebih tua cenderung memiliki lebih banyak produk perbankan.

Cek korelasi setelah adanya fitur baru :



Dapat dilihat bahwa masih ada feature yang redundan dari hasil fitur engineering. Maka dari itu, kami akan mempertimbangkan apakah tetap akan memakai fitur tersebut atau dihapus saja.

3. Git

Link repository : https://github.com/team-predictive-pioneers/STAGE-2