Unsupervised Learning

January 5, 2024

Coursebook: Unsupervised Learning

- Bagian 6 Audit Analytics untuk Bank Rakyat Indonesia
- Durasi: 7 Jam
- *Last Updated*: December 2023

- *Coursebook* ini disusun dan dikurasi oleh tim produk dan instruktur dari Algoritma Data Science School

1 Background

Coursebook ini merupakan bagian dari **BRI Audit Analytics** yang disiapkan oleh Algoritma. *Coursebook* ini ditujukan hanya untuk khalayak terbatas, yaitu individu dan organisasi yang menerima *coursebook* ini langsung dari organisasi pelatihan. Tidak boleh direproduksi, didistribusikan, diterjemahkan, atau diadaptasi dalam bentuk apapun di luar individu dan organisasi ini tanpa izin.

Algoritma adalah pusat pendidikan *data science* yang berbasis di Jakarta. Kami menyelenggarakan *workshop* dan program pelatihan untuk membantu para profesional dan mahasiswa dalam menguasai berbagai sub-bidang *data science* yaitu: *data visualization, machine learning,* statistik, dan lain sebagainya.

1.1 Training Objectives

Pada workshop ini, Anda akan mempelajari salah satu case yang cukup umum diselesaikan dengan *machine learning* yakni *unsupervised learning* yang banyak digunakan mulai dari PCA (Principal Component Analysis), hingga Clustering, dan pendekatan penemuan pola lainnya di mana variabel target tidak diketahui atau ditentukan. Tujuan kami adalah mengembangkan intuisi yang kuat di balik masalah dimensi, mekanisme yang kami miliki, dan pada akhirnya memperkuat pemahaman kami dengan mengerjakan dua skenario bisnis kehidupan nyata yang paling umum.

Anda akan mempelajari siklus proses dalam membangun model *machine learning* aplikasi dari *unsupervised learning*. Workshop ini akan berfokus pada PCA, hingga melakukan pembuatan model untuk *anomaly detection* menggunakan *Logistic Outlier Factor* (LOF). Dengan durasi selama 7 jam, pembelajaran pada workshop ini terbagi dalam beberapa modul:

- ***Data Preprocessing***
  - *Principles and Motivation*
  - *Variance and matrix covariance*
  - *Eigenvalues and eigenvector*

  - *Principal Component Analysis (PCA)*

- ***Exploratory Data Analysis***
  - *Biplot visualization*
    - *Individual factor map*
    - *Variable factor map*

- ***Anomaly Detection***
  - *Understanding Local Outlier Factor (LOF)*
  - *Practicing Local Outlier Factor (LOF)*

2 Unsupervised Learning

Dalam Spesialisasi Machine Learning, kita telah mempelajari algoritma yang sangat berguna dalam situasi regresi dan klasifikasi. Secara umum, kita memahami cara mencari parameter untuk X1, X2,…Xn untuk menjelaskan atau memprediksi respons "target" Y.

Pada pembelajaran tanpa pengawasan (*unsupervised learning*), situasinya berbeda karena tidak ada respons Y atau target. Fokusnya adalah menemukan struktur/pola antara X1, X2,…Xn dengan tujuan untuk mengidentifikasi peluang pengurangan dimensi atau pengelompokan. *Unsupervised Learning* sering dianggap sebagai proses eksplorasi karena sulit menentukan model atau formulasi yang "benar" tanpa *"ground truth"* sebagai tolok ukurnya. Oleh karenanya, teknik seperti *confusion matrix* dan AUC tidak berlaku karena kurangnya label *"ground truth"*.

Meskipun begitu, metode *unsupervised learning* tetap masih sangat bermanfaat, terutama dalam konteks pengelompokan dan reduksi dimensi. Dalam case ini, kita akan menyelami algoritma *unsupervised learning* seperti PCA dan *anomaly detection*, serta memahami mengapa metode semacam ini merupakan alat penting untuk ditambahkan ke dalam tools anda.

3 Pengurangan Dimensi

3.1 Prinsip dan Motivasi

Penerapan *machine learning* semakin meluas di bidang yang melibatkan data berdimensi tinggi seperti pengenalan angka tulisan tangan, *Internet of Things* (IoT), dan pengenalan wajah. Oleh karena itu, sebagai seorang ilmuan data modern yang bekerja dengan teknologi ini dihadapkan pada masalah "dimensi" yang memerlukan solusi metodis, bukan hanya untuk mengurangi dimensi data tetapi juga meminimalkan kehilangan informasi.

Sebagai contoh motivasi, bayangkan kasus gambar beresolusi rendah dibawah ini. Lakukan pencarian sederhana di Google untuk wajah hitam putih berukuran 40x40 piksel[^1]. Saat setiap gambar dianggap sebagai inputan/masukkan, kita memiliki kumpulan data dengan 1.600 dimensi.

Jika Anda memperbesar wajah tersebut, kita dapat melihat 1.600 piksel individual. Saat menangani gambar skala abu-abu / *gray scale*, strateginya dapat dengan memberikan nilai kecerahan pada skala 0 hingga 1 pada masing-masing 1.600 kolom, dengan 0 untuk "putih penuh" dan 1 untuk "hitam penuh", dan bergantung pada saturasi atau kecerahan setiap piksel - berikan nilai di antaranya.

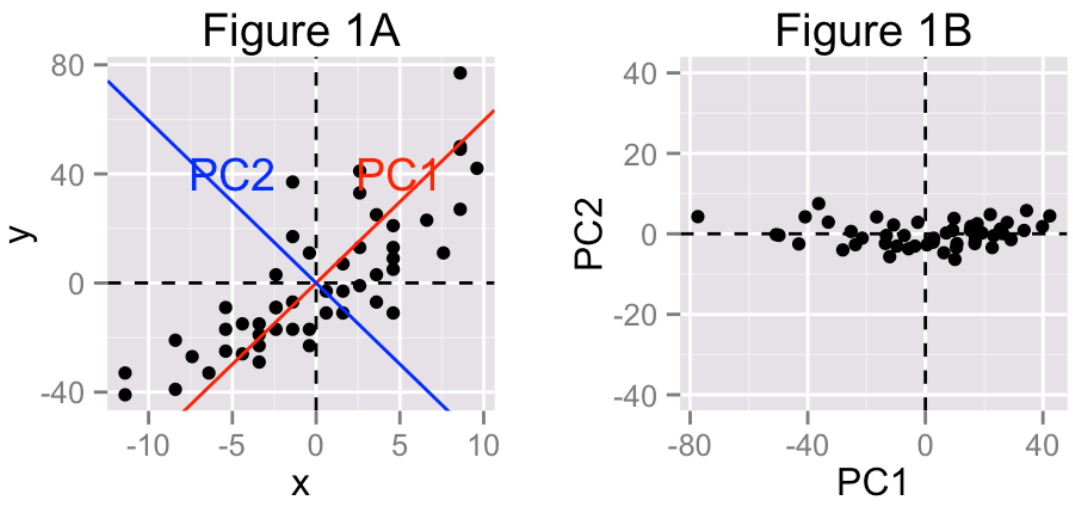
Ada banyak makalah dan sumber bagus yang membahas topik penggunaan PCA dalam kompresi gambar, seperti yang ditulis oleh para peneliti di *Institute of Chemical Technology, Prague*[^2], *The one by Czech Institute of Informatics, Robotics and Cybernetics*[^3], dan yang ada disini[^4].

Gambar berukuran 40x40 mungkin tidak menarik. Jika Anda membuat klasifikasi gambar menggunakan foto dari *iPhone 7 Plus*, yang mana memiliki resolusi 1.920 x 1.080 piksel (lebih dari 2 juta dimensi). Dan itu untuk satu pengamatan. Ingat dari materi minggu lalu *Practical Statistic* bahwa cara kita mengukur "informasi" adalah melalui nilai varians-nya, sehingga metode reduksi dimensi pada dasarnya berkaitan dengan mewakili sebanyak mungkin "varians" dalam dimensi sesedikit mungkin.

Hasil dari transformasi ini adalah data asli kita (matriks X) diwakili oleh matriks yang ditransformasikan secara linier, Z, dengan Z biasanya merupakan matriks dengan dimensi yang jauh lebih sedikit (biasanya <10) daripada X. Kolom pertama dari Z menjelaskan sebagian besar varians dalam X, dan kolom kedua menjelaskan jumlah varians yang lebih kecil dari kolom pertama, dan seterusnya hingga kolom terakhir.

Tujuan PCA adalah mencari Q, sehingga transformasi linier dapat dilakukan.

PCA akan mencari korelasi dalam data kita dan menggunakan redundansi tersebut untuk membuat matriks Z baru dengan dimensi yang cukup untuk menjelaskan sebagian besar varians dalam data asli. Variabel baru matriks Z disebut **komponen utama**.



Jika Anda melihat Gambar 1A, menunjukkan data asli yang berada pada bidang dengan koordinat x dan y. Dua dimensi (masing-masing x dan y) diperlukan untuk menggambarkan varians dalam data ini sepenuhnya.

Namun, seandainya kita mengidentifikasi dua sumbu lain untuk mendeskripsikan data yang sama, dan salah satunya ortogonal langsung terhadap sumbu lainnya: sekarang kita dapat mengukur varians dalam data hanya dengan menggunakan sumbu baru ini, dimana (kita menyebutnya **komponen utama**).

Komponen utama pertama (PC1) adalah komponen utama yang menjelaskan varians paling banyak dalam data. Komponen utama kedua (PC2) menjelaskan varians sisa dalam data. Dalam gambar 1B, PC1 adalah garis lurus yang menghubungkan semua titik data. Garis ini menjelaskan varians paling banyak dalam data, yaitu varians yang disebabkan oleh perbedaan nilai x.

PC2 tegak lurus terhadap PC1. PC2 menjelaskan varians sisa dalam data, yaitu varians yang disebabkan oleh perbedaan nilai y.

Jika kita menganggap PC2 konstan, kita dapat memproyeksikan semua titik data ke PC1. Proyeksi ini akan mengurangi dimensi data dari dua menjadi satu.

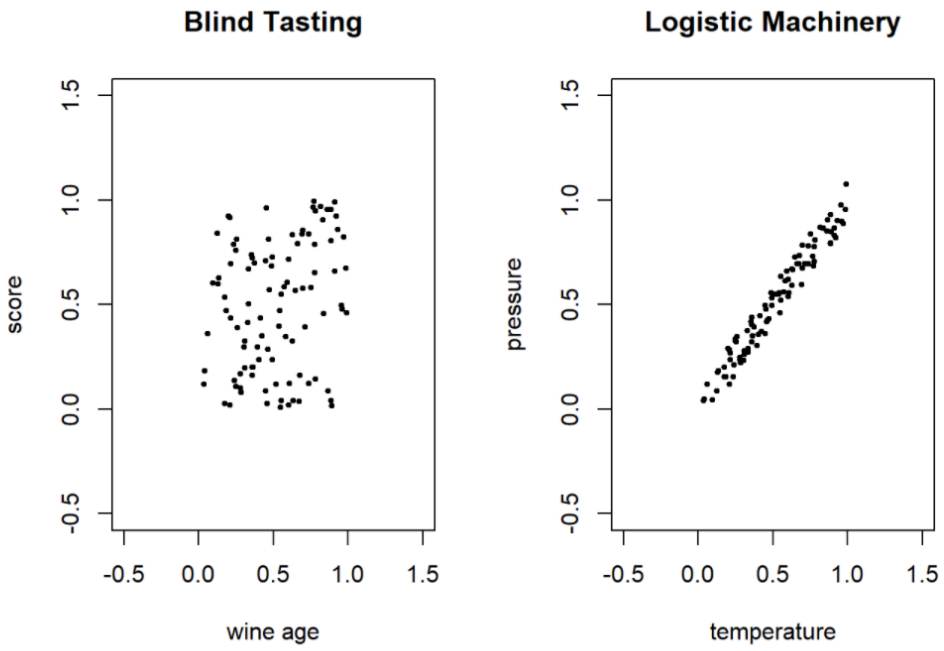
Proses ini disebut proyeksi ortogonal. Proyeksi ortogonal adalah proses memindahkan titik data ke arah sumbu tertentu, dengan mempertahankan jarak dari titik data ke sumbu tersebut.

Dalam kasus ini, proyeksi ortogonal akan mengurangi varians yang dijelaskan oleh PC2. Namun, karena varians yang dijelaskan oleh PC2 relatif kecil, kehilangan varians ini dapat diterima.

Aplikasi PCA lainnya:

- Penemuan pola pada data berdimensi tinggi
- Identifikasi variabel yang sangat berkorelasi dengan variabel lain
- Memvisualisasikan data berdimensi tinggi

Dive Deeper: Manakah dari dua kumpulan data berikut yang paling terbantu oleh Analisis Komponen Utama (PCA)?



4 Import Library

[5]:
```
# Menyembunyikan Peringatan
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter(action='ignore', category=FutureWarning)

# Komputasi Matematika dan Array
import math
import numpy as np

# Persiapan Data
import pandas as pd

# Visualisasi Data
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go

# Paket Machine Learning
import scipy.stats as stats
from scipy.stats import norm, skew
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from pyod.models.lof import LOF
```

```
# Paket Lainnya
from numpy.linalg import eig
from helper import biplot_pca
plt.style.use('seaborn-v0_8-whitegrid')
```

---

5 Step-by-Step: PCA Analysis

Dalam langkah-langkah ini, kita akan melakukan analisis Principal Component Analysis (PCA), termasuk proses hingga pembuatan dataframe baru dari hasil komponen utama (Principal Components).

Kita akan menggunakan dataset `credit100.csv` yang sama pada course sebelumnya.

[6]:
```
credit_raw = pd.read_csv('data_input/credit100.csv')
credit_raw.head()
```

[6]:

	fraud_bool	income	name_email_similarity	prev_address_months_count	current_address_months_count	customer_age	days_since_request	intended_balcon_amount	payment_type	zip_count_4w	…	has_other_cards	proposed_credit_limit	foreign_request	source	session_length_in_minutes	device_os	keep_alive_session
0	0	0.9	0.615786	-1	51	40	0.010337	-0.699550	AB	2659	…	1	1500.0	0	INTERNET			
1	0	0.6	0.072461	-1	125	20	0.006974	-0.959468	AC	5599	…	0	200.0	0	INTERNET			
2	1	0.5	0.076592	-1	37	50	0.006319	-1.219933	AB	1352	…	0	1500.0	0	INTERNET			
3	0	0.3	0.255433	23	19	20	0.009267	3.242242	AA	3743	…	0	200.0	0	INTERNET			
4	0	0.9	0.192362	-1	34	30	0.019240	-1.131560	AB	546	…	1	1000.0	0	INTERNET			

```
0                   13.079806     linux                   0
1                    6.666551   windows                   0
2                    4.492668   windows                   1
3                    2.321885     other                   1
4                    4.845147     linux                   1

   device_distinct_emails_8w device_fraud_count  month
0                          1                  0      1
1                          1                  0      4
2                          2                  0      0
3                          1                  0      3
4                          1                  0      5

[5 rows x 32 columns]
```

Deskripsi Dataset:

Berikut adalah interpretasi keterangan kolom-kolom pada dataset:

1. **fraud_bool:** Label Kecurangan (1 jika kecurangan, 0 jika sah).
2. **income:** Pendapatan tahunan pelamar dalam kuartil. Rentang antara [0.1, 0.9].
3. **name_email_similarity:** Metrik kesamaan antara email dan nama pelamar. Nilai tinggi menunjukkan kesamaan tinggi. Rentang antara [0,1].
4. **prev_address_months_count:** Jumlah bulan di alamat terdaftar sebelumnya dari pelamar, yaitu tempat tinggal sebelumnya pelamar, jika berlaku. Rentang antara [-1,380] bulan (-1 adalah nilai yang hilang).
5. **current_address_months_count:** Bulan di alamat terdaftar saat ini dari pelamar. Rentang antara [-1,429] bulan (-1 adalah nilai yang hilang).
6. **customer_age:** Usia pelamar dalam rentang per dekade (misalnya, 20-29 direpresentasikan sebagai 20). Rentang antara [10 - 90] tahun.
7. **days_since_request:** Jumlah hari sejak aplikasi diajukan. Rentang antara [0,79] hari.
8. **intended_balcon_amount:** Jumlah awal yang ditransfer untuk aplikasi. Rentang antara $[-16, 114]$. (Nilai negative merupakan nilai yang hilang).
9. **payment_type:** Jenis rencana pembayaran kredit. 5 kemungkinan nilai (anonim).
10. **zip_count_4w:** Jumlah aplikasi dalam kode pos yang sama dalam 4 minggu terakhir. Rentang antara [1, 6830].
11. **velocity_6h:** Kecepatan total aplikasi yang dibuat dalam 6 jam terakhir, rata-rata jumlah aplikasi per jam dalam 6 jam terakhir. Rentang antara $[-175, 16818]$.
12. **velocity_24h:** Kecepatan total aplikasi yang dibuat dalam 24 jam terakhir, rata-rata jumlah aplikasi per jam dalam 24 jam terakhir. Rentang antara [1297, 9586]
13. **velocity_4w:** Kecepatan total aplikasi yang dibuat dalam 4 minggu terakhir, rata-rata jumlah aplikasi per jam dalam 4 minggu terakhir. Rentang antara [2825, 7020].

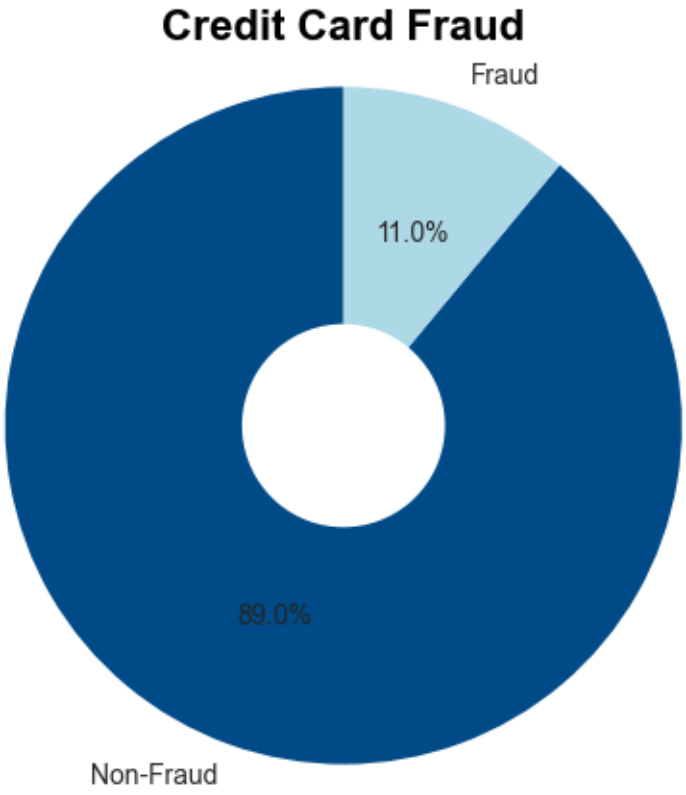
14. **bank_branch_count_8w:** Jumlah total aplikasi di cabang bank terpilih dalam 8 minggu terakhir. Rentang antara [0, 2404].
15. **date_of_birth_distinct_emails_4w:** Jumlah email untuk pelamar dengan tanggal lahir yang sama dalam 4 minggu terakhir. Rentang antara [0, 39].
16. **employment_status:** Status pekerjaan pelamar. 7 kemungkinan nilai (anonim).
17. **credit_risk_score:** Skor risiko aplikasi secara internal. Rentang antara [−191, 389].
18. **email_is_free:** Domain email aplikasi (gratis atau berbayar).
19. **housing_status:** Status tempat tinggal saat ini pelamar. 7 kemungkinan nilai (anonim).
20. **phone_home_valid:** Validitas nomor telepon rumah yang diberikan.
21. **phone_mobile_valid:** Validitas nomor telepon seluler yang diberikan.
22. **bank_months_count:** Berapa umur akun sebelumnya (jika ada) dalam bulan. Rentang antara [-1,32] bulan (-1 adalah nilai yang hilang).
23. **has_other_cards:** Jika pelamar memiliki kartu lain dari perusahaan perbankan yang sama.
24. **proposed_credit_limit:** Batas kredit yang diajukan oleh pelamar. Rentang antara [200,2000].
25. **foreign_request:** Jika negara asal permintaan berbeda dari negara bank.
26. **source:** Sumber aplikasi online. Baik browser (INTERNET) atau aplikasi seluler (TELEAPP).
27. **session_length_in_minutes:** Panjang sesi pengguna di situs web perbankan dalam menit. Rentang antara [-1,107] menit.
28. **device_os:** Sistem operasi perangkat yang melakukan permintaan. Nilai mungkin: Windows, macOS, Linux, X11, atau lainnya.
29. **keep_alive_session:** Pilihan pengguna tentang logout sesi.
30. **device_distinct_emails_8w:** Jumlah email yang berbeda dalam situs web perbankan dari perangkat yang digunakan dalam 8 minggu terakhir. Rentang antara [−1, 2] emails (-1 merupakan nilai yang hilang).
31. **device_fraud_count:** Jumlah aplikasi penipuan dengan perangkat yang digunakan. Rentang antara [0,1]. Pada dataset hanya ada 0 sehingga tidak digunakan.
32. **month:** Bulan di mana aplikasi diajukan. Rentang antara [0,7].

5.1 Exploratory Data Analysis

5.1.1 Cek Jumlah Transaksi

[7]:
```
# Hitung proporsi non-fraud dan fraud
values = credit_raw['fraud_bool'].value_counts(normalize=True)
total_normal = credit_raw.loc[credit_raw['fraud_bool'] == 0].shape[0]
```

```
total_fraudulent = credit_raw.loc[credit_raw['fraud_bool'] == 1].shape[0]

# Persiapkan data untuk plot
labels = ['Non-Fraud', 'Fraud']
colors = ['#004B87', 'LightBlue']

# Buat plot pie
plt.figure(figsize=(5, 5))
plt.pie(values, labels=labels, autopct='%1.1f%%', colors=colors, startangle=90,␣
↪wedgeprops=dict(width=0.7))

# Atur proporsi lingkaran agar berbentuk lingkaran
plt.axis('equal')

# Atur judul dan tampilan plot
plt.title('Credit Card Fraud', fontsize=16, fontweight='bold', color='black')
plt.show()
```



NOTE:

Dataset yang kita gunakan dalam course ini menyertakan kolom label yang mencerminkan kelas transaksi fraud dan no-fraud. Meskipun demikian, fokus utama analisis

kita adalah pada eksplorasi struktur data dan pengidentifikasian pola tanpa memperhatikan label kelas. Tujuan utama course ini adalah memahami teknik `unsupervised learning`, anomaly detection, serta bagaimana menerapkannya untuk mendapatkan wawasan yang berharga dari dataset, terlepas dari ketidakseimbangan kelas. Dengan demikian, kita akan fokus pada teknik unsupervised learning dan pemahaman yang mendalam tentang struktur data.

5.1.2 Screening dataset secara cepat

[8]:
```
credit_raw.head(5).T
```

[8]:

	0	1	2	\
fraud_bool	0	0	1	
income	0.9	0.6	0.5	
name_email_similarity	0.615786	0.072461	0.076592	
prev_address_months_count	-1	-1	-1	
current_address_months_count	51	125	37	
customer_age	40	20	50	
days_since_request	0.010337	0.006974	0.006319	
intended_balcon_amount	-0.69955	-0.959468	-1.219933	
payment_type	AB	AC	AB	
zip_count_4w	2659	5599	1352	
velocity_6h	11007.70883	6304.262836	1357.413551	
velocity_24h	7448.797416	4373.093592	6667.029662	
velocity_4w	5576.303564	4721.100025	6318.876263	
bank_branch_count_8w	2030	1	15	
date_of_birth_distinct_emails_4w	7	13	6	
employment_status	CA	CA	CF	
credit_risk_score	253	112	199	
email_is_free	1	1	1	
housing_status	BA	BE	BC	
phone_home_valid	1	1	1	
phone_mobile_valid	1	0	1	
bank_months_count	9	-1	28	
has_other_cards	1	0	0	
proposed_credit_limit	1500.0	200.0	1500.0	
foreign_request	0	0	0	
source	INTERNET	INTERNET	INTERNET	
session_length_in_minutes	13.079806	6.666551	4.492668	
device_os	linux	windows	windows	
keep_alive_session	0	0	1	
device_distinct_emails_8w	1	1	2	
device_fraud_count	0	0	0	
month	1	4	0	

	3	4
fraud_bool	0	0

income	0.3	0.9
name_email_similarity	0.255433	0.192362
prev_address_months_count	23	-1
current_address_months_count	19	34
customer_age	20	30
days_since_request	0.009267	0.01924
intended_balcon_amount	3.242242	-1.13156
payment_type	AA	AB
zip_count_4w	3743	546
velocity_6h	5234.262304	3589.153453
velocity_24h	6139.489579	4574.395571
velocity_4w	4201.725919	4318.258913
bank_branch_count_8w	1	16
date_of_birth_distinct_emails_4w	14	5
employment_status	CA	CA
credit_risk_score	63	275
email_is_free	1	1
housing_status	BC	BC
phone_home_valid	0	1
phone_mobile_valid	1	1
bank_months_count	16	1
has_other_cards	0	1
proposed_credit_limit	200.0	1000.0
foreign_request	0	0
source	INTERNET	INTERNET
session_length_in_minutes	2.321885	4.845147
device_os	other	linux
keep_alive_session	1	1
device_distinct_emails_8w	1	1
device_fraud_count	0	0
month	3	5

5.1.3 Data Types Inspection and Preparation

[9]:
```
credit_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   fraud_bool                      100000 non-null  int64
 1   income                          100000 non-null  float64
 2   name_email_similarity           100000 non-null  float64
 3   prev_address_months_count       100000 non-null  int64
 4   current_address_months_count    100000 non-null  int64
 5   customer_age                    100000 non-null  int64
 6   days_since_request              100000 non-null  float64
```

```
 7   intended_balcon_amount            100000 non-null  float64
 8   payment_type                      100000 non-null  object
 9   zip_count_4w                      100000 non-null  int64
 10  velocity_6h                       100000 non-null  float64
 11  velocity_24h                      100000 non-null  float64
 12  velocity_4w                       100000 non-null  float64
 13  bank_branch_count_8w              100000 non-null  int64
 14  date_of_birth_distinct_emails_4w  100000 non-null  int64
 15  employment_status                 100000 non-null  object
 16  credit_risk_score                 100000 non-null  int64
 17  email_is_free                     100000 non-null  int64
 18  housing_status                    100000 non-null  object
 19  phone_home_valid                  100000 non-null  int64
 20  phone_mobile_valid                100000 non-null  int64
 21  bank_months_count                 100000 non-null  int64
 22  has_other_cards                   100000 non-null  int64
 23  proposed_credit_limit             100000 non-null  float64
 24  foreign_request                   100000 non-null  int64
 25  source                            100000 non-null  object
 26  session_length_in_minutes         100000 non-null  float64
 27  device_os                         100000 non-null  object
 28  keep_alive_session                100000 non-null  int64
 29  device_distinct_emails_8w         100000 non-null  int64
 30  device_fraud_count                100000 non-null  int64
 31  month                             100000 non-null  int64
dtypes: float64(9), int64(18), object(5)
memory usage: 24.4+ MB
```

[10]:
```
credit_raw.nunique()
```

[10]:
```
fraud_bool                          2
income                              9
name_email_similarity           99989
prev_address_months_count         357
current_address_months_count      404
customer_age                        9
days_since_request              99886
intended_balcon_amount          99943
payment_type                        5
zip_count_4w                     5455
velocity_6h                     99983
velocity_24h                    99993
velocity_4w                     99972
bank_branch_count_8w             2140
date_of_birth_distinct_emails_4w   38
employment_status                   7
credit_risk_score                 509
```

```
email_is_free                  2
housing_status                 7
phone_home_valid               2
phone_mobile_valid             2
bank_months_count             32
has_other_cards                2
proposed_credit_limit         12
foreign_request                2
source                         2
session_length_in_minutes  99780
device_os                      5
keep_alive_session             2
device_distinct_emails_8w      4
device_fraud_count             1
month                          8
dtype: int64
```

Observasi:

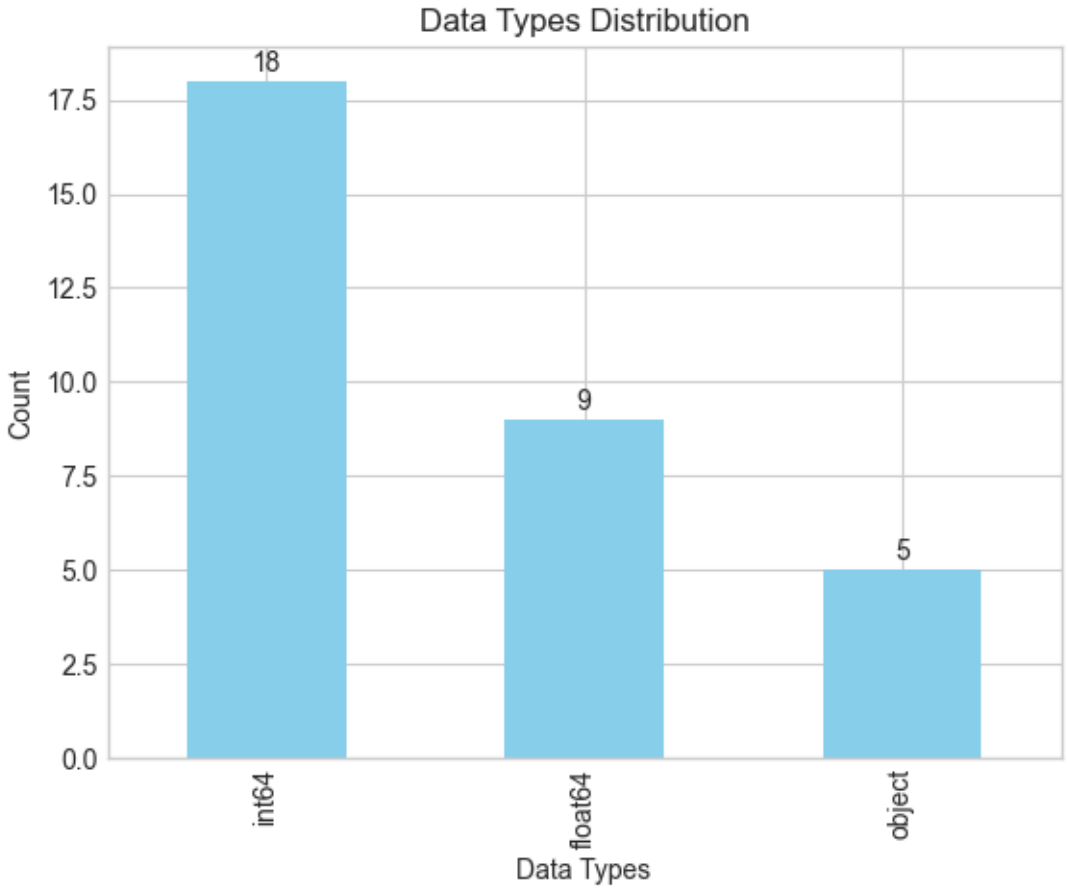
Berdasarkan dataset yang kita gunakan, untuk menggunakan algoritma *unsupervised learning* seperti *Local Outlier Factor (LOF)*, KMeans, dll., Anda perlu memastikan bahwa tipe data setiap kolom sesuai dengan kebutuhan algoritma tersebut. Berikut adalah beberapa langkah yang bisa Anda lakukan untuk membenarkan tipe data:

1. **Pengecekan dan Koreksi Tipe Data:**
   - Pastikan bahwa kolom-kolom dengan nilai unik yang rendah dan bertipe integer yang seharusnya bersifat kategorikal diubah menjadi tipe data kategorikal. Contohnya: `fraud_bool`, `employment_status`, `housing_status`, `phone_home_valid`, `phone_mobile_valid`, `has_other_cards`, `foreign_request`, dan `source`.
   - Pastikan bahwa kolom dengan nilai unik yang rendah dan bertipe float.
2. **Pengecekan dan Koreksi Nilai yang Anomali:**
   - Periksa nilai-nilai unik dalam kolom-kolom dengan nilai unik yang rendah untuk memastikan tidak ada nilai yang tidak valid atau anomali. Jika ditemukan, perbaiki atau hapus nilai tersebut.
   - Periksa kolom dengan nilai float atau integer apakah ada nilai yang diluar rentang yang masuk akal.

Pastikan Anda menyesuaikan langkah-langkah di atas sesuai dengan karakteristik dan kebutuhan data Anda. Jika Anda memiliki kolom lain yang memerlukan penanganan khusus, pastikan untuk memeriksa dan menyesuaikan tipe datanya. Setelah ini, Anda dapat melanjutkan dengan langkah-langkah *unsupervised learning* seperti PCA, LOF, dan KMeans seperti yang sudah dijelaskan sebelumnya.

[11]:
```
# Menghitung distribusi tipe data
data_type_counts = credit_raw.dtypes.astype(str).value_counts()

# Membuat plot
ax = data_type_counts.plot(kind='bar', color='skyblue')
```

```
# Menambahkan label pada setiap bar
for i, count in enumerate(data_type_counts):
    ax.text(i, count + 0.1, str(count), ha='center', va='bottom')

# Menambahkan judul dan label pada sumbu
plt.title("Data Types Distribution")
plt.xlabel("Data Types")
plt.ylabel("Count")

# Menampilkan plot
plt.show()
```



5.2 Data Preprocessing

[12]:
```
credit_clean = credit_raw.copy()
```

1. Check Missing Values

[13]:
```
credit_clean.isna().sum()
```

[13]:
```
fraud_bool                          0
income                              0
name_email_similarity               0
prev_address_months_count           0
current_address_months_count        0
customer_age                        0
days_since_request                  0
intended_balcon_amount              0
payment_type                        0
zip_count_4w                        0
velocity_6h                         0
velocity_24h                        0
velocity_4w                         0
bank_branch_count_8w                0
date_of_birth_distinct_emails_4w    0
employment_status                   0
credit_risk_score                   0
email_is_free                       0
housing_status                      0
phone_home_valid                    0
phone_mobile_valid                  0
bank_months_count                   0
has_other_cards                     0
proposed_credit_limit               0
foreign_request                     0
source                              0
session_length_in_minutes           0
device_os                           0
keep_alive_session                  0
device_distinct_emails_8w           0
device_fraud_count                  0
month                               0
dtype: int64
```

2. Cek Duplikat Data

[14]:
```
# cek duplicated value
credit_clean.duplicated().sum()
```

[14]:
```
0
```

3. Feature Selection

[15]:
```
# device_fraud_count : karena isinya 0 semua
credit_clean.drop('device_fraud_count', axis = 1, inplace=True)
```

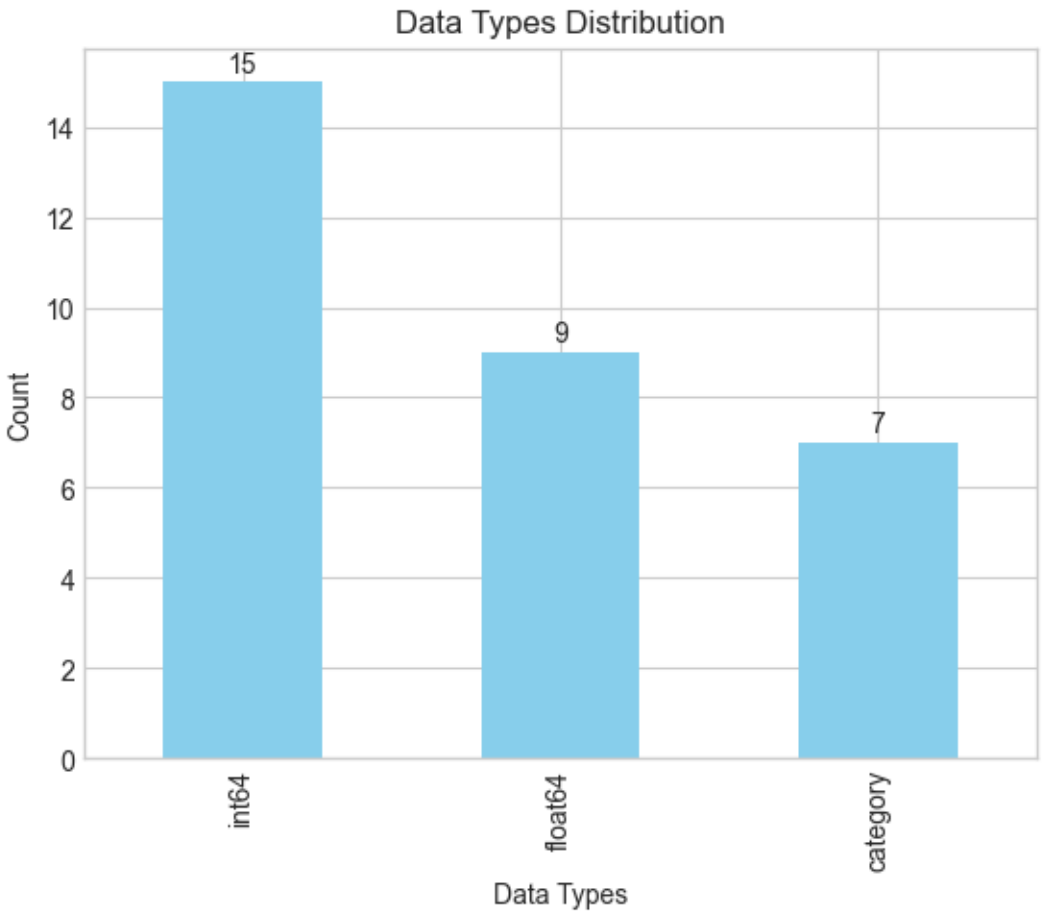
4. Data wrangling

[16]:
```
# Contoh untuk beberapa kolom yang bersifat kategorikal
categorical_columns = ['payment_type','employment_status',
                       'housing_status', 'has_other_cards',
                       'foreign_request','source', 'device_os']

# Ubah tipe data menjadi kategorikal
credit_clean[categorical_columns] = credit_clean[categorical_columns].
 ↪astype('category')
credit_clean.dtypes
```

[16]:
```
fraud_bool                          int64
income                            float64
name_email_similarity             float64
prev_address_months_count           int64
current_address_months_count        int64
customer_age                        int64
days_since_request                float64
intended_balcon_amount            float64
payment_type                     category
zip_count_4w                        int64
velocity_6h                       float64
velocity_24h                      float64
velocity_4w                       float64
bank_branch_count_8w                int64
date_of_birth_distinct_emails_4w    int64
employment_status                category
credit_risk_score                   int64
email_is_free                       int64
housing_status                   category
phone_home_valid                    int64
phone_mobile_valid                  int64
bank_months_count                   int64
has_other_cards                  category
proposed_credit_limit             float64
foreign_request                  category
source                           category
session_length_in_minutes         float64
device_os                        category
keep_alive_session                  int64
device_distinct_emails_8w           int64
month                               int64
dtype: object
```

[17]:
```
# Menghitung distribusi tipe data
data_type_counts = credit_clean.dtypes.astype(str).value_counts()

# Membuat plot
```

```
ax = data_type_counts.plot(kind='bar', color='skyblue')

# Menambahkan label pada setiap bar
for i, count in enumerate(data_type_counts):
    ax.text(i, count + 0.1, str(count), ha='center', va='bottom')

# Menambahkan judul dan label pada sumbu
plt.title("Data Types Distribution")
plt.xlabel("Data Types")
plt.ylabel("Count")

# Menampilkan plot
plt.show()
```



Observasi

Terdapat kumpulan data berisi fitur kategorikal. Meskipun ada berbagai metode yang tersedia untuk menangani data kategorikal, demi mencapai tujuan kami membandingkan regresi logistik dan mesin peningkatan gradien, kami telah memutuskan untuk mengubah fitur ini menjadi nilai

numerik.

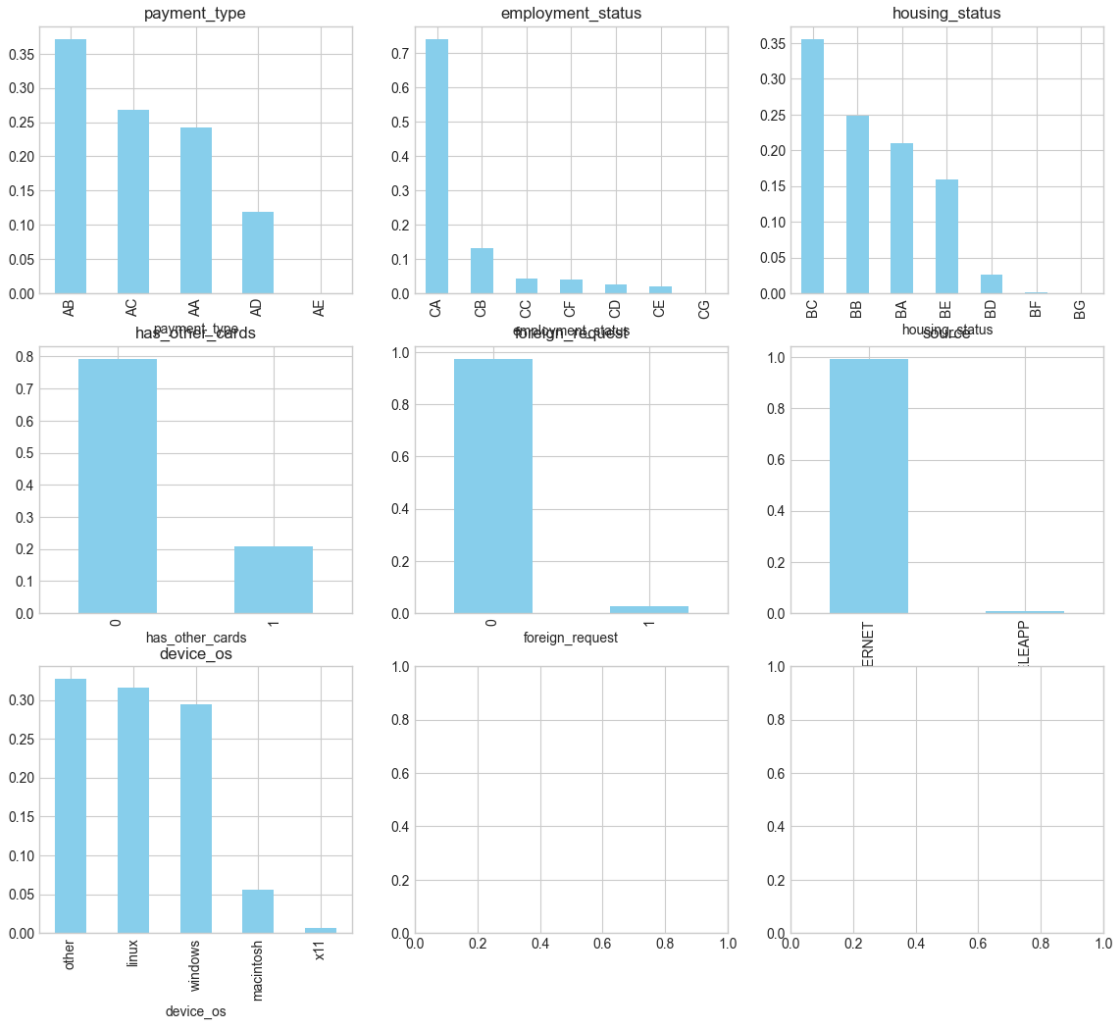
5. Ambil variabel dataset numerik

[18]:
```
credit_clean.dtypes
```

[18]:
```
fraud_bool                          int64
income                            float64
name_email_similarity             float64
prev_address_months_count           int64
current_address_months_count        int64
customer_age                        int64
days_since_request                float64
intended_balcon_amount            float64
payment_type                     category
zip_count_4w                        int64
velocity_6h                       float64
velocity_24h                      float64
velocity_4w                       float64
bank_branch_count_8w                int64
date_of_birth_distinct_emails_4w    int64
employment_status                category
credit_risk_score                   int64
email_is_free                       int64
housing_status                   category
phone_home_valid                    int64
phone_mobile_valid                  int64
bank_months_count                   int64
has_other_cards                  category
proposed_credit_limit             float64
foreign_request                  category
source                           category
session_length_in_minutes         float64
device_os                        category
keep_alive_session                  int64
device_distinct_emails_8w           int64
month                               int64
dtype: object
```

[19]:
```
# Memilih hanya 7 kolom kategori
categories = credit_clean.columns[credit_clean.dtypes == 'category']

# Menentukan jumlah baris dan kolom untuk subplot
num_plots = len(categories)
num_rows = 3
num_cols = math.ceil(num_plots / num_rows)

# Membuat subplot
```

```
fig, axes = plt.subplots(num_rows, num_cols, figsize=(14, 12))  # Sesuaikan␣
 ↪ukuran subplot

# Iterasi melalui kolom kategori dan membuat subplot
for i, x in enumerate(categories):
    row = i // num_cols
    col = i % num_cols

    # Memeriksa apakah kolom memiliki data sebelum membuat plot
    if not credit_clean[x].isnull().all():
        credit_clean[x].value_counts(normalize=True).plot(kind='bar',␣
 ↪color='skyblue', ax=axes[row, col])
        axes[row, col].set_title(x)

# Menampilkan subplot
plt.show()
```



6. Data Scaling

Visualisasi Plot Skewness:

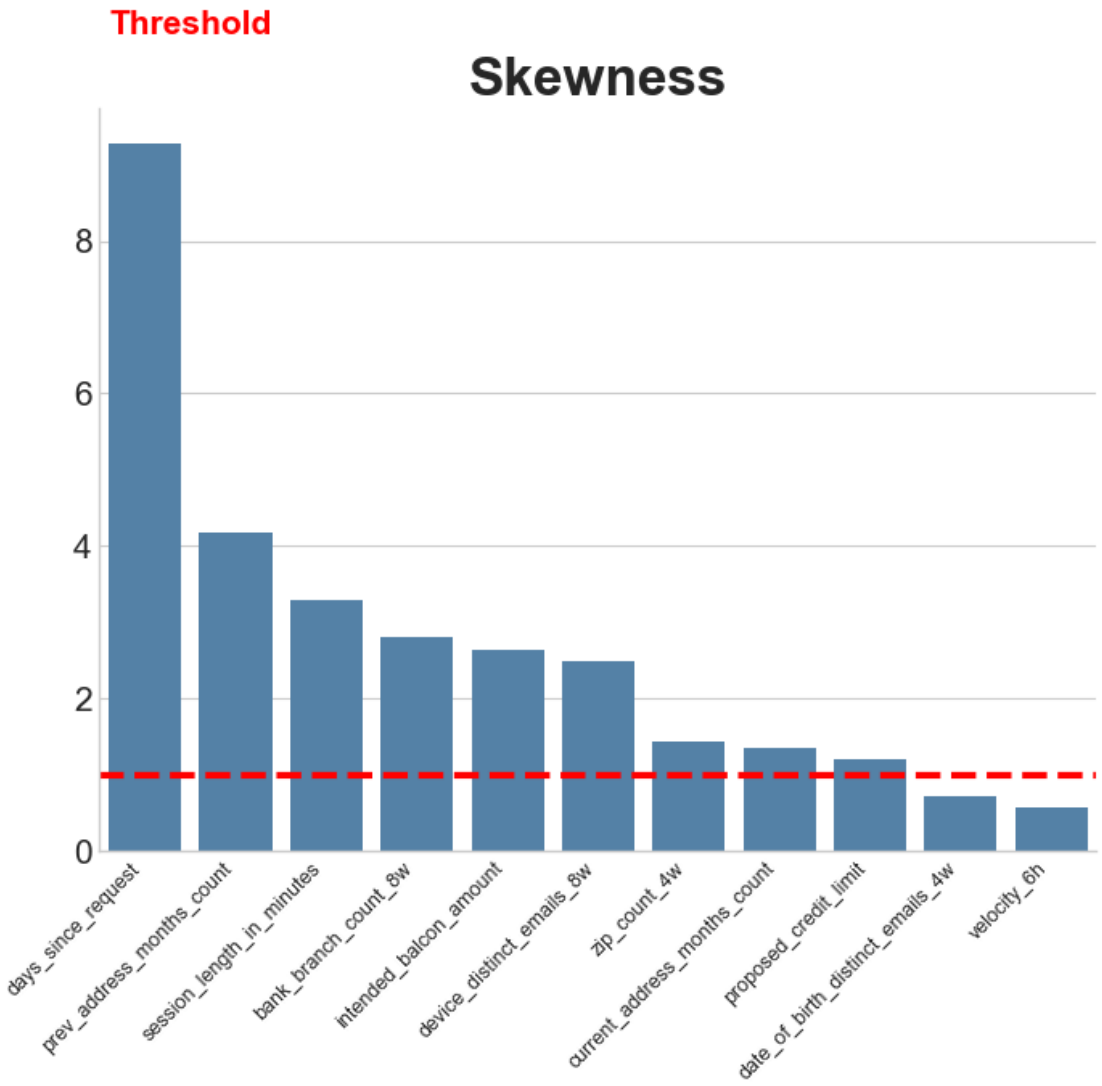
Sebelum melakukan scaling data, kita pertama-tama mengevaluasi skewness dari kolom-kolom numerik dalam dataset.

[20]:
```
# Pilih hanya kolom numerik
numerical_cols = credit_clean.select_dtypes(include=['int64', 'float64']).
 ↪columns

# Sisihkan kolom 'fraud_bool'
numerical_cols = numerical_cols[numerical_cols != 'fraud_bool']
```

[21]:
```
numerical_cols
```

[21]:
```
Index(['income', 'name_email_similarity', 'prev_address_months_count',
       'current_address_months_count', 'customer_age', 'days_since_request',
       'intended_balcon_amount', 'zip_count_4w', 'velocity_6h', 'velocity_24h',
       'velocity_4w', 'bank_branch_count_8w',
       'date_of_birth_distinct_emails_4w', 'credit_risk_score',
       'email_is_free', 'phone_home_valid', 'phone_mobile_valid',
       'bank_months_count', 'proposed_credit_limit',
       'session_length_in_minutes', 'keep_alive_session',
       'device_distinct_emails_8w', 'month'],
      dtype='object')
```

[22]:
```
plt.figure(figsize=(8, 6))
skew_features = credit_clean[numerical_cols].apply(lambda x: skew(x))
skew_features = skew_features[skew_features > 0.5].sort_values(ascending=False)␣
 ↪# treshold 0.5
ax = sns.barplot(x=skew_features.index, y=skew_features.values,␣
 ↪color='SteelBlue')
ax.set_ylabel('', fontsize=20)
ax.set_xlabel('', fontsize=20)
ax.tick_params(axis='both', labelsize=15)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right', fontsize=9)
ax.axhline(y=1, color='red', linestyle='--', linewidth=3)
ax.set_title('Skewness', ha = 'center', weight='bold', fontsize=24)
ax.text(0.01, 1.1, 'Threshold', color='red', transform=ax.transAxes,␣
 ↪fontsize=15, weight='bold')
sns.despine()
plt.gca().set_facecolor('white')
plt.show()
```



Observasi:

- Skewness mengukur tingkat asimetri distribusi data. Nilai skewness di atas 0.5 menunjukkan adanya kemiringan (asimetri) dalam distribusi data.
- Plot bar di atas menampilkan nilai skewness untuk setiap kolom numerik yang memiliki skewness di atas threshold 0.5.
- Garis merah putus-putus pada nilai skewness 1 berfungsi sebagai batas threshold; kolom-kolom di atas batas ini dianggap memiliki skewness tinggi. Kolom-kolom dengan skewness di atas threshold dapat mengindikasikan adanya ketidaknormalan dalam distribusi data. Kolom-kolom dengan skewness tinggi mungkin memerlukan teknik scaling atau transformasi data untuk meningkatkan kinerja model.
- Nilai skewness positif menunjukkan ekor distribusi yang lebih panjang di sisi kanan, sementara nilai negatif menunjukkan ekor distribusi yang lebih panjang di sisi kiri.

Melakukan scaling data

Setelah memahami skewness data, kita dapat melanjutkan dengan tahap kedua, yaitu melakukan scaling data menggunakan z-score normalization. Hal ini dilakukan untuk menghilangkan penyimpangan skewness dan memastikan distribusi data mendekati distribusi normal standar. Rangkuman kalimat untuk tahap kedua dapat mencakup proses scaling data menggunakan z-score normalization untuk memastikan konsistensi dalam rentang nilai dan meningkatkan keterbacaan model.

[Optional] Math Behind Z-Score Normalization (StandardScaler) in sklearn:

Z-Score Normalization (StandardScaler) diimplementasikan dalam scikit-learn dengan menggunakan kelas `StandardScaler`. Formula matematis dari Z-Score Normalization adalah sebagai berikut:

$$Z = \frac{X - \mu}{\sigma}$$

- (Z) adalah nilai hasil normalisasi (z-score) dari suatu observasi.
- (X) adalah nilai observasi asli.
- μ adalah rata-rata dari seluruh nilai observasi dalam suatu fitur.
- σ adalah deviasi standar dari seluruh nilai observasi dalam suatu fitur.

Penjelasan: 1. **Mean (μ):** - Rata-rata dari seluruh nilai dalam suatu fitur. - Dihitung dengan menjumlahkan semua nilai dan membaginya dengan jumlah total nilai.

2. **Standard Deviation (σ):**
   - Deviasi standar mengukur sejauh mana nilai-nilai dalam suatu fitur tersebar dari rata-rata.
   - Dihitung dengan menghitung akar kuadrat dari varians.
3. **Z-Score ((Z)):**
   - Memberikan informasi tentang seberapa jauh nilai observasi berada dari rata-rata dalam satuan deviasi standar.
   - Jika (Z) positif, itu berarti nilai observasi berada di atas rata-rata. Jika (Z) negatif, itu berarti nilai observasi berada di bawah rata-rata.

Alasan Penggunaan Scaling pada Reduksi Dimensi: - Konsistensi Skala: - Algoritma reduksi dimensi, seperti PCA (Principal Component Analysis), sensitif terhadap skala variabel. Scaling memastikan bahwa semua variabel berada pada rentang yang serupa, sehingga tidak ada variabel yang mendominasi proses reduksi dimensi.

- **Interpretasi yang Mudah:**
  - Skala yang seragam memudahkan interpretasi bobot relatif dari masing-masing komponen utama setelah reduksi dimensi.
- **Konvergensi Algoritma:**
  - Beberapa algoritma reduksi dimensi, seperti gradient descent pada tugas t-SNE, dapat lebih cepat mencapai konvergensi dengan data yang telah di-scaled.
- **Mengatasi Skewness dan Outlier:**
  - Scaling dapat membantu mengurangi dampak skewness dan nilai ekstrem (outlier) dalam data, meningkatkan keandalan analisis reduksi dimensi.

Dengan melakukan Z-Score Normalization sebelum menerapkan metode reduksi dimensi, kita memastikan bahwa seluruh variabel berada pada skala yang seragam, memfasilitasi interpretasi

dan meningkatkan performa algoritma reduksi dimensi.

[23]:
```
credit_num = credit_clean[numerical_cols]
```

[24]:
```
credit_num.head()
```

[24]:

	income	name_email_similarity	prev_address_months_count	current_address_months_count	customer_age	days_since_request	intended_balcon_amount	zip_count_4w	velocity_6h	velocity_24h	…	credit_risk_score	email_is_free	phone_home_valid	phone_mobile_valid	bank_months_count	proposed_credit_limit	session_length_in_minutes	keep_alive_session	device_distinct_emails_8w	month
0	0.9	0.615786	-1	51	40	0.010337	-0.699550	2659	11007.708830	7448.797416	…	253	1	1	1	9	1500.0	13.079806	0	1	1
1	0.6	0.072461	-1	125	20	0.006974	-0.959468	5599	6304.262836	4373.093592	…	112	1	1	0	-1	200.0	6.666551	0	1	4
2	0.5	0.076592	-1	37	50	0.006319	-1.219933	1352	1357.413551	6667.029662	…	199	1	1	1	28	1500.0	4.492668	1	2	0
3	0.3	0.255433	23	19	20	0.009267	3.242242	3743	5234.262304	6139.489579	…	63	1	0	1	16	200.0	2.321885	1	1	3
4	0.9	0.192362	-1	34	30	0.019240	-1.131560	546	3589.153453	4574.395571	…	275	1	1	1	1	1000.0	4.845147	1	1	5

```
[5 rows x 23 columns]
```

Menggunakan Z-score standardization untuk scaling dataset numerik dengan fungsi StandardScaler() pada library sklearn:

[25]:
```
credit_scaled = StandardScaler().fit_transform(credit_num.values)
credit_scaled
```

[25]:
```
array([[ 1.12580168,  0.45453446, -0.38732389, …, -1.11467683,
        -0.12426586, -1.03803432],
       [ 0.09476593, -1.41007296, -0.38732389, …, -1.11467683,
        -0.12426586,  0.31203178],
       [-0.24891265, -1.39589477, -0.38732389, …,  0.89712101,
         4.87436093, -1.48805636],
       …,
       [ 0.43844452, -1.54122323, -0.38732389, …,  0.89712101,
        -0.12426586,  0.76205381],
       [-0.24891265, -1.29201166, -0.38732389, …, -1.11467683,
         4.87436093, -1.48805636],
       [ 0.7821231 , -0.40150292,  0.37934726, …,  0.89712101,
        -0.12426586, -0.58801229]])
```

[26]:
```
credit_scaled = pd.DataFrame(credit_scaled, columns=[numerical_cols])
```

[27]:
```
credit_scaled.head()
```

[27]:

	income	name_email_similarity	prev_address_months_count	current_address_months_count	customer_age	days_since_request	intended_balcon_amount	zip_count_4w	velocity_6h	velocity_24h	…
0	1.125802	0.454534	-0.387324	-0.431527	0.459337	-0.188833	-0.442254	1.075569	1.790845	1.827162	…
1	0.094766	-1.410073	-0.387324	0.406739	-1.165255	-0.189457	-0.455380	4.005273	0.226647	-0.256489	…
2	-0.248913	-1.395895	-0.387324	-0.590118	1.271634	-0.189578	-0.468534	-0.226854	-1.418498	1.297549	…
3	-0.936270	-0.782141	0.170255	-0.794021	-1.165255	-0.189032	-0.243188	2.155773	-0.129197	0.940165	…
4	1.125802	-0.998590	-0.387324	-0.624102	-0.352959	-0.187183	-0.464071	-1.030032	-0.676301	-0.120116	…

```
   credit_risk_score email_is_free phone_home_valid phone_mobile_valid  \
0           1.623733      0.916752         1.233515           0.359129
1          -0.324384      0.916752         1.233515          -2.784511
2           0.877646      0.916752         1.233515           0.359129
3          -1.001389      0.916752        -0.810691           0.359129
4           1.927695      0.916752         1.233515           0.359129

   bank_months_count proposed_credit_limit session_length_in_minutes  \
0          -0.144869               1.860207                  0.655308
1          -0.965382              -0.673720                 -0.118348
2           1.414106               1.860207                 -0.380592
3           0.429490              -0.673720                 -0.642461
4          -0.801279               0.885620                 -0.338071

   keep_alive_session device_distinct_emails_8w     month
0           -1.114677                -0.124266 -1.038034
1           -1.114677                -0.124266  0.312032
2            0.897121                 4.874361 -1.488056
3            0.897121                -0.124266 -0.137990
4            0.897121                -0.124266  0.762054

[5 rows x 23 columns]
```

Kita telah mengetahui sebelumnya bahwa korelasi dalam data membuat beberapa variabel menjadi mubazir. Pasca-PCA, matriks Z harus berupa matriks yang setiap dimensinya tidak berkorelasi.

Diskusi: Manakah dari plot berikut yang menunjukkan variabel yang tidak berkorelasi?

Variabel *Age* dan *Sale Price* pada `Sale Price of Vehicle` memiliki korelasi negatif. Jika Anda menggambar garis regresi di atasnya, garis tersebut menunjukkan kemiringan ke bawah, dan korelasi antara *Age* dan *Sale Price* akan bernilai mendekati -1.

Variabel *temperature* dan *presure* pada `Logistic Machinery` memiliki korelasi positif. Jika Anda memplot garis regresi di atasnya, garis tersebut menunjukkan kemiringan ke atas, dan korelasi antara *temperature* dan *presure* akan bernilai mendekati 1.

Secara intuitif, kita secara visual menyimpulkan bahwa variabel `Blind Tasting` tidak berkorelasi karena mengetahui nilai x tidak memberi kita indikasi berapa nilai y.

Sekarang jika kita mengikuti contoh di bagian **pca1.png** (dari Prinsip dan Motivasi) dan memilih kombinasi sumbu baru untuk `Logistic Machinery` (yang merupakan garis ungu berlabel `v1` dan `v2` di bawah ini gambar), maka distribusi suhu dan tekanan menjadi tidak berkorelasi berdasarkan dua sumbu baru:

5.2.1 [Opsional] Nilai Eigen dan Vektor Eigen

Jika Anda memiliki paparan terbatas pada Aljabar Matriks - silakan lewati sub-bagian ini. Konsep yang disajikan dalam sub-bagian ini tidak akan dinilai, pada kenyataannya R memiliki fungsi bawaan yang sangat berguna yang membantu Anda menghitung vektor eigen dan nilai eigen sehingga pengerjaan bagian dalam sebagian besar bersifat opsional bagi praktisi mana pun.

Dalam teori aljabar linier dan matriks, kita mempelajari bahwa sebuah vektor jika dikalikan dengan matriks, akan mengubah arahnya. Ketika kita mengambil sebuah vektor (2,3) dan mengalikannya dengan skala, katakanlah 2, maka vektor yang dihasilkan adalah (4,6) - skalar menskalakan vektor tersebut dengan besaran 2.

Sekarang jika kita mengambil vektor yang sama dan mengalikannya dengan matriks, vektor yang dikerjakan akan berubah arah:

$$\begin{pmatrix} 11 \\ 16 \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

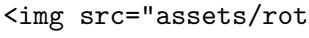
Secara sederhana, kita dapat memandang operasi di atas sebagai suatu matriks yang mentransformasi vektor menjadi vektor lain: vektor hasilnya (umumnya) memiliki arah dan panjang yang berbeda dari vektor asli. Namun, terdapat beberapa pengecualian penting terhadap hal ini:

- Matriks yang bertindak pada suatu vektor tanpa mengubahnya sama sekali disebut **matriks identitas**. Matriks identitas adalah matriks persegi di mana semua elemen diagonal utama bernilai satu dan semua elemen lainnya bernilai nol.

  Matriks (2,3) yang dikalikan dengan matriks identitas berukuran 2 akan menghasilkan matriks (2,3); baik arah maupun besarnya tidak berubah.

- Matriks yang memutar setiap vektor sejauh sudut tetap disebut **matriks rotasi**. Arah vektor berubah, tetapi tidak besarnya. Ambil matriks A di bawah ini dan kalikan dengan (-3, 2) untuk mendapatkan (3, -2). Jika Anda menggambar kedua vektor tersebut, Anda akan melihat bahwa keduanya memiliki panjang yang sama dengan vektor kedua diputar sejauh 180 derajat dari yang pertama. Kalikan A dengan (-20, 18) dan vektor akan berubah menjadi (20, -18), lagi-lagi diputar sejauh 180 derajat. Ternyata, A adalah matriks rotasi yang akan mengubah suatu vektor sejauh 180 derajat tanpa mengubah besarnya.

  $$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} -3 \\ 2 \end{pmatrix}$$



- Untuk sebagian besar matriks, terdapat vektor tertentu yang disebut **vektor eigen** yang arahnya tidak berubah (kecuali jika dikalikan dengan -1, maka arahnya terbalik) ketika dioperasikan oleh matriks. Hasil perkalian matriks dengan vektor tersebut, vektor eigen, memiliki efek yang sama dengan mengalikan vektor tersebut dengan suatu skalar. Skalar ini adalah konstanta dan secara resmi disebut sebagai **nilai eigen**. Ketika kita mengalikan matriks A dengan vektor eigen x, hasilnya akan berupa konstanta λ dikalikan dengan x. Persamaannya adalah:

  $Ax = \lambda x$, di mana angka λ adalah nilai eigen dari A.

$$\begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 12 \\ 8 \end{pmatrix} = 4 \begin{pmatrix} 3 \\ 2 \end{pmatrix} 2 \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix} \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 24 \\ 16 \end{pmatrix} = 4 \begin{pmatrix} 6 \\ 4 \end{pmatrix}$$

Perhatikan bagaimana vektor eigen yang diperbesar tetap menjadi vektor eigen.

5.2.2 Intuisi dan Praktik: PCA

Ingat bahwa sebelumnya kita menyebutkan secara intuitif, tujuan dari PCA adalah memilih sumbu baru (dimensi baru) untuk plot kita, dan dimensi baru ini dari data kita disebut sebagai **komponen utama**? Dimensi baru ini seharusnya tidak memiliki korelasi satu sama lain (dalam konteks dimensionalitas, korelasi menyebabkan redundansi!). Jika kita memiliki matriks kovariansi:

[28]:
```
a = np.matrix('1.04, 0.77; 0.77, 0.68')
a
```

[28]:
```
matrix([[1.04, 0.77],
        [0.77, 0.68]])
```

Ingat dari course Practical Statistics bahwa ini berarti varians dari variabel pertama adalah 1,04, varians dari variabel kedua adalah 0,68, dan kovarians antara keduanya adalah 0,77.

Sekarang `A` adalah matriks simetris persegi, sehingga dapat didiagonalisasi dengan memilih sistem koordinat baru yang diberikan oleh eigenvector-nya, eigenvalue yang sesuai kemudian akan menggantikan nilai-nilai di diagonal. Dengan menggunakan sistem koordinat baru ini, matriks kovarians A kita akan terlihat seperti:

[29]:
```
np.cov(a)
```

[29]:
```
array([[0.03645, 0.01215],
       [0.01215, 0.00405]])
```

[30]:
```
w,v = eig(a)
print('E-value:', w)
print('E-vector', v)
```

```
E-value: [1.65075913 0.06924087]
E-vector [[ 0.78346326 -0.6214381 ]
 [ 0.6214381   0.78346326]]
```

5.2.3 Aplikasi PCA: Reduksi Dimensi pada Data credit

Pertama, mari kita lihat matriks kovariansi dari dataset `credit_scaled`:

7. Cek korelasi data

[31]:
```
# cek kembali data yang sudah di pca
credit_scaled.cov()
```

[31]:

	income	name_email_similarity	\
income	1.000010	-0.048731	
name_email_similarity	-0.048731	1.000010	
prev_address_months_count	0.008809	-0.009854	
current_address_months_count	-0.017391	0.038234	
customer_age	0.137962	-0.066315	
days_since_request	-0.012759	-0.006397	
intended_balcon_amount	0.045168	0.035757	

zip_count_4w	-0.079997	0.025874
velocity_6h	-0.100350	0.027620
velocity_24h	-0.106405	0.038162
velocity_4w	-0.108241	0.042983
bank_branch_count_8w	0.005063	-0.003919
date_of_birth_distinct_emails_4w	-0.075545	0.036560
credit_risk_score	0.195343	0.033618
email_is_free	-0.013721	-0.075405
phone_home_valid	0.012676	0.003814
phone_mobile_valid	0.010555	0.015459
bank_months_count	-0.001819	-0.011884
proposed_credit_limit	0.135009	0.053971
session_length_in_minutes	-0.051490	0.000112
keep_alive_session	-0.058315	0.031243
device_distinct_emails_8w	-0.002465	-0.029268
month	0.119413	-0.048978

	prev_address_months_count
income	0.008809
name_email_similarity	-0.009854
prev_address_months_count	1.000010
current_address_months_count	-0.270593
customer_age	-0.078717
days_since_request	0.080489
intended_balcon_amount	0.002354
zip_count_4w	-0.027014
velocity_6h	0.004293
velocity_24h	0.017160
velocity_4w	0.009710
bank_branch_count_8w	-0.033813
date_of_birth_distinct_emails_4w	0.064815
credit_risk_score	-0.032034
email_is_free	-0.023588
phone_home_valid	-0.045306
phone_mobile_valid	0.010982
bank_months_count	-0.048139
proposed_credit_limit	-0.042681
session_length_in_minutes	0.046938
keep_alive_session	0.041924
device_distinct_emails_8w	0.004498
month	-0.005397

	current_address_months_count	customer_age
income	-0.017391	0.137962
name_email_similarity	0.038234	-0.066315
prev_address_months_count	-0.270593	-0.078717
current_address_months_count	1.000010	0.159740

```
customer_age                                 0.159740     1.000010
days_since_request                          -0.061145    -0.033966
intended_balcon_amount                       0.042259    -0.027465
zip_count_4w                                 0.040709    -0.016129
velocity_6h                                  0.021910    -0.028680
velocity_24h                                 0.015533    -0.006880
velocity_4w                                  0.022781    -0.004876
bank_branch_count_8w                         0.046643     0.041201
date_of_birth_distinct_emails_4w            -0.185124    -0.437193
credit_risk_score                            0.115030     0.192823
email_is_free                               -0.073249     0.014226
phone_home_valid                             0.115907     0.188598
phone_mobile_valid                          -0.097486    -0.150076
bank_months_count                            0.066281     0.012682
proposed_credit_limit                        0.149883     0.175815
session_length_in_minutes                   -0.027386     0.029887
keep_alive_session                          -0.059696    -0.051321
device_distinct_emails_8w                    0.004002     0.042828
month                                       -0.019638     0.007649

                            days_since_request  intended_balcon_amount  \
income                                -0.012759                0.045168
name_email_similarity                 -0.006397                0.035757
prev_address_months_count              0.080489                0.002354
current_address_months_count          -0.061145                0.042259
customer_age                          -0.033966               -0.027465
days_since_request                     1.000010                0.022183
intended_balcon_amount                 0.022183                1.000010
zip_count_4w                           0.006127                0.007608
velocity_6h                            0.033093                0.045158
velocity_24h                           0.021932                0.070400
velocity_4w                            0.020836                0.059332
bank_branch_count_8w                  -0.019784                0.100776
date_of_birth_distinct_emails_4w       0.015786               -0.012100
credit_risk_score                     -0.085697               -0.014962
email_is_free                          0.007625               -0.020873
phone_home_valid                      -0.043488                0.006145
phone_mobile_valid                     0.009163                0.042329
bank_months_count                     -0.027671                0.175411
proposed_credit_limit                 -0.064816                0.033850
session_length_in_minutes              0.049374                0.018501
keep_alive_session                     0.000793               -0.006148
device_distinct_emails_8w              0.011413               -0.024972
month                                 -0.017966               -0.049242

                            zip_count_4w  velocity_6h  velocity_24h  …  \
income                       -0.079997    -0.100350    -0.106405  …
```

name_email_similarity	0.025874	0.027620	0.038162	…
prev_address_months_count	-0.027014	0.004293	0.017160	…
current_address_months_count	0.040709	0.021910	0.015533	…
customer_age	-0.016129	-0.028680	-0.006880	…
days_since_request	0.006127	0.033093	0.021932	…
intended_balcon_amount	0.007608	0.045158	0.070400	…
zip_count_4w	1.000010	0.133936	0.198823	…
velocity_6h	0.133936	1.000010	0.463804	…
velocity_24h	0.198823	0.463804	1.000010	…
velocity_4w	0.298165	0.400854	0.540729	…
bank_branch_count_8w	0.003664	0.020905	0.041774	…
date_of_birth_distinct_emails_4w	0.121708	0.122253	0.149496	…
credit_risk_score	-0.101330	-0.151244	-0.153062	…
email_is_free	0.024346	0.019873	0.026393	…
phone_home_valid	-0.038401	-0.028620	-0.046243	…
phone_mobile_valid	0.007479	-0.015739	-0.032518	…
bank_months_count	0.049366	0.017426	0.021521	…
proposed_credit_limit	-0.013902	-0.035302	0.003048	…
session_length_in_minutes	0.043676	0.053628	0.064216	…
keep_alive_session	0.017006	0.013564	0.007827	…
device_distinct_emails_8w	0.026515	0.031013	0.036795	…
month	-0.281729	-0.410802	-0.550383	…

	credit_risk_score	email_is_free	\
income	0.195343	-0.013721	
name_email_similarity	0.033618	-0.075405	
prev_address_months_count	-0.032034	-0.023588	
current_address_months_count	0.115030	-0.073249	
customer_age	0.192823	0.014226	
days_since_request	-0.085697	0.007625	
intended_balcon_amount	-0.014962	-0.020873	
zip_count_4w	-0.101330	0.024346	
velocity_6h	-0.151244	0.019873	
velocity_24h	-0.153062	0.026393	
velocity_4w	-0.169070	0.046455	
bank_branch_count_8w	-0.018674	-0.006948	
date_of_birth_distinct_emails_4w	-0.156319	0.034843	
credit_risk_score	1.000010	-0.016162	
email_is_free	-0.016162	1.000010	
phone_home_valid	-0.013571	-0.019829	
phone_mobile_valid	-0.002204	0.030560	
bank_months_count	-0.060329	0.006312	
proposed_credit_limit	0.622599	0.000535	
session_length_in_minutes	-0.029040	0.041217	
keep_alive_session	-0.055943	-0.036244	
device_distinct_emails_8w	-0.028274	0.004740	
month	0.176883	-0.058587	

	phone_home_valid	phone_mobile_valid	\
income	0.012676	0.010555	
name_email_similarity	0.003814	0.015459	
prev_address_months_count	-0.045306	0.010982	
current_address_months_count	0.115907	-0.097486	
customer_age	0.188598	-0.150076	
days_since_request	-0.043488	0.009163	
intended_balcon_amount	0.006145	0.042329	
zip_count_4w	-0.038401	0.007479	
velocity_6h	-0.028620	-0.015739	
velocity_24h	-0.046243	-0.032518	
velocity_4w	-0.047811	-0.031044	
bank_branch_count_8w	0.064747	-0.015609	
date_of_birth_distinct_emails_4w	-0.147730	0.102194	
credit_risk_score	-0.013571	-0.002204	
email_is_free	-0.019829	0.030560	
phone_home_valid	1.000010	-0.275398	
phone_mobile_valid	-0.275398	1.000010	
bank_months_count	0.057916	0.005845	
proposed_credit_limit	-0.014093	-0.025538	
session_length_in_minutes	-0.044417	-0.004365	
keep_alive_session	0.054068	0.023697	
device_distinct_emails_8w	-0.004077	-0.046356	
month	0.048598	0.035254	

	bank_months_count	proposed_credit_limit	\
income	-0.001819	0.135009	
name_email_similarity	-0.011884	0.053971	
prev_address_months_count	-0.048139	-0.042681	
current_address_months_count	0.066281	0.149883	
customer_age	0.012682	0.175815	
days_since_request	-0.027671	-0.064816	
intended_balcon_amount	0.175411	0.033850	
zip_count_4w	0.049366	-0.013902	
velocity_6h	0.017426	-0.035302	
velocity_24h	0.021521	0.003048	
velocity_4w	0.035290	0.038256	
bank_branch_count_8w	0.144942	0.003451	
date_of_birth_distinct_emails_4w	-0.005195	-0.072189	
credit_risk_score	-0.060329	0.622599	
email_is_free	0.006312	0.000535	
phone_home_valid	0.057916	-0.014093	
phone_mobile_valid	0.005845	-0.025538	
bank_months_count	1.000010	-0.026512	
proposed_credit_limit	-0.026512	1.000010	
session_length_in_minutes	0.013237	0.003169	

keep_alive_session	0.006517	-0.053385
device_distinct_emails_8w	0.004568	-0.002564
month	-0.027446	-0.038215

	session_length_in_minutes	keep_alive_session
income	-0.051490	-0.058315
name_email_similarity	0.000112	0.031243
prev_address_months_count	0.046938	0.041924
current_address_months_count	-0.027386	-0.059696
customer_age	0.029887	-0.051321
days_since_request	0.049374	0.000793
intended_balcon_amount	0.018501	-0.006148
zip_count_4w	0.043676	0.017006
velocity_6h	0.053628	0.013564
velocity_24h	0.064216	0.007827
velocity_4w	0.076175	0.027273
bank_branch_count_8w	0.005549	0.011107
date_of_birth_distinct_emails_4w	-0.033820	0.061008
credit_risk_score	-0.029040	-0.055943
email_is_free	0.041217	-0.036244
phone_home_valid	-0.044417	0.054068
phone_mobile_valid	-0.004365	0.023697
bank_months_count	0.013237	0.006517
proposed_credit_limit	0.003169	-0.053385
session_length_in_minutes	1.000010	-0.056252
keep_alive_session	-0.056252	1.000010
device_distinct_emails_8w	0.078448	-0.053139
month	-0.076507	-0.027120

	device_distinct_emails_8w	month
income	-0.002465	0.119413
name_email_similarity	-0.029268	-0.048978
prev_address_months_count	0.004498	-0.005397
current_address_months_count	0.004002	-0.019638
customer_age	0.042828	0.007649
days_since_request	0.011413	-0.017966
intended_balcon_amount	-0.024972	-0.049242
zip_count_4w	0.026515	-0.281729
velocity_6h	0.031013	-0.410802
velocity_24h	0.036795	-0.550383
velocity_4w	0.045437	-0.847785
bank_branch_count_8w	0.003071	-0.036585
date_of_birth_distinct_emails_4w	-0.021827	-0.238472
credit_risk_score	-0.028274	0.176883
email_is_free	0.004740	-0.058587
phone_home_valid	-0.004077	0.048598
phone_mobile_valid	-0.046356	0.035254

```
bank_months_count                          0.004568 -0.027446
proposed_credit_limit                     -0.002564 -0.038215
session_length_in_minutes                  0.078448 -0.076507
keep_alive_session                        -0.053139 -0.027120
device_distinct_emails_8w                  1.000010 -0.047966
month                                     -0.047966  1.000010

[23 rows x 23 columns]
```

Terdapat 23 kolom numerik yang sudah dilakukan scaling menggunakan StandardScaler

[32]:
```
# sebelum di scaling
credit_num.cov().round(2)
```

[32]:
```
                                  income  name_email_similarity  \
income                              0.08                  -0.00
name_email_similarity              -0.00                   0.08
prev_address_months_count           0.11                  -0.12
current_address_months_count       -0.45                   0.98
customer_age                        0.49                  -0.24
days_since_request                 -0.02                  -0.01
intended_balcon_amount              0.26                   0.21
zip_count_4w                      -23.36                   7.57
velocity_6h                       -87.80                  24.20
velocity_24h                      -45.70                  16.41
velocity_4w                       -29.18                  11.60
bank_branch_count_8w                0.67                  -0.52
date_of_birth_distinct_emails_4w   -0.11                   0.05
credit_risk_score                   4.11                   0.71
email_is_free                      -0.00                  -0.01
phone_home_valid                    0.00                   0.00
phone_mobile_valid                  0.00                   0.00
bank_months_count                  -0.01                  -0.04
proposed_credit_limit              20.15                   8.07
session_length_in_minutes          -0.12                   0.00
keep_alive_session                 -0.01                   0.00
device_distinct_emails_8w          -0.00                  -0.00
month                               0.08                  -0.03

                                  prev_address_months_count  \
income                                                 0.11
name_email_similarity                                 -0.12
prev_address_months_count                           1852.74
current_address_months_count                       -1028.18
customer_age                                         -41.71
days_since_request                                    18.69
intended_balcon_amount                                 2.01
zip_count_4w                                       -1166.86
```

velocity_6h	555.58
velocity_24h	1090.26
velocity_4w	387.23
bank_branch_count_8w	-662.61
date_of_birth_distinct_emails_4w	14.09
credit_risk_score	-99.80
email_is_free	-0.51
phone_home_valid	-0.95
phone_mobile_valid	0.15
bank_months_count	-25.25
proposed_credit_limit	-942.52
session_length_in_minutes	16.75
keep_alive_session	0.90
device_distinct_emails_8w	0.04
month	-0.52

	current_address_months_count	customer_age	\
income	-0.45	0.49	
name_email_similarity	0.98	-0.24	
prev_address_months_count	-1028.18	-41.71	
current_address_months_count	7792.98	173.60	
customer_age	173.60	151.56	
days_since_request	-29.12	-2.26	
intended_balcon_amount	73.87	-6.70	
zip_count_4w	3606.32	-199.26	
velocity_6h	5815.84	-1061.68	
velocity_24h	2024.10	-125.03	
velocity_4w	1863.30	-55.62	
bank_branch_count_8w	1874.55	230.92	
date_of_birth_distinct_emails_4w	-82.54	-27.18	
credit_risk_score	734.96	171.81	
email_is_free	-3.22	0.09	
phone_home_valid	5.01	1.14	
phone_mobile_valid	-2.74	-0.59	
bank_months_count	71.31	1.90	
proposed_credit_limit	6788.15	1110.43	
session_length_in_minutes	-20.04	3.05	
keep_alive_session	-2.62	-0.31	
device_distinct_emails_8w	0.07	0.11	
month	-3.85	0.21	

	days_since_request	intended_balcon_amount	\
income	-0.02	0.26	
name_email_similarity	-0.01	0.21	
prev_address_months_count	18.69	2.01	
current_address_months_count	-29.12	73.87	
customer_age	-2.26	-6.70	

days_since_request	29.11	2.37
intended_balcon_amount	2.37	392.10
zip_count_4w	33.17	151.17
velocity_6h	536.86	2688.80
velocity_24h	174.66	2057.74
velocity_4w	104.15	1088.53
bank_branch_count_8w	-48.59	908.49
date_of_birth_distinct_emails_4w	0.43	-1.21
credit_risk_score	-33.46	-21.44
email_is_free	0.02	-0.21
phone_home_valid	-0.11	0.06
phone_mobile_valid	0.02	0.27
bank_months_count	-1.82	42.33
proposed_credit_limit	-179.40	343.88
session_length_in_minutes	2.21	3.04
keep_alive_session	0.00	-0.06
device_distinct_emails_8w	0.01	-0.10
month	-0.22	-2.17

	zip_count_4w	velocity_6h	velocity_24h	\
income	-23.36	-87.80	-45.70	
name_email_similarity	7.57	24.20	16.41	
prev_address_months_count	-1166.86	555.58	1090.26	
current_address_months_count	3606.32	5815.84	2024.10	
customer_age	-199.26	-1061.68	-125.03	
days_since_request	33.17	536.86	174.66	
intended_balcon_amount	151.17	2688.80	2057.74	
zip_count_4w	1007050.97	404153.34	294516.86	
velocity_6h	404153.34	9041762.56	2058631.09	
velocity_24h	294516.86	2058631.09	2178930.66	
velocity_4w	277228.51	1116780.94	739532.00	
bank_branch_count_8w	1673.99	28618.34	28073.15	
date_of_birth_distinct_emails_4w	616.87	1856.68	1114.56	
credit_risk_score	-7359.78	-32915.99	-16352.80	
email_is_free	12.17	29.77	19.41	
phone_home_valid	-18.85	-42.10	-33.39	
phone_mobile_valid	2.39	-15.06	-15.27	
bank_months_count	603.76	638.60	387.17	
proposed_credit_limit	-7157.28	-54458.99	2308.14	
session_length_in_minutes	363.32	1336.75	785.77	
keep_alive_session	8.48	20.27	5.74	
device_distinct_emails_8w	5.32	18.66	10.87	
month	-628.23	-2744.88	-1805.31	

	…	credit_risk_score	email_is_free	\
income	…	4.11	-0.00	
name_email_similarity	…	0.71	-0.01	

prev_address_months_count	…	-99.80	-0.51
current_address_months_count	…	734.96	-3.22
customer_age	…	171.81	0.09
days_since_request	…	-33.46	0.02
intended_balcon_amount	…	-21.44	-0.21
zip_count_4w	…	-7359.78	12.17
velocity_6h	…	-32915.99	29.77
velocity_24h	…	-16352.80	19.41
velocity_4w	…	-11337.78	21.44
bank_branch_count_8w	…	-615.34	-1.58
date_of_birth_distinct_emails_4w	…	-57.14	0.09
credit_risk_score	…	5238.57	-0.58
email_is_free	…	-0.58	0.25
phone_home_valid	…	-0.48	-0.00
phone_mobile_valid	…	-0.05	0.00
bank_months_count	…	-53.22	0.04
proposed_credit_limit	…	23118.60	0.14
session_length_in_minutes	…	-17.42	0.17
keep_alive_session	…	-2.01	-0.01
device_distinct_emails_8w	…	-0.41	0.00
month	…	28.45	-0.06

	phone_home_valid	phone_mobile_valid	\
income	0.00	0.00	
name_email_similarity	0.00	0.00	
prev_address_months_count	-0.95	0.15	
current_address_months_count	5.01	-2.74	
customer_age	1.14	-0.59	
days_since_request	-0.11	0.02	
intended_balcon_amount	0.06	0.27	
zip_count_4w	-18.85	2.39	
velocity_6h	-42.10	-15.06	
velocity_24h	-33.39	-15.27	
velocity_4w	-21.67	-9.15	
bank_branch_count_8w	14.42	-2.26	
date_of_birth_distinct_emails_4w	-0.37	0.16	
credit_risk_score	-0.48	-0.05	
email_is_free	-0.00	0.00	
phone_home_valid	0.24	-0.04	
phone_mobile_valid	-0.04	0.10	
bank_months_count	0.35	0.02	
proposed_credit_limit	-3.54	-4.17	
session_length_in_minutes	-0.18	-0.01	
keep_alive_session	0.01	0.00	
device_distinct_emails_8w	-0.00	-0.00	
month	0.05	0.02	

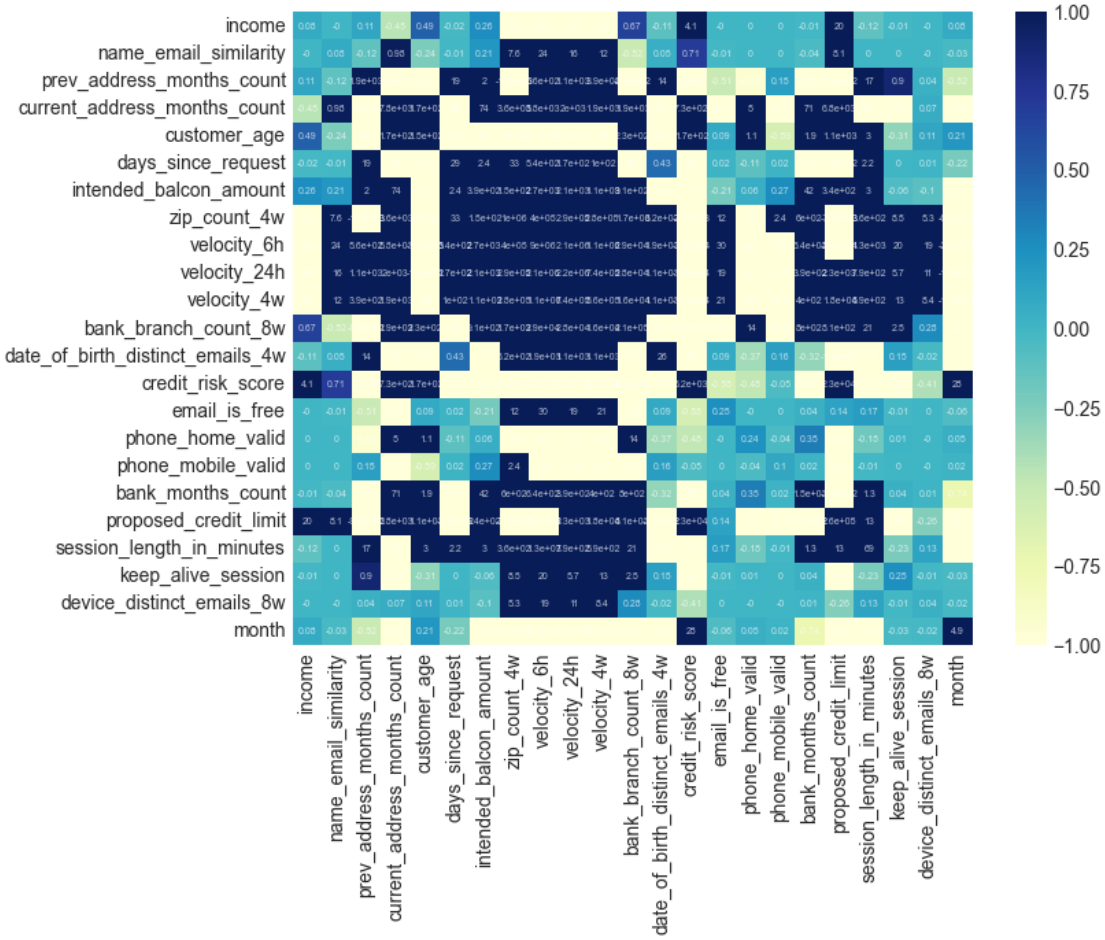
	bank_months_count	proposed_credit_limit	\
income	-0.01	20.15	
name_email_similarity	-0.04	8.07	
prev_address_months_count	-25.25	-942.52	
current_address_months_count	71.31	6788.15	
customer_age	1.90	1110.43	
days_since_request	-1.82	-179.40	
intended_balcon_amount	42.33	343.88	
zip_count_4w	603.76	-7157.28	
velocity_6h	638.60	-54458.99	
velocity_24h	387.17	2308.14	
velocity_4w	398.50	18184.64	
bank_branch_count_8w	804.22	806.08	
date_of_birth_distinct_emails_4w	-0.32	-187.06	
credit_risk_score	-53.22	23118.60	
email_is_free	0.04	0.14	
phone_home_valid	0.35	-3.54	
phone_mobile_valid	0.02	-4.17	
bank_months_count	148.54	-165.77	
proposed_credit_limit	-165.77	263210.30	
session_length_in_minutes	1.34	13.48	
keep_alive_session	0.04	-13.61	
device_distinct_emails_8w	0.01	-0.26	
month	-0.74	-43.57	

	session_length_in_minutes	\
income	-0.12	
name_email_similarity	0.00	
prev_address_months_count	16.75	
current_address_months_count	-20.04	
customer_age	3.05	
days_since_request	2.21	
intended_balcon_amount	3.04	
zip_count_4w	363.32	
velocity_6h	1336.75	
velocity_24h	785.77	
velocity_4w	585.06	
bank_branch_count_8w	20.94	
date_of_birth_distinct_emails_4w	-1.42	
credit_risk_score	-17.42	
email_is_free	0.17	
phone_home_valid	-0.18	
phone_mobile_valid	-0.01	
bank_months_count	1.34	
proposed_credit_limit	13.48	
session_length_in_minutes	68.72	
keep_alive_session	-0.23	

```
device_distinct_emails_8w                          0.13
month                                             -1.41

                                  keep_alive_session  \
income                                         -0.01
name_email_similarity                           0.00
prev_address_months_count                       0.90
current_address_months_count                   -2.62
customer_age                                   -0.31
days_since_request                              0.00
intended_balcon_amount                         -0.06
zip_count_4w                                    8.48
velocity_6h                                    20.27
velocity_24h                                    5.74
velocity_4w                                    12.56
bank_branch_count_8w                            2.51
date_of_birth_distinct_emails_4w                0.15
credit_risk_score                              -2.01
email_is_free                                  -0.01
phone_home_valid                                0.01
phone_mobile_valid                              0.00
bank_months_count                               0.04
proposed_credit_limit                         -13.61
session_length_in_minutes                      -0.23
keep_alive_session                              0.25
device_distinct_emails_8w                      -0.01
month                                          -0.03

                                  device_distinct_emails_8w    month
income                                            -0.00     0.08
name_email_similarity                             -0.00    -0.03
prev_address_months_count                          0.04    -0.52
current_address_months_count                       0.07    -3.85
customer_age                                       0.11     0.21
days_since_request                                 0.01    -0.22
intended_balcon_amount                            -0.10    -2.17
zip_count_4w                                       5.32  -628.23
velocity_6h                                       18.66 -2744.88
velocity_24h                                      10.87 -1805.31
velocity_4w                                        8.42 -1745.46
bank_branch_count_8w                               0.28   -37.01
date_of_birth_distinct_emails_4w                  -0.02    -2.68
credit_risk_score                                 -0.41    28.45
email_is_free                                      0.00    -0.06
phone_home_valid                                  -0.00     0.05
phone_mobile_valid                                -0.00     0.02
bank_months_count                                  0.01    -0.74
```

```
proposed_credit_limit                -0.26   -43.57
session_length_in_minutes             0.13    -1.41
keep_alive_session                   -0.01    -0.03
device_distinct_emails_8w             0.04    -0.02
month                                -0.02     4.94

[23 rows x 23 columns]
```

[33]:
```
# alternatif menggunakan seaborn heatmap, sebelum dilakukan scaled

plt.figure(figsize=(8, 6), dpi=100)
sns.heatmap(credit_num.cov().round(2), vmin=-1, vmax=1, annot=True,␣
 ↪cmap='YlGnBu',
            annot_kws={"size": 5, "color":'white', "alpha":0.7, "ha": 'center',␣
 ↪"va": 'center'});
```

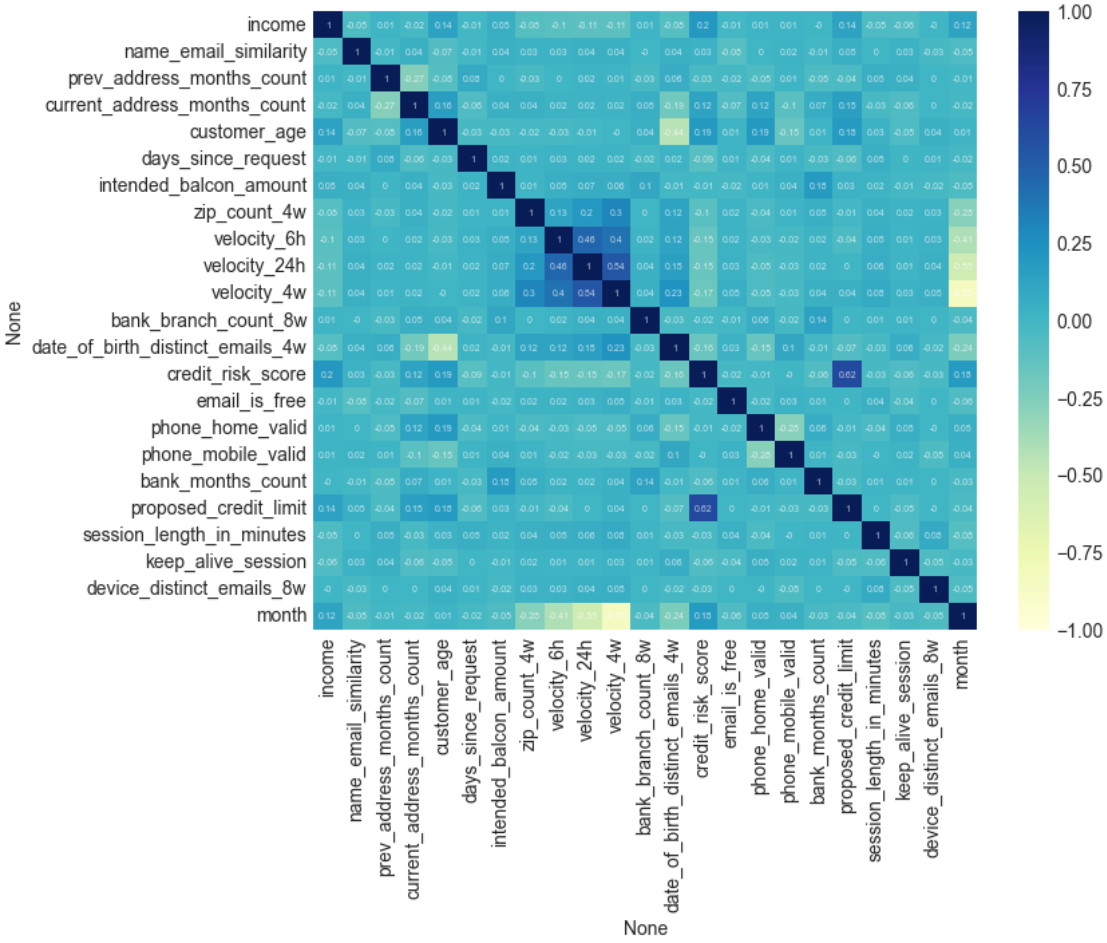


Di atas adalah distribusi nilai covariance dari data yang belum distandarisasi (scale). Variance dari masing-masing variabel berbeda jauh karena range/skala dari tiap variabel berbeda, begitupun

covariance. **Nilai variance dan covariance dipengaruhi oleh skala dari data**. Semakin tinggi skala, nilai variance atau covariance akan semakin tinggi.

Data dengan perbedaan skala antar variabel yang tinggi tidak baik untuk langsung dianalisis PCA karena dapat menimbulkan bias.

[34]:
```
# alternatif menggunakan seaborn heatmap, setelah dilakukan scaled
plt.figure(figsize=(8, 6), dpi=100)
sns.heatmap(credit_scaled.cov().round(2), vmin=-1, vmax=1, annot=True,␣
 ↪cmap='YlGnBu',
            annot_kws={"size": 5, "color":'white', "alpha":0.7, "ha": 'center',␣
 ↪"va": 'center'});
```



5.2.4 Principal Component Analysis menggunakan library sklearn

Setelah melakukan scaling, kita dapat menerapkan PCA pada data yang telah diubah skala, seperti yang ditunjukkan dalam kode berikut: menggunakan `credit_scaled`

[35]:
```
# membuat model PCA
pca_all = PCA(svd_solver='full') # implementasi full svd sehingga mendapatkan␣
 ↪semua PC yang terbentuk
pca_all.fit(credit_scaled.values)
```

[35]:
```
PCA(svd_solver='full')
```

[36]:
```
pca_all.n_components_
```

[36]:
```
23
```

`pca.components_` : berisi nilai *eigen vector* yang akan dijadikan formula untuk PC baru

[37]:
```
pd.DataFrame(pca_all.components_)
```

[37]:

	0	1	2	3	4	5	6	\
0	-0.151037	0.040877	0.030732	-0.037863	-0.115780	0.043330	0.050360	
1	0.131211	0.027067	-0.200700	0.359607	0.431709	-0.111579	0.068807	
2	-0.178849	-0.101274	-0.134318	0.157163	0.167072	0.021271	0.036078	
3	-0.009587	0.213531	-0.262366	0.222816	-0.246917	-0.170547	0.438431	
4	0.242364	-0.279994	0.235604	-0.217804	0.135823	0.228995	0.375115	
5	0.188357	0.155747	0.531049	-0.309087	0.038336	0.119368	0.185062	
6	-0.235646	0.516401	0.077805	0.167756	-0.062101	0.314409	0.132201	
7	0.172621	-0.112981	-0.114172	0.128873	0.214158	0.456829	0.127624	
8	-0.417507	0.173197	0.009419	-0.031988	0.112507	0.106199	-0.056853	
9	-0.021937	0.202547	-0.046761	0.028459	-0.244928	0.266050	0.170098	
10	0.121905	-0.113744	-0.121785	0.111820	-0.058301	0.598057	-0.014284	
11	-0.206624	-0.062817	-0.166514	0.069482	-0.104654	0.332235	-0.175366	
12	-0.199769	-0.423826	0.131820	0.065009	-0.081208	0.085576	-0.264096	
13	-0.308344	-0.514776	-0.014358	0.156806	-0.173163	-0.032976	0.304332	
14	0.547238	-0.026695	-0.390116	0.038245	-0.128592	-0.002828	-0.044823	
15	-0.108939	-0.127094	0.013075	0.044177	-0.046784	-0.124393	0.580360	
16	0.111762	-0.046885	0.215445	0.223301	-0.199189	-0.009768	-0.079542	
17	0.128360	0.074671	0.020785	0.057311	-0.070542	-0.070303	0.020096	
18	0.170970	0.010739	0.499058	0.697312	-0.107183	-0.046154	-0.096163	
19	-0.088448	0.041605	0.061080	0.081162	0.543356	-0.009293	0.092915	
20	-0.047239	0.024611	0.028759	0.094929	0.384675	0.012462	0.020061	
21	0.037005	-0.000327	0.004650	-0.028080	0.006940	-0.021828	-0.033059	
22	-0.007561	0.006024	-0.004642	-0.002325	0.000733	-0.003045	-0.007773	

	7	8	9	…	13	14	15	16	\
0	0.239616	0.352018	0.414803	…	-0.231746	0.043622	-0.072833	0.010492	
1	0.069862	0.106972	0.156302	…	0.352513	-0.015874	0.202134	-0.217273	
2	-0.007396	0.016889	-0.022723	…	-0.462617	-0.047896	0.425957	-0.324666	
3	0.041106	-0.051532	-0.049706	…	0.021995	-0.146909	-0.062562	0.254078	
4	-0.055353	0.001611	0.019117	…	-0.024789	0.225189	-0.162471	0.169431	
5	-0.155093	0.035027	0.060782	…	0.103966	-0.252733	0.311583	-0.251187	
6	-0.021946	0.028791	-0.000634	…	-0.009434	-0.491929	-0.118447	0.005524	

```
7  -0.103286  0.183392  0.118983  … -0.113106 -0.098916 -0.166988  0.262103
8   0.119617 -0.082872 -0.066426  …  0.085306  0.473197  0.042277  0.120429
9  -0.164871 -0.000459 -0.012297  …  0.028219  0.368136  0.228230 -0.389565
10  0.477876 -0.240225 -0.167987  …  0.059772  0.036078  0.067682 -0.094456
11 -0.476396  0.171030  0.080309  …  0.090041  0.111344 -0.059934  0.027938
12  0.244599 -0.113872 -0.049655  …  0.079829 -0.389076 -0.079060 -0.149737
13 -0.314724  0.141569  0.050763  …  0.107626 -0.116529  0.011363 -0.124214
14 -0.141151  0.000537 -0.025960  … -0.065052 -0.086060  0.105649 -0.075051
15  0.257813 -0.166999 -0.047259  … -0.036045  0.071306  0.029780 -0.047163
16  0.303979  0.580973  0.094537  …  0.082986  0.086032  0.364064  0.264860
17  0.170067  0.277484  0.086507  …  0.003160  0.130061 -0.606180 -0.572511
18 -0.180817 -0.249173 -0.076325  … -0.123527  0.179212 -0.104604  0.040148
19 -0.036077  0.332556 -0.534780  … -0.059841 -0.020769 -0.104467 -0.013200
20 -0.001482 -0.289882  0.653228  … -0.018125  0.003599 -0.023472  0.024701
21 -0.002987  0.001374 -0.007820  … -0.716723 -0.005595 -0.014360  0.013627
22 -0.016671  0.006550  0.010857  … -0.015082  0.010368 -0.001510 -0.003617

          17        18        19        20        21        22
0   0.037212 -0.092509  0.070897  0.039108  0.037528 -0.483264
1   0.066798  0.417700  0.032357 -0.100973  0.054958 -0.179102
2   0.211714 -0.476955  0.000565  0.062404  0.064336  0.065935
3   0.451115  0.046819 -0.207881  0.112532 -0.234753  0.050471
4   0.330485 -0.021074  0.360367 -0.250153  0.206724  0.020152
5   0.023646  0.110266 -0.144659  0.397021 -0.183692 -0.041902
6  -0.073182  0.004529  0.397022 -0.132294  0.218359  0.068536
7  -0.182277 -0.146097 -0.243146  0.002718 -0.503271 -0.011479
8   0.040502  0.095741  0.395842  0.409462 -0.364502  0.017052
9  -0.001579  0.071445 -0.025815 -0.523836 -0.284846  0.011334
10  0.150392  0.103756 -0.173721  0.250832  0.276410 -0.006428
11 -0.200212  0.109315 -0.179733  0.145609  0.274389  0.011601
12 -0.005072  0.090865  0.173012 -0.238445 -0.409997  0.000138
13  0.143333  0.150016  0.145484  0.221379  0.043604  0.034776
14 -0.190783 -0.072610  0.552374  0.246501 -0.119041 -0.027070
15 -0.683521 -0.020119 -0.034851  0.025647  0.075767  0.007789
16 -0.065003  0.020630  0.056199 -0.032660  0.040016  0.295578
17  0.054067 -0.057457 -0.038607  0.184381 -0.036715  0.192215
18 -0.019240 -0.086775  0.019397  0.071578  0.001405 -0.136164
19 -0.033328 -0.033491  0.010747 -0.013980 -0.008865 -0.012687
20 -0.005242 -0.047001  0.025411  0.011798  0.003196  0.251476
21 -0.016415  0.682026 -0.010833  0.004329 -0.015434  0.070253
22 -0.004895  0.012329 -0.000285 -0.000011  0.001875  0.711832

[23 rows x 23 columns]
```

Rasio Varians yang Dijelaskan

Rasio varians yang dijelaskan memberikan informasi proporsi tentang seberapa banyak informasi yang diatributkan pada setiap PC. Berdasarkan informasi di bawah ini, kita mendapatkan informasi

bahwa PC1 mengandung sekitar 13,18% informasi, PC2 sebanyak 0,08%, dan PC3 sebanyak 0,06%.

[38]:
```
pca_all.explained_variance_ratio_
```

[38]:
```
array([0.1318414 , 0.08761348, 0.06657119, 0.05687241, 0.05320949,
       0.0503121 , 0.04685173, 0.04350163, 0.04271713, 0.04131351,
       0.04033185, 0.03925077, 0.03842389, 0.03796283, 0.03597653,
       0.03405586, 0.03011896, 0.02978514, 0.02918552, 0.02171452,
       0.02086706, 0.01493503, 0.00658796])
```

Untuk lebih mudah dipahami, kita dapat menghitung proporsi kumulatif dari PC1 hingga PC20. Berikut adalah output yang menyatakan bahwa seluruh dimensi dalam PC mengandung sekitar 95.77% informasi dari dataset asli kita.

[39]:
```
np.cumsum(np.round(pca_all.explained_variance_ratio_, decimals=4)*100)
```

[39]:
```
array([ 13.18,  21.94,  28.6 ,  34.29,  39.61,  44.64,  49.33,  53.68,
        57.95,  62.08,  66.11,  70.04,  73.88,  77.68,  81.28,  84.69,
        87.7 ,  90.68,  93.6 ,  95.77,  97.86,  99.35, 100.01])
```

Observasi:

Dalam output yang diberikan diatas, nilai pada setiap posisi adalah kumulatif dari rasio varians yang dijelaskan oleh tiap komponen utama. Ketika kita melihat persentase kumulatif tersebut, kita mencari titik di mana persentase varians yang dijelaskan sudah cukup tinggi untuk kebutuhan analisis kita.

Dari hasil yang diberikan, tampaknya sekitar 95.77% dari varians dapat dijelaskan oleh 20 komponen utama pertama. Oleh karena itu, Anda mungkin memilih untuk mengambil 20 komponen utama ini, yang mencakup sebagian besar varians data Anda.

Namun, keputusan akhir bergantung pada kebutuhan spesifik analisis Anda. Beberapa praktik umum adalah memilih jumlah komponen utama yang menjelaskan sekitar 80-95% dari varians total, tetapi Anda dapat memilih nilai yang sesuai dengan tujuan dan konteks analisis Anda.

[40]:
```
# Membuat model PCA  dengan jumlah komponen utama sekitar 95%
pca = PCA(n_components = 0.95,
          svd_solver='full')
pca.fit(credit_scaled.values)

# Nilai dimensi baru dari PCA Model
credit_pca20 = pd.DataFrame(pca.fit_transform(credit_scaled),
                            columns=[f'PC{i}' for i in range(1, pca.n_components_␣
 ↪+ 1)])
credit_pca20.head()
```

[40]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	\
0	1.692755	3.323030	-0.889182	0.568785	1.539485	0.669139	-0.882377	
1	0.853039	0.094970	1.362371	-1.279626	-1.114656	-1.041867	-1.349552	
2	0.933672	2.583289	-0.083890	-1.519840	1.423196	-0.311211	-1.014184	

```
3  1.463710 -2.074312  0.009756  0.335156 -0.323293 -0.674480 -1.214947
4 -2.179767  0.666897 -1.064235 -0.738059 -0.202709  0.682306 -1.596558

        PC8       PC9      PC10      PC11      PC12      PC13      PC14  \
0 -0.816128  0.464054  0.338666 -0.942384  1.925547  1.243251 -2.648092
1 -1.287512 -0.509089  1.731860  2.152441 -1.882548  1.821081 -0.446959
2 -2.772364 -0.670852 -2.078549  2.107466  1.285194 -2.136757  1.083076
3 -0.397285  0.673089 -0.550966  1.103438 -0.722769  0.301028  0.111886
4  0.048435  0.442683 -0.068521  0.133925  0.679665 -0.644503  0.599069

       PC15      PC16      PC17      PC18      PC19      PC20
0  0.622580  0.200997  1.858074 -0.260204 -1.150616 -0.780847
1  0.185748  1.788388  1.661765  1.457426 -0.587829 -0.324188
2 -1.209996 -0.610770 -1.318451 -1.986781 -0.225584 -1.288779
3 -0.864106  0.344133  0.582058  0.913164 -0.487811 -0.710987
4  0.769384  0.144125  0.493707 -0.715133 -0.387881 -1.316316
```

[41]:
```
credit_pca20.shape
```

[41]:
```
(100000, 20)
```

[42]:
```
credit_scaled.head(3)
```

[42]:
```
     income name_email_similarity prev_address_months_count  \
0  1.125802              0.454534                -0.387324
1  0.094766             -1.410073                -0.387324
2 -0.248913             -1.395895                -0.387324

  current_address_months_count customer_age days_since_request  \
0                    -0.431527     0.459337          -0.188833
1                     0.406739    -1.165255          -0.189457
2                    -0.590118     1.271634          -0.189578

  intended_balcon_amount zip_count_4w velocity_6h velocity_24h  …  \
0              -0.442254     1.075569    1.790845     1.827162  …
1              -0.455380     4.005273    0.226647    -0.256489  …
2              -0.468534    -0.226854   -1.418498     1.297549  …

  credit_risk_score email_is_free phone_home_valid phone_mobile_valid  \
0          1.623733      0.916752         1.233515           0.359129
1         -0.324384      0.916752         1.233515          -2.784511
2          0.877646      0.916752         1.233515           0.359129

  bank_months_count proposed_credit_limit session_length_in_minutes  \
0         -0.144869              1.860207                 0.655308
1         -0.965382             -0.673720                -0.118348
2          1.414106              1.860207                -0.380592
```

```
  keep_alive_session device_distinct_emails_8w     month
0          -1.114677                -0.124266 -1.038034
1          -1.114677                -0.124266  0.312032
2           0.897121                 4.874361 -1.488056

[3 rows x 23 columns]
```

5.3 Visualisasi PCA

5.3.1 Visualisasi Biplot

Biplot adalah visualisasi data multivariat yang menggabungkan **individual factor map** dan **variables factor map**.

PCA tidak hanya berguna untuk dimensionality reduction namun baik untuk visualisasi high-dimensional data. Visualisasi dapat menggunakan **biplot** yang menampilkan:

1. **Individual factor map**, yaitu sebaran data secara keseluruhan menggunakan 2 PC. Tujuannya untuk:

- observasi yang serupa
- outlier dari keseluruhan data

2. **Variables factor map**, yaitu plot yang menunjukkan korelasi antar variable dan kontribusinya terhadap PC.

[43]:
```
sample_plot = credit_scaled.iloc[0:50, :] # ambil sample data
sample_plot.head(3)
```

[43]:
```
     income name_email_similarity prev_address_months_count  \
0  1.125802              0.454534                -0.387324
1  0.094766             -1.410073                -0.387324
2 -0.248913             -1.395895                -0.387324

  current_address_months_count customer_age days_since_request  \
0                    -0.431527     0.459337          -0.188833
1                     0.406739    -1.165255          -0.189457
2                    -0.590118     1.271634          -0.189578

  intended_balcon_amount zip_count_4w velocity_6h velocity_24h  …  \
0              -0.442254     1.075569    1.790845     1.827162  …
1              -0.455380     4.005273    0.226647    -0.256489  …
2              -0.468534    -0.226854   -1.418498     1.297549  …

  credit_risk_score email_is_free phone_home_valid phone_mobile_valid  \
0          1.623733      0.916752         1.233515           0.359129
1         -0.324384      0.916752         1.233515          -2.784511
2          0.877646      0.916752         1.233515           0.359129
```

```
  bank_months_count proposed_credit_limit session_length_in_minutes  \
0         -0.144869              1.860207                  0.655308
1         -0.965382             -0.673720                 -0.118348
2          1.414106              1.860207                 -0.380592

  keep_alive_session device_distinct_emails_8w     month
0          -1.114677                -0.124266 -1.038034
1          -1.114677                -0.124266  0.312032
2           0.897121                 4.874361 -1.488056

[3 rows x 23 columns]
```

[44]:
```
sample_plot.shape
```

[44]:
```
(50, 23)
```

[45]:
```
# ambil nama tiap kolom balance_saled sesuai urutan di df
features = list(sample_plot.columns.values)
# hasil PCA. sama seperti objek transform_ di atas, tapi beda nama kolom
components = pca.fit_transform(sample_plot)
# dot product/perkalian matriks antara eigenvector dengan akar kuadrat eigen␣
 ↪value
loadings = pca.components_.T * np.sqrt(pca.explained_variance_)

# Pilih fitur dengan loadings terbesar pada PC1 dan PC2
important_features = [features[i][0] for i in np.abs(loadings[:, :5]).
 ↪argmax(axis=0)]

fig = px.scatter(components, x=0, y=1, hover_name=list(range(len(components))))

for i, feature in enumerate(features):
    # Hanya tambahkan garis panah dan anotasi untuk fitur yang penting
    if feature[0] in important_features:
        fig.add_shape(
            type='line',
            x0=0, y0=0,
            x1=loadings[i, 0],
            y1=loadings[i, 1],
        )

        fig.add_annotation(
            x=loadings[i, 0],
            y=loadings[i, 1],
            ax=0, ay=0,
            xanchor="center",
            yanchor="bottom",
            text=features[i][0],
```

```
        )

        fig.add_annotation(
            x=loadings[i, 0],
            y=loadings[i, 1],
            ax=0, ay=0,
            xref='x', yref='y',
            axref='x', ayref='y',
            text='',  # if you want only the arrow
            showarrow=True,
            arrowhead=4,
            arrowsize=1.5,
            arrowwidth=1,
            arrowcolor='black'
        )

# Plotly figure layout
fig.update_layout(title='Biplot PCA', title_x=0.5, width=800, height=600)
fig.update_xaxes(title_text='PC1')
fig.update_yaxes(title_text='PC2')
fig.show()
```

Individual Factor Map

Pada **individual factor map**, observasi diplot berdasarkan skor mereka pada PC1 dan PC2. Observasi yang serupa akan berada dekat satu sama lain, sedangkan observasi yang berbeda akan berada jauh satu sama lain.

Pada biplot yang Anda kirimkan, observasi 1, 2, dan 3 berada dekat satu sama lain, menunjukkan bahwa mereka memiliki karakteristik yang mirip. Observasi 4, 5, dan 6 juga berada dekat satu sama lain, menunjukkan bahwa mereka juga memiliki karakteristik yang mirip. Observasi 7, 8, dan 9 berada jauh dari observasi lainnya, menunjukkan bahwa mereka memiliki karakteristik yang berbeda.

Outlier

Outlier adalah observasi yang berada jauh dari distribusi data lainnya. Pada biplot, outlier biasanya terletak di tepi plot. Pada biplot diatas, observasi 42 adalah outlier. Observasi ini berada jauh di tepi plot, menunjukkan bahwa observasi ini memiliki karakteristik yang sangat berbeda dari observasi lainnya.

Variables Factor Map

Pada **variables factor map**, variabel diplot berdasarkan skor mereka pada PC1 dan PC2. Korelasi antar variabel dapat dilihat dari sudut antar dua panah

- Panah saling berdekatan (sudut antar panah < 90), maka korelasi positif
- Panah saling tegak lurus (sudut antar panah $= 90$), maka tidak berkorelasi
- Panah saling bertolak belakang (sudut antar panah mendekati 180), maka korelasi negatif

Contoh interpretasi plot diatas:

- Variabel `bank_branch_count_8w` dan `phone_mobile_valid` berkorelasi positif. Variabel ini berada dekat satu sama lain, menunjukkan bahwa variabel ini cenderung meningkat atau menurun bersama-sama.
- Variabel `bank_branch_count_8w` dan `month` tidak berkorelasi.
- Variabel `month` dan `name_email_similarity` berkorelasi negatif. Variabel ini berada jauh satu sama lain, menunjukkan bahwa variabel ini cenderung meningkat atau menurun secara berlawanan.

Kesimpulan Berdasarkan biplot diatas, dapat disimpulkan bahwa: ...

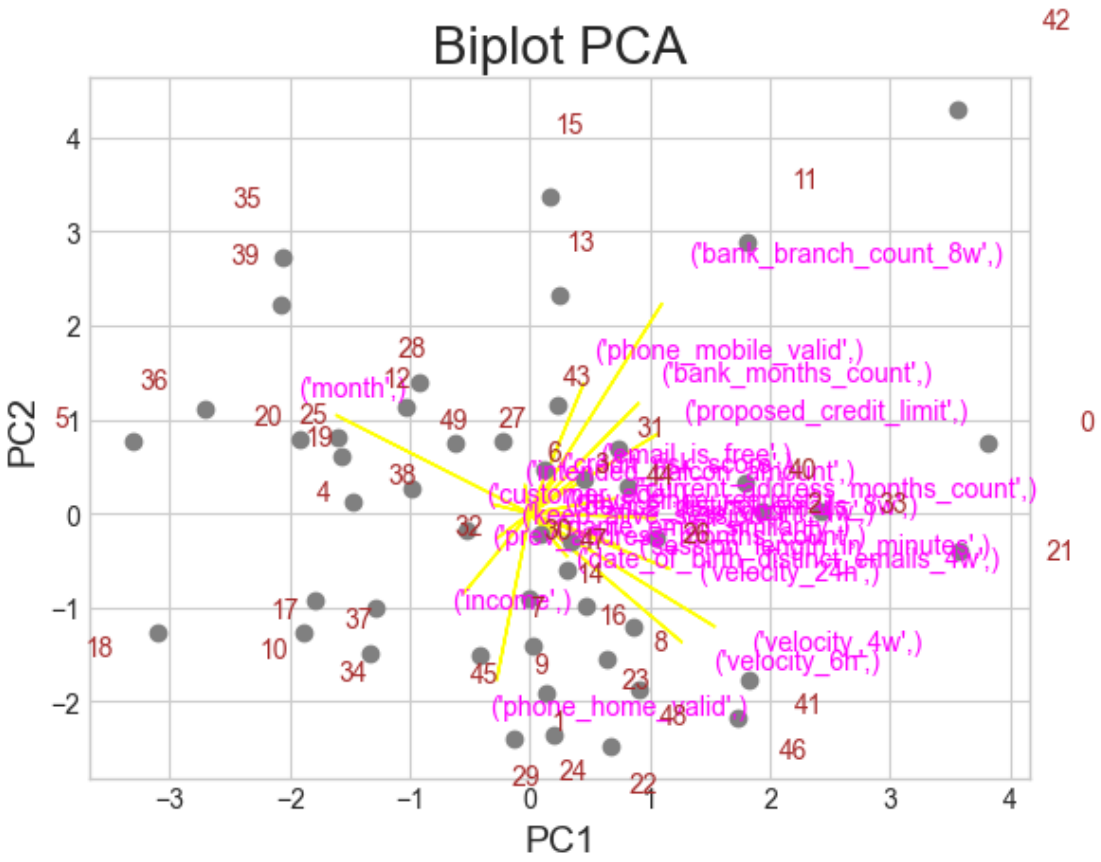
5.3.2 [Optional] Visualisasi explained variance menggunakan library]

[46]:
```
exp_var_cumul = np.cumsum(np.round(pca.explained_variance_ratio_,␣
↪decimals=4)*100)

px.area(
    x=range(1, exp_var_cumul.shape[0] + 1),
    y=exp_var_cumul,
    labels={"x": "Jumlah Components", "y": "Explained Variance"}
)
```

Plot di atas adalah plot kumulatif varians yang dijelaskan oleh komponen utama dalam analisis PCA. Pada sumbu x, kita memiliki jumlah komponen utama, sedangkan sumbu y menunjukkan persentase varians yang dijelaskan. Kurva area menunjukkan akumulasi persentase varians seiring penambahan komponen utama. Poin penting pada kurva adalah di mana Anda memutuskan jumlah komponen yang akan diambil untuk menjelaskan sejumlah tertentu dari varians total dalam data.

5.3.3 [Optional] Simple biplot : helper.py]

[47]:
```
biplot_pca(sample_plot)
```



5.3.4 [Optional] Scree Plot Proporsi Eigenvalue]

[48]:
```
import plotly.graph_objects as go

# Dapatkan nilai eigenvalue dari komponen utama
eigenvalues = pca.explained_variance_

# Hitung proporsi variasi yang dijelaskan oleh setiap komponen utama
explained_var_ratio = pca.explained_variance_ratio_

# Buat scree plot menggunakan plotly
fig = go.Figure()

# Plot eigenvalue
fig.add_trace(go.Scatter(x=list(range(1, len(eigenvalues) + 1)),
                         y=eigenvalues, mode='lines+markers',␣
 ↪name='Eigenvalue'))

# Plot proporsi variasi yang dijelaskan
```

```
fig.add_trace(go.Scatter(x=list(range(1, len(explained_var_ratio) + 1)),
                         y=explained_var_ratio, mode='lines+markers',
                         name='Explained Variance Ratio'))

# Atur layout dan tampilkan
fig.update_layout(title='Scree Plot',
                  xaxis_title='Principal Component (PC)',
                  yaxis_title='Eigenvalue / Explained Variance Ratio',
                  showlegend=True,
                  width=800, height=620)

fig.show()
```

Dalam contoh ini, scree plot menunjukkan eigenvalue dan proporsi variasi yang dijelaskan oleh setiap komponen utama. Anda dapat mengatur `ncp` sesuai dengan jumlah komponen utama yang ingin Anda tampilkan pada plot. Namun, dalam hal ini kita pakai semua PC nya yakni 23

[49]:
```
# Buat dataframe untuk hasil PCA
pca_df = credit_pca20.copy()#pd.DataFrame(data=x_pca, columns=[f'PC{i+1}' for i␣
 ↪in range(x_pca.shape[1])])

# Tambahkan kolom kelas (jika ada)
# Jika x memiliki kolom kelas, ganti 'class_column_name' dengan nama kolomnya
pca_df['fraud_bool'] = credit_clean['fraud_bool'].astype(str)

# Buat scatter plot PCA
fig = px.scatter(pca_df, x='PC1', y='PC2',
                 title='Plot PCA berdasarkan Label',
                 color='fraud_bool',
                #   color=outliers.astype(str),
                 color_discrete_map={'0': '#a6c4ff', '1': '#ffa07a'})

# Atur layout dan tampilkan plot
fig.update_layout(xaxis_title='PC 1', yaxis_title='PC 2',
                  width=800, height=600)
fig.show()
```

Tujuan dari plot PCA berdasarkan label dengan PC1 dan PC2 adalah untuk memvisualisasikan distribusi data pada dimensi-dimensi utama (PC1 dan PC2) setelah proses reduksi dimensi menggunakan metode PCA. Plot tersebut juga memberikan pemahaman visual tentang bagaimana kelas atau label data tersebar dalam ruang yang direduksi dimensinya, membantu mengidentifikasi pola atau cluster yang mungkin ada dalam data.

Important:

Jika pada data awal tidak ada kolom label (misalnya, kolom target seperti 'fraud_bool'), maka melakukan plot PCA berdasarkan label tidak akan memberikan informasi yang relevan atau bermakna. Plot tersebut berguna ketika kita ingin melihat pola atau cluster berdasarkan kelas atau label tertentu dalam data.

Jika tidak ada kolom label, lebih relevan untuk fokus pada visualisasi distribusi data berdasarkan komponen utama (PC1 dan PC2) tanpa mempertimbangkan kelas atau label. Plot tersebut dapat memberikan wawasan tentang struktur internal data tanpa melibatkan informasi label. Dengan demikian, fokusnya akan pada variasi dan pola intrinsik dalam data, bukan pada pemisahan berdasarkan kelas.

[50]:
```
# Dapatkan loadings dari PCA
loadings = pca.components_

# Buat dataframe untuk loadings
loadings_df = pd.DataFrame(data=loadings.T, columns=[f'PC{i+1}' for i in␣
 ↪range(loadings.shape[0])])

# Tambahkan kolom nama variabel
loadings_df['Variable'] = credit_scaled.columns

# Tampilkan loadings yang signifikan (misalnya, absolute loadings > 0.3)
significant_loadings = loadings_df[abs(loadings_df['PC1']) > 0.2]
significant_loadings
```

[50]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	\
7	0.272998	-0.011632	0.170400	0.219698	0.207326	-0.062567	0.481160	
8	0.332235	-0.318315	0.022183	0.002883	0.061321	0.090515	-0.271476	
9	0.305289	-0.137153	-0.099439	-0.094422	-0.047025	-0.088929	0.107691	
10	0.404106	-0.279941	-0.028704	-0.125352	-0.010460	0.015621	-0.002203	
11	0.288576	0.518364	-0.167788	0.239467	-0.186757	-0.261848	-0.153593	
17	0.237831	0.272516	0.245491	0.111663	0.020680	0.173127	-0.058940	
18	0.278637	0.196126	-0.415571	-0.105264	0.019541	-0.064495	0.230005	
22	-0.423333	0.242621	0.002793	0.179051	0.087539	0.029009	0.008577	

	PC8	PC9	PC10	PC11	PC12	PC13	PC14	\
7	0.035475	-0.124596	0.036945	0.196703	0.512680	0.095425	0.129882	
8	-0.418818	-0.129885	-0.283481	0.093240	-0.022176	-0.330322	0.140538	
9	-0.006446	0.037211	-0.243017	0.145835	-0.341255	0.347956	0.078378	
10	0.192010	-0.040707	0.027677	-0.291184	0.092809	0.236445	-0.158558	
11	0.170006	-0.333268	-0.197671	0.162653	-0.021419	-0.011554	0.176552	
17	0.166192	0.658737	-0.307781	0.062013	0.033613	-0.342220	-0.005277	
18	-0.136513	0.119629	0.321415	0.075557	-0.037261	-0.196290	0.072416	
22	-0.069880	-0.147456	-0.135594	0.192729	-0.137465	0.064606	-0.092848	

	PC15	PC16	PC17	Variable
7	-0.207396	-0.047797	0.075471	(zip_count_4w,)
8	0.091688	-0.234790	-0.010717	(velocity_6h,)
9	-0.284732	-0.329304	0.400055	(velocity_24h,)
10	0.306045	0.084556	-0.107509	(velocity_4w,)
11	0.306944	0.112792	0.027198	(bank_branch_count_8w,)
17	0.005918	0.084181	0.082705	(bank_months_count,)

```
18 -0.163517 -0.012256 -0.180826  (proposed_credit_limit,)
22 -0.142119 -0.032587  0.104566                  (month,)
```

Observasi:

Interpretasi loadings dari hasil PCA dapat memberikan pemahaman tentang kontribusi setiap variabel terhadap pembentukan setiap komponen utama (PC). Di sini, saya akan memberikan contoh interpretasi untuk beberapa variabel yang memiliki loadings signifikan pada PC1:

1. **zip_count_4w:** Variabel ini memiliki loading positif yang cukup besar pada PC1 (0.27). Ini menunjukkan bahwa nilai tinggi pada `zip_count_4w` berkontribusi positif terhadap nilai PC1. Dengan kata lain, observasi dengan nilai `zip_count_4w` yang tinggi cenderung memiliki nilai PC1 yang tinggi.
2. **velocity_6h:** Variabel ini memiliki loading positif yang kuat pada PC1 (0.33). Hal ini menandakan bahwa nilai tinggi pada `velocity_6h` berhubungan positif dengan nilai PC1. Observasi dengan `velocity_6h` yang tinggi cenderung memiliki nilai PC1 yang tinggi.
3. **velocity_24h:** Variabel ini memiliki loading negatif yang signifikan pada PC1 (-0.31). Ini menandakan bahwa nilai tinggi pada `velocity_24h` berhubungan negatif dengan nilai PC1. Dengan kata lain, observasi dengan nilai `velocity_24h` yang tinggi cenderung memiliki nilai PC1 yang rendah.
4. **velocity_4w:** Variabel ini memiliki loading positif yang signifikan pada PC1 (0.40). Ini menunjukkan bahwa nilai tinggi pada `velocity_4w` berhubungan positif dengan nilai PC1. Observasi dengan `velocity_4w` yang tinggi cenderung memiliki nilai PC1 yang tinggi.

Demikian pula, Anda dapat melanjutkan untuk menganalisis loadings pada PC lainnya. Loadings yang tinggi (baik positif maupun negatif) menunjukkan sejauh mana variabel tersebut berkontribusi terhadap pembentukan komponen utama tersebut. Variabel dengan loadings tinggi pada suatu PC dapat dianggap sebagai variabel yang memiliki pengaruh besar pada PC tersebut.

6 Studi Kasus Local Outlier Factor (LOF): Deteksi Anomali dengan Menggunakan Library PyOD

6.1 Overview

Anomaly detection adalah teknik yang digunakan untuk mengidentifikasi data yang tidak sesuai dengan pola normal data. Teknik ini dapat digunakan untuk berbagai tujuan, seperti deteksi fraud, deteksi gangguan pada sistem, dan deteksi penyimpangan data. Anomali dapat merugikan suatu perusahaan, terutama dalam kondisi di mana observasi tidak biasa dapat menjadi indikator adanya masalah serius. Untuk mengatasi tantangan ini, deteksi anomali menjadi kritis, dan salah satu pendekatan yang dapat digunakan adalah algoritma *Local Outlier Factor* (LOF).

Penerapan deteksi anomali memiliki dampak besar di berbagai industri, termasuk *e-commerce* dan industri perbankan. Mencegah terjadinya anomali atau situasi tak lazim yang dapat berkembang menjadi masalah lebih besar akan membantu perusahaan mencegah kerugian keuangan.

Dalam kasus ini, kita akan membahas secara mendalam mengenai deteksi anomali menggunakan teknis *unsupervised learning* yang mana dataset diasumsikan tidak memimlikilabel dengan menggunakan algoritma *Local Outlier Factor* (LOF), konsep dasar di balik algoritma ini, dan bagaimana

algoritma ini bekerja untuk mendeteksi observasi yang tidak biasa. Penggunaan library `PyOD` dalam implementasi LOF akan menjadi fokus utama, memperkuat kehandalan dan fleksibilitas deteksi anomali dalam konteks pemrograman Python.

Referensi: PyOD Documentation. https://pyod.readthedocs.io/en/latest/.

7 Local Outlier Factor

Dalam deteksi anomali , *Local Outlier Factor* (LOF) adalah algoritma yang diusulkan oleh Markus M. Breunig, Hans-Peter Kriegel , Raymond T. Ng dan Jörg Sander pada tahun 2000 untuk menemukan titik data anomali dengan mengukur deviasi lokal dari titik data tertentu sehubungan dengan tetangganya. [1]

LOF adalah algoritma *unsupervied learning* salah satu teknik anomaly detection yang populer. Teknik ini bekerja dengan cara menghitung seberapa jauh suatu data dari tetangga terdekatnya. Data yang memiliki nilai LOF yang tinggi dianggap sebagai outlier. Implementasi LOF dapat diaplikasikan pada berbagai kondisi, seperti mengidentifikasi titik-titik tak lazim dalam suatu dataset. Selain itu, algoritma ini dapat digunakan untuk mendeteksi kondisi anomali, yang penting untuk mencegah situasi yang tidak diinginkan berkembang lebih jauh.

Kesesuaian LOF untuk Deteksi Fraud:

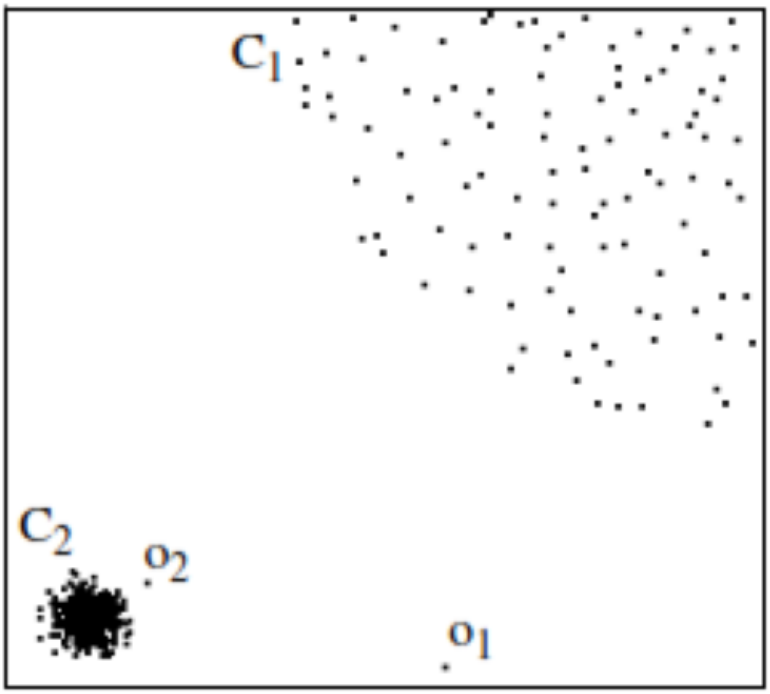
LOF dapat menjadi pilihan yang baik untuk deteksi fraud karena memiliki beberapa kelebihan, yaitu:

- Efektif dalam menemukan outlier lokal: LOF dapat mengidentifikasi outlier yang tidak dapat ditemukan oleh metode global, seperti outlier yang berada di dalam cluster yang padat.
- Tidak sensitif terhadap distribusi data: LOF dapat bekerja dengan baik pada data dengan distribusi yang tidak normal.
- Mudah diimplementasikan: LOF dapat diimplementasikan dengan mudah menggunakan library Python seperti Pyod.

Namun, LOF juga memiliki beberapa kekurangan, yaitu:

- Dapat menjadi lambat untuk data yang besar: LOF memerlukan komputasi yang cukup berat untuk dataset yang besar.
- Memerlukan pemilihan parameter yang tepat: Parameter k (jumlah tetangga terdekat) yang digunakan dalam LOF dapat mempengaruhi hasil deteksi outlier.

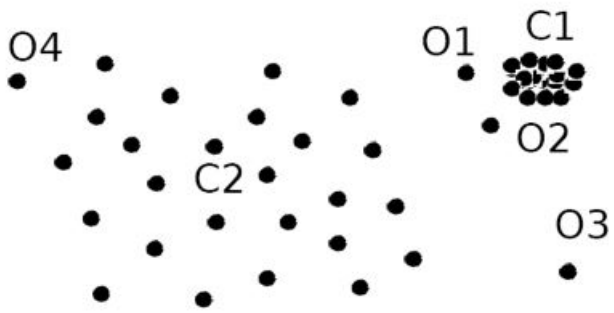
Lof akan menghitung kepadatan lokal suatu titik dibandingkan dengan kepadatan lokal k tetangganya. Misalkan kita mempunyai contoh pada gambar berikut.



Pada gambar di atas terlihat bahwa titik o1 merupakan outlier karena jarak titik tersebut jauh lebih besar dibandingkan titik terdekat. Pertanyaannya adalah, bagaimana kita memberi label o2? Dengan algoritma faktor outlier lokal, kita dapat menghitung skor o2 dan melihat apakah itu outlier atau bukan. Faktor outlier lokal akan menghasilkan skor dengan interpretasi sebagai berikut:

- Jika LOF > 1, kemungkinan besar merupakan titik outlier
- Jika LOF < 1, kemungkinan besar titik tersebut bukan merupakan titik outlier

Jika titik tersebut kurang padat dibandingkan area dengan kepadatan tinggi di C2, kemungkinan besar titik tersebut merupakan outlier. Semakin terisolasi suatu titik dibandingkan dengan titik tetangganya yang memiliki kepadatan tinggi, semakin besar kemungkinan suatu observasi dikategorikan sebagai outlier.



Contoh lainnya adalah o1, dan o2 dapat dianggap sebagai outlier, dan istilah untuk outlier ini bersifat lokal. o3 merupakan outlier global, sedangkan o4 bukan merupakan outlier karena lingkungan sekitar titik tidak sepadat kepadatan di dekat o1, o2, atau o3.

PyOD LOF Documentation. https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.lof

Dataset yang digunakan masih sama, menggunakan dataset dari Bank Account Fraud dari kaggle yang sudah dilakukan PCA sebelumnya dengan mempertahankan 20PC

7.1 Membuat Model LOF

Disini kita menggunakan semua observasi dengan 20 PC

[51]:
```
credit_pca20.head()
```

[51]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	\
0	1.692755	3.323030	-0.889182	0.568785	1.539485	0.669139	-0.882377	
1	0.853039	0.094970	1.362371	-1.279626	-1.114656	-1.041867	-1.349552	
2	0.933672	2.583289	-0.083890	-1.519840	1.423196	-0.311211	-1.014184	
3	1.463710	-2.074312	0.009756	0.335156	-0.323293	-0.674480	-1.214947	
4	-2.179767	0.666897	-1.064235	-0.738059	-0.202709	0.682306	-1.596558	

	PC8	PC9	PC10	PC11	PC12	PC13	PC14	\
0	-0.816128	0.464054	0.338666	-0.942384	1.925547	1.243251	-2.648092	
1	-1.287512	-0.509089	1.731860	2.152441	-1.882548	1.821081	-0.446959	
2	-2.772364	-0.670852	-2.078549	2.107466	1.285194	-2.136757	1.083076	
3	-0.397285	0.673089	-0.550966	1.103438	-0.722769	0.301028	0.111886	
4	0.048435	0.442683	-0.068521	0.133925	0.679665	-0.644503	0.599069	

	PC15	PC16	PC17	PC18	PC19	PC20
0	0.622580	0.200997	1.858074	-0.260204	-1.150616	-0.780847
1	0.185748	1.788388	1.661765	1.457426	-0.587829	-0.324188

```
2 -1.209996 -0.610770 -1.318451 -1.986781 -0.225584 -1.288779
3 -0.864106  0.344133  0.582058  0.913164 -0.487811 -0.710987
4  0.769384  0.144125  0.493707 -0.715133 -0.387881 -1.316316
```

[52]:
```
# Menggunakan LOF dari PyOD (Python Outlier Detection)
lof_model = LOF()
outliers = lof_model.fit_predict(credit_pca20)
```

Pada kode di atas, kita menggunakan model LOF (Local Outlier Factor) dari modul yang disebut sebagai `LOF` dari library Pyod, tetapi kode lengkapnya tidak diberikan). Model LOF adalah algoritma deteksi outlier yang bekerja dengan menilai tingkat kedekatan setiap titik data dengan tetangganya. Output dari `fit_predict(credit_pca20)` adalah label yang menunjukkan setiap titik data sebagai outlier (label -1) atau tidak (label 1).

Sejauh ini, Kita sudah benar memasukkan hasil PCA (`credit_pca20`) sebagai input untuk deteksi outlier menggunakan LOF. Namun, beberapa hal yang perlu diperhatikan:

1. **Jumlah Cluster:**
   - Model LOF memerlukan informasi tentang jumlah cluster atau tetangga yang digunakan untuk menghitung faktor lokal. Jika Kita tidak memberikan parameter ini, LOF akan menganggap bahwa seluruh dataset adalah satu cluster.
2. **Penyesuaian Parameter:**
   - Terkadang, performa LOF dapat dipengaruhi oleh parameter yang diatur, seperti `n_neighbors` (jumlah tetangga yang digunakan dalam perhitungan). Pastikan untuk menyesuaikan parameter-parameter ini sesuai dengan karakteristik dataset Kita.
3. **Interpretasi Hasil:**
   - Setelah Kita mendapatkan label outlier, Kita dapat menggunakan informasi ini untuk analisis lebih lanjut. Label 1 menunjukkan bahwa titik data tersebut dianggap sebagai outlier.

Cek jumlah antara LOF 0 (normal) dan LOF 1 (anomali)

[53]:
```
unique_labels, label_counts = np.unique(outliers, return_counts=True)
print("Label Counts:", dict(zip(unique_labels, label_counts)))
```

```
Label Counts: {0: 90000, 1: 10000}
```

[ ]:
```
# Membuat model LOF dengan treshold
lof_model = LOF(contamination=0.05, n_neighbors=40)

# Fitting model pada data hasil Scaled
outliers = lof_model.fit_predict(credit_pca20)
```

[ ]:
```
unique_labels, label_counts = np.unique(outliers, return_counts=True)
print("Label Counts:", dict(zip(unique_labels, label_counts)))
```

[ ]:
```
# Menghitung nilai LOF
lof_scores = lof_model.decision_function(credit_pca20)

# Menampilkan distribusi nilai LOF
```

```
plt.figure(figsize=(8, 6))
sns.histplot(lof_scores, bins=50, kde=True, color='skyblue')
plt.title('Distribution of LOF Scores')
plt.xlabel('LOF Score')
plt.ylabel('Frequency')
plt.show()
```

0 mewakili inliers (data normal), 1 mewakili outliers (data anomali).

Dalam hal ini, Anda dapat menganggap data dengan nilai LOF 1 sebagai anomali. Dengan memeriksa distribusi nilai LOF, Anda dapat melihat seberapa baik model dapat memisahkan inliers dan outliers berdasarkan skor LOF yang diberikan. Jika distribusinya lebih terfokus di sekitar nilai 1, itu mungkin menunjukkan bahwa model secara efektif memisahkan data menjadi dua kategori tersebut.

7.2 Cek Index Anomali

[ ]:
```
# cek index yang dideteksi merupakan anomali
outlier_indices = np.where(outliers == 1)[0]
outlier_indices
```

7.3 Visualisasi Plot LOF

[ ]:
```
# menampilkan plot anomali
plt.figure(figsize=(10, 6))
sns.scatterplot(x=credit_pca20['PC1'],
                y=credit_pca20['PC2'],
                hue=outliers,
                palette='coolwarm')
plt.title('Hasil Local Outlier Factor')
plt.xlabel(f'PC 1 ({pca.explained_variance_ratio_[0]*100:.2f}%)')
plt.ylabel(f'PC 2 ({pca.explained_variance_ratio_[1]*100:.2f}%)')
plt.show()
```

[ ]:
```
# Plot hasil LOF menggunakan Plotly Express
fig = px.scatter(credit_pca20, x='PC1', y='PC2',
                 color=outliers.astype(str),
                 color_discrete_map={'0': '#a6c4ff', '1': '#ffa07a'},
                 title='LOF Results',
                 labels={'PC1': f'PC 1 ({pca.explained_variance_ratio_[0]*100:.
 ↪2f}%)',
                         'PC2': f'PC 2 ({pca.explained_variance_ratio_[1]*100:.
 ↪2f}%)'})

# Menampilkan plot
fig.update_layout(width=800, height=600)
fig.show()
```

7.4 LOF Menggunakan Treshold

7.4.1 Buat model LOF menggunakan treshold 500 observasi

Kita coba pada sebagian dataset, yakni 500 observasi dan menggunakan semua PC

[ ]:
```
credit_lof = credit_pca20.iloc[0:500,:]
credit_lof.shape
```

[ ]:
```
# Set seed (kunci)
np.random.seed(123)

# Set the outlier detection and thresholding methods
lof_model1 = LOF(contamination= 0.05,
                 n_neighbors = 40)
outliers1 = lof_model1.fit_predict(credit_lof)
```

Cek jumlah antara LOF 0 (normal) dan LOF 1 (anomali)

[ ]:
```
unique_labels1, label_counts1 = np.unique(outliers1, return_counts=True)
print("Label Counts:", dict(zip(unique_labels1, label_counts1)))
```

7.4.2 Visualisasi Plot LOF (treshold)

[ ]:
```
# Plot hasil LOF menggunakan Plotly Express
fig = px.scatter(credit_lof, x='PC1', y='PC2',
                 color=outliers1.astype(str),
                 color_discrete_map={'0': '#a6c4ff', '1': '#ffa07a'},
                 title='LOF Results',
                 labels={'PC1': f'PC 1 ({pca.explained_variance_ratio_[0]*100:.
 ↪2f}%)',
                         'PC2': f'PC 2 ({pca.explained_variance_ratio_[1]*100:.
 ↪2f}%)'})

# Menampilkan plot
fig.update_layout(width=800, height=600)
fig.show()
```

7.4.3 Cek Index Anomali

[ ]:
```
anomaly_indices = np.where(outliers1 == 1)[0]
anomaly_data = credit_lof.iloc[anomaly_indices]
anomaly_data
```

Kode diatas bertujuan untuk menemukan indeks indeks dari data yang dianggap sebagai anomali oleh model LOF. Setelah itu, data anomali tersebut diekstrak dari DataFrame `credit_lof` yang merupakan hasil PCA.

Interpretasi hasilnya dapat dilakukan sebagai berikut:

1. **anomaly_indices**: Array ini berisi indeks-indeks dari anomali dalam array **outliers1**. Indeks-indeks ini sesuai dengan baris-baris dalam DataFrame **credit_lof** yang merupakan data hasil dari PCA.
2. **anomaly_data**: DataFrame ini berisi data aktual (baris-baris) dari DataFrame **credit_lof** yang diidentifikasi sebagai anomali oleh model LOF. DataFrame ini sudah berada dalam ruang hasil transformasi PCA.

Interpretasi dari hasil ini melibatkan pemahaman terhadap karakteristik atau pola yang membuat titik-titik ini dianggap sebagai anomali. Anda mungkin ingin menganalisis nilai-nilai fitur dalam DataFrame **anomaly_data** dan membandingkannya dengan distribusi keseluruhan data. Fitur-fitur dengan nilai ekstrem atau pola yang sangat berbeda dari mayoritas data mungkin menjadi indikator anomali.

Berikut langkah-langkah yang bisa diambil:

- Periksa nilai-nilai fitur dalam **anomaly_data** dan bandingkan dengan data normal.
- Visualisasikan anomali dalam ruang hasil transformasi PCA untuk melihat apakah ada pola yang jelas.
- Periksa fitur-fitur asli dalam data anomali untuk memahami sifat anomali tersebut.

Misalnya, Anda dapat menggunakan teknik visualisasi seperti scatter plot atau visualisasi relevan lainnya untuk menjelajahi anomali lebih lanjut.

7.4.4 [Optional] Membandingkan kolom label dengan hasil lof

[ ]:
```
# Buat DataFrame baru dengan kolom fraud_bool dari credit_clean500
df_combined_500 = pd.concat([credit_lof, credit_clean.loc[0:
 ↪499,['fraud_bool']]], axis=1)
df_combined_500
```

[ ]:
```
df_combined_500['lof_score'] = outliers1
df_combined_500
```

[ ]:
```
(df_combined_500['fraud_bool'].value_counts() / len(df_combined_500)) * 100
```

[ ]:
```
# Buat scatter plot dengan ukuran titik berdasarkan kolom 'Outlier'
plt.figure(figsize=(10, 8))
sns.scatterplot(x='PC1', y='PC2', hue='fraud_bool', size='lof_score',
                sizes=(200, 30), data=df_combined_500, palette='coolwarm')
plt.title('LOF Score Distribution')
plt.show()
```

Dari visualisasi di atas kita dapat melihat bahwa transaksi asli dan penipuan mempunyai pola yang berbeda-beda. Semakin tinggi skor lof yang dimilikinya, maka titik tersebut semakin tebal dan menonjol.

Aturan praktis skor lof mengatakan bahwa jika skor LOF = 1, kemungkinan besar skor tersebut merupakan outlier. Entah bagaimana, ambang batas dapat disesuaikan dengan distribusi data. Mari kita lihat dulu statistik skor LOFnya.

hasil dari model LOF adalah 0 dan 1, ini menunjukkan bahwa setiap sampel dianggap sebagai outlier (1) atau bukan outlier (0) berdasarkan skor LOF yang dihitung. Secara umum, model LOF pada PyOD akan memberikan label -1 untuk outlier dan 1 untuk sampel normal. Oleh karena itu, Anda mungkin perlu melakukan penyesuaian atau interpretasi tergantung pada bagaimana label outlier diatur pada dataset atau implementasi LOF yang digunakan.

[ ]:
```
print(df_combined_500['fraud_bool'].value_counts())
print(df_combined_500['lof_score'].value_counts())
```

[ ]:
```
# Hitung nilai untuk pie plot
fraud_counts = df_combined_500['fraud_bool'].value_counts()
lof_counts = df_combined_500['lof_score'].value_counts()

# Atur subplot
fig, axs = plt.subplots(1, 2, figsize=(12, 6))

# Pie plot untuk kolom 'fraud_bool'
axs[0].pie(fraud_counts, labels=fraud_counts.index, autopct='%1.1f%%',␣
 ↪startangle=90, colors=['#a6c4ff', '#ffa07a'])
axs[0].set_title('Distribusi Berdasarkan Fraud')

# Pie plot untuk kolom 'lof_score'
axs[1].pie(lof_counts, labels=lof_counts.index, autopct='%1.1f%%',␣
 ↪startangle=90, colors=['#a6c4ff', '#ffa07a'])
axs[1].set_title('Distribusi Berdasarkan LOF')

plt.show()
```

7.5 Important Note Local Outlier Factors (LOF)

Pada banyak kasus mengenai deteksi outlier atau anomali memang tidak dilengkapi dengan data yang ada label/targetnya. Namun, jika dataset yang digunakan memiliki label/target, kita bisa melihat dan menilai hasil kualitas dari deteksi tersebut. LOF disini sebetulnya berguna untuk menilai skor yang tidak biasa tanpa kita mengetahui kondisi ground truth / kondisi sebenarnya di lapangan berdasarkan pengamatannya. Untuk kasus ini, ia tidak tahu nilai sebenarnya dari suatu dataset apakah fraud/outlier atau tida

Beberapa catatan penting tentang memprediksi kondisi outlier atau anomali adalah terlalu sedikitnya kasus dalam mendeteksi penyakit atau perilaku yang berubah dengan cepat dalam transaksi penipuan. Jika kita menghadapi situasi seperti ini, kita tidak dapat hanya menggunakan algoritma pembelajaran mesin tanpa supervisi untuk mendeteksi. Beberapa teknik yang kokoh perlu dinilai untuk mendapatkan hasil yang lebih baik.

Local outlier factor akan menghasilkan angka dalam interval rasio. Jika skornya lebih dari 1, itu berarti kerapatan tetangga lebih tinggi. Interpretasi dari suatu rasio dapat disesuaikan berdasarkan rekomendasi bisnis pengguna daripada hanya mengartikan lof > 1 sebagai outlier. Jika pengguna ingin lebih selektif dalam mendeteksi pengamatan outlier, pengguna dapat meningkatkan ambang batas skor lof berdasarkan domain bisnis.

Additional Information

Berikut ini jika kita melakukan LOF pada data asli yang sudah di pilih kolom numerik saja dan sudah di scaled:

[ ]:
```
# Membuat model LOF dengan treshold
lof_model = LOF(contamination=0.05, n_neighbors=40)

# Fitting model pada data hasil Scaled
outliers = lof_model.fit_predict(credit_scaled)
```

[ ]:
```
# cek index yang dideteksi merupakan anomali
outlier_indices = np.where(outliers == 1)[0]
outlier_indices
```

[ ]:
```
unique_labels, label_counts = np.unique(outliers, return_counts=True)
print("Label Counts:", dict(zip(unique_labels, label_counts)))
```

[ ]:
```
# Membuat scatter plot menggunakan plotly.graph_objects
fig = go.Figure()

# Menambahkan trace scatter
scatter = go.Scatter(
    x=credit_scaled['credit_risk_score'].squeeze(),
    y=credit_scaled['proposed_credit_limit'].squeeze(),
    mode='markers',
    marker=dict(
        color=outliers,
        colorscale=[[0, '#a6c4ff'], [1, '#ffa07a']]
    ),
    text=credit_scaled.index,  # Menampilkan indeks pada hover
)

fig.add_trace(scatter)

# Menyesuaikan layout
fig.update_layout(
    title='Hasil Local Outlier Factor',
    xaxis=dict(title='credit_risk_score'),
    yaxis=dict(title='proposed_credit_limit'),
    hovermode='closest',
)
fig.update_layout(width=800, height=600)
# Menampilkan plot
fig.show()
```

8 Summary

Outlier adalah kondisi di mana suatu pengamatan dalam suatu dataset berbeda dari sisa data. Untuk mendeteksi apakah suatu pengamatan adalah outlier atau tidak, beberapa metode dapat digunakan.

Kita dapat menggunakan `local outlier factor` untuk menangani situasi di mana kita dapat menggunakan deteksi outlier dalam data multivariabel. Algoritma ini akan menggunakan perbandingan kepadatan pengamatan terhadap kepadatan pengamatan tetangga K-terdekat.

LOF membutuhkan K sebagai atribut untuk menghitung pengamatan tetangganya, dan dengan mensimulasikan K yang berbeda, algoritma ini akan menghasilkan skor lof yang berbeda. Semakin banyak nilai K yang digunakan, semakin stabil algoritma dapat mendeteksi outlier. Meskipun algoritma ini digunakan untuk mendeteksi pengamatan outlier, ia dapat digunakan untuk mendeteksi situasi anomali. Meskipun LOF dapat digunakan untuk menilai masalah pembelajaran mesin yang terawasi, perhatikan bahwa beberapa latar belakang kasus bisnis perlu diperhatikan, seperti proporsi dataset yang tidak seimbang.

Setelah data PCA, kita tidak bisa lagi mengetahui secara pasti kolom-kolom mana yang signifikan memberikan identitas bahwa observasi tersebut anomali. Namun, kita masih bisa memperkirakannya dengan menggunakan beberapa metode, antara lain:

Analisis komponen utama. Analisis komponen utama dapat digunakan untuk melihat bagaimana masing-masing komponen utama berkontribusi terhadap anomali. Komponen utama yang memiliki kontribusi yang signifikan terhadap anomali dapat dianggap sebagai kolom-kolom yang signifikan memberikan identitas anomali.

- Misalkan, kita memiliki data PCA dengan dua komponen utama, yaitu PC1 dan PC2. Jika komponen PC1 memiliki kontribusi yang signifikan terhadap anomali, maka kita dapat memperkirakan bahwa kolom-kolom yang memiliki korelasi yang tinggi dengan PC1 juga signifikan memberikan identitas anomali.

9 Glossary

- *Variance*: Ukuran seberapa jauh sebuah kumpulan bilangan tersebar. Nilai variance yang mendekati 0 menunjukkan data tidak banyak beragam.
- *Std (Standard Deviation)*: Akar kuadrat dari variance, dapat digunakan untuk menentukan keragaman data. Semakin kecil nilai *Std* maka data semakin tidak beragam dan sebaliknya.
- *Covariance*: Nilai yang menunjukkan hubungan antara variansi pada variable X dan variansi pada variable Y. Tidak dapat di interpretasikan, dikarenakan memiliki nilai negatif tak hingga sampai positif tak hingga.
- *Correlation*: Nilai yang menunjukkan hubungan antar variable X dengan variable Y. Memiliki rentang nilai -1 sampai 1. Nilai korelasi mendekati 1 menujukkan variable X dan Y memiliki hubungan positif kuat, sedangkan nilai korelasi mendekati -1 menunjukkan variable X dan Y memiliki hubungan negatif kuat.

10 Referensi

- In Depth: Principal Component Analysis (Jake VanderPlas)
- Intuisi PCA

- PCA (chapter 3)
- The Importance of Feature Scaling before PCA