

Annissa Claryta Berliana Putri (50421177)

CREDITA

Aplikasi Pendeteksi Kelayakan Debitur

Latar Belakang

Tahukah anda?

Pengajuan **kredit** oleh debitur **meningkat**



Hal ini bisa menjadi **peluang** bagi pihak yang memiliki **usaha kredit** untuk meningkatkan pendapatan

***Kreditur** : pihak yang memberikan pinjaman atau kredit | **Debitur** : pihak yang menerima pinjaman kredit

Masih banyak orang **mengajukan kredit** tetapi **tidak mampu membayar** tagihan

Masalah



Hal ini dapat **menurunkan** pendapatan kreditur



menyebabkan perusahaan mengalami **kerugian**



dan pada akhirnya **bangkrut**

Tantangan

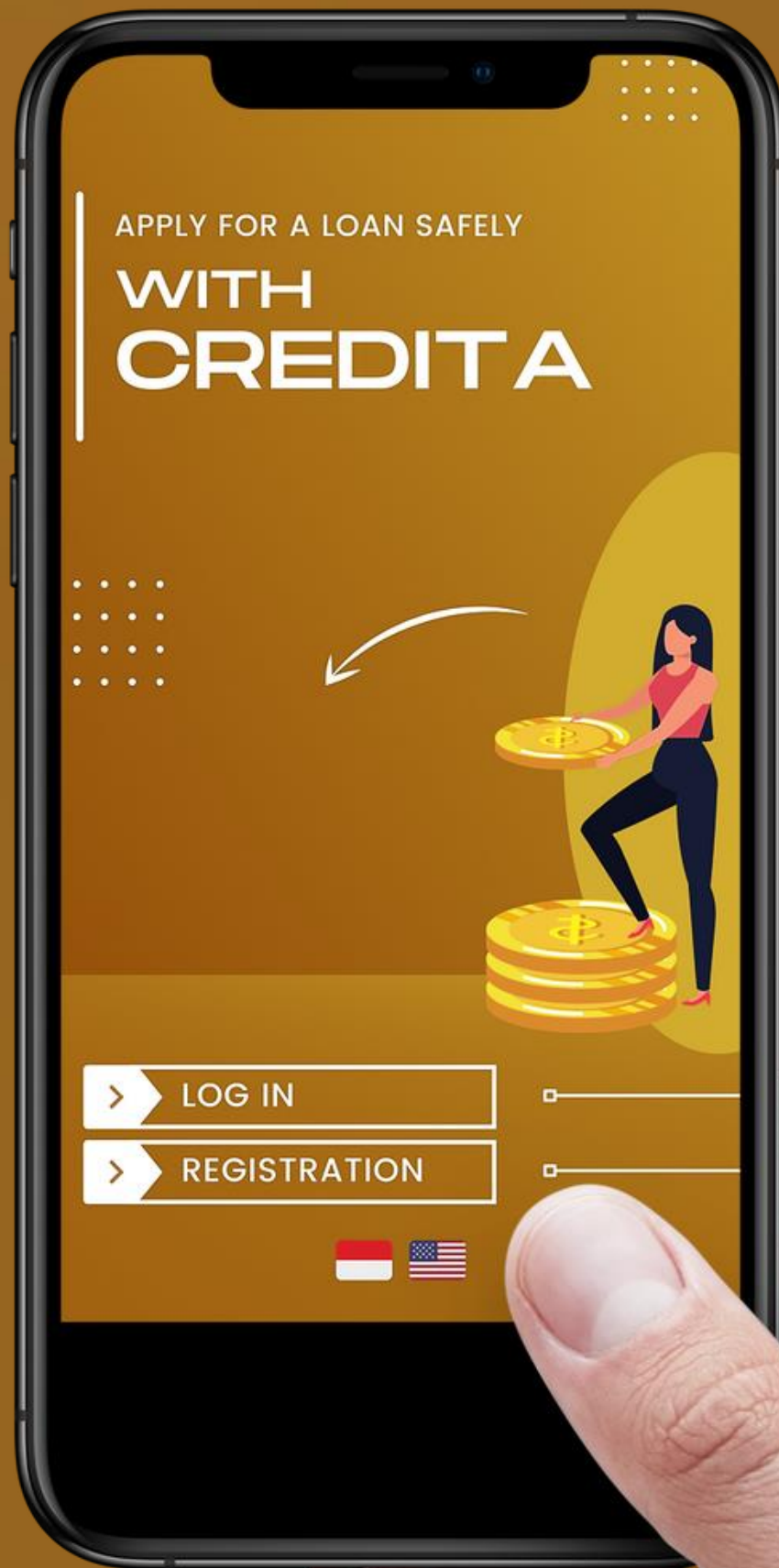
Bagaimana untuk **mengoptimalkan** penerimaan pengajuan kredit?

Solusi

Dengan melakukan **pengolahan data debitur** yang telah melakukan pinjaman...

Aplikasi **CREDITA**

Pendeteksi Kelayakan Pengajuan Kredit oleh Debitur dengan menggunakan Teknologi
Machine Learning



Implementasi **Machine Learning** dalam Bidang **Keuangan**

Metode

Teknologi **Machine Learning** adalah mesin yang dikembangkan untuk bisa **belajar dengan sendirinya** tanpa arahan dari penggunanya

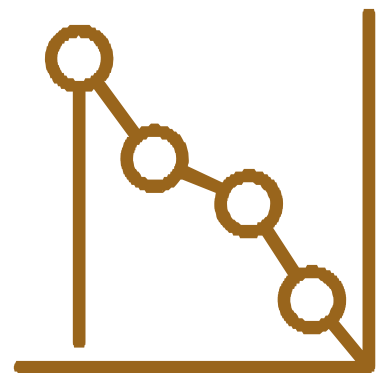
Perkembangan ML **sangat pesat** dan penggunaannya dapat diimplementasikan dalam **berbagai hal**



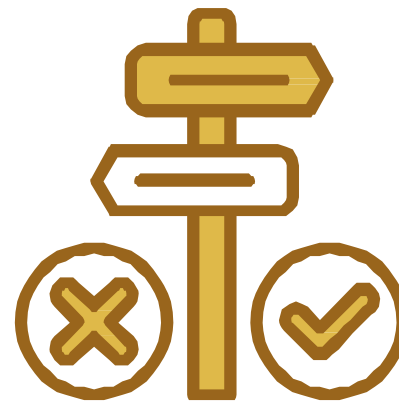
Keuangan



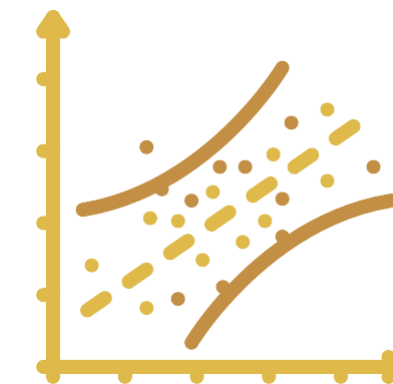
Dengan membandingkan **tiga model...**



**Logistic
Regression**



Decision Tree

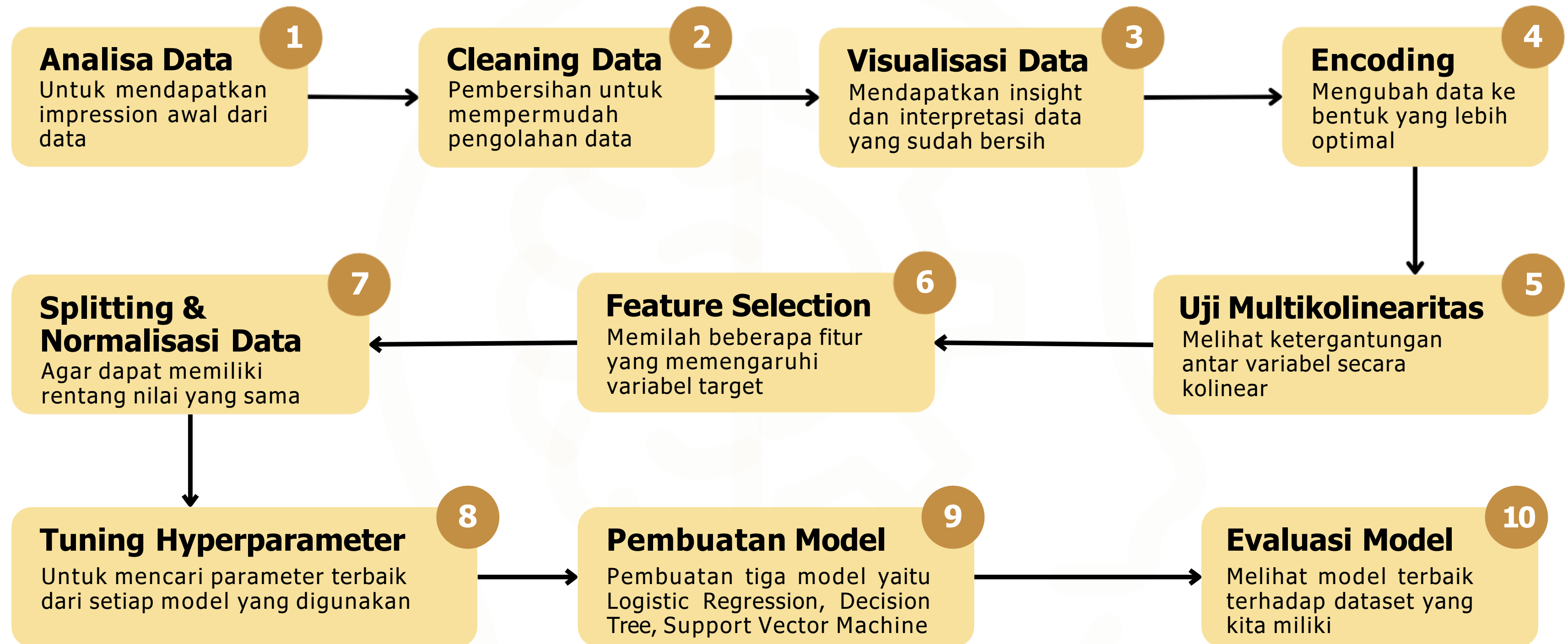


**Support
Vector
Machine**

dan akan dicari **model terbaik** untuk prediksi yang lebih **akurat** dan **tepercaya**

Langkah-langkah **Pembuatan Model** pada **Machine Learning**

Metode



Data debitur pada Dataset

Analisis Data

#	Column	
---	-----	
0	person_age	Umur (tahun)
1	person_income	Pendapatan Tahunan
2	person_home_ownership	Kepemilikan Rumah
3	person_emp_length	Lama Kerja (tahun)
4	loan_intent	Niat Pinjaman
5	loan_grade	Kelas Pinjaman
6	loan_amnt	Jumlah Pinjaman
7	loan_int_rate	Suku Bunga
8	loan_status	Status Pinjaman (non default: 0, default: 1)
9	loan_percent_income	Persen Pendapatan
10	cb_person_default_on_file	Histori Default
11	cb_person_cred_hist_length	Panjang Riwayat Kredit

Ukuran data : **32581 baris x 12 kolom**

Variabel **Dependent** dan **Independent**

Analisis Data

	person_age	person_income	person_home_ownership	person_emp_length	loan_intent	loan_grade	loan_amnt	loan_int_rate	loan_status	loan_percent_income	cb_person_default_on_file	cb_person_cred_hist_length
0	22	59000	RENT	123.0	PERSONAL	D	35000	16.02	1	0.59	Y	3
1	21	9600	OWN	5.0	EDUCATION	B	1000	11.14	0	0.10	N	2
2	25	9600	MORTGAGE	1.0	MEDICAL	C	5500	12.87	1	0.57	N	3
3	23	65500	RENT	4.0	MEDICAL	C	35000	15.23	1	0.53	N	2
4	24	54400	RENT	8.0	MEDICAL	C	35000	14.27	1	0.55	Y	4

Variabel Target
(Dependent)

Keterangan Variabel Target: **loan_status**

- ✓ 1 (non-default) maka debitur tersebut **tidak berisiko**
- ✗ 2 (default) menandakan debitur tersebut **berisiko**



Kolom lainnya akan menjadi **variabel bebas** (Independent)

Karakteristik Statistik Data

Analisis Data

Menunjukkan rangkuman statistik seperti rata-rata, median, dan data statistik lainnya

	person_age	person_income	person_emp_length	loan_amnt	loan_int_rate	loan_status	loan_percent_income	cb_person_cred_hist_length
count	32581.000000	3.258100e+04	31686.000000	32581.000000	29465.000000	32581.000000	32581.000000	32581.000000
mean	27.734600	6.607485e+04	4.789686	9589.371106	11.011695	0.218164	0.170203	5.804211
std	6.348078	6.198312e+04	4.142630	6322.086646	3.240459	0.413006	0.106782	4.055001
min	20.000000	4.000000e+03	0.000000	500.000000	5.420000	0.000000	0.000000	2.000000
25%	23.000000	3.850000e+04	2.000000	5000.000000	7.900000	0.000000	0.090000	3.000000
50%	26.000000	5.500000e+04	4.000000	8000.000000	10.990000	0.000000	0.150000	4.000000
75%	30.000000	7.920000e+04	7.000000	12200.000000	13.470000	0.000000	0.230000	8.000000
max	144.000000	6.000000e+06	123.000000	35000.000000	23.220000	1.000000	0.830000	30.000000

terdapat kemungkinan **outliers** dimana...

Data Maksimum dari person_age sebesar **144 tahun**

Data Maksimum dari person_emp_length sebesar **123 tahun**



Mendeteksi Missing Value

Cleaning Data

Missing value merupakan **nilai atribut yang kosong** pada objek data

🔍 Pencarian

```
[ ] #mencari jumlah missing value pda data
df.isna().sum()
```

```
person_age          0
person_income       0
person_home_ownership 0
person_emp_length   895
loan_intent          0
loan_grade          0
loan_amnt           0
loan_int_rate       3116
loan_status         0
loan_percent_income 0
cb_person_default_on_file 0
cb_person_cred_hist_length 0
dtype: int64
```

Nilai pada data
Hilang!

Terdapat **2 kolom** yang mengandung missing value

person_emp_length & loan_int_rate

🔧 Penanganan

Akan dilakukan penanganan dengan
imputasi data

Missing value akan diganti dengan
nilai rata-rata data

```
#mengubah missing value person_emp_length menjadi rata-rata
persempmean = round(df['person_emp_length'].mean())
df['person_emp_length'] = df['person_emp_length'].fillna(persempmean)
```

```
#mengubah missing value loan_int_rate menjadi rata-rata
loanmean = df['loan_int_rate'].mean()
df['loan_int_rate'] = df['loan_int_rate'].fillna(loanmean)
```

Ketika dilakukan pengecekan missing value kembali...

```
person_age          0  loan_amnt          0
person_income       0  loan_int_rate       0
person_home_ownership 0  loan_status          0
person_emp_length   0  loan_percent_income 0
loan_intent         0  cb_person_default_on_file 0
loan_grade         0  cb_person_cred_hist_length 0
```

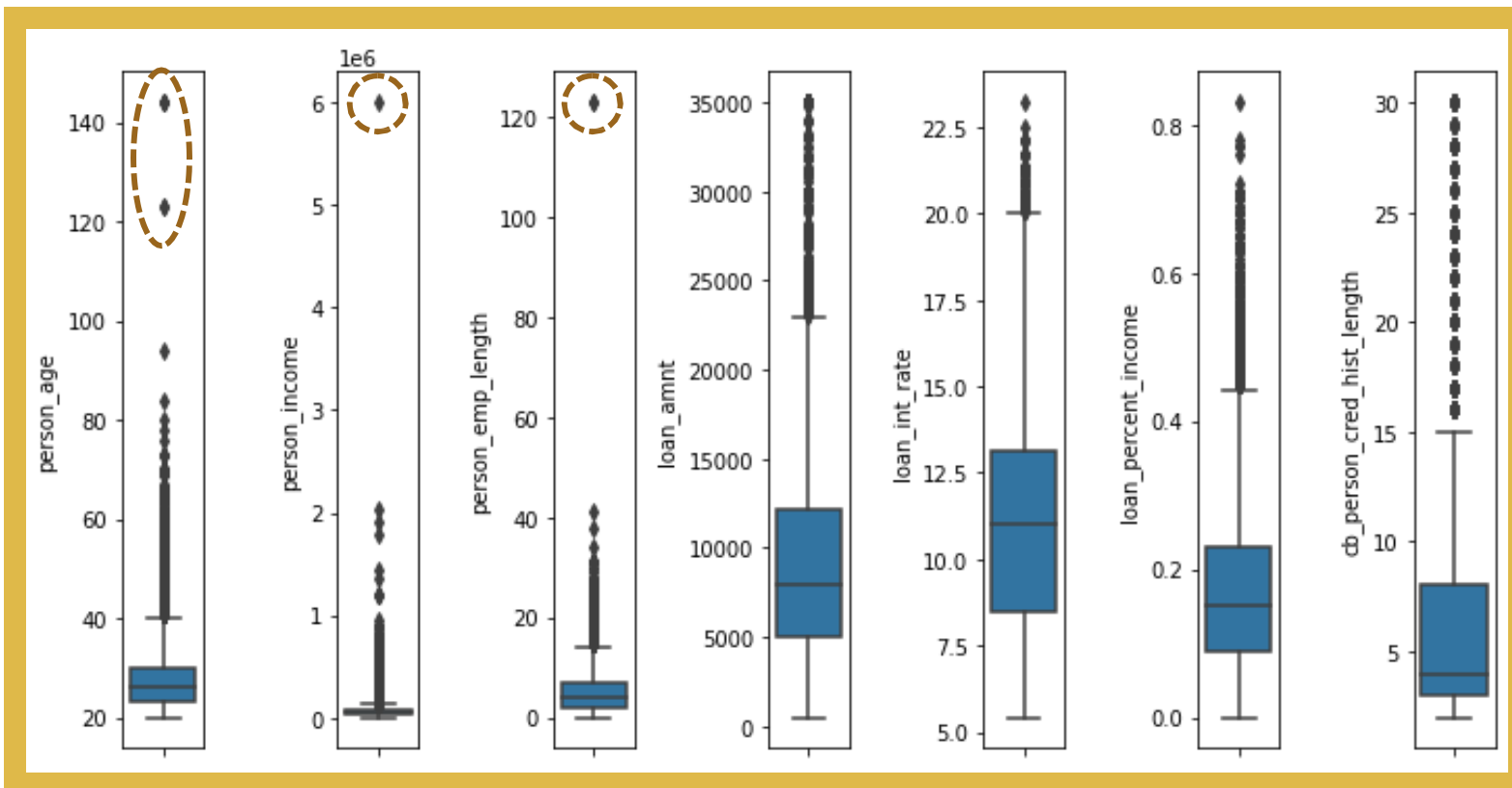
Sudah **tidak ada data kosong**

Mendeteksi Outliers

Cleaning Data

Outlier adalah data observasi yang muncul dengan **nilai-nilai ekstrim**

Pencarian



Terdeteksi **Nilai Ekstrim!**

Terdapat **3 kolom** yang mengandung outliers

person_age, person_income, person_emp_length

Penanganan

Akan dilakukan penanganan dengan **membuang outliers**

```
#memeriksa data person_income yang menjadi outlier
df.loc[df['person_income'] == 6000000]

#menghapus row
df.drop([32297], axis=0, inplace=True)
```

Drop data **person_income** sebesar 6000000

```
df.loc[df['person_age'] > 100]
#menghapus row
df.drop([81, 183, 575, 747], axis=0, inplace=True)
```

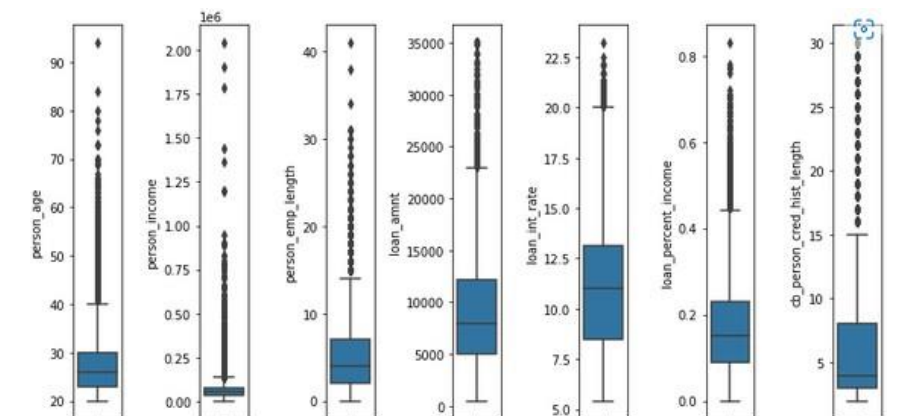
Drop data **person_age** yang lebih besar dari 100

```
df.loc[df['person_emp_length'] > df['person_age']]
#menghapus row
df.drop([0, 210], axis=0, inplace=True)
```

Drop data **person_emp_length** yang lebih besar dari person_age

Ketika dilakukan pengecekan outliers kembali...

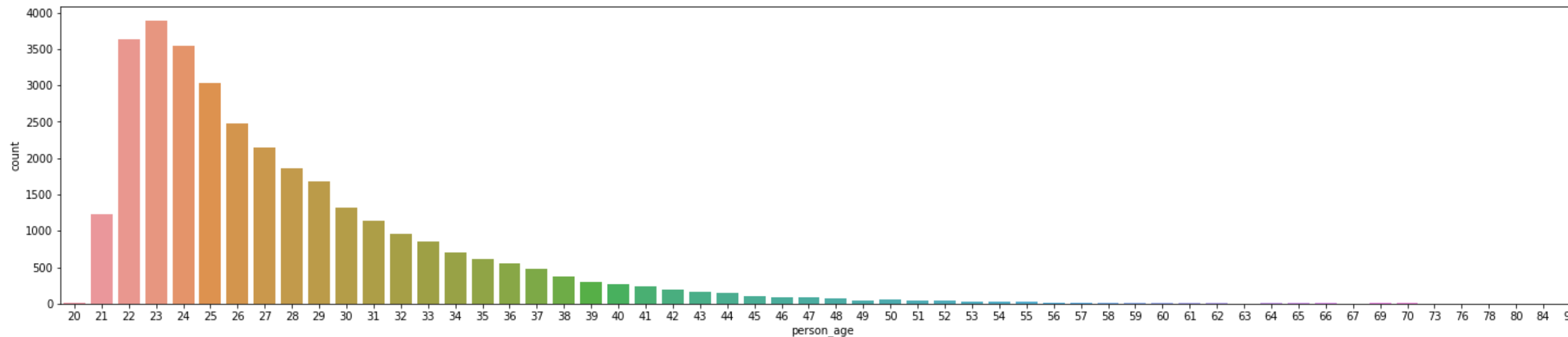
Sudah **tidak ada outliers**



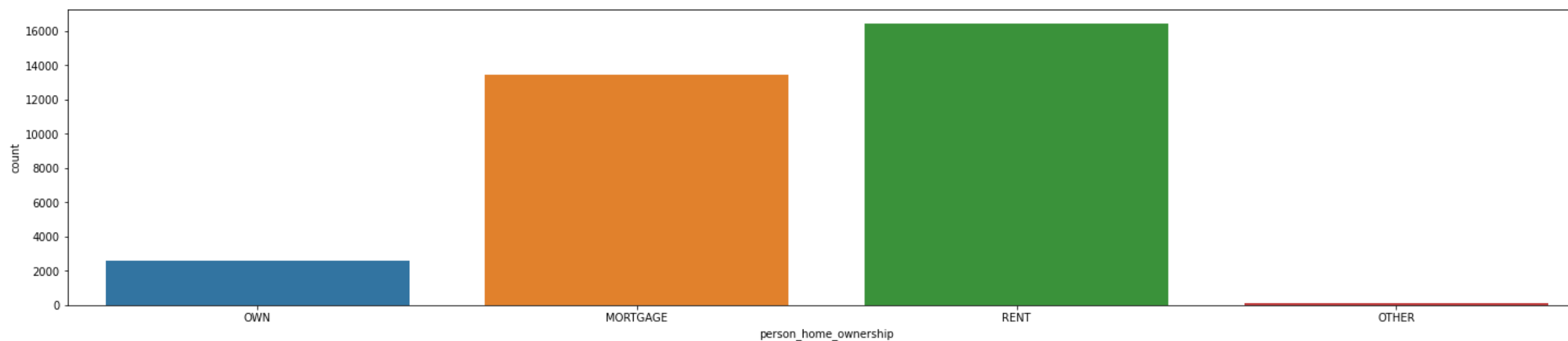
Melakukan **Interpretasi Data**

Mengkonversikan kumpulan data menjadi hal yang **lebih sederhana** untuk disajikan

Visualisasi Data



Sebagian besar debitur **berusia produktif** yakni pada rentang 21-40 tahun dan jumlah debitur terbanyak ada pada usia **23 tahun**

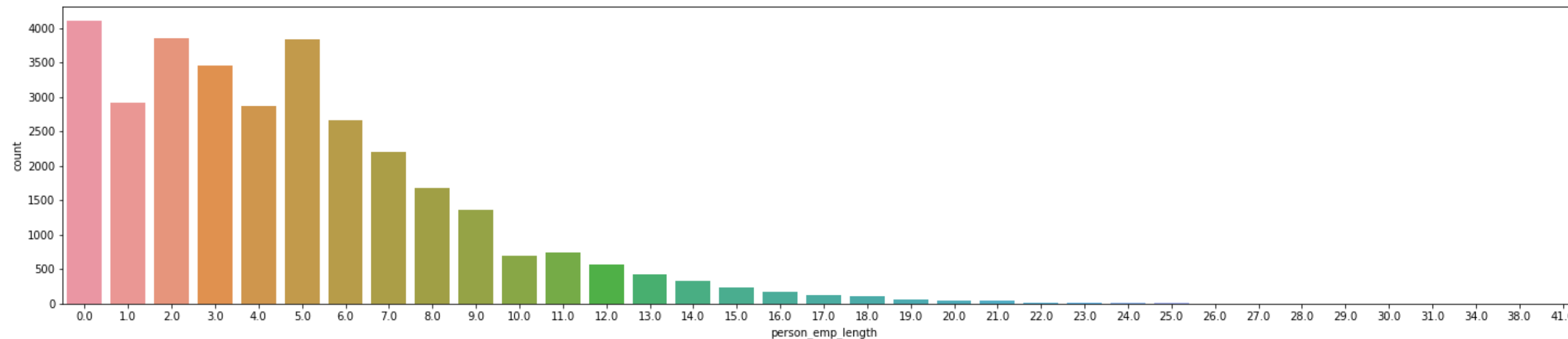


Urutan jenis kepemilikan rumah debitur dari tertinggi hingga terendah adalah **rent (sewa), mortgage (gadai), own (milik pribadi), dan other (lain-lain)**

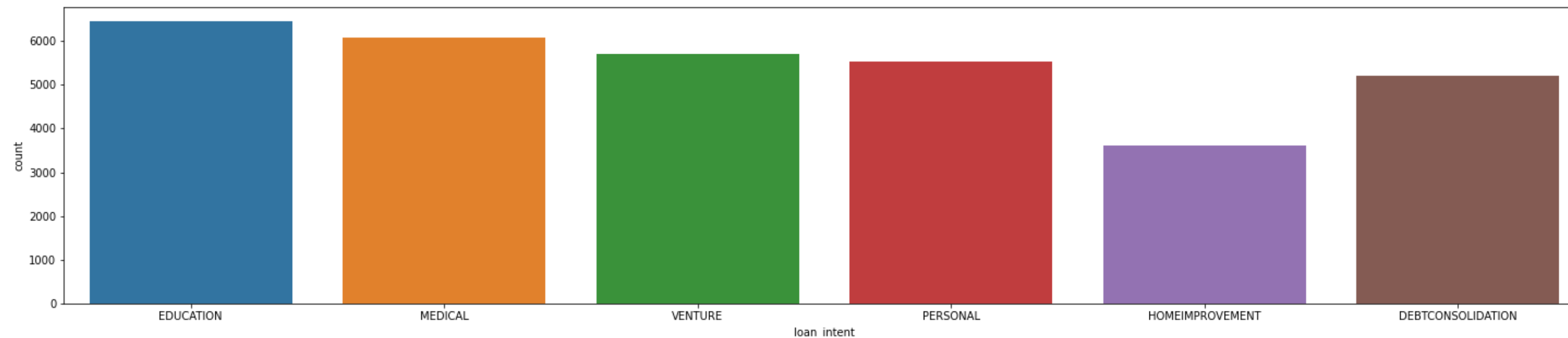
Melakukan **Interpretasi Data**

Mengkonversikan kumpulan data menjadi hal yang **lebih sederhana** untuk disajikan

Visualisasi Data



Debitur memiliki lama bekerja berkisar **0 hingga 20 tahun** dan debitur terbanyak memiliki lama kerja **0 tahun**

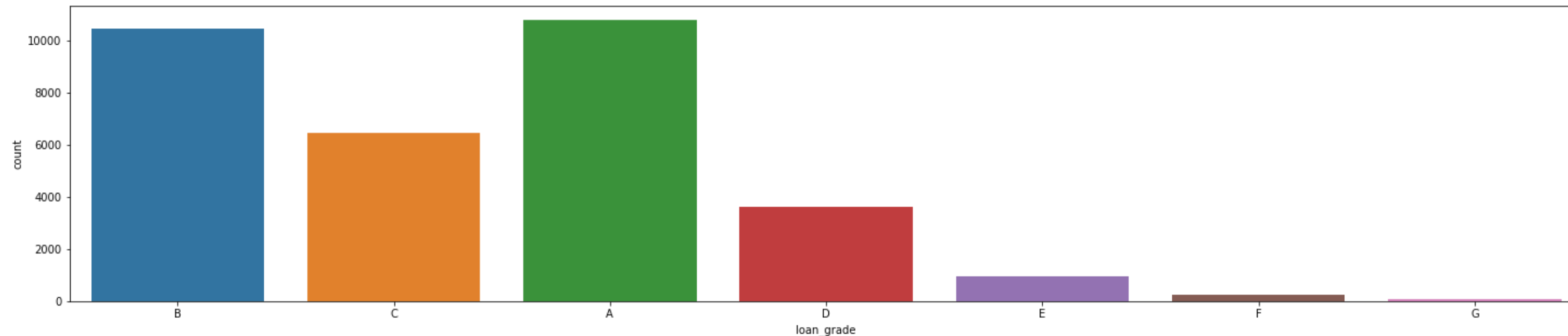


Mayoritas tujuan kredit adalah untuk **kebutuhan edukasi, diikuti biaya medis, usaha, kebutuhan pribadi, pembayaran hutang, dan renovasi rumah**

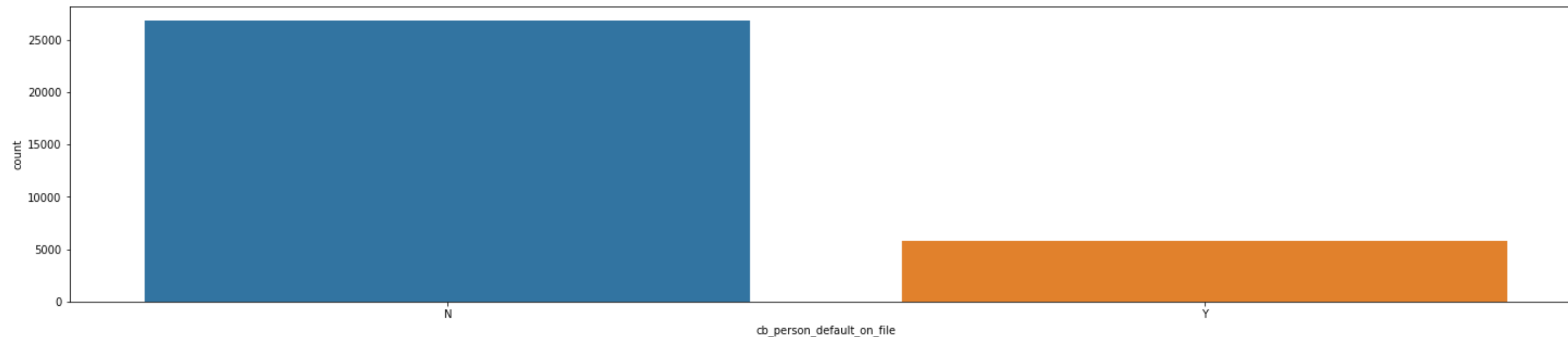
Melakukan **Interpretasi Data**

Mengkonversikan kumpulan data menjadi hal yang **lebih sederhana** untuk disajikan

Visualisasi Data



Grade tertinggi yang menjadi pilihan kreditur adalah **grade A** yang kemudian **disusul oleh grade B** dimana grade A memiliki risk yang paling kecil

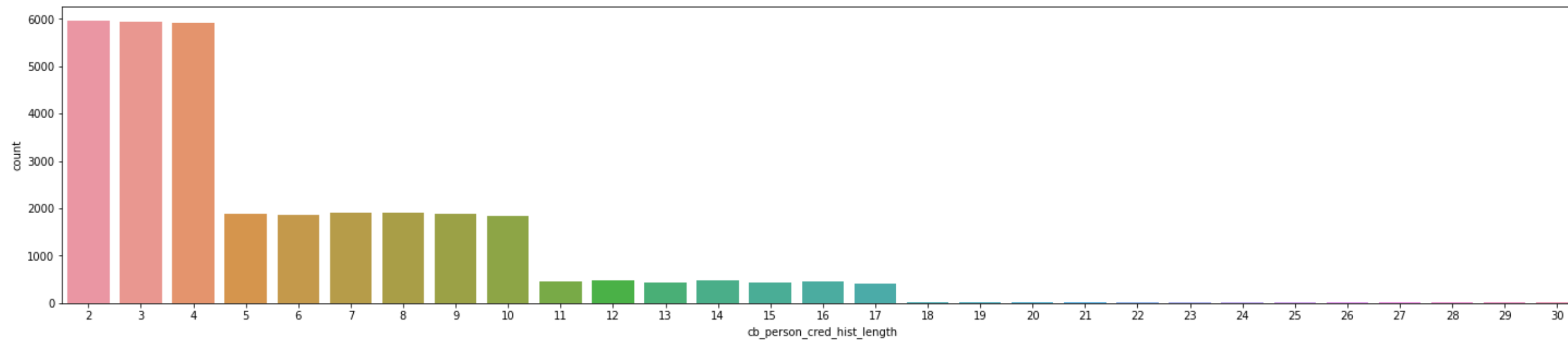


Lebih banyak debitur dengan riwayat **berhasil membayar pinjaman** dibanding dengan debitur yang pernah mengalami kegagalan pembayaran kredit

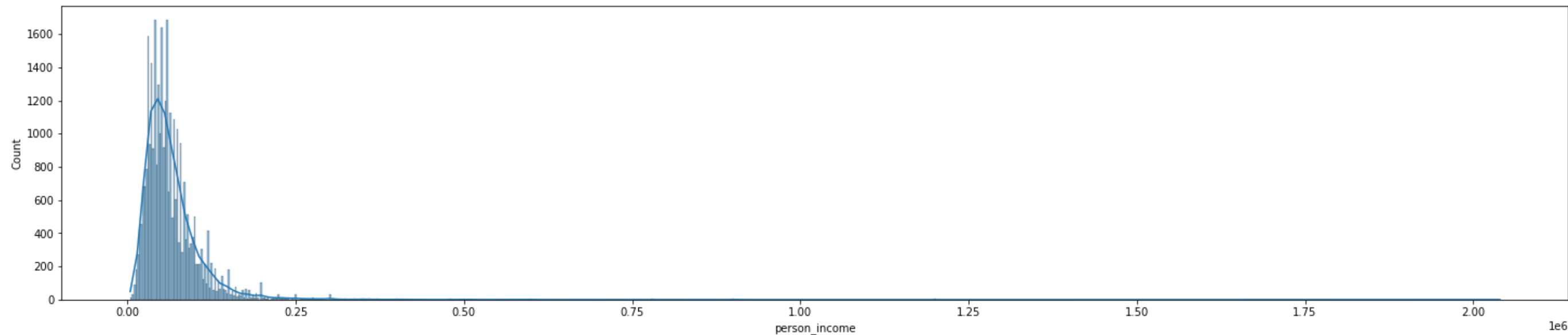
Melakukan **Interpretasi Data**

Mengkonversikan kumpulan data menjadi hal yang **lebih sederhana** untuk disajikan

Visualisasi Data



Mayoritas panjang riwayat kredit debitur berada di rentang waktu **2-4 tahun**

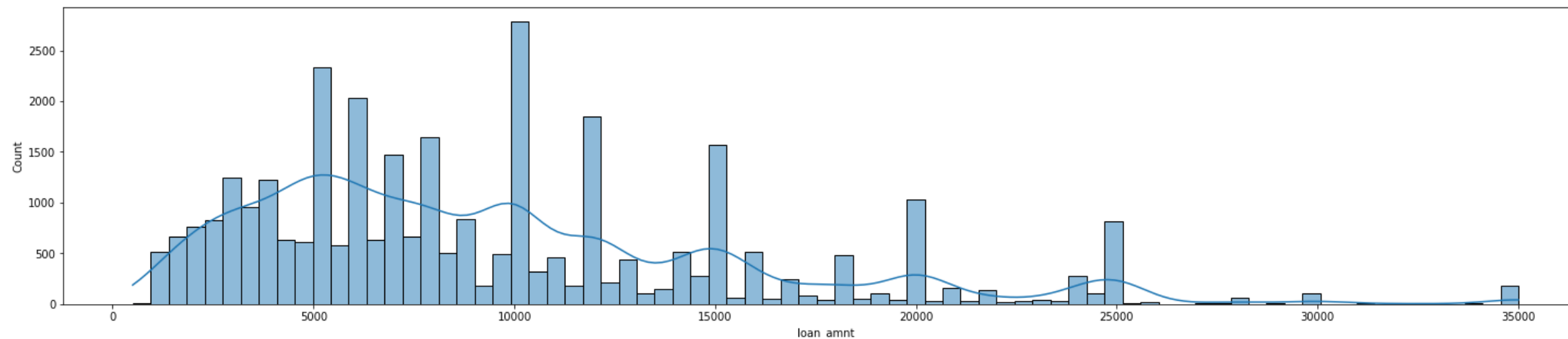


Pendapatan tahunan debitur berada pada rentang puluhan hingga ratusan ribu dan paling banyak berada di kisaran **60000**

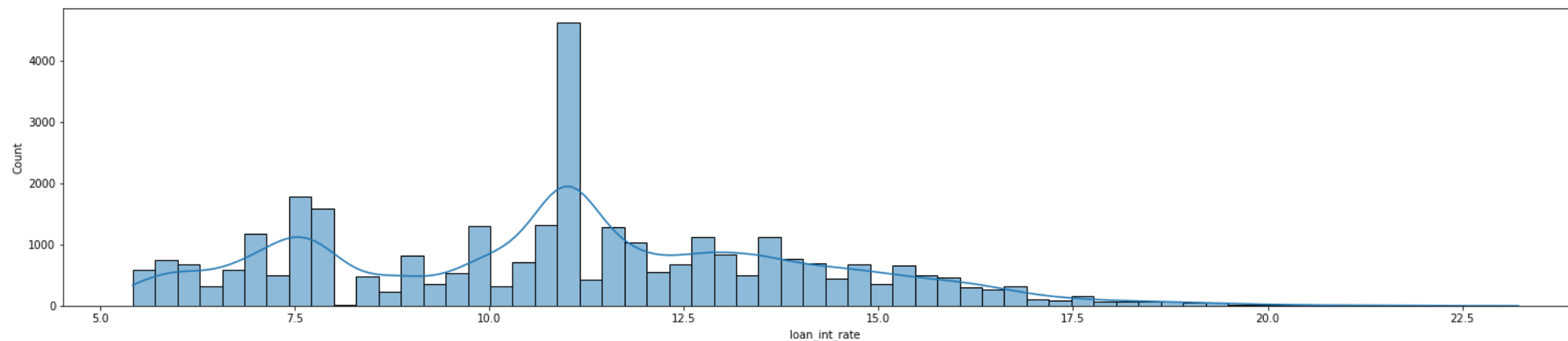
Melakukan **Interpretasi Data**

Mengkonversikan kumpulan data menjadi hal yang **lebih sederhana** untuk disajikan

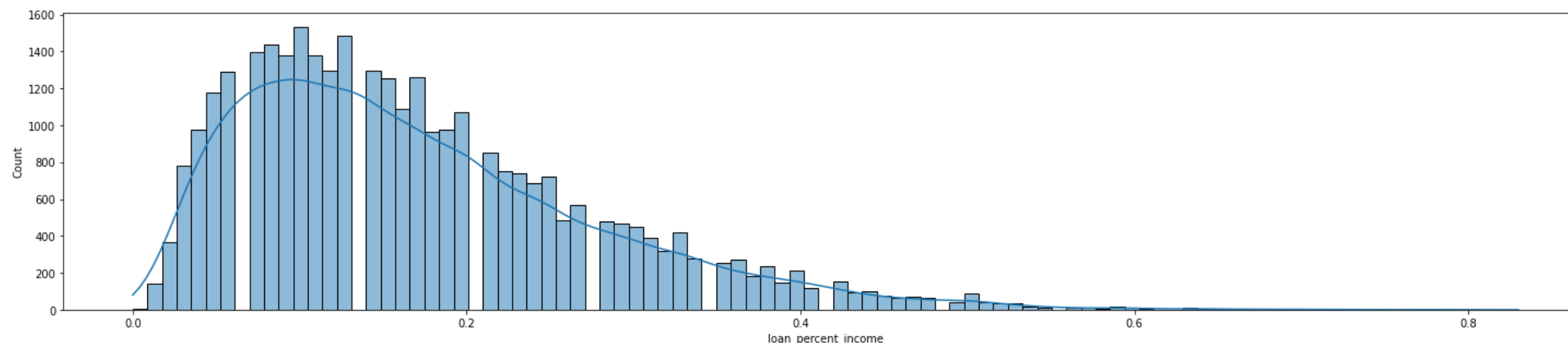
Visualisasi Data



Jumlah pinjaman tergolong bervariasi dengan pinjaman tertinggi berada di kisaran **10000**



Suku bunga pinjaman juga bervariasi berkisar dari **5.42% ke 23.2%** dengan suku bunga yang paling banyak dipilih adalah **11%**



Persenan pendapatan berada di rentang **0 hingga 0.83** dengan modus data dipegang oleh **0.1**

Melakukan **Encoding**

Mengubah fitur yang bertipe object menjadi numerik

Encoding

🔍 Pencarian

```
df.select_dtypes(include=['object'])
```

Hasil:

	person_home_ownership	loan_intent	loan_grade	cb_person_default_on_file
1	OWN	EDUCATION	B	N
2	MORTGAGE	MEDICAL	C	N
3	RENT	MEDICAL	C	N
4	RENT	MEDICAL	C	Y
5	OWN	VENTURE	A	N
...
32576	MORTGAGE	PERSONAL	C	N
32577	MORTGAGE	PERSONAL	A	N
32578	RENT	HOMEIMPROVEMENT	B	N
32579	MORTGAGE	PERSONAL	B	N
32580	RENT	MEDICAL	B	N

Terdapat 4 **kolom** yang bertipe object
**person_home_ownership, loan_intent,
loan_grade, cb_person_default_on_file**

🔑 Penanganan

Akan dilakukan penanganan dengan
melakukan encoding

One Hot Encoding dilakukan untuk kolom-kolom bertipe **nominal**

```
# OneHot encoding
i = ['person_home_ownership', 'loan_intent', 'cb_person_default_on_file']
OneHot = pd.get_dummies(df, columns = i)
df = OneHot
```

Ordinal Encoding dilakukan untuk kolom-kolom bertipe **ordinal**

```
# Ordinal encoding
df.loan_grade = pd.Categorical(df.loan_grade)
df['loan_grade'] = df.loan_grade.cat.codes
```

Ketika dilakukan pengecekan
tipe kembali...

Sudah **tidak ada data
yang bertipe object**

```
Data columns (total 21 columns):
#      Column      Non-Null Count  Dtype
---  -
0      person_age    32574 non-null  int64
1      person_income  32574 non-null  int64
2      person_emp_length  32574 non-null  int64
3      loan_grade      32574 non-null  int8
4      loan_amnt       32574 non-null  float64
5      loan_int_rate    32574 non-null  float64
6      loan_status      32574 non-null  int64
7      loan_percent_income  32574 non-null  float64
8      cb_person_cred_hist_length  32574 non-null  int64
9      person_home_ownership_MORTGAGE  32574 non-null  uint8
10     person_home_ownership_OTHER      32574 non-null  uint8
11     person_home_ownership_OWN        32574 non-null  uint8
12     person_home_ownership_RENT       32574 non-null  uint8
13     loan_intent_DEBTCONSOLIDATION     32574 non-null  uint8
14     loan_intent_EDUCATION             32574 non-null  uint8
15     loan_intent_HOMEIMPROVEMENT       32574 non-null  uint8
16     loan_intent_MEDICAL               32574 non-null  uint8
17     loan_intent_PERSONAL              32574 non-null  uint8
18     loan_intent_VENTURE               32574 non-null  uint8
19     cb_person_default_on_file_N       32574 non-null  uint8
20     cb_person_default_on_file_Y       32574 non-null  uint8
dtypes: float64(2), int64(6), int8(1), uint8(12)
memory usage: 2.6 MB
```

Melakukan Uji Multikolinearitas

Uji Multikolinearitas

Untuk melihat bahwa tidak ada fitur (variabel independen) yang **saling berkorelasi satu sama lain** dalam sebuah model regresi

🔍 Dengan Variance Inflation Factor (VIF)

	feature	VIF
10	person_home_ownership_OTHER	inf
11	person_home_ownership_OWN	inf
19	cb_person_default_on_file_N	inf
18	loan_intent_VENTURE	inf
17	loan_intent_PERSONAL	inf
16	loan_intent_MEDICAL	inf
15	loan_intent_HOMEIMPROVEMENT	inf
14	loan_intent_EDUCATION	inf
13	loan_intent_DEBTCONSOLIDATION	inf
12	person_home_ownership_RENT	inf
20	cb_person_default_on_file_Y	inf
9	person_home_ownership_MORTGAGE	inf
3	loan_grade	5.489208
5	loan_int_rate	4.840994
0	person_age	4.438182
8	cb_person_cred_hist_length	4.374340
7	loan_percent_income	2.824750
4	loan_amnt	2.637642
1	person_income	1.813296
6	loan_status	1.469172
2	person_emp_length	1.108154

🔑 Penjelasan

VIF's Score = inf, mengartikan bahwa terdapat korelasi sempurna pada di antara variabel yang ada sehingga variabel tersebut **redundant**.

Akan tetapi, kasus ini terjadi karna fitur yang memiliki **VIF's Score = inf** merupakan fitur hasil dari OneHot Encoding, sehingga memiliki korelasi yang sempurna. Maka fitur-fitur tersebut tetap akan digunakan.

Untuk fitur lainnya memiliki nilai multikolinearitas < 10 , maka **tidak terjadi multikolinearitas**

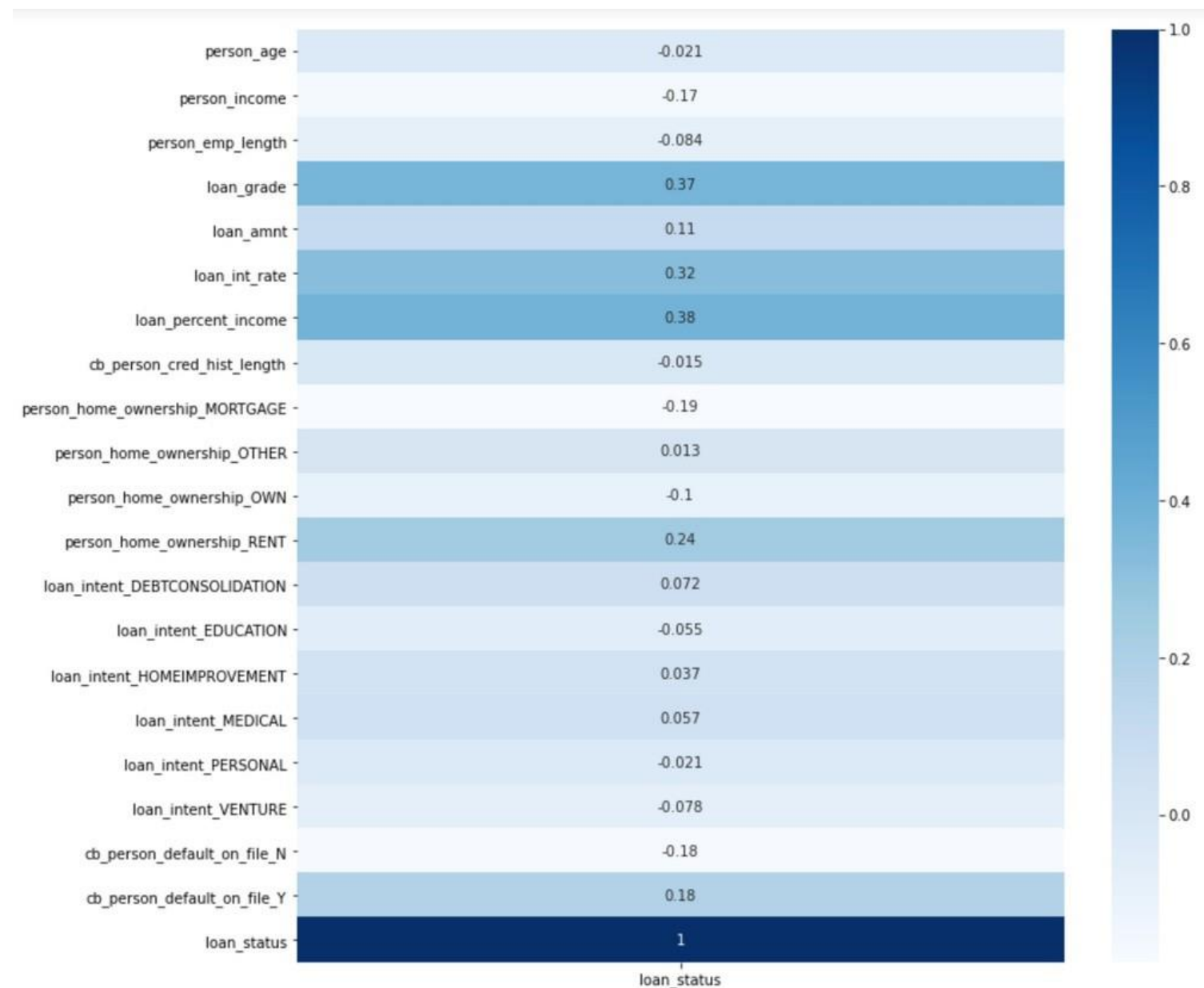
Kesimpulan

Kita dapat menggunakan **semua fitur** untuk tahap feature selection

Feature Selection dengan **Pearson Correlation**

Pearson Correlation merupakan metode yang digunakan untuk mencari hubungan antara independent variable dengan dependent variable.

🔍 Hasil



🔑 Penjelasan

Telah didapatkan korelasi antara fitur-fitur dengan variabel target yaitu 'loan status'. Kemudian akan dicari mean dari seluruh nilai korelasi dan variabel dengan korelasi yang lebih besar dari nilai mean akan dipilih.

Mean dari korelasi tiap variabel yang ada adalah **0.140147762993015**.
Sehingga, **variabel yang dipilih sebagai hasil seleksi fitur adalah**

person_income	0.168485
loan_grade	0.373042
loan_int_rate	0.319302
loan_percent_income	0.379250
person_home_ownership_MORTGAGE	0.187607
person_home_ownership_RENT	0.238416
cb_person_default_on_file_N	0.179002
cb_person_default_on_file_Y	0.179002

Splitting dan Normalisasi Data

Hasil

```
# set X and y
X = df[['person_income', 'loan_grade', 'loan_int_rate', 'loan_percent_income',
        'person_home_ownership_MORTGAGE', 'person_home_ownership_RENT',
        'cb_person_default_on_file_N', 'cb_person_default_on_file_Y']]
y = df['loan_status']
```

```
# train - test split dengan perbandingan 8 : 2
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```
# Normalisasi dengan MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
X_train = sc.fit_transform(X_train) # fit X_train
X_test = sc.transform(X_test) #transform X_test
```

Splitting dan Normalisasi Data

Penjelasan

X adalah dataframe berisi **fitur-fitur yang terpilih dari proses feature selection**

Y adalah **variabel target**

Membagi data train dan data tes dengan perbandingan **8:2**

26064

6516

Min Max Scaler dilakukan untuk **mengubah data numerik menjadi range [0,1]**

Melakukan **Tuning Hyperparameter**

Untuk **mencari parameter terbaik** dari setiap model yang digunakan

Tuning Hyperparameter

GridSearchCV

GridSearchCV adalah teknik untuk mencari **nilai parameter terbaik** dari kumpulan parameter grid yang diberikan

```
# scoring model and parameters, pencarian model dan parameter terbaik dengan menggunakan GridSearchCV
from sklearn.model_selection import GridSearchCV
for model_name, mp in model_params.items():
    clf = GridSearchCV(mp['model'], mp['params'], cv=3, return_train_score=False)
    clf.fit(X_train, y_train) # fit GridSearch ke X_train dan y_train
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })
best = pd.DataFrame(scores, columns=['model', 'best_score', 'best_params'])
```

	model	best_score	best_params
0	Logistic_Regression	0.850301	{'C': 0.1, 'multi_class': 'multinomial', 'pena...
1	decision_tree	0.892590	{'criterion': 'gini', 'max_depth': 8, 'max_fea...
2	SVM_Classifier	0.872558	{'gamma': 'scale', 'kernel': 'poly'}

Hasil

- **Model Logistic Regression**

Best Score :0.850301

Best Parameter :

'C': 0.1, 'multi_class': 'multinomial', 'penalty': 'none', 'solver': 'sag'

- **Model Decision Tree**

Best Score :0.892590

Best Parameter :

'criterion': 'gini', 'max_depth': 8, 'max_features': 'log2', 'splitter': 'best'

- **Model Support Vector Machine**

Best Score :0.872558

Best Parameter :

'gamma': 'scale', 'kernel': 'poly'

Logistic Regression

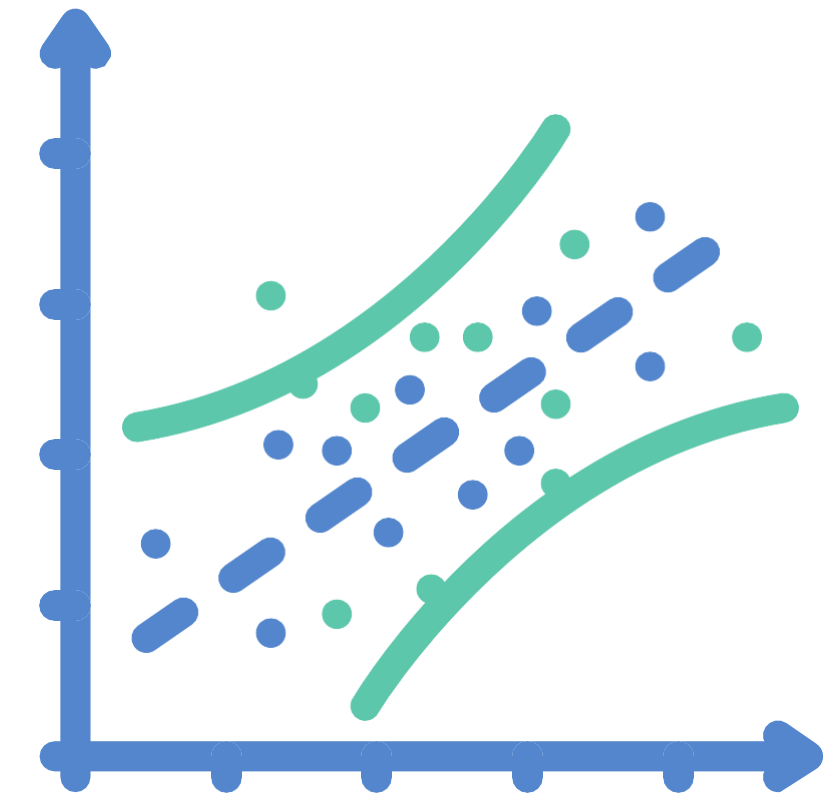
Pembuatan Model untuk **Klasifikasi**



Metode

- Regresi Logistik menggunakan **konsep regresi untuk** memprediksi suatu nilai yang bersifat **kategorik**
- Regresi Logistik memberikan output berupa **probabilitas suatu item** termasuk kelas tertentu

Pembuatan Model



Coding

```
# Membentuk Model Logistic Regression
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(C = 0.1, solver = 'sag', multi_class = 'ovr', penalty = 'none').fit(X_train, y_train)
```

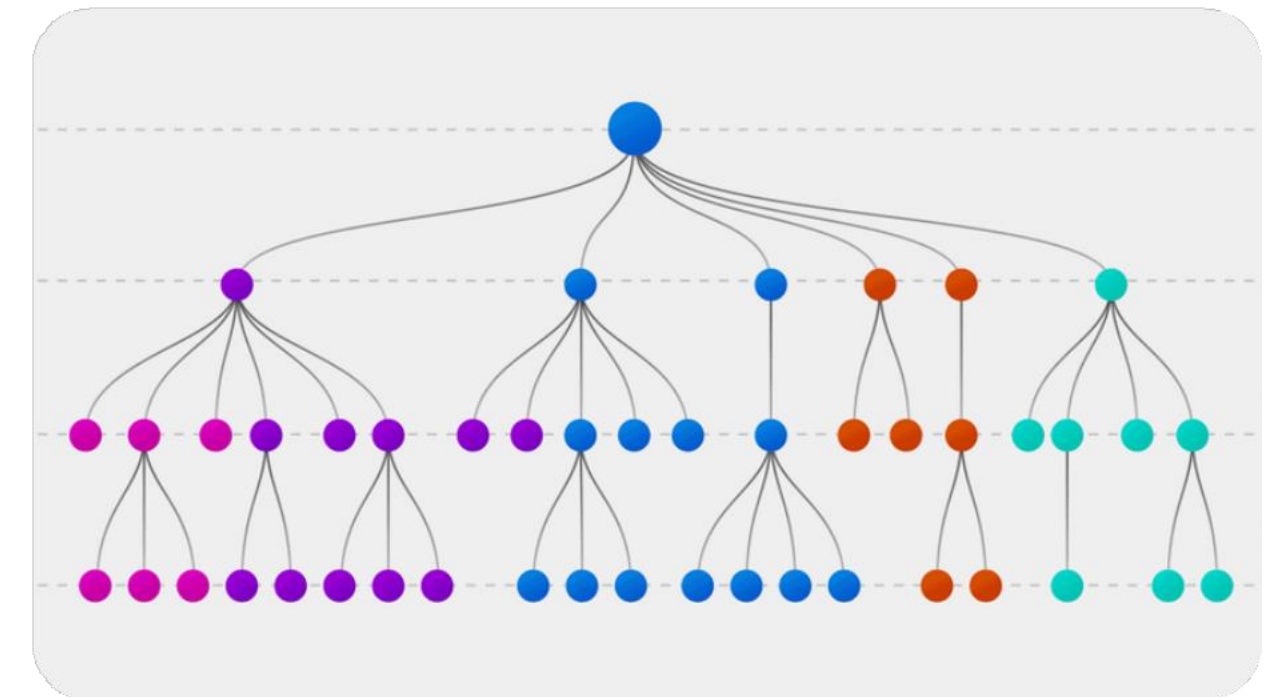
Decision Tree

Pembuatan Model untuk **Klasifikasi**

Pembuatan Model

Metode

- Decision tree adalah model prediksi menggunakan **struktur pohon** atau struktur berhirarki



Coding

```
# Membentuk Model Decision Tree
from sklearn.tree import DecisionTreeClassifier
credittree = DecisionTreeClassifier(criterion = 'entropy', max_depth = 8, random_state = 0, splitter = 'best')
```

Support Vector Machine (SVM)

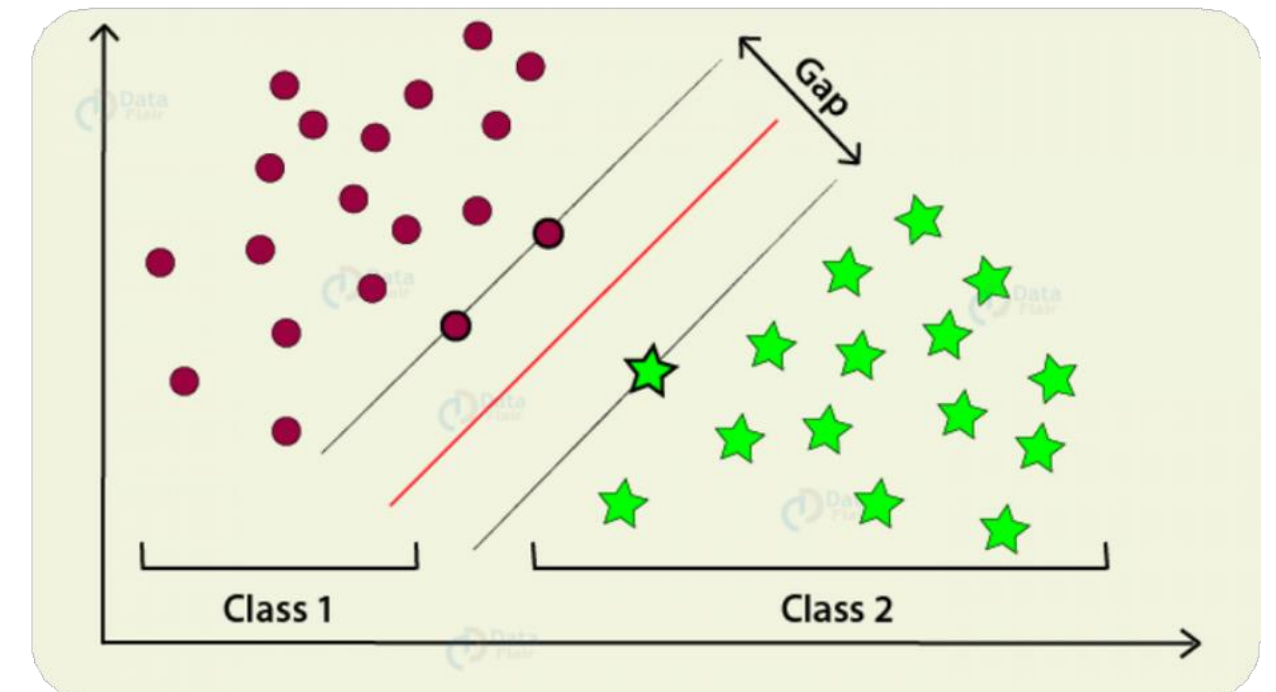
Pembuatan Model untuk **Klasifikasi**

Pembuatan Model



Metode

- SVM adalah algoritma supervised learning untuk **klasifikasi** dengan cara menemukan **separator** berupa **hyperplane**



Coding

```
# Membentuk Model Support Vector Machines (SVM)
from sklearn import svm
clf = svm.SVC(C=1, gamma = 'scale', kernel='poly')
```

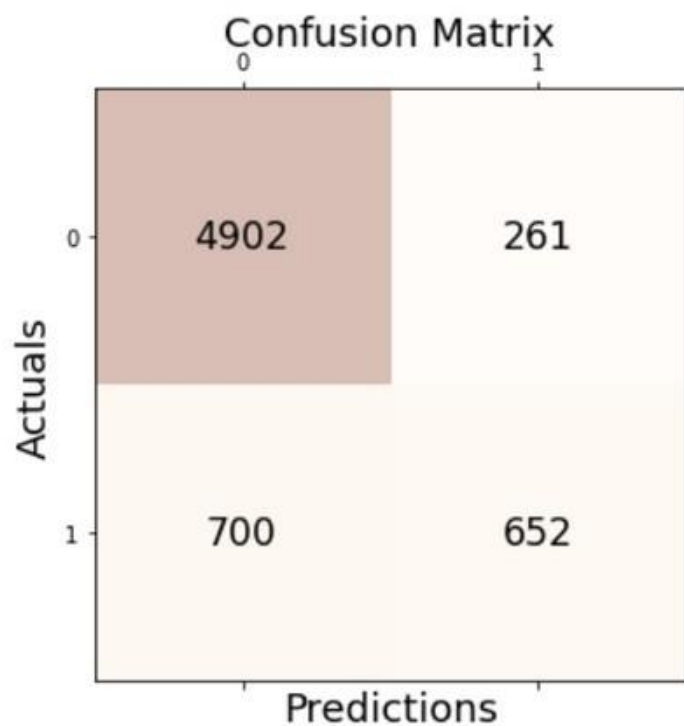
Evaluasi Model

Melihat Model terbaik pada dataset

Evaluasi Model

Logistic Regression

Confusion Matrix Logistic Regression Model



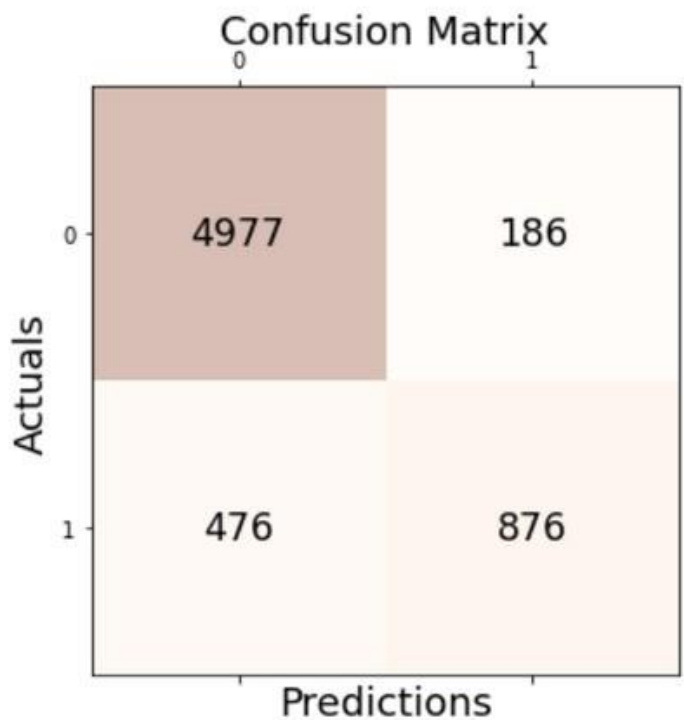
Classification Report Logistic Regression Model

	precision	recall	f1-score	support
0	0.88	0.95	0.91	5163
1	0.71	0.48	0.58	1352
accuracy			0.85	6515
macro avg	0.79	0.72	0.74	6515
weighted avg	0.84	0.85	0.84	6515

f1-score: 0.9107292150487692

Decision Tree

Confusion Matrix Decision Tree Model



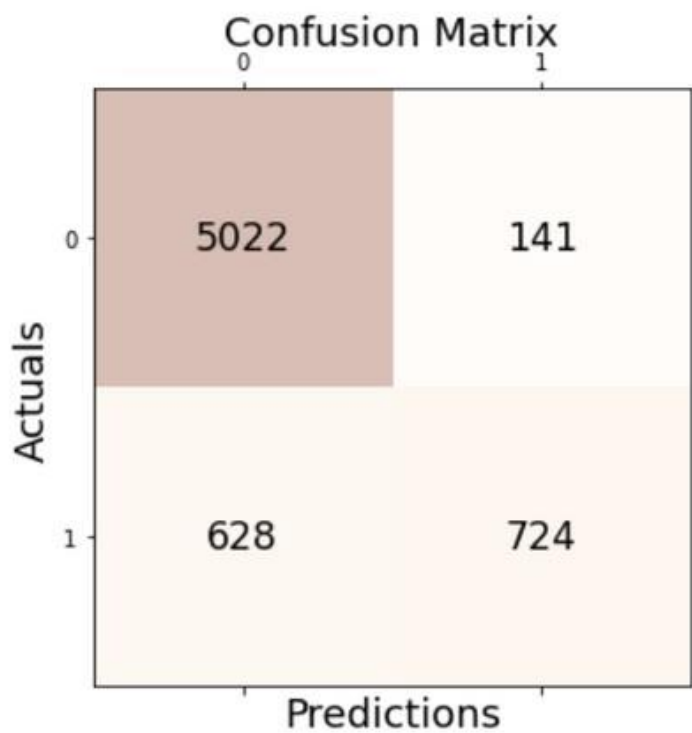
Classification Report Decision Tree Model

	precision	recall	f1-score	support
0	0.91	0.96	0.94	5163
1	0.82	0.65	0.73	1352
accuracy			0.90	6515
macro avg	0.87	0.81	0.83	6515
weighted avg	0.89	0.90	0.89	6515

f1-score: 0.9376412961567445

Support Vector Machines

Confusion Matrix SVM Model



Classification Report SVM Model

	precision	recall	f1-score	support
0	0.89	0.97	0.93	5163
1	0.84	0.54	0.65	1352
accuracy			0.88	6515
macro avg	0.86	0.75	0.79	6515
weighted avg	0.88	0.88	0.87	6515

f1-score: 0.9288819014149634

Evaluasi Model - Confusion Matrix

Melihat Model terbaik pada dataset

		Predictions	
		0	1
Actuals	0	True Negative	False Positive
	1	False Negative	True Positive

1) True Negative : Model memprediksi data ada di kelas **Negatif** dan yang sebenarnya data memang ada di kelas **Negatif**.

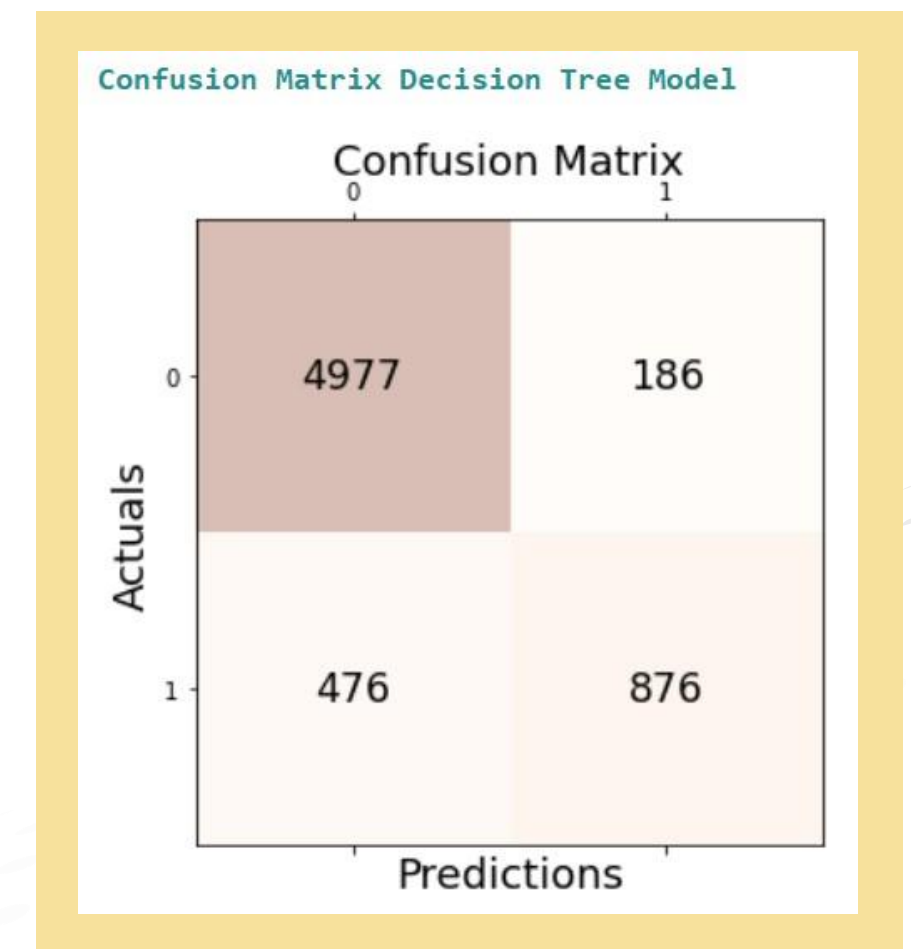
2) False Negative : Model memprediksi data ada di kelas **Negatif** dan yang sebenarnya data memang ada di kelas **Positif**.

3) True Positive : Model memprediksi data ada di kelas **Positif** dan yang sebenarnya data memang ada di kelas **Positif**.

4) False Positive : Model memprediksi data ada di kelas **Positif** dan yang sebenarnya data memang ada di kelas **Negatif**.

Evaluasi Model

Decision Tree



Evaluasi Model - Classification Report

Melihat **model terbaik** pada dataset

Evaluasi Model

Penjelasan

Decision Tree

```
Classification Report Decision Tree Model
      precision    recall  f1-score   support

     0       0.91      0.96      0.94       5163
     1       0.82      0.65      0.73       1352

 accuracy      0.90      0.90      0.90       6515
 macro avg     0.87      0.81      0.83       6515
weighted avg     0.89      0.90      0.89       6515

f1-score: 0.9376412961567445
```

Precision : Perbandingan antara True Positive dengan banyaknya data yang diprediksi positif

$$precision = \frac{TP}{TP + FP}$$

Recall : Perbandingan antara True Positive dengan banyaknya data yang sebenarnya positif

$$recall = \frac{TP}{TP + FN}$$

F1-score : Harmonic mean dari precision dan recall

$$\frac{1}{F1} = \frac{1}{2} \left(\frac{1}{precision} + \frac{1}{recall} \right)$$

Support : Jumlah data test yang digunakan

Model terbaik yang kita peroleh menggunakan

Decision Tree

parameter 'criterion': 'gini', 'max_depth': 8, 'max_features': 'log2', 'splitter': 'best'

Interpretasi dari hasil yang kami peroleh memiliki kesimpulan bahwa **untuk mengetahui debitur layak menerima kredit atau tidak** dengan memperhatikan

1 Jumlah Pendapatan Tahunan

2 Kelas Pinjaman

3 Suku Bunga

4 Persen Pendapatan

5 Kepemilikan Rumah Gadai/Sewa

6 Riwayat Gagal Bayar

Python Programming

Credit Risk Prediction Programming Algorithm

Person Income: 9600

Loan Grade: c

Loan Interest Rate: 12.87

Loan Percent Income: 0.57

Person Home Ownership: mortGage

CB Person Default on File: n

Kelayakan Penerima Nasabah Kredit:

Tidak Layak

Credit Risk Prediction Programming Algorithm

Person Income: 120000

Loan Grade: A

Loan Interest Rate: 7.49

Loan Percent Income: 0.15

Person Home Ownership: mortgage

CB Person Default on File: n

Kelayakan Penerima Nasabah Kredit:

Layak

Credit Risk Prediction Programming Algorithm

Person Income: 76000

Loan Grade: B

Loan Interest Rate: 10.99


Loan Percent Income: 0.46

Person Home Ownership: RENT

CB Person Default on File: N

Kelayakan Penerima Nasabah Kredit:

Tidak Layak



CREDITA APPLICATION

INCOME
9600
*in million per month

LOAN GRADE
C

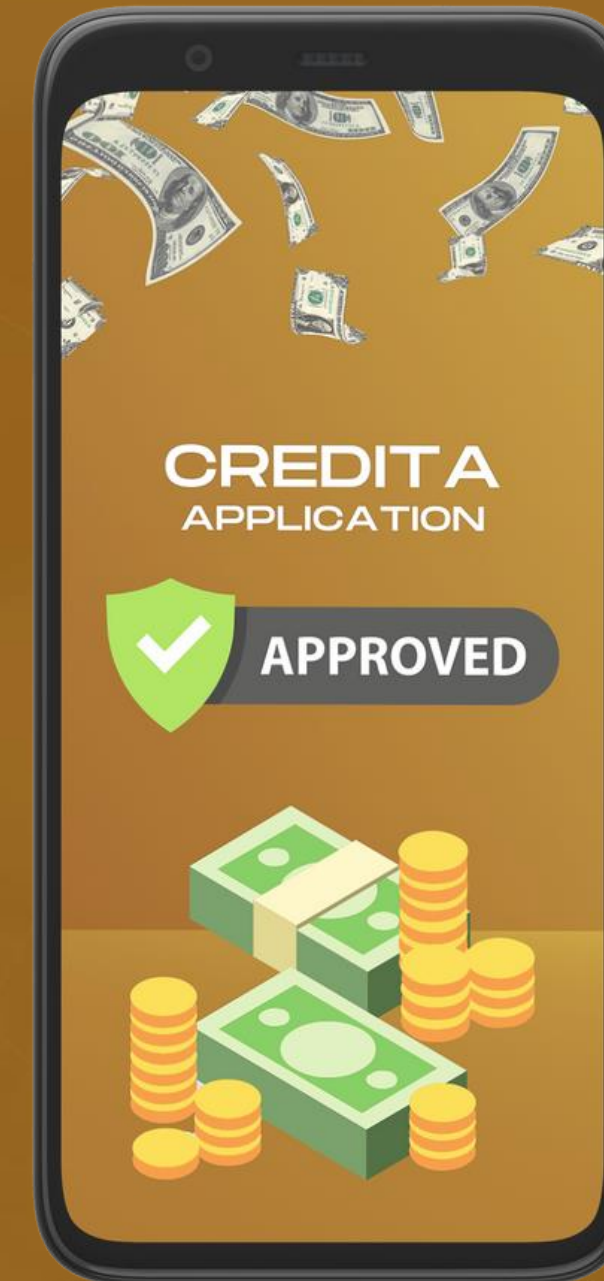
LOAN INTEREST RATE
12.87

LOAN PERCENT INCOME
0.57

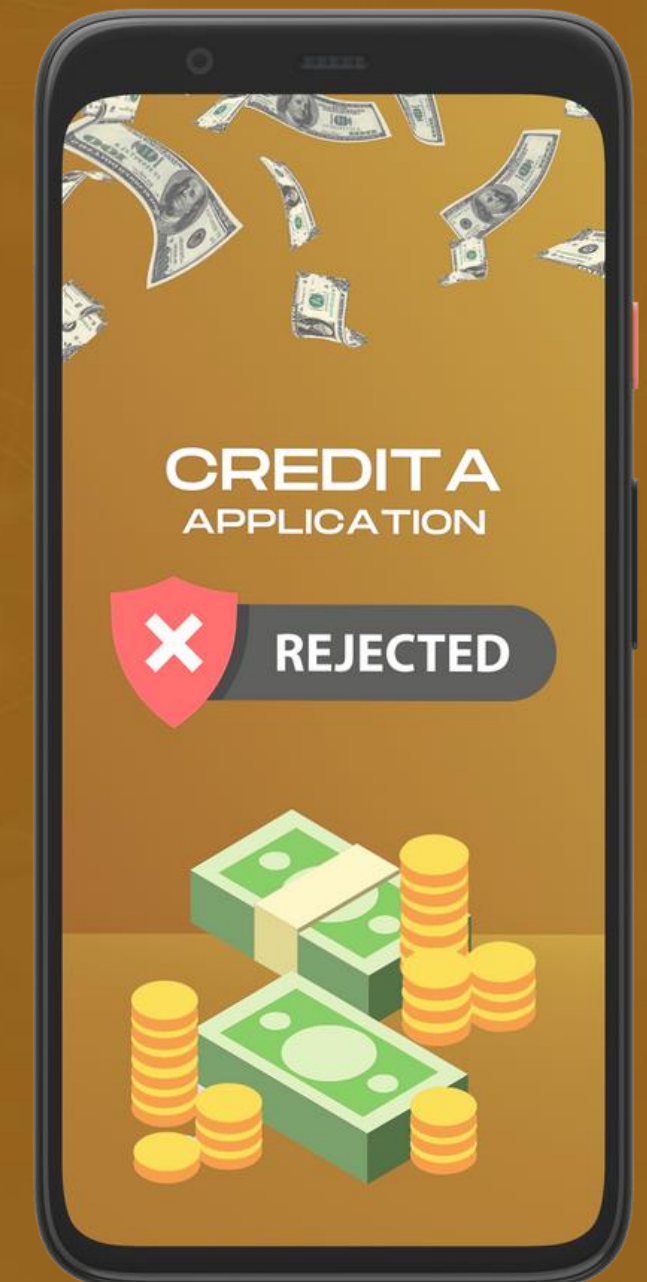
HOME OWNERSHIP
MORTGAGE

CB PERSON DEFAULT ON FILE
N

SUBMIT



Implementasi



Mata Kuliah Sains Data – **SCMA602017**

**TERIMA
KASIH**

Program **Pendeteksi Kelayakan Debitur**