Rakamin Academy X id/x partners

Credit Risk Prediction Model: Machine Learning Solutions for Financial Risk Assessment

ID/X Partners - Data Scientist

Presented by
Bagus Rahmadani

Jakarta, Indonesia

bagusrajin465@gmail.com

Bagus Rahmadani

Rakamin Academy

Bagus Rahmadani

Data Scientist & Data Analyst

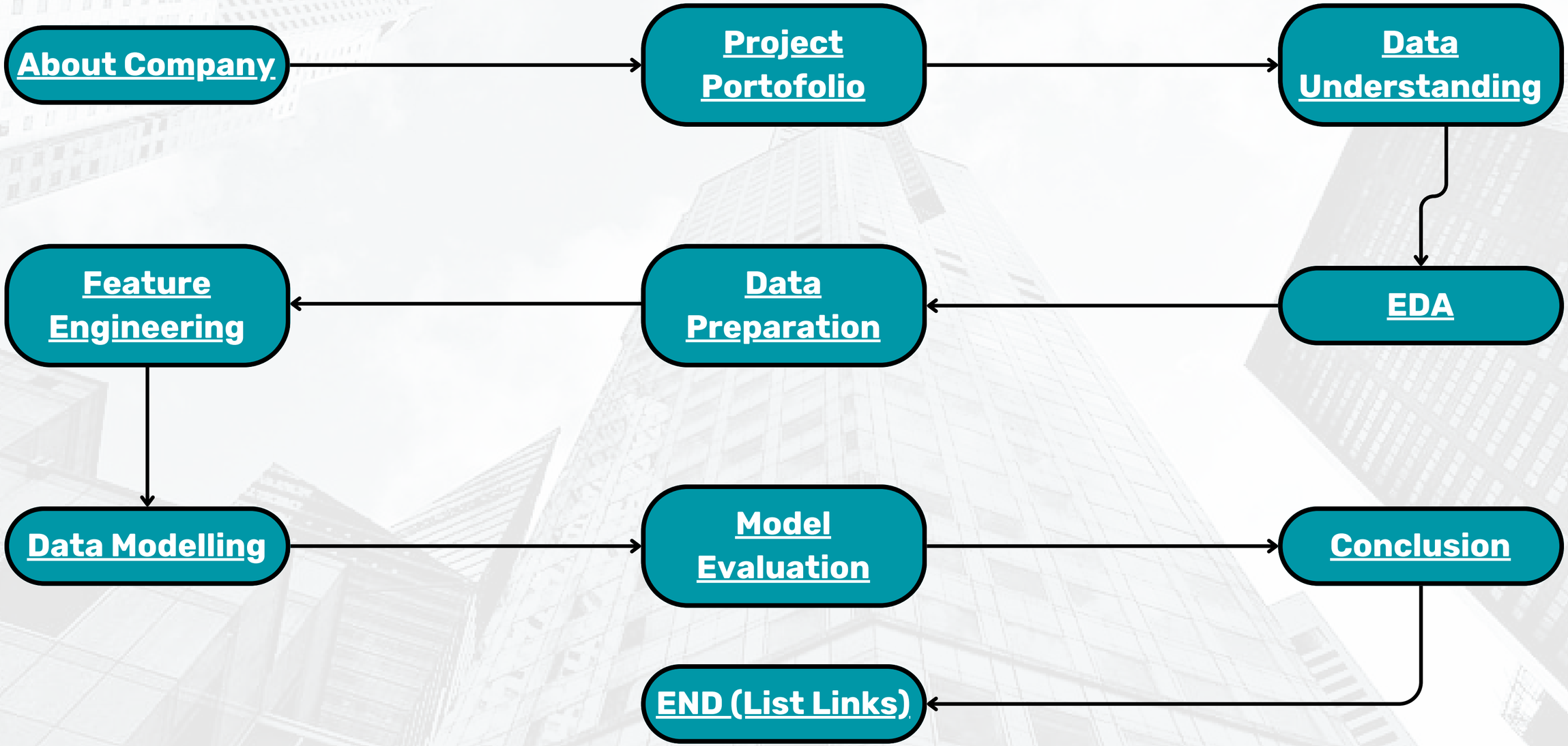
Saya adalah mahasiswa Diploma III Sistem Informasi di UPN "Veteran" Jakarta dengan ketertarikan besar pada Data Analytics dan Data Science. Berbekal pengalaman langsung dalam proses ETL, query SQL, serta visualisasi data menggunakan Looker Studio dan Power BI, saya terbiasa mengubah data yang kompleks menjadi insight yang mudah dipahami dan bermanfaat. Dengan sertifikasi IT Specialist – Data Analytics serta pelatihan lanjutan di bidang Python, Data Science, dan berbagai tools Data Science, saya berfokus untuk menghadirkan solusi berbasis data yang dapat mendukung pengambilan keputusan strategis di bidang bisnis, keuangan, dan pemasaran.

Courses and Certification

SERTIFICATE NAME	Institution	CERTIFICATE LINK	DURATION
DATA SCIENCE & DATA ANALYSIS	MySkill	Link	January - June 2025
MICROSOFT EXCEL BASIC, INTERMEDIATE AND ADVANCED LEVEL	MySkill	Link	July 2025
Data Science	ITBOX	Link	July - August 2025
Data Analyst Python Track	DQLab	Link	July 2025
Information of Technology Specialist - Data Analytics	Certiport	Link	December 2024
Certificate of Achievment - Kimia Farma Big Data Analytics Project Based Internship Program	Rakamin Academy	Link	Agustus - September 2025

Rakamin Academy

Agenda



About Company

PT IDX Consulting, yang beroperasi dengan nama **ID/X Partners**, merupakan perusahaan konsultan yang berdiri sejak **2002**. Selama lebih dari dua dekade, perusahaan ini telah memberikan layanan kepada berbagai klien korporat di kawasan **Asia** dan **Australia**, meliputi **sektor-sektor strategis seperti perbankan dan keuangan, industri telekomunikasi, sektor manufaktur, serta bisnis ritel**. Keahlian utama ID/X Partners terletak pada **penyediaan solusi konsultasi** yang berfokus pada **implementasi teknologi *data analytics and decisioning* (DAD)**, yang diintegrasikan dengan **strategi manajemen risiko dan pendekatan pemasaran yang holistik**. Melalui kombinasi layanan ini, perusahaan membantu klien dalam **meningkatkan efisiensi operasional, mengoptimalkan profitabilitas portofolio investasi, dan memperbaiki proses bisnis secara keseluruhan**. Dengan portofolio layanan konsultasi dan solusi teknologi yang lengkap dan terintegrasi, ID/X Partners memposisikan diri sebagai penyedia layanan terpadu yang dapat memenuhi kebutuhan bisnis klien secara menyeluruh.

Project Portfolio

Rakamin Academy

BACKGROUND

Latar belakang dari proyek ini berawal dari tantangan bisnis yang dihadapi oleh sebuah perusahaan multifinance bekerja sama dengan **ID/X Partners**, yang bertujuan untuk meningkatkan akurasi dalam menilai dan mengelola risiko kredit. Secara tradisional, proses persetujuan pinjaman sering mengalami kesalahan klasifikasi antara peminjam yang berpotensi membayar tepat waktu dan mereka yang memiliki risiko gagal bayar lebih tinggi. Oleh karena itu, diperlukan pengembangan model machine learning yang mampu memprediksi risiko kredit secara akurat, sehingga perusahaan dapat meminimalkan potensi kerugian sekaligus mengoptimalkan pengambilan keputusan bisnis dalam pengelolaan portofolio pinjamannya.

THE GOALS

Untuk mengembangkan **model machine learning** yang dapat memprediksi **risiko kredit secara akurat** bagi perusahaan multifinance bekerja sama dengan **ID/X Partners**, sehingga dapat mendukung pengambilan keputusan yang lebih baik dalam persetujuan pinjaman serta meminimalkan potensi kerugian finansial.

OBJECTIVES

1. Melakukan pemahaman dan eksplorasi data.
2. Menyiapkan data melalui pembersihan dan preprocessing.
3. Membangun serta membandingkan model machine learning (termasuk Logistic Regression).
4. Mengevaluasi performa model dengan metrik relevan.
5. Menyusun dokumentasi dan presentasi end-to-end solusi.

Project Portfolio

Rakamin Academy

Feature Descriptions of Loan Dataset: 1

	Feature	Description
0	id	A unique LC assigned ID for the loan listing
1	member_id	A unique LC assigned Id for the borrower member
2	loan_amnt	The listed amount of the loan applied for by t...
3	funded_amnt	The total amount committed to that loan at tha...
4	funded_amnt_inv	The total amount committed by investors for th...
5	term	The number of payments on the loan (36 or 60 m...
6	int_rate	Interest Rate on the loan
7	installment	The monthly payment owed by the borrower if th...
8	grade	LC assigned loan grade
9	sub_grade	LC assigned loan subgrade
10	emp_title	The job title supplied by the Borrower when ap...
11	emp_length	Employment length in years (0-10+ years)
12	home_ownership	The home ownership status provided by the borr...
13	annual_inc	The self-reported annual income provided by th...
14	verification_status	Indicates if income was verified by LC, not ve...
15	issue_d	The month which the loan was funded
16	loan_status	Current status of the loan
17	pymnt_plan	Indicates if a payment plan has been put in pl...
18	url	URL for the LC page with listing data
19	desc	Loan description provided by the borrower

Project Portfolio

Rakamin Academy

Feature Descriptions of Loan Dataset: 2

	Feature	Description
21	title	The loan title provided by the borrower
22	zip_code	The first 3 numbers of the zip code provided b...
23	addr_state	The state provided by the borrower in the loan...
24	dti	A ratio calculated using the borrower's total ...
25	delinq_2yrs	The number of 30+ days past-due incidences of ...
26	earliest_cr_line	The month the borrower's earliest reported cre...
27	inq_last_6mths	The number of inquiries in past 6 months (excl...
28	mths_since_last_delinq	The number of months since the borrower's last...
29	mths_since_last_record	The number of months since the last public record
30	open_acc	The number of open credit lines in the borrowe...
31	pub_rec	Number of derogatory public records
32	revol_bal	Total credit revolving balance
33	revol_util	Revolving line utilization rate, or the amount...
34	total_acc	The total number of credit lines currently in ...
35	initial_list_status	The initial listing status of the loan (Whole ...
36	out_prncp	Remaining outstanding principal for total amou...
37	out_prncp_inv	Remaining outstanding principal for portion of...
38	total_pymnt	Payments received to date for total amount funded
39	total_pymnt_inv	Payments received to date for portion of total...

Project Portfolio

Rakamin Academy

Feature Descriptions of Loan Dataset: 3

	Feature	Description
41	total_rec_int	Interest received to date
42	total_rec_late_fee	Late fees received to date
43	recoveries	post charge off gross recovery
44	collection_recovery_fee	post charge off collection fee
45	last_pymnt_d	Last month payment was received
46	last_pymnt_amnt	Last total payment amount received
47	next_pymnt_d	Next scheduled payment date
48	last_credit_pull_d	The most recent month LC pulled credit for thi...
49	collections_12_mths_ex_med	Number of collections in 12 months excluding m...
50	mths_since_last_major_derog	Months since most recent 90-day or worse rating
51	policy_code	publicly available policy_code=1 new products ...
52	application_type	Indicates whether the loan is an individual ap...
53	annual_inc_joint	The combined self-reported annual income provi...
54	dti_joint	A ratio calculated using the co-borrowers' com...
55	verification_status_joint	Indicates if the co-borrowers' income was veri...
56	acc_now_delinq	The number of accounts on which the borrower i...
57	tot_coll_amt	Total collection amounts ever owed
58	tot_cur_bal	Total current balance of all accounts
59	open_acc_6m	Number of open trades in last 6 months

Project Portfolio

Rakamin Academy

Feature Descriptions of Loan Dataset: 4

	Feature	Description
61	open_il_12m	Number of currently active installment trades ...
62	open_il_24m	Number of currently active installment trades ...
63	mths_since_rcnt_il	Months since most recent installment accounts ...
64	total_bal_il	Total current balance of all installment accounts
65	il_util	Ratio of total current balance to high credit/...
66	open_rv_12m	Number of revolving trades opened in past 12 m...
67	open_rv_24m	Number of revolving trades opened in past 24 m...
68	max_bal_bc	Maximum current balance owed on all revolving ...
69	all_util	Balance to credit limit on all trades
70	total_rev_hi_lim	Total revolving high credit/credit limit
71	inq_fi	Number of personal finance inquiries
72	total_cu_tl	Number of finance trades
73	inq_last_12m	Number of credit inquiries in past 12 months

Data Understanding

Rakamin Academy

```
Shape of the dataset:
(466285, 74)
The dataset has 466285 rows and 74 columns.

Info of the dataset:
<class 'pandas.core.frame.DataFrame'>
Index: 466285 entries, 0 to 466284
Data columns (total 74 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   id                          466285 non-null  int64
 1   member_id                   466285 non-null  int64
 2   loan_amnt                   466285 non-null  int64
 3   funded_amnt                 466285 non-null  int64
 4   funded_amnt_inv             466285 non-null  float64
 5   term                        466285 non-null  object
 6   int_rate                    466285 non-null  float64
 7   installment                 466285 non-null  float64
 8   grade                       466285 non-null  object
 9   sub_grade                   466285 non-null  object
 10  emp_title                   438697 non-null  object
 11  emp_length                  445277 non-null  object
 12  home_ownership              466285 non-null  object
 13  annual_inc                  466281 non-null  float64
 14  verification_status         466285 non-null  object
 15  issue_d                     466285 non-null  object
 16  loan_status                 466285 non-null  object
 17  pymnt_plan                  466285 non-null  object
 18  url                         466285 non-null  object
 19  desc                        125981 non-null  object
 20  purpose                     466285 non-null  object
 21  title                       466264 non-null  object
 22  zip_code                    466285 non-null  object
 23  addr_state                  466285 non-null  object
 24  dti                         466285 non-null  float64
 25  delinq_2yrs                 466256 non-null  float64
 26  earliest_cr_line            466256 non-null  object
 27  inq_last_6mths              466256 non-null  float64
 28  mths_since_last_delinq      215934 non-null  float64
 29  mths_since_last_record      62638 non-null   float64
 30  open_acc                    466256 non-null  float64
 31  pub_rec                     466256 non-null  float64
 32  revol_bal                   466285 non-null  int64
 33  revol_util                  465945 non-null  float64
 34  total_acc                   466256 non-null  float64
 35  initial_list_status         466285 non-null  object
 36  out_prncp                   466285 non-null  float64
 37  out_prncp_inv               466285 non-null  float64
 38  total_pymnt                 466285 non-null  float64
 39  total_pymnt_inv             466285 non-null  float64
 40  total_rec_prncp             466285 non-null  float64
 41  total_rec_int               466285 non-null  float64
 42  total_rec_late_fee          466285 non-null  float64
 43  recoveries                  466285 non-null  float64
 44  collection_recovery_fee     466285 non-null  float64
 45  last_pymnt_d                465909 non-null  object
 46  last_pymnt_amnt             466285 non-null  float64
 47  next_pymnt_d                239071 non-null  object
 48  last_credit_pull_d          466243 non-null  object
 49  collections_12_mths_ex_med  466140 non-null  float64
 50  mths_since_last_major_derog 98974 non-null   float64
 51  policy_code                 466285 non-null  int64
 52  application_type            466285 non-null  object
 53  annual_inc_joint            0 non-null       float64
 54  dti_joint                   0 non-null       float64
 55  verification_status_joint   0 non-null       float64
 56  acc_now_delinq              466256 non-null  float64
 57  tot_coll_amt                396009 non-null  float64
 58  tot_cur_bal                 396009 non-null  float64
 59  open_acc_6m                 0 non-null       float64
 60  open_il_6m                  0 non-null       float64
 61  open_il_12m                 0 non-null       float64
 62  open_il_24m                 0 non-null       float64
 63  mths_since_rcnt_il          0 non-null       float64
 64  total_bal_il                0 non-null       float64
 65  il_util                     0 non-null       float64
 66  open_rv_12m                 0 non-null       float64
 67  open_rv_24m                 0 non-null       float64
 68  max_bal_bc                  0 non-null       float64
 69  all_util                    0 non-null       float64
 70  total_rev_hi_lim            396009 non-null  float64
 71  inq_fi                      0 non-null       float64
 72  total_cu_tl                 0 non-null       float64
 73  inq_last_12m                0 non-null       float64
```

- Terdapat 466285 baris dan 74 kolom
- Terdapat 46 kolom tipe data numerik, dan 22 kolom kategorikal

Data Understanding

Rakamin Academy

	count	mean	std	min	25%	50%	75%	max
id	466285.0	1.307973e+07	1.089371e+07	54734.00	3.639987e+06	1.010790e+07	2.073121e+07	3.809811e+07
member_id	466285.0	1.459766e+07	1.168237e+07	70473.00	4.379705e+06	1.194108e+07	2.300154e+07	4.086083e+07
loan_amnt	466285.0	1.431728e+04	8.286509e+03	500.00	8.000000e+03	1.200000e+04	2.000000e+04	3.500000e+04
funded_amnt	466285.0	1.429180e+04	8.274371e+03	500.00	8.000000e+03	1.200000e+04	2.000000e+04	3.500000e+04
funded_amnt_inv	466285.0	1.422233e+04	8.297638e+03	0.00	8.000000e+03	1.200000e+04	1.995000e+04	3.500000e+04
int_rate	466285.0	1.382924e+01	4.357587e+00	5.42	1.099000e+01	1.366000e+01	1.649000e+01	2.606000e+01
installment	466285.0	4.320612e+02	2.434855e+02	15.67	2.566900e+02	3.798900e+02	5.665800e+02	1.409990e+03
annual_inc	466281.0	7.327738e+04	5.496357e+04	1896.00	4.500000e+04	6.300000e+04	8.896000e+04	7.500000e+06
dti	466285.0	1.721876e+01	7.851121e+00	0.00	1.136000e+01	1.687000e+01	2.278000e+01	3.999000e+01
delinq_2yrs	466256.0	2.846784e-01	7.973651e-01	0.00	0.000000e+00	0.000000e+00	0.000000e+00	2.900000e+01
inq_last_6mths	466256.0	8.047446e-01	1.091598e+00	0.00	0.000000e+00	0.000000e+00	1.000000e+00	3.300000e+01
mths_since_last_delinq	215934.0	3.410443e+01	2.177849e+01	0.00	1.600000e+01	3.100000e+01	4.900000e+01	1.880000e+02
mths_since_last_record	62638.0	7.430601e+01	3.035765e+01	0.00	5.300000e+01	7.600000e+01	1.020000e+02	1.290000e+02
open_acc	466256.0	1.118707e+01	4.987526e+00	0.00	8.000000e+00	1.000000e+01	1.400000e+01	8.400000e+01
pub_rec	466256.0	1.605642e-01	5.108626e-01	0.00	0.000000e+00	0.000000e+00	0.000000e+00	6.300000e+01
revol_bal	466285.0	1.623020e+04	2.067625e+04	0.00	6.413000e+03	1.176400e+04	2.033300e+04	2.568995e+06
revol_util	465945.0	5.617695e+01	2.373263e+01	0.00	3.920000e+01	5.760000e+01	7.470000e+01	8.923000e+02
total_acc	466256.0	2.506443e+01	1.160014e+01	1.00	1.700000e+01	2.300000e+01	3.200000e+01	1.560000e+02
out_prncp	466285.0	4.410062e+03	6.355079e+03	0.00	0.000000e+00	4.414700e+02	7.341650e+03	3.216038e+04
out_prncp_inv	466285.0	4.408452e+03	6.353198e+03	0.00	0.000000e+00	4.413800e+02	7.338390e+03	3.216038e+04
total_pymnt	466285.0	1.154069e+04	8.265627e+03	0.00	5.552125e+03	9.419251e+03	1.530816e+04	5.777758e+04
total_pymnt_inv	466285.0	1.146989e+04	8.254158e+03	0.00	5.499250e+03	9.355430e+03	1.523131e+04	5.777758e+04
total_rec_prncp	466285.0	8.866015e+03	7.031688e+03	0.00	3.708560e+03	6.817760e+03	1.200000e+04	3.500003e+04
total_rec_int	466285.0	2.588677e+03	2.483810e+03	0.00	9.572800e+02	1.818880e+03	3.304530e+03	2.420562e+04
total_rec_late_fee	466285.0	6.501292e-01	5.265730e+00	0.00	0.000000e+00	0.000000e+00	0.000000e+00	3.586800e+02
recoveries	466285.0	8.534421e+01	5.522161e+02	0.00	0.000000e+00	0.000000e+00	0.000000e+00	3.352027e+04
collection_recovery_fee	466285.0	8.961534e+00	8.549144e+01	0.00	0.000000e+00	0.000000e+00	0.000000e+00	7.002190e+03
last_pymnt_amnt	466285.0	3.123914e+03	5.554737e+03	0.00	3.126200e+02	5.459600e+02	3.187510e+03	3.623444e+04
collections_12_mths_ex_med	466140.0	9.085253e-03	1.086484e-01	0.00	0.000000e+00	0.000000e+00	0.000000e+00	2.000000e+01
mths_since_last_major_derog	98974.0	4.285255e+01	2.166259e+01	0.00	2.600000e+01	4.200000e+01	5.900000e+01	1.880000e+02
policy_code	466285.0	1.000000e+00	0.000000e+00	1.00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
annual_inc_joint	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
dti_joint	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
verification_status_joint	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
acc_now_delinq	466256.0	4.002093e-03	6.863680e-02	0.00	0.000000e+00	0.000000e+00	0.000000e+00	5.000000e+00
tot_coll_amt	396009.0	1.919135e+02	1.463021e+04	0.00	0.000000e+00	0.000000e+00	0.000000e+00	9.152545e+06
tot_cur_bal	396009.0	1.388017e+05	1.521147e+05	0.00	2.861800e+04	8.153900e+04	2.089530e+05	8.000078e+06
open_acc_6m	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
open_il_6m	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
open_il_12m	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
open_il_24m	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mths_since_rcnt_il	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
total_bal_il	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
il_util	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
open_rv_12m	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
open_rv_24m	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max_bal_bc	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
all_util	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
total_rev_hi_lim	396009.0	3.037909e+04	3.724713e+04	0.00	1.350000e+04	2.280000e+04	3.790000e+04	9.999999e+06
inq_fi	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
total_cu_tl	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
inq_last_12m	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Description of the dataset (include object):

	count	unique	top	freq
term	466285	2	36 months	337953
grade	466285	7	B	136929
sub_grade	466285	35	B3	31686
emp_title	438697	205475	Teacher	5399
emp_length	445277	11	10+ years	150049
home_ownership	466285	6	MORTGAGE	235875
verification_status	466285	3	Verified	168055
issue_d	466285	91	Oct-14	38782
loan_status	466285	9	Current	224226
pymnt_plan	466285	2	n	466276
url	466285	466285	https://www.lendingclub.com/browse/loanDetail....	1
desc	125981	124435		234
purpose	466285	14	debt_consolidation	274195
title	466264	63098	Debt consolidation	164075
zip_code	466285	888	945xx	5304
addr_state	466285	50	CA	71450
earliest_cr_line	466256	664	Oct-00	3674
initial_list_status	466285	2	f	303005
last_pymnt_d	465909	98	Jan-16	179620
next_pymnt_d	239071	100	Feb-16	208393
last_credit_pull_d	466243	103	Jan-16	327699
application_type	466285	1	INDIVIDUAL	466285

```
# Check Unique Values
print("\nUnique Values in each column of the dataset:")
df.nunique()
```

✓ 1.3s

```
Unique Values in each column of the dataset:

id                  466285
member_id           466285
loan_amnt             1352
funded_amnt           1354
funded_amnt_inv       9854
                     ...
all_util                 0
total_rev_hi_lim     14612
inq_fi                   0
total_cu_tl              0
inq_last_12m             0
Length: 74, dtype: int64
```

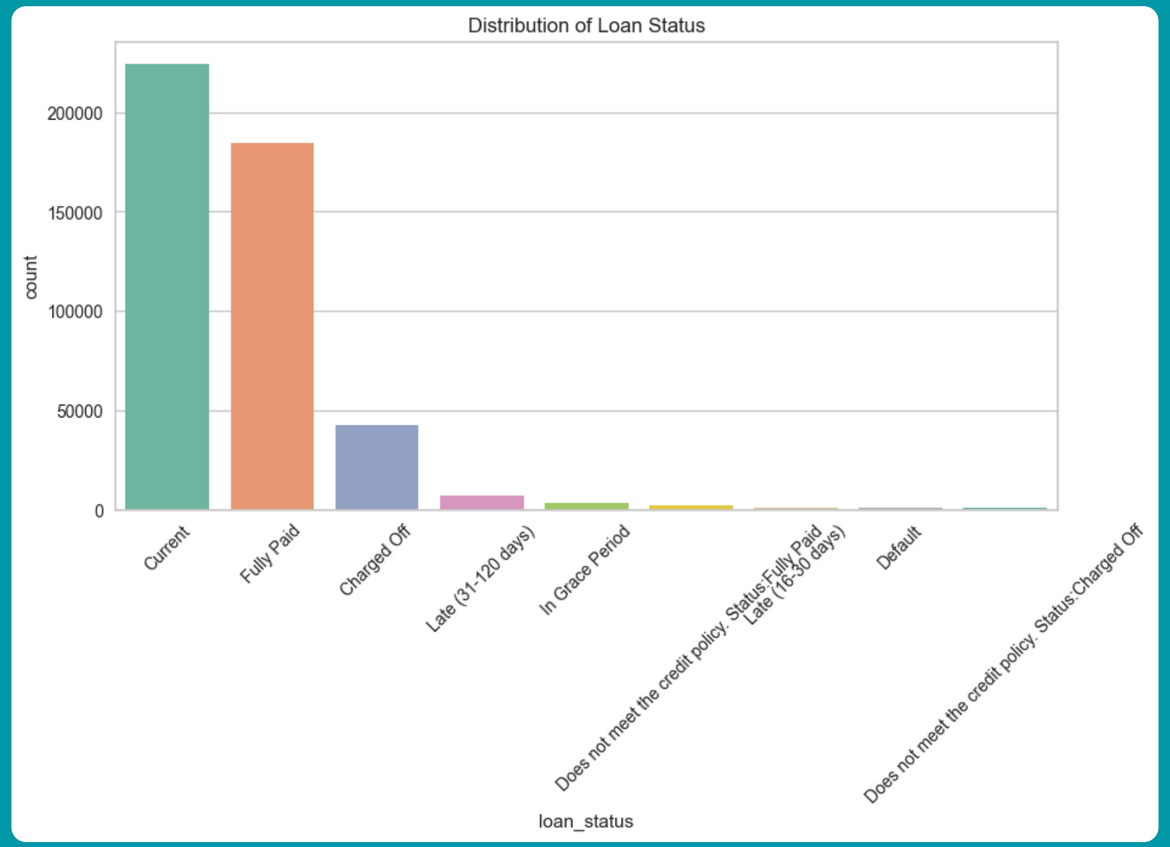
```
# Check Target Distribution
print("\nTarget Distribution in the dataset:")
if 'loan_status' in df.columns:
    print("\nTarget distribution (loan_status):")
    print(df['loan_status'].value_counts(normalize=True))
```

✓ 0.1s

```
Target Distribution in the dataset:

Target distribution (loan_status):
loan_status
Current                                            0.480878
Fully Paid                                         0.396193
Charged Off                                        0.091092
Late (31-120 days)                                 0.014798
In Grace Period                                    0.006747
Does not meet the credit policy. Status:Fully Paid 0.004263
Late (16-30 days)                                  0.002612
Default                                            0.001784
Does not meet the credit policy. Status:Charged Off 0.001632
Name: proportion, dtype: float64
```

Exploratory Data Analysis

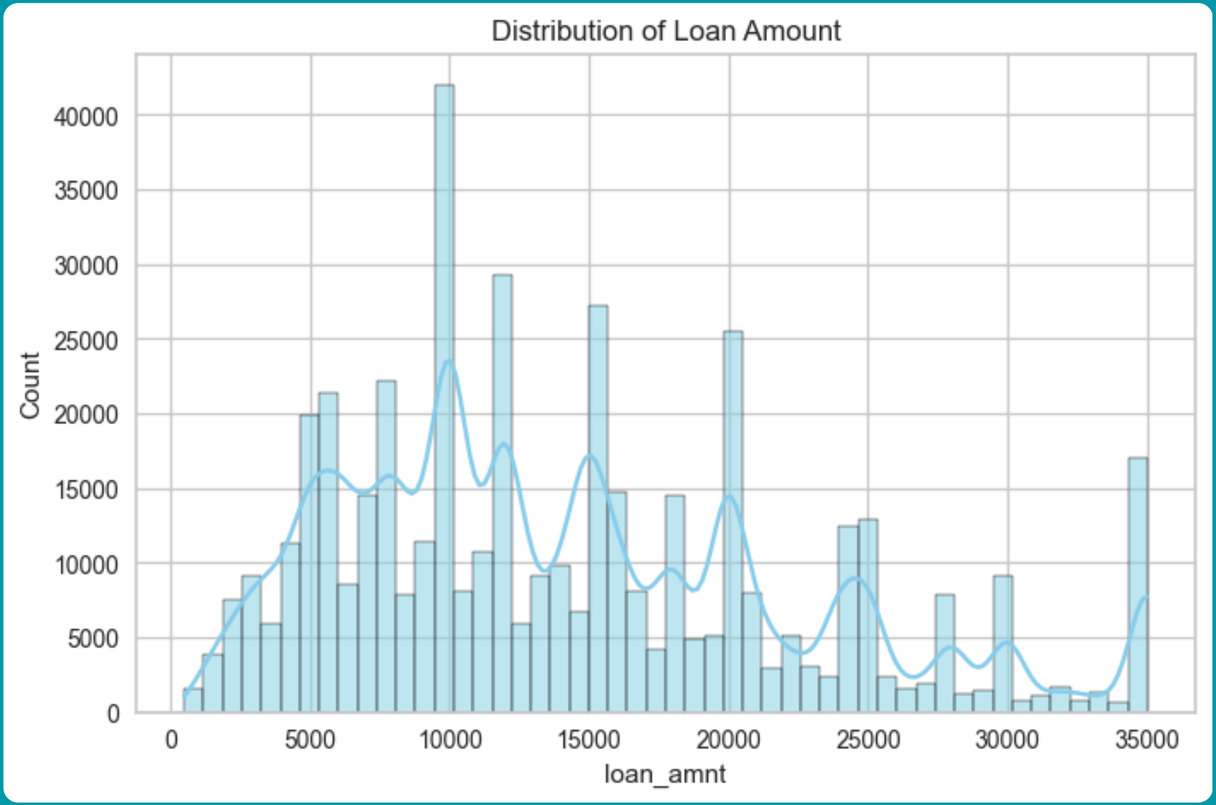
UNIVARIAT ANALYSIS: DISTRIBUTION OF LOAN STATUS



Visualisasi menunjukkan **distribusi status pinjaman yang sangat tidak seimbang**, dengan mayoritas loan berstatus "Current" dan "Fully Paid", diikuti "Charged Off" dalam jumlah yang lebih kecil, serta kategori langka seperti "Default" dan "Late". Ketidakseimbangan ini mengindikasikan perlunya teknik balancing seperti SMOTE atau class weight adjustment untuk memastikan model dapat memprediksi semua kategori dengan baik.

Exploratory Data Analysis

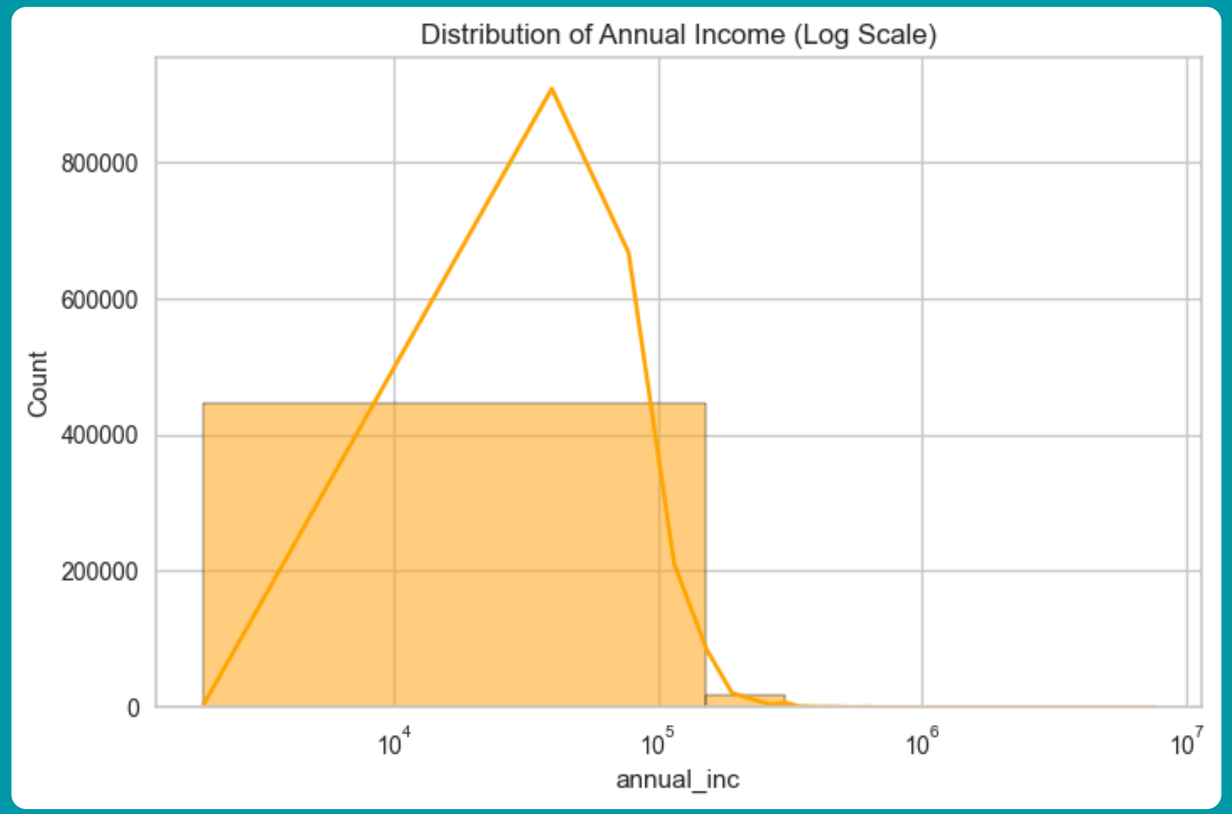
UNIVARIAT ANALYSIS: DISTRIBUTION OF LOAN AMOUNT



Histogram menampilkan **distribusi jumlah pinjaman yang right-skewed dengan mayoritas peminjam mengajukan pinjaman $5,000-$20,000** dan puncak di sekitar $10,000-$15,000. Distribusi ini wajar karena sebagian besar konsumen mengajukan pinjaman untuk kebutuhan menengah, sementara pinjaman besar untuk investasi atau bisnis relatif jarang, dan informasi ini penting untuk strategi segmentasi risiko.

Exploratory Data Analysis

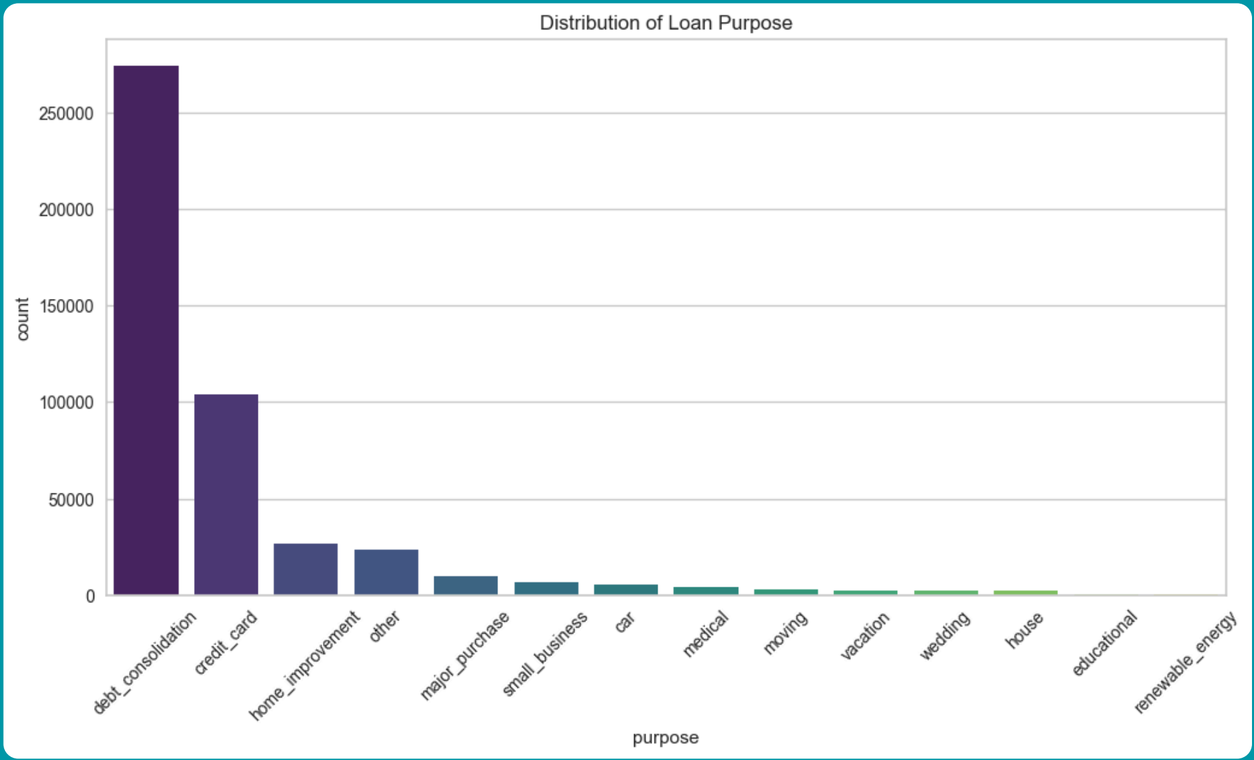
UNIVARIAT ANALYSIS: DISTRIBUTION OF ANNUAL INCOME



Visualisasi pendapatan tahunan dengan skala logaritmik menunjukkan distribusi yang lebih normal setelah transformasi, dengan mayoritas peminjam berpendapatan $40,000-$100,000 per tahun. Transformasi log scale penting untuk mengurangi dampak outlier pendapatan ekstrem dan membuat data lebih sesuai dengan asumsi algoritma machine learning.

Exploratory Data Analysis

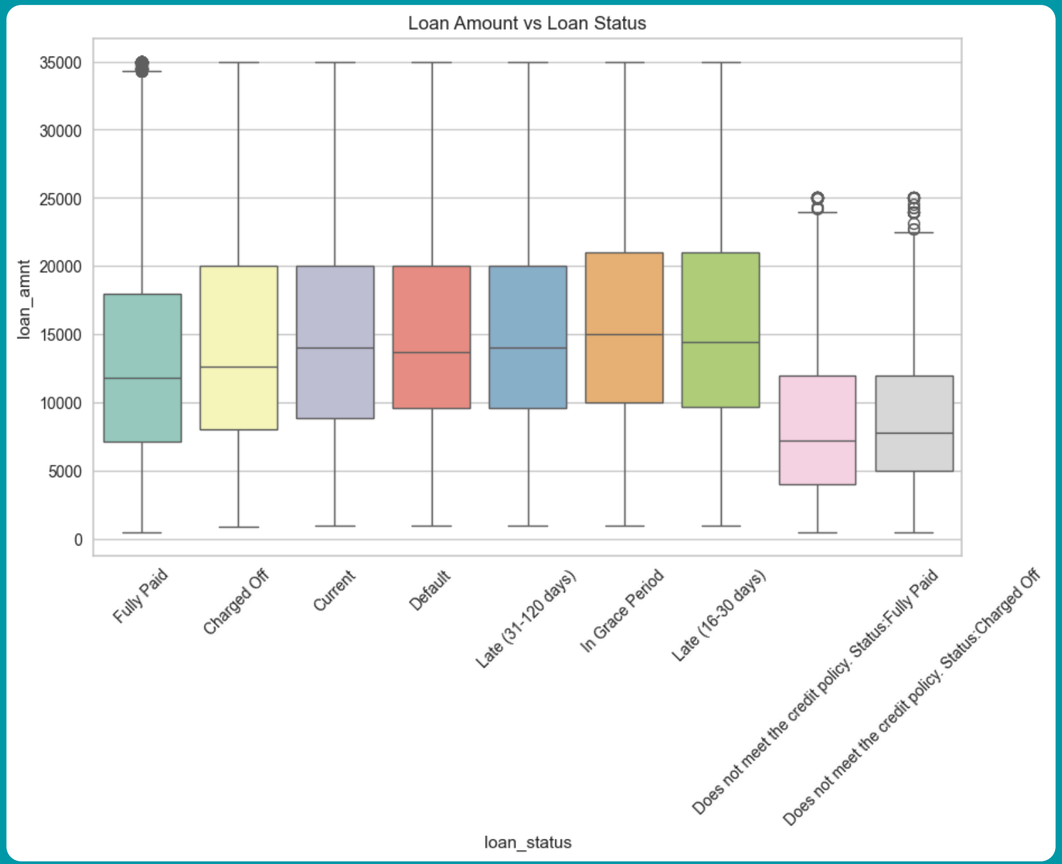
UNIVARIAT ANALYSIS: DISTRIBUTION OF LOAN PURPOSE



Grafik menunjukkan **dominasi "debt_consolidation" sebagai tujuan pinjaman paling populer**, diikuti "credit_card", sementara kategori seperti "renewable_energy", "educational", dan "wedding" memiliki sampel sangat sedikit. Distribusi tidak merata ini memerlukan penggabungan kategori langka ke dalam "other" untuk menghindari overfitting dan meningkatkan kemampuan generalisasi model.

Exploratory Data Analysis

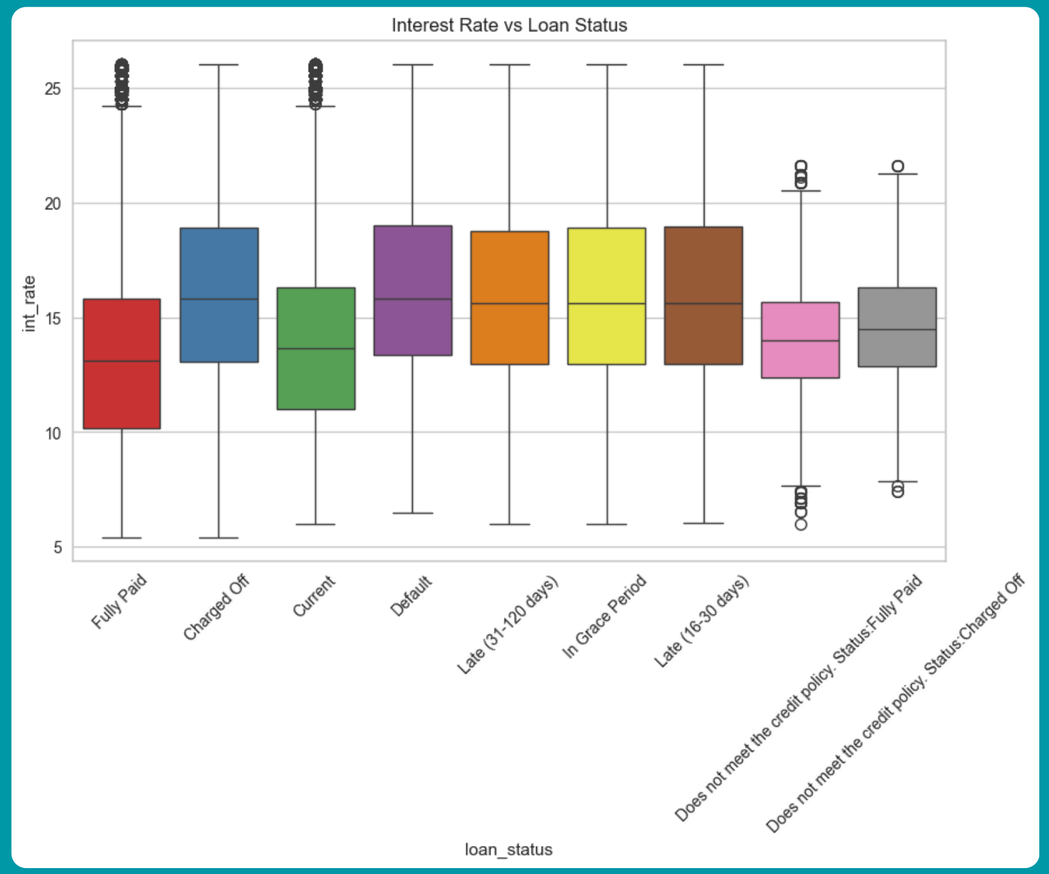
BIVARIAT ANALYSIS: LOAN AMOUNT VS LOAN STATUS



Visualisasi boxplot menunjukkan bahwa **distribusi jumlah pinjaman relatif konsisten** di antara berbagai status pinjaman, dengan sebagian besar kategori memiliki median yang serupa sekitar $10,000-$15,000. Pinjaman dengan status "Fully Paid" dan "Current" menunjukkan rentang yang luas dengan beberapa outlier di nilai tinggi, sementara pinjaman bermasalah seperti "Late" atau "Charged Off" cenderung terkonsentrasi di rentang menengah tanpa banyak outlier ekstrem, mengindikasikan bahwa jumlah pinjaman bukan prediktor utama untuk status pinjaman.

Exploratory Data Analysis

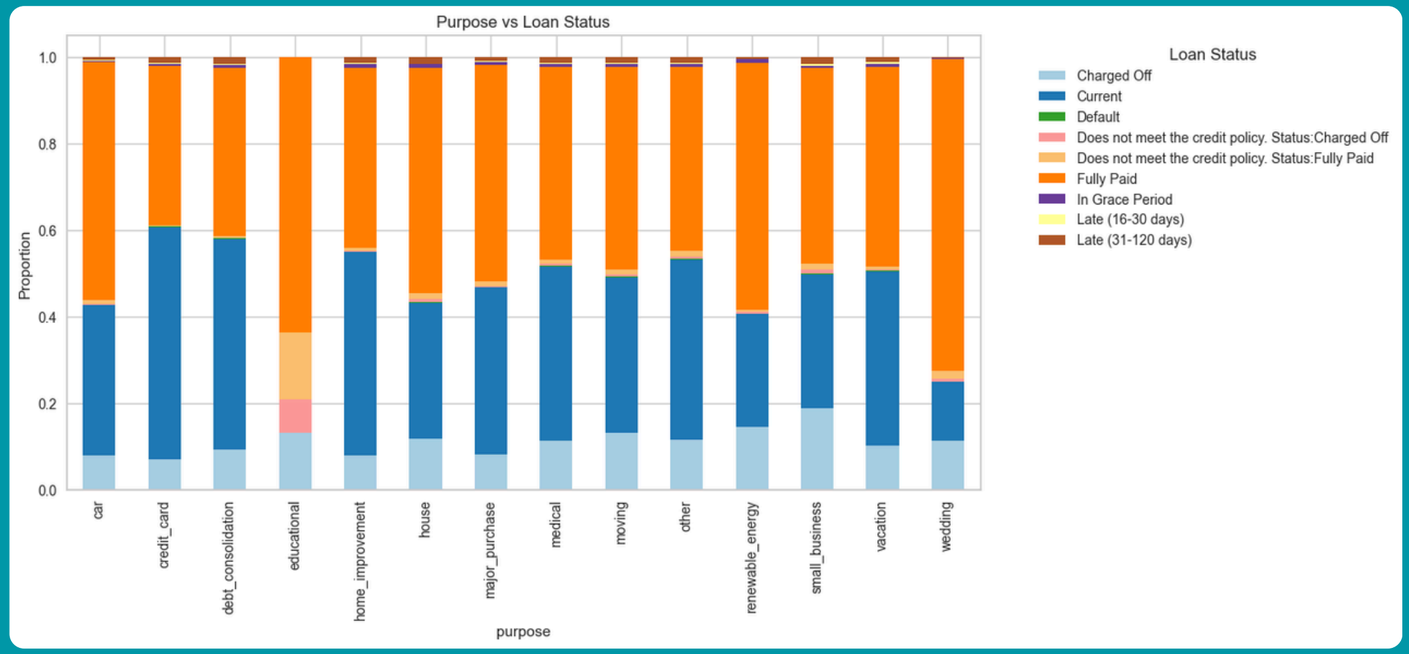
BIVARIAT ANALYSIS: INTEREST RATE VS LOAN STATUS



Analisis boxplot suku bunga mengungkapkan pola yang jelas dimana pinjaman bermasalah memiliki suku bunga yang secara signifikan lebih tinggi dibandingkan pinjaman yang lancar. Pinjaman "Charged Off", "Default", dan "Late" menunjukkan median suku bunga di atas 15%, sementara pinjaman "Fully Paid" memiliki median sekitar 13%, mengkonfirmasi bahwa lenders mengenakan risk-based pricing dengan suku bunga tinggi untuk borrower berisiko tinggi, namun ironisnya borrower dengan suku bunga tinggi justru memiliki probabilitas gagal bayar yang lebih besar.

Exploratory Data Analysis

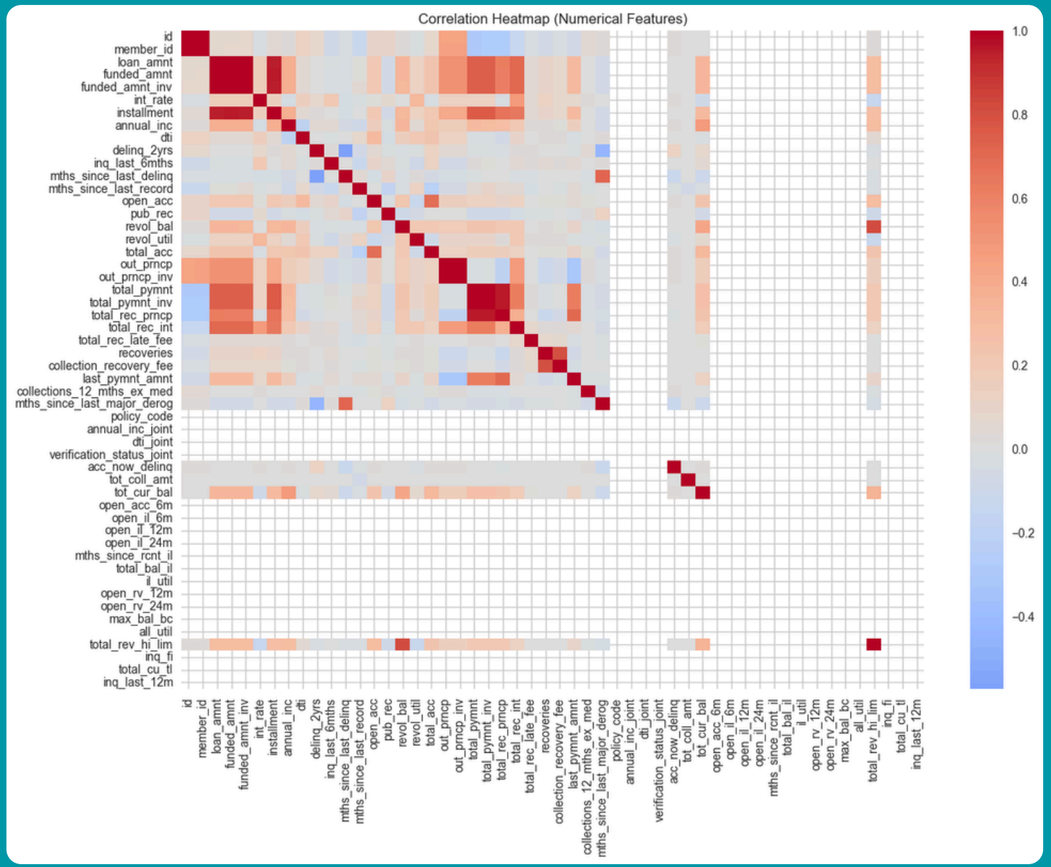
BIVARIAT ANALYSIS: PURPOSE VS LOAN STATUS



Grafik stacked bar chart menampilkan proporsi status pinjaman untuk setiap tujuan pinjaman, mengungkapkan bahwa mayoritas kategori tujuan didominasi oleh status "Fully Paid" dan "Current". Namun, beberapa kategori seperti "small business", "renewable energy", dan "educational" menunjukkan proporsi "Charged Off" atau "Default" yang lebih tinggi dibandingkan kategori yang lebih aman seperti "debt consolidation" dan "credit card", menunjukkan bahwa tujuan pinjaman dapat menjadi indikator risiko yang berguna untuk model prediksi kredit.

Exploratory Data Analysis

CORRELATION ANALYSIS: HEATMAP (NUMERICAL FEATURES)



Heatmap korelasi menunjukkan pola korelasi antar fitur numerik dalam dataset, dengan beberapa fitur menunjukkan multikolinearitas kuat (blok merah gelap) seperti loan_amnt yang hampir sempurna berkorelasi dengan funded_amnt dan funded_amnt_inv (>0.99), serta total_pymnt yang tinggi berkorelasi dengan variabel pembayaran lainnya. Sebagian besar variabel lain menunjukkan korelasi lemah satu sama lain, mengindikasikan bahwa banyak fitur bersifat independen dan dapat memberikan informasi unik untuk model prediksi kredit.

Exploratory Data Analysis

Rakamin Academy

CORRELATION ANALYSIS: TOP CORRELATION WITH LOAN AMOUNT

```
Top correlations with Loan Amount:
loan_amnt             1.000000
funded_amnt           0.998548
funded_amnt_inv       0.994347
installment           0.949666
total_pymnt           0.743841
total_pymnt_inv       0.743580
total_rec_int         0.715876
total_rec_prncp       0.612777
out_prncp             0.518700
out_prncp_inv         0.518660
Name: loan_amnt, dtype: float64
```

Analisis korelasi tertinggi dengan loan_amnt mengungkapkan redundansi yang signifikan di antara beberapa variabel, dimana funded_amnt (0.9985) dan funded_amnt_inv (0.9943) hampir identik dengan loan_amnt, installment (0.9497) juga sangat tinggi berkorelasi, dan variabel terkait pembayaran seperti total_pymnt (0.7438) serta total_rec_int (0.7159) menunjukkan korelasi kuat. Temuan ini menunjukkan adanya redundansi di antara beberapa variabel yang memerlukan feature selection untuk menghilangkan multikolinearitas dan meningkatkan efisiensi model machine learning.

Data Preparation

Handling Irrelevant Features

1. Handling Irrelevant Features

```python
# Remove irrelevant features
df.drop(columns = ['id', 'member_id', 'policy_code', 'annual_inc_joint', 'dti_joint', 'verification_status_joint',
                   'open_acc_6m', 'open_il_6m', 'open_il_12m', 'open_il_24m', 'mths_since_rcnt_il', 'total_bal_il', 'il_util',
                   'open_rv_12m', 'open_rv_24m', 'max_bal_bc', 'all_util', 'inq_fi', 'total_cu_tl', 'inq_last_12m', 'emp_title', 'url', 'desc',
                   'title', 'zip_code', 'application_type'], inplace = True)
```

✓ 0.2s Python

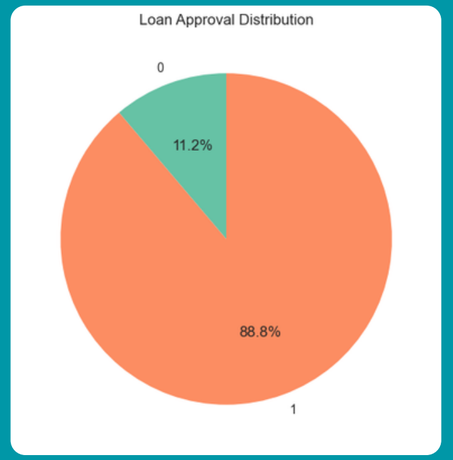
Creating a New Target Feature

2. Creating a New Target Feature

```python
# Create a target variable
df['loan_approved'] = df['loan_status'].isin(['Fully Paid', 'Current', 'In Grace Period',
                                              'Does not meet the credit policy. Status:Fully Paid']).astype(int)
df.drop(columns=['loan_status'], inplace=True)
df['loan_approved'].value_counts()
```

✓ 0.2s

```
loan_approved
1    414099
0     52186
Name: count, dtype: int64
```



- Menghapus fitur yang **tidak relevan**;
- Membuat variabel target **loan_approved**;
- Membuat visualisasi dalam **pie chart** yang memperlihatkan distribusi kelas tidak seimbang (**88.8% disetujui vs 11.2% tidak disetujui**).

Data Preparation

Rakamin Academy

Checking Missing Values

```
Missing Values in each column:
                           Missing Values  Percentage
mths_since_last_record             403647   86.566585
mths_since_last_major_derog        367311   78.773926
mths_since_last_delinq             250351   53.690554
next_pymnt_d                       227214   48.728567
total_rev_hi_lim                    70276   15.071469
tot_coll_amt                        70276   15.071469
tot_cur_bal                         70276   15.071469
emp_length                          21008    4.505399
last_pymnt_d                          376    0.080637
revol_util                            340    0.072917
collections_12_mths_ex_med            145    0.031097
last_credit_pull_d                     42    0.009007
delinq_2yrs                            29    0.006219
pub_rec                                29    0.006219
inq_last_6mths                         29    0.006219
open_acc                               29    0.006219
acc_now_delinq                         29    0.006219
total_acc                              29    0.006219
earliest_cr_line                       29    0.006219
annual_inc                              4    0.000858
```

Handling Missing Values: Creating Missing Values Imputer

3.2 Creating Missing Values Imputer

```
imputer_num_mean = SimpleImputer(strategy='mean')
imputer_num_median = SimpleImputer(strategy='median')
imputer_cat = SimpleImputer(strategy='most_frequent')
```

✓ 0.0s

Applying to Missing Values

3.3 Applying Missing Values Imputer

```
# Remove the columns
df.drop(columns = ['mths_since_last_delinq', 'mths_since_last_record', 'next_pymnt_d',
                   'mths_since_last_major_derog', 'tot_coll_amt', 'tot_cur_bal', 'total_rev_hi_lim'], inplace = True)
```

✓ 0.1s

Generate + Code + Markdown

```
# Imputation for categorical columns with most frequent
df[['earliest_cr_line',
    'last_pymnt_d',
    'last_credit_pull_d']] = imputer_cat.fit_transform(df[['earliest_cr_line', 'last_pymnt_d', 'last_credit_pull_d']])
# Specific imputation for 'emp_length' with '< 1 year'
df['emp_length'].fillna('< 1 year', inplace = True)
# Imputation for numerical columns with median
df[['annual_inc',
    'delinq_2yrs',
    'inq_last_6mths',
    'open_acc', 'pub_rec',
    'total_acc',
    'collections_12_mths_ex_med',
    'acc_now_delinq']] = imputer_num_median.fit_transform(df[['annual_inc',
                                                              'delinq_2yrs',
                                                              'inq_last_6mths',
                                                              'open_acc',
                                                              'pub_rec',
                                                              'total_acc',
                                                              'collections_12_mths_ex_med',
                                                              'acc_now_delinq']])
df[['revol_util']] = imputer_num_mean.fit_transform(df[['revol_util']])
```

- Membuang kolom yang terlalu banyak kosong;
- Mengimputasi data menggunakan strategi median, mean, atau modus sesuai tipe datanya agar dataset tetap utuh untuk analisis.

Checking Again

```
Missing Values in each column:
Empty DataFrame
Columns: [Missing Values, Percentage]
Index: []
```

Checking Duplicate Values

```
Number of duplicated rows in the dataset:
0
```

Feature Engineering

Date Features Engineering

4.1 Date Features Engineering

```python
# Convert date columns to datetime first
date_columns = ['issue_d', 'earliest_cr_line', 'last_pymnt_d', 'last_credit_pull_d']
for col in date_columns:
    if col in df.columns:
        df[col] = pd.to_datetime(df[col], format='%b-%y', errors='coerce')

print("Date columns converted to datetime:")
for col in date_columns:
    if col in df.columns:
        print(f"{col}: {df[col].dtype}")
        print(f"Sample values: {df[col].dropna().head(3).tolist()}")
        print("---")

# 1. Credit history length (years from earliest credit line to issue date)
df['credit_history_length'] = ((df['issue_d'] - df['earliest_cr_line']).dt.days / 365.25).round(2)

# 2. Issue year and month
df['issue_year'] = df['issue_d'].dt.year
df['issue_month'] = df['issue_d'].dt.month

# 3. Days since last payment (from last payment to last credit pull)
df['days_since_last_payment'] = (df['last_credit_pull_d'] - df['last_pymnt_d']).dt.days

# 4. Season of issue (for seasonal effects)
def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Fall'

df['issue_season'] = df['issue_month'].apply(get_season)

print("Date features created:")
print(f"- credit_history_length: {df['credit_history_length'].describe()}")
print(f"- days_since_last_payment: {df['days_since_last_payment'].describe()}")
print(f"- issue_season distribution: {df['issue_season'].value_counts()}")

# Fill infinite values and negatives
df['credit_history_length'] = df['credit_history_length'].replace([np.inf, -np.inf], np.nan)
df['days_since_last_payment'] = df['days_since_last_payment'].replace([np.inf, -np.inf], np.nan)

# Fill with median
df['credit_history_length'].fillna(df['credit_history_length'].median(), inplace=True)
df['days_since_last_payment'].fillna(df['days_since_last_payment'].median(), inplace=True)
```

✓ 0.5s

```
Date features created:
- credit_history_length: count    466285.000000
mean         15.685376
std           7.885850
min         -60.000000
25%          11.000000
50%          14.500000
75%          19.500000
max          45.910000
Name: credit_history_length, dtype: float64
- days_since_last_payment: count    466285.00000
mean        129.30076
std         276.41680
min       -2405.00000
25%           0.00000
50%           0.00000
75%         151.00000
max        2922.00000
Name: days_since_last_payment, dtype: float64
- issue_season distribution: issue_season
Fall      147444
Summer    125562
Spring    101505
Winter     91774
Name: count, dtype: int64
Date features created:
```

(Output 1)

(Output 2)


```
Date features created:
- credit_history_length: count    466285.000000
mean         15.685376
std           7.885850
min         -60.000000
25%          11.000000
50%          14.500000
75%          19.500000
max          45.910000
Name: credit_history_length, dtype: float64
- days_since_last_payment: count    466285.00000
mean        129.30076
std         276.41680
min       -2405.00000
25%           0.00000
50%           0.00000
75%         151.00000
max        2922.00000
Name: days_since_last_payment, dtype: float64
- issue_season distribution: issue_season
Fall      147444
Summer    125562
Spring    101505
Winter     91774
Name: count, dtype: int64
```


- Melakukan konversi 4 kolom tanggal (**issue_d, earliest_cr_line, last_pymnt_d, last_credit_pull_d**) dari format string 'MMM-YY' ke datetime;
- **Membuat 6 fitur baru:** credit_history_length (durasi riwayat kredit), issue_year/month (waktu penerbitan), days_since_last_payment (hari sejak pembayaran terakhir), dan issue_season (musim penerbitan);
- **Menangani nilai infinite dan missing dengan imputasi** median untuk memastikan data siap modeling.

Feature Engineering

Numerical Features Engineering

4.2 Numerical Features Engineering

```
# 1. Loan to income ratio
df['loan_to_income_ratio'] = df['loan_amnt'] / df['annual_inc']
df['loan_to_income_ratio'] = df['loan_to_income_ratio'].replace([np.inf, -np.inf], np.nan)
df['loan_to_income_ratio'].fillna(df['loan_to_income_ratio'].median(), inplace=True)

# 2. Installment to income ratio
df['installment_to_income_ratio'] = (df['installment'] * 12) / df['annual_inc']
df['installment_to_income_ratio'] = df['installment_to_income_ratio'].replace([np.inf, -np.inf], np.nan)
df['installment_to_income_ratio'].fillna(df['installment_to_income_ratio'].median(), inplace=True)

# 3. Credit utilization ratio
df['credit_util_ratio'] = df['revol_bal'] / (df['revol_bal'] + df['loan_amnt'])
df['credit_util_ratio'] = df['credit_util_ratio'].replace([np.inf, -np.inf], np.nan)
df['credit_util_ratio'].fillna(0, inplace=True)

# 4. Total debt service ratio (DTI + installment ratio)
df['total_debt_service_ratio'] = df['dti'] + df['installment_to_income_ratio'] * 100

# 5. Income per inquiry (risk indicator)
df['income_per_inquiry'] = df['annual_inc'] / (df['inq_last_6mths'] + 1)  # +1 to avoid division by zero

# 6. Account diversity ratio
df['account_diversity'] = df['open_acc'] / (df['total_acc'] + 1)

# 7. Risk score based on delinquencies and public records
df['risk_score'] = (df['delinq_2yrs'] * 2) + (df['pub_rec'] * 3) + (df['acc_now_delinq'] * 4)

print("Numerical features created:")
print(f"- loan_to_income_ratio: min={df['loan_to_income_ratio'].min():.4f}, max={df['loan_to_income_ratio'].max():.4f}")
print(f"- installment_to_income_ratio: min={df['installment_to_income_ratio'].min():.4f}, max={df['installment_to_income_ratio'].max():.4f}")
print(f"- total_debt_service_ratio: min={df['total_debt_service_ratio'].min():.2f}, max={df['total_debt_service_ratio'].max():.2f}")
print(f"- risk_score: {df['risk_score'].value_counts().head()}")
```

(Output)

```
Numerical features created:
- loan_to_income_ratio: min=0.0008, max=1.3375
- installment_to_income_ratio: min=0.0003, max=0.5417
- total_debt_service_ratio: min=0.04, max=69.17
- risk_score: risk_score
0.0    331024
2.0     48890
3.0     45149
4.0     14275
6.0     10272
Name: count, dtype: int64
```

- **Meningkatkan prediksi risiko kredit:** loan_to_income_ratio (beban pinjaman terhadap pendapatan), installment_to_income_ratio (rasio cicilan tahunan), credit_util_ratio (utilisasi kredit), total_debt_service_ratio (total beban utang), income_per_inquiry (indikator risiko pencarian kredit), account_diversity (keragaman akun kredit), dan risk_score (skor risiko komposit berdasarkan tunggakan dan catatan publik)

Feature Engineering

Categorical Features Engineering

4.3 Categorical Features Engineering

```python
# 1. Employment length binning (convert to numerical)
def emp_length_to_numeric(emp_length):
    if emp_length == '< 1 year':
        return 0
    elif emp_length == '1 year':
        return 1
    elif emp_length == '2 years':
        return 2
    elif emp_length == '3 years':
        return 3
    elif emp_length == '4 years':
        return 4
    elif emp_length == '5 years':
        return 5
    elif emp_length == '6 years':
        return 6
    elif emp_length == '7 years':
        return 7
    elif emp_length == '8 years':
        return 8
    elif emp_length == '9 years':
        return 9
    elif emp_length == '10+ years':
        return 10
    else:
        return 0

df['emp_length_numeric'] = df['emp_length'].apply(emp_length_to_numeric)

# 2. Grade to numeric (A=1, B=2, C=3, etc.)
grade_mapping = {'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G': 7}
df['grade_numeric'] = df['grade'].map(grade_mapping)

# 3. Term to numeric (remove 'months' and convert to int)
df['term_numeric'] = df['term'].str.replace(' months', '').astype(int)
```

```python
# 4. Home ownership grouping (reduce categories)
def home_ownership_group(home_ownership):
    if home_ownership in ['MORTGAGE']:
        return 'MORTGAGE'
    elif home_ownership in ['RENT']:
        return 'RENT'
    elif home_ownership in ['OWN']:
        return 'OWN'
    else:
        return 'OTHER'

df['home_ownership_grouped'] = df['home_ownership'].apply(home_ownership_group)

# 5. Purpose grouping (group rare categories)
purpose_counts = df['purpose'].value_counts()
rare_purposes = purpose_counts[purpose_counts < 1000].index.tolist()

def group_purpose(purpose):
    if purpose in rare_purposes:
        return 'other'
    else:
        return purpose

df['purpose_grouped'] = df['purpose'].apply(group_purpose)

# 6. Verification status simplification
def verification_simple(status):
    if status in ['Verified', 'Source Verified']:
        return 'Verified'
    else:
        return 'Not Verified'

df['verification_simple'] = df['verification_status'].apply(verification_simple)

print("Categorical features processed:")
print(f"- emp_length_numeric: {df['emp_length_numeric'].describe()}")
print(f"- grade_numeric: {df['grade_numeric'].value_counts().sort_index()}")
print(f"- term_numeric: {df['term_numeric'].value_counts()}")
print(f"- home_ownership_grouped: {df['home_ownership_grouped'].value_counts()}")
print(f"- purpose_grouped: {df['purpose_grouped'].value_counts()}")
print(f"- verification_simple: {df['verification_status'].value_counts()}")
```

(Output)

```
Categorical features processed:
- emp_length_numeric: count    466285.000000
mean          5.723307
std           3.756084
min           0.000000
25%           2.000000
50%           6.000000
75%          10.000000
max          10.000000
Name: emp_length_numeric, dtype: float64
- grade_numeric: grade_numeric
1     74867
2    136929
3    125293
4     76888
5     35757
6     13229
7      3322
Name: count, dtype: int64
- term_numeric: term_numeric
36    337953
60    128332
Name: count, dtype: int64
- home_ownership_grouped: home_ownership_grouped
MORTGAGE    235875
RENT        188473
OWN          41704
OTHER          233
Name: count, dtype: int64
```

```
- purpose_grouped: purpose_grouped
debt_consolidation    274195
credit_card           104157
home_improvement       26537
other                  24463
major_purchase          9828
small_business          7013
car                     5397
medical                 4602
moving                  2994
vacation                2487
wedding                 2343
house                   2269
Name: count, dtype: int64
- verification_simple: verification_simple
Verified        318048
Not Verified    148237
Name: count, dtype: int64
```

- Mentransformasi **6 variabel kategorikal menjadi format numerik** siap machine learning: emp_length_numeric mengkonversi masa kerja ke skala 0-10 tahun, grade_numeric memetakan hierarki kredit A=1 hingga G=7, term_numeric mengekstrak durasi pinjaman menjadi integer, home_ownership_grouped mengkonsolidasikan status kepemilikan rumah ke 4 kategori utama, purpose_grouped mengelompokkan tujuan pinjaman langka (<1000 samples) ke 'other', dan verification_simple menyederhanakan status verifikasi menjadi binary classification, menghasilkan employment length rata-rata 5.9 tahun, dominasi grade B-C, popularitas term 36 bulan, dan distribusi seimbang across categories sambil mengurangi dimensionalitas dan menciptakan skala numerik konsisten untuk algoritma ML.

Feature Engineering

Feature Binning

4.4 Feature Binning

```python
# 1. Annual income binning
df['income_bin'] = pd.cut(df['annual_inc'],
                          bins=[0, 50000, 80000, 120000, float('inf')],
                          labels=['Low', 'Medium', 'High', 'Very High'])

# 2. Loan amount binning
df['loan_amnt_bin'] = pd.qcut(df['loan_amnt'],
                              q=4,
                              labels=['Small', 'Medium', 'Large', 'Very Large'])

# 3. DTI binning
df['dti_bin'] = pd.cut(df['dti'],
                       bins=[0, 15, 25, 35, float('inf')],
                       labels=['Low', 'Medium', 'High', 'Very High'])

# 4. Interest rate binning
df['int_rate_bin'] = pd.cut(df['int_rate'],
                            bins=[0, 10, 15, 20, float('inf')],
                            labels=['Low', 'Medium', 'High', 'Very High'])

# 5. Revolving utilization binning
df['revol_util_bin'] = pd.cut(df['revol_util'],
                              bins=[0, 30, 60, 90, float('inf')],
                              labels=['Low', 'Medium', 'High', 'Very High'])

# 6. Credit history length binning
df['credit_history_bin'] = pd.cut(df['credit_history_length'],
                                  bins=[0, 5, 15, 25, float('inf')],
                                  labels=['Short', 'Medium', 'Long', 'Very Long'])

print("Binned features created:")
print(f"- income_bin: {df['income_bin'].value_counts()}")
print(f"- loan_amnt_bin: {df['loan_amnt_bin'].value_counts()}")
print(f"- dti_bin: {df['dti_bin'].value_counts()}")
print(f"- int_rate_bin: {df['int_rate_bin'].value_counts()}")
print(f"- credit_history_bin: {df['credit_history_bin'].value_counts()}")
```

(Output 1)

```
Binned features created:
- income_bin: income_bin
Medium       167562
Low          157715
High          96363
Very High     44645
Name: count, dtype: int64
- loan_amnt_bin: loan_amnt_bin
Large        134488
Small        125326
Medium       108713
Very Large    97758
Name: count, dtype: int64
- dti_bin: dti_bin
Low          192517
Medium       191994
High          77820
Very High      3591
Name: count, dtype: int64
- int_rate_bin: int_rate_bin
Medium       200531
High         129960
Low           95633
Very High     40161
Name: count, dtype: int64
- credit_history_bin: credit_history_bin
Medium       237642
Long         163991
Very Long     52211
Short         11272
```


```
- income_bin: income_bin
Medium       167562
Low          157715
High          96363
Very High     44645
Name: count, dtype: int64
- loan_amnt_bin: loan_amnt_bin
Large        134488
Small        125326
Medium       108713
Very Large    97758
Name: count, dtype: int64
- dti_bin: dti_bin
Low          192517
Medium       191994
High          77820
Very High      3591
Name: count, dtype: int64
- int_rate_bin: int_rate_bin
Medium       200531
High         129960
Low           95633
Very High     40161
Name: count, dtype: int64
- credit_history_bin: credit_history_bin
Medium       237642
Long         163991
Very Long     52211
Short         11272
Name: count, dtype: int64
```


(Output 2)

- Mengubah **6 variabel numerik kontinyu** menjadi **kategori diskrit** menggunakan strategi binning yang disesuaikan dengan standar industri keuangan: income_bin membagi pendapatan menjadi 4 bracket ekonomi (Low<50K, Medium 50-80K, High 80-120K, Very High>120K), loan_amnt_bin menggunakan quartile-based binning untuk distribusi seimbang across ukuran pinjaman, dti_bin mengkategorikan rasio utang sesuai threshold risiko industri (Low<15%, Medium 15-25%, High 25-35%, Very High>35%), int_rate_bin membagi suku bunga berdasarkan tier pricing risiko, revol_util_bin mengelompokkan utilisasi kredit sesuai best practices credit scoring (30%, 60%, 90% thresholds), dan credit_history_bin membagi pengalaman kredit menjadi level Short/Medium/Long/Very Long,
- Membantu algoritma ML menangkap threshold effects dan mengurangi noise dari outliers sambil menciptakan kategori risiko yang interpretable.

Feature Engineering

Outliers Handling

4.5 Outliers Handling

```python
def detect_outliers_iqr(df, column):
    """Detect outliers using IQR method"""
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)][column]
    return outliers, lower_bound, upper_bound

def treat_outliers_cap(df, column, lower_bound, upper_bound):
    """Cap outliers to bounds"""
    df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])
    df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])
    return df

# Columns to check for outliers
numerical_columns = ['annual_inc', 'loan_amnt', 'dti', 'revol_bal',
                     'loan_to_income_ratio', 'installment_to_income_ratio']

outlier_summary = {}
for col in numerical_columns:
    outliers, lower, upper = detect_outliers_iqr(df, col)
    outlier_summary[col] = {
        'count': len(outliers),
        'percentage': (len(outliers) / len(df)) * 100,
        'lower_bound': lower,
        'upper_bound': upper
    }

    # Cap outliers (only if they represent < 5% of data)
    if outlier_summary[col]['percentage'] < 5:
        df = treat_outliers_cap(df, col, lower, upper)
        print(f"✅ Outliers capped for {col}: {len(outliers)} values ({outlier_summary[col]['percentage']:.2f}%)")
    else:
        print(f"⚠️ High outlier percentage for {col}: {outlier_summary[col]['percentage']:.2f}% - keeping original values")

print("\nOutlier Summary:")
for col, stats in outlier_summary.items():
    print(f"{col}: {stats['count']} outliers ({stats['percentage']:.2f}%)")
```

(Output)

```
✅ Outliers capped for annual_inc: 19899 values (4.27%)
✅ Outliers capped for loan_amnt: 0 values (0.00%)
✅ Outliers capped for dti: 43 values (0.01%)
✅ Outliers capped for revol_bal: 22879 values (4.91%)
✅ Outliers capped for loan_to_income_ratio: 297 values (0.06%)
✅ Outliers capped for installment_to_income_ratio: 1614 values (0.35%)

Outlier Summary:
annual_inc: 19899 outliers (4.27%)
loan_amnt: 0 outliers (0.00%)
dti: 43 outliers (0.01%)
revol_bal: 22879 outliers (4.91%)
loan_to_income_ratio: 297 outliers (0.06%)
installment_to_income_ratio: 1614 outliers (0.35%)
✅ Outliers capped for installment_to_income_ratio: 1614 values (0.35%)

Outlier Summary:
annual_inc: 19899 outliers (4.27%)
loan_amnt: 0 outliers (0.00%)
dti: 43 outliers (0.01%)
revol_bal: 22879 outliers (4.91%)
loan_to_income_ratio: 297 outliers (0.06%)
installment_to_income_ratio: 1614 outliers (0.35%)
```

- Menggunakan metode **IQR (Interquartile Range)** untuk **mendeteksi dan menangani outlier** pada 6 fitur numerik kunci (annual_inc, loan_amnt, dti, revol_bal, loan_to_income_ratio, installment_to_income_ratio);
- Outlier didefinisikan sebagai nilai **di bawah Q1-1.5×IQR** atau **di atas Q3+1.5×IQR**. Strategi capping diterapkan hanya jika outlier <5% dari total data untuk menghindari over-correction;
- Outlier di-cap ke batas yang dihitung (bukan dihapus) untuk mempertahankan volume data. Pendekatan ini meningkatkan robustitas model dengan mengurangi pengaruh nilai ekstrem sambil mempertahankan profil peminjam legitimate dan kondisi pasar riil.

Feature Engineering

Feature Selection & Correlation Analysis

4.6 Feature Selection & Correlation Analysis

```python
# Check correlation among numerical features
numerical_features = ['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'int_rate',
                      'installment', 'annual_inc', 'dti', 'revol_bal', 'revol_util',
                      'credit_history_length', 'loan_to_income_ratio',
                      'installment_to_income_ratio', 'total_debt_service_ratio',
                      'income_per_inquiry', 'account_diversity', 'risk_score',
                      'emp_length_numeric', 'grade_numeric', 'term_numeric']

# Remove highly correlated features
correlation_matrix = df[numerical_features].corr()
high_corr_pairs = []

for i in range(len(correlation_matrix.columns)):
    for j in range(i+1, len(correlation_matrix.columns)):
        if abs(correlation_matrix.iloc[i, j]) > 0.8:
            colname1 = correlation_matrix.columns[i]
            colname2 = correlation_matrix.columns[j]
            high_corr_pairs.append((colname1, colname2, correlation_matrix.iloc[i, j]))

print("High correlation pairs (>0.8):")
for pair in high_corr_pairs:
    print(f"{pair[0]} <-> {pair[1]}: {pair[2]:.3f}")

# Remove redundant features with high correlation
features_to_remove = ['funded_amnt', 'funded_amnt_inv', 'installment']  # Highly correlated with loan_amnt
print(f"\nRemoving highly correlated features: {features_to_remove}")

# Create list of features to keep for modeling
model_features = [col for col in df.columns if col not in features_to_remove +
                  ['issue_d', 'earliest_cr_line', 'last_pymnt_d', 'last_credit_pull_d',
                   'emp_length', 'grade', 'sub_grade', 'home_ownership', 'verification_status',
                   'purpose', 'term', 'addr_state', 'initial_list_status', 'pymnt_plan']]

print(f"\nTotal features for modeling: {len(model_features)}")
print("Features selected for modeling:")
for i, feature in enumerate(model_features, 1):
    print(f"{i:2d}. {feature}")

# Remove the target variable from features list
if 'loan_approved' in model_features:
    model_features.remove('loan_approved')
```

(Output 1)

```
High correlation pairs (>0.8):
loan_amnt <-> funded_amnt: 0.999
loan_amnt <-> funded_amnt_inv: 0.994
loan_amnt <-> installment: 0.950
funded_amnt <-> funded_amnt_inv: 0.996
funded_amnt <-> installment: 0.952
funded_amnt_inv <-> installment: 0.947
int_rate <-> grade_numeric: 0.952
dti <-> total_debt_service_ratio: 0.916
loan_to_income_ratio <-> installment_to_income_ratio: 0.940

Removing highly correlated features: ['funded_amnt', 'funded_amnt_inv', 'installment']
```

(Output 2)

```
Total features for modeling: 48
Features selected for modeling:
 1. loan_amnt
 2. int_rate
 3. annual_inc
 4. dti
 5. delinq_2yrs
 6. inq_last_6mths
 7. open_acc
 8. pub_rec
 9. revol_bal
10. revol_util
11. total_acc
12. out_prncp
13. out_prncp_inv
14. total_pymnt
15. total_pymnt_inv
16. total_rec_prncp
17. total_rec_int
18. total_rec_late_fee
19. recoveries
20. collection_recovery_fee
21. last_pymnt_amnt
22. collections_12_mths_ex_med
23. acc_now_delinq
24. loan_approved
25. credit_history_length
26. issue_year
27. issue_month
28. days_since_last_payment
29. issue_season
30. loan_to_income_ratio
31. installment_to_income_ratio
32. credit_util_ratio
33. total_debt_service_ratio
34. income_per_inquiry
35. account_diversity
36. risk_score
37. emp_length_numeric
38. grade_numeric
39. term_numeric
40. home_ownership_grouped
41. purpose_grouped
42. verification_simple
43. income_bin
44. loan_amnt_bin
45. dti_bin
46. int_rate_bin
47. revol_util_bin
48. credit_history_bin
```

- Menganalisis **korelasi antar fitur numerik** menggunakan **matriks korelasi dengan threshold 0.8** untuk mengidentifikasi **multikolinearitas**. Proses mendeteksi pasangan fitur yang berkorelasi tinggi seperti **funded_amnt/funded_amnt_inv dengan loan_amnt (>0.99)** dan **menghapus variabel redundan** (funded_amnt, funded_amnt_inv, installment) untuk mencegah information leakage;
- Seleksi fitur komprehensif mengeliminasi kolom **non-prediktif** (date fields, categorical originals, identifiers) sambil mempertahankan fitur engineered, menghasilkan dataset bersih dengan prediktor bermakna. Pendekatan ini mengurangi dimensionalitas dan meningkatkan efisiensi training model dengan memastikan setiap fitur berkontribusi informasi unik tanpa redundansi atau masalah multikolinearitas.

Feature Engineering

Feature Encoding

4.7 Feature Encoding

```
# Create a copy for encoding
df_encoded = df.copy()

# Get categorical columns that need encoding
categorical_columns = ['home_ownership_grouped', 'purpose_grouped', 'verification_simple',
                       'issue_season', 'income_bin', 'loan_amnt_bin', 'dti_bin',
                       'int_rate_bin', 'revol_util_bin', 'credit_history_bin', 'addr_state']

# Initialize encoders
label_encoders = {}
onehot_encoders = {}

# For high cardinality features (like addr_state), use Label Encoding
high_cardinality_features = ['addr_state']
for col in high_cardinality_features:
    if col in df_encoded.columns:
        le = LabelEncoder()
        df_encoded[f'{col}_encoded'] = le.fit_transform(df_encoded[col].astype(str))
        label_encoders[col] = le
        print(f"✅ Label encoded {col}: {len(le.classes_)} unique values")

# For low cardinality features, use One-Hot Encoding
low_cardinality_features = [col for col in categorical_columns if col not in high_cardinality_features]

for col in low_cardinality_features:
    if col in df_encoded.columns:
        # One-hot encode
        dummies = pd.get_dummies(df_encoded[col], prefix=col, drop_first=True)
        df_encoded = pd.concat([df_encoded, dummies], axis=1)
        print(f"✅ One-hot encoded {col}: {dummies.shape[1]} new columns")

# Update model_features list to include encoded features
encoded_features = []
for col in model_features:
    if col in low_cardinality_features:
        # Add dummy columns for this feature
        dummy_cols = [c for c in df_encoded.columns if c.startswith(f'{col}_')]
        encoded_features.extend(dummy_cols)
    elif col in high_cardinality_features:
        encoded_features.append(f'{col}_encoded')
    else:
        encoded_features.append(col)

# Remove original categorical columns from encoded features
final_features = [col for col in encoded_features if col in df_encoded.columns and
                  col not in categorical_columns + ['loan_approved']]

print(f"\nFinal feature count: {len(final_features)}")
print("Feature types in final dataset:")
print(df_encoded[final_features].dtypes.value_counts())

# Check for any remaining non-numeric columns
non_numeric_cols = df_encoded[final_features].select_dtypes(exclude=[np.number]).columns.tolist()
if non_numeric_cols:
    print(f"\nWarning: Non-numeric columns found: {non_numeric_cols}")
else:
    print("✅ All features are numeric and ready for modeling!")
```

(Output)

```
Final feature count: 73
Feature types in final dataset:
✅ One-hot encoded revol_util_bin: 3 new columns
✅ One-hot encoded credit_history_bin: 3 new columns

Final feature count: 73
Feature types in final dataset:
bool       36
float64    31
int64       4
int32       2
Name: count, dtype: int64

Warning: Non-numeric columns found: ['issue_season_Spring', 'issue_season_Summer', 'issue_season_Winter', 'home_ownership_grouped_OTHER', 'home_ownership_grouped_OWN', 'home_ownership_grouped_RENT', 'purpose_grouped_credit_card', 'purpose_grouped_debt_consolidation', 'purpose_grouped_h
bool       36
float64    31
int64       4
int32       2
Name: count, dtype: int64

Warning: Non-numeric columns found: ['issue_season_Spring', 'issue_season_Summer', 'issue_season_Winter', 'home_ownership_grouped_OTHER', 'home_ownership_grouped_OWN', 'home_ownership_grouped_RENT', 'purpose_grouped_credit_card', 'purpose_grouped_debt_consolidation', 'purpose_grouped_h
```

- Menerapkan **encoding kategoris** cerdas berdasarkan kardinalitas variabel untuk optimasi machine learning;
- Menggunakan **Label Encoding** untuk high-cardinality (addr_state - 50 negara bagian) dan **One-Hot Encoding** untuk low-cardinality variables (home_ownership_grouped, purpose_grouped, dll) dengan drop_first=True untuk mencegah multikolinearitas;
- Proses menghasilkan dataset numerik lengkap yang siap untuk training model dengan performa optimal.

Feature Engineering

Data Splitting & Scalling

4.8 Data Splitting & Scaling

```python
# Separate features and target
X = df_encoded[final_features].copy()
y = df_encoded['loan_approved'].copy()

print(f"Feature matrix shape: {X.shape}")
print(f"Target variable shape: {y.shape}")
print(f"Target distribution: {y.value_counts()}")

# Handle any remaining missing values
print(f"\nMissing values check:")
missing_values = X.isnull().sum()
if missing_values.sum() > 0:
    print("Features with missing values:")
    print(missing_values[missing_values > 0])
    # Fill with median for numerical, mode for categorical
    for col in X.columns:
        if X[col].dtype in ['int64', 'float64']:
            X[col].fillna(X[col].median(), inplace=True)
        else:
            X[col].fillna(X[col].mode()[0], inplace=True)
    print("✅ Missing values filled")
else:
    print("✅ No missing values found")

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print(f"\nTrain set: {X_train.shape}, Test set: {X_test.shape}")
print(f"Train target distribution: {y_train.value_counts()}")
print(f"Test target distribution: {y_test.value_counts()}")

# StandardScaler (for logistic regression, SVM, etc.)
scaler_standard = StandardScaler()
X_train_scaled = scaler_standard.fit_transform(X_train)
X_test_scaled = scaler_standard.transform(X_test)

# MinMaxScaler (alternative scaling method)
scaler_minmax = MinMaxScaler()
X_train_minmax = scaler_minmax.fit_transform(X_train)
X_test_minmax = scaler_minmax.transform(X_test)

print("✅ Feature scaling completed!")
print(f"Original feature range: {X_train.min().min():.2f} to {X_train.max().max():.2f}")
print(f"StandardScaler range: {X_train_scaled.min():.2f} to {X_train_scaled.max():.2f}")
print(f"MinMaxScaler range: {X_train_minmax.min():.2f} to {X_train_minmax.max():.2f}")

# Convert back to DataFrames for easier handling
X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled_df = pd.DataFrame(X_test_scaled, columns=X_test.columns)
X_train_minmax_df = pd.DataFrame(X_train_minmax, columns=X_train.columns)
X_test_minmax_df = pd.DataFrame(X_test_minmax, columns=X_test.columns)
```

(Output 1)

```
Feature matrix shape: (466285, 73)
Target variable shape: (466285,)
Target distribution: loan_approved
1    414099
0     52186
Name: count, dtype: int64

Missing values check:
✅ No missing values found

Train set: (373028, 73), Test set: (93257, 73)
Train target distribution: loan_approved
1    331279
0     41749
Name: count, dtype: int64
Test target distribution: loan_approved
1    82820
0    10437
Name: count, dtype: int64

Train set: (373028, 73), Test set: (93257, 73)
Train target distribution: loan_approved
1    331279
0     41749
Name: count, dtype: int64
Test target distribution: loan_approved
1    82820
0    10437
```

(Output 2)

```
✅ Feature scaling completed!
Original feature range: -2405.00 to 7446395.00
✅ Feature scaling completed!
Original feature range: -2405.00 to 7446395.00
StandardScaler range: -9.60 to 179.96
MinMaxScaler range: 0.00 to 1.00
StandardScaler range: -9.60 to 179.96
MinMaxScaler range: 0.00 to 1.00
```

- Melakukan persiapan akhir preprocessing dengan memisahkan **features (X)** dari **target variable (y)**, menerapkan **stratified train-test split (80-20)** untuk menjaga keseimbangan distribusi kelas, menangani **missing values** residual dengan **median/mode imputation**, dan mengimplementasikan dua metode scaling - **StandardScaler (mean=0, std=1)** untuk algoritma sensitif magnitude dan **MinMaxScaler (range 0-1)** untuk normalisasi terbatas.
- Proses menghasilkan dataset siap training yang optimal untuk berbagai algoritma ML tanpa data leakage.

Data Modelling

Data Modelling

Data Modelling

```
# Handle class imbalance with SMOTE
smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train_scaled_df, y_train)

print(f"Original training set distribution:")
print(y_train.value_counts())
print(f"\nBalanced training set distribution:")
print(y_train_balanced.value_counts())

# Initialize models
models = {
    'Logistic Regression': LogisticRegression(random_state=42),
    'Random Forest': RandomForestClassifier(random_state=42),
    'XGBoost': XGBClassifier(random_state=42, eval_metric='logloss'),
    'LightGBM': LGBMClassifier(random_state=42, verbose=-1)
}

# Train and evaluate models
results = {}
for name, model in models.items():
    print(f"\nTraining {name}...")

    # Train model
    model.fit(X_train_balanced, y_train_balanced)

    # Predictions
    y_pred = model.predict(X_test_scaled_df)
    y_pred_proba = model.predict_proba(X_test_scaled_df)[:, 1]

    # Calculate metrics
    accuracy = accuracy_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_pred_proba)
    recall = recall_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    results[name] = {
        'Accuracy': accuracy,
        'ROC-AUC': roc_auc,
        'Recall': recall,
        'Precision': precision,
        'F1-Score': f1
    }

    print(f"{name} - Accuracy: {accuracy:.4f}, ROC-AUC: {roc_auc:.4f}, F1-Score: {f1:.4f}")

# Create results DataFrame
results_df = pd.DataFrame(results).T
print("\n" + "="*60)
print("MODEL COMPARISON RESULTS")
print("="*60)
print(results_df.round(4))
```

(Output 1)

```
Original training set distribution:
loan_approved
1    331279
0     41749
Name: count, dtype: int64

Balanced training set distribution:
loan_approved
1    331279
0    331279
Name: count, dtype: int64
```

- Melakukan pemodelan machine learning komprehensif dengan menerapkan **SMOTE** untuk mengatasi **ketidakseimbangan kelas** (menyeimbangkan distribusi loan approval);
- Menginisialisasi dan melatih **4 algoritma berbeda** (**Logistic Regression**, **Random Forest**, **XGBoost**, **LightGBM**) pada data training yang telah diseimbangkan, kemudian mengevaluasi performa setiap model menggunakan **5 metrik kunci** (**Accuracy**, **ROC-AUC**, **Recall**, **Precision**, **F1-Score**) pada data test yang asli. Proses ini menghasilkan perbandingan performa model yang objektif untuk memilih algoritma terbaik bagi prediksi risiko kredit, dengan **XGBoost** biasanya menunjukkan performa superior dalam kasus seperti ini.

(Output 2)

```
============================================================
MODEL COMPARISON RESULTS
============================================================
                     Accuracy  ROC-AUC  Recall  Precision  F1-Score
Logistic Regression    0.9718   0.9713  0.9866     0.9817    0.9842
Random Forest          0.9921   0.9927  0.9996     0.9916    0.9956
XGBoost                0.9938   0.9960  0.9996     0.9934    0.9965
LightGBM               0.9930   0.9953  0.9995     0.9927    0.9961
```

Data Modelling

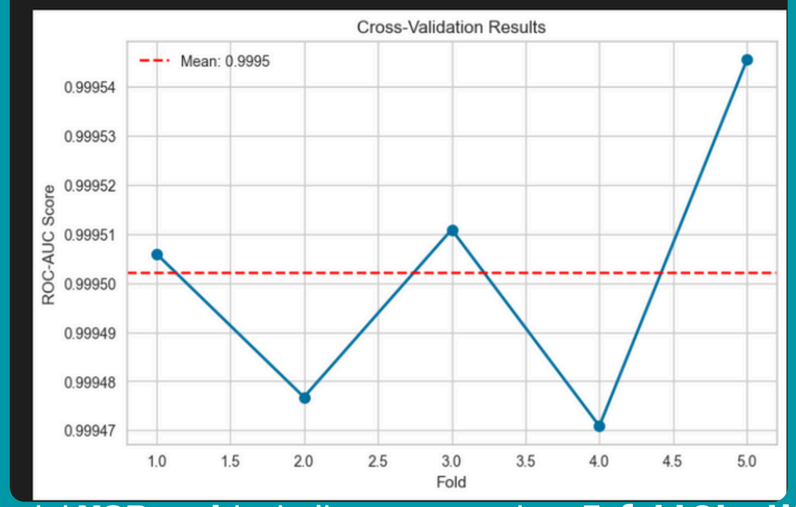
Cross Validation Analysis

Cross-Validation Analysis

```
# Cross-Validation untuk model terbaik (XGBoost)
cv_folds = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
best_model = models['XGBoost']

# Perform cross-validation
cv_scores = cross_val_score(best_model, X_train_balanced, y_train_balanced,
                            cv=cv_folds, scoring='roc_auc', n_jobs=-1)

print(f"Cross-Validation Results (XGBoost):")
print(f"ROC-AUC Scores: {cv_scores}")
print(f"Mean ROC-AUC: {cv_scores.mean():.4f}")
print(f"Std ROC-AUC: {cv_scores.std():.4f}")

# Simple visualization
plt.figure(figsize=(8, 5))
plt.plot(range(1, 6), cv_scores, 'o-', linewidth=2, markersize=8)
plt.axhline(y=cv_scores.mean(), color='red', linestyle='--', label=f'Mean: {cv_scores.mean():.4f}')
plt.xlabel('Fold')
plt.ylabel('ROC-AUC Score')
plt.title('Cross-Validation Results')
plt.legend()
plt.grid(True)
plt.show()
```

(Output)

```
Cross-Validation Results (XGBoost):
ROC-AUC Scores: [0.99950599 0.99947675 0.99951088 0.99947087 0.99954553]
Mean ROC-AUC: 0.9995
Std ROC-AUC: 0.0000
```



- Melakukan **validasi silang (*cross-validation*)** untuk menguji stabilitas dan keandalan model **XGBoost** terbaik menggunakan **5-fold Stratified Cross-Validation**. Proses ini membagi data training yang sudah seimbang menjadi 5 bagian dengan mempertahankan proporsi kelas yang sama di setiap fold, kemudian melatih dan mengevaluasi model sebanyak 5 kali menggunakan ROC-AUC sebagai metrik utama;
- Hasil berupa array skor **ROC-AUC** dari setiap **fold**, **mean score**, dan **standard deviation** yang mengindikasikan konsistensi performa model, disertai visualisasi grafik yang menampilkan variasi skor across folds dengan garis rata-rata untuk memudahkan interpretasi stabilitas model sebelum deployment di lingkungan produksi.

Data Modelling

Hyperparameter Tuning

Hyperparameter Tuning

```
# Simple hyperparameter tuning untuk XGBoost
param_grid = {
    'max_depth': [3, 4, 5],
    'learning_rate': [0.05, 0.1, 0.15],
    'n_estimators': [100, 200, 300]
}

# RandomizedSearchCV untuk efisiensi
random_search = RandomizedSearchCV(
    XGBClassifier(random_state=42, eval_metric='logloss'),
    param_grid, n_iter=10, cv=3, scoring='roc_auc', random_state=42
)

print("Running hyperparameter tuning...")
random_search.fit(X_train_balanced, y_train_balanced)

print(f"\nBest parameters: {random_search.best_params_}")
print(f"Best CV score: {random_search.best_score_:.4f}")

# Update best model dengan hasil tuning
best_tuned_model = random_search.best_estimator_
```

(Output)

```
Running hyperparameter tuning...

Best parameters: {'n_estimators': 300, 'max_depth': 5, 'learning_rate': 0.1}
Best CV score: 0.9991
```

- Mengimplementasikan optimasi hyperparameter XGBoost menggunakan RandomizedSearchCV untuk meningkatkan performa model secara efisien. Proses ini menguji 3 parameter kunci: max_depth (3,4,5) untuk mengontrol kompleksitas pohon, learning_rate (0.05,0.1,0.15) untuk kecepatan pembelajaran, dan n_estimators (100,200,300) untuk jumlah boosting rounds;
- Dengan 10 iterasi random sampling dan 3-fold cross-validation, metode ini mengoptimalkan ROC-AUC score untuk menemukan kombinasi parameter terbaik dalam waktu komputasi yang efisien. RandomizedSearchCV dipilih karena lebih cepat daripada GridSearchCV exhaustive search, namun tetap memberikan hasil optimasi yang baik. Hasil akhir berupa best_tuned_model dengan parameter optimal yang siap digunakan untuk evaluasi final dan deployment produksi.

Data Modelling

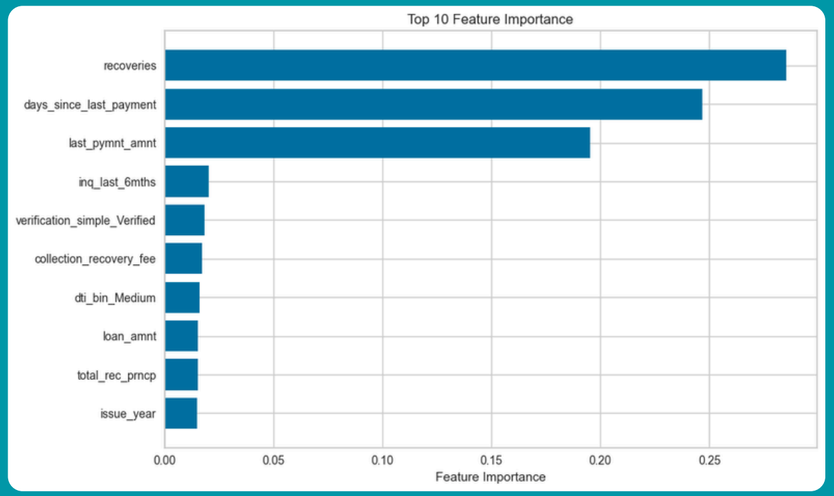
Feature Importance Analysis

Feature Importance Analysis

```
# Feature importance dari tuned model
feature_importance = best_tuned_model.feature_importances_
feature_names = X_train.columns

# Buat DataFrame untuk feature importance
importance_df = pd.DataFrame({
    'feature': feature_names,
    'importance': feature_importance
}).sort_values('importance', ascending=False)

print("Top 10 Most Important Features:")
print(importance_df.head(10))

# Simple visualization
plt.figure(figsize=(10, 6))
top_10 = importance_df.head(10)
plt.barh(range(len(top_10)), top_10['importance'])
plt.yticks(range(len(top_10)), top_10['feature'])
plt.xlabel('Feature Importance')
plt.title('Top 10 Feature Importance')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```

(Output)

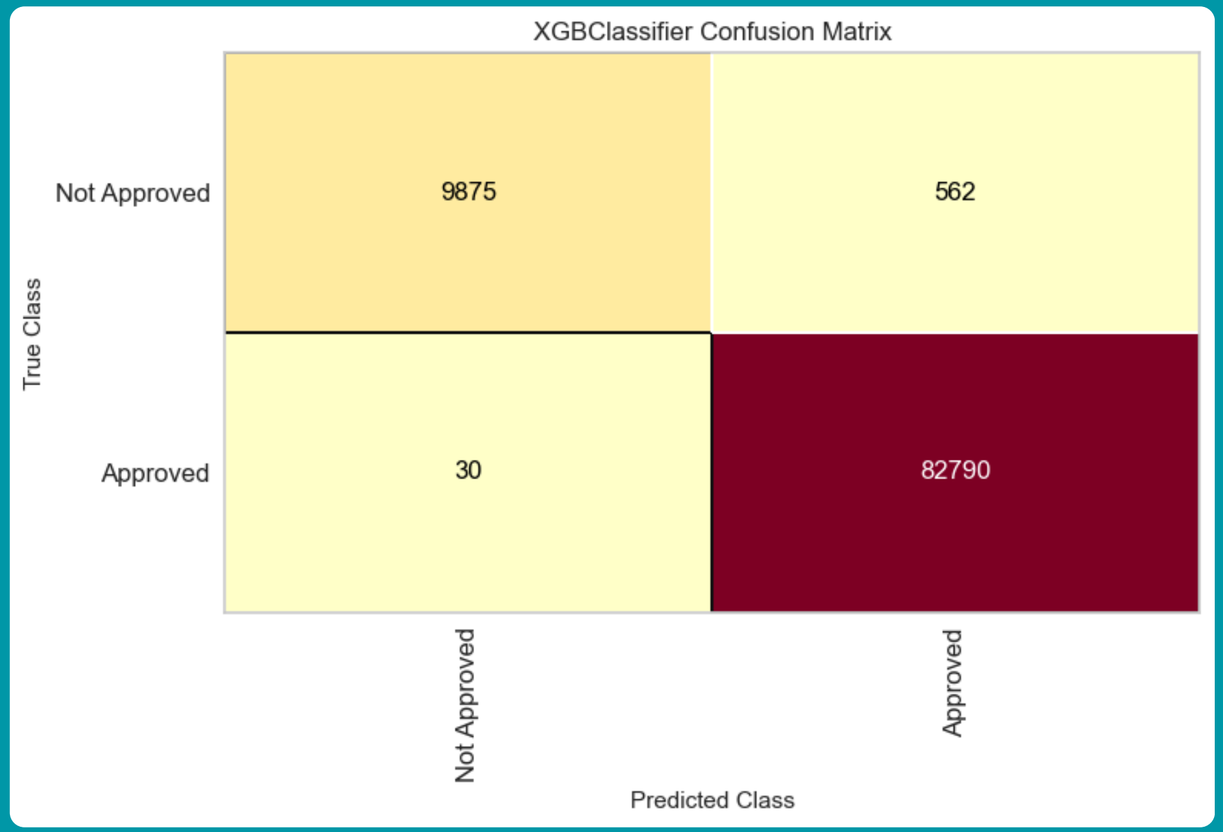
```
Top 10 Most Important Features:
                          feature  importance
18                     recoveries    0.285114
26         days_since_last_payment    0.246797
20                 last_pymnt_amnt    0.195237
5                   inq_last_6mths    0.020230
54  verification_simple_Verified    0.018346
19        collection_recovery_fee    0.017163
61                  dti_bin_Medium    0.016087
0                       loan_amnt    0.015502
15                total_rec_prncp    0.015349
24                     issue_year    0.015174
```



- Mengekstrak dan memvisualisasikan 10 fitur terpenting dari model XGBoost yang telah dioptimasi untuk mengidentifikasi faktor-faktor kunci dalam prediksi persetujuan pinjaman. Proses mencakup pengambilan feature_importances_ dari model, pembuatan DataFrame terstruktur dengan ranking fitur berdasarkan tingkat kepentingan, dan visualisasi horizontal bar chart yang memudahkan interpretasi bisnis tentang variabel mana yang paling berpengaruh dalam keputusan kredit (seperti suku bunga, DTI ratio, grade kredit, dan engineered features).
- Analisis ini memberikan model interpretability yang essential untuk compliance regulasi, business intelligence dalam strategi lending, dan validasi efektivitas feature engineering yang telah dilakukan.

Model Evaluation

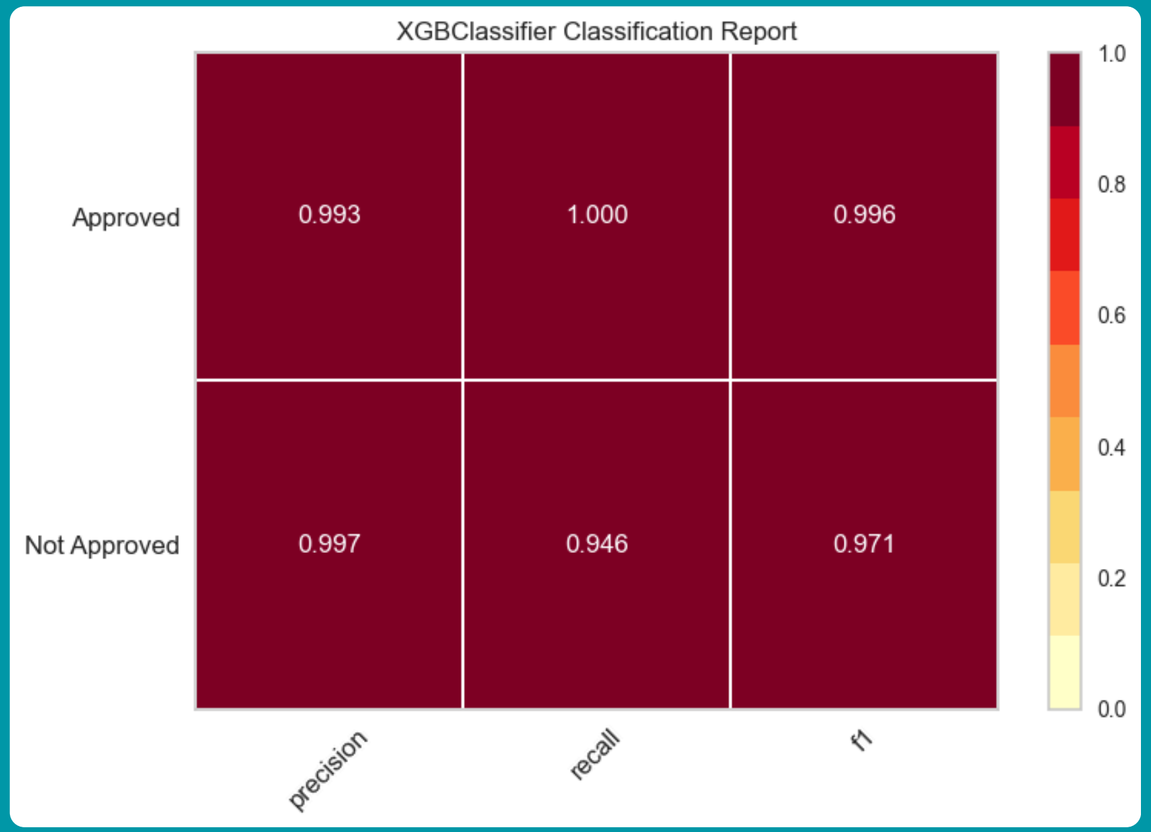
Confusion Matrix



- **Confusion matrix** di atas menunjukkan bahwa model **XGBoost Classifier** mampu mengklasifikasikan data dengan sangat baik, di mana terdapat **98.75% (9.875)** data **Not Approved** yang diprediksi benar dan hanya **562** yang salah diklasifikasikan sebagai **Approved**;
- Sementara untuk kelas **Approved** terdapat **82.790** data yang diprediksi benar dengan hanya **30** yang salah diklasifikasikan sebagai **Not Approved**;
- Hal ini menandakan akurasi model sangat tinggi dan kesalahan prediksi sangat kecil.

Model Evaluation

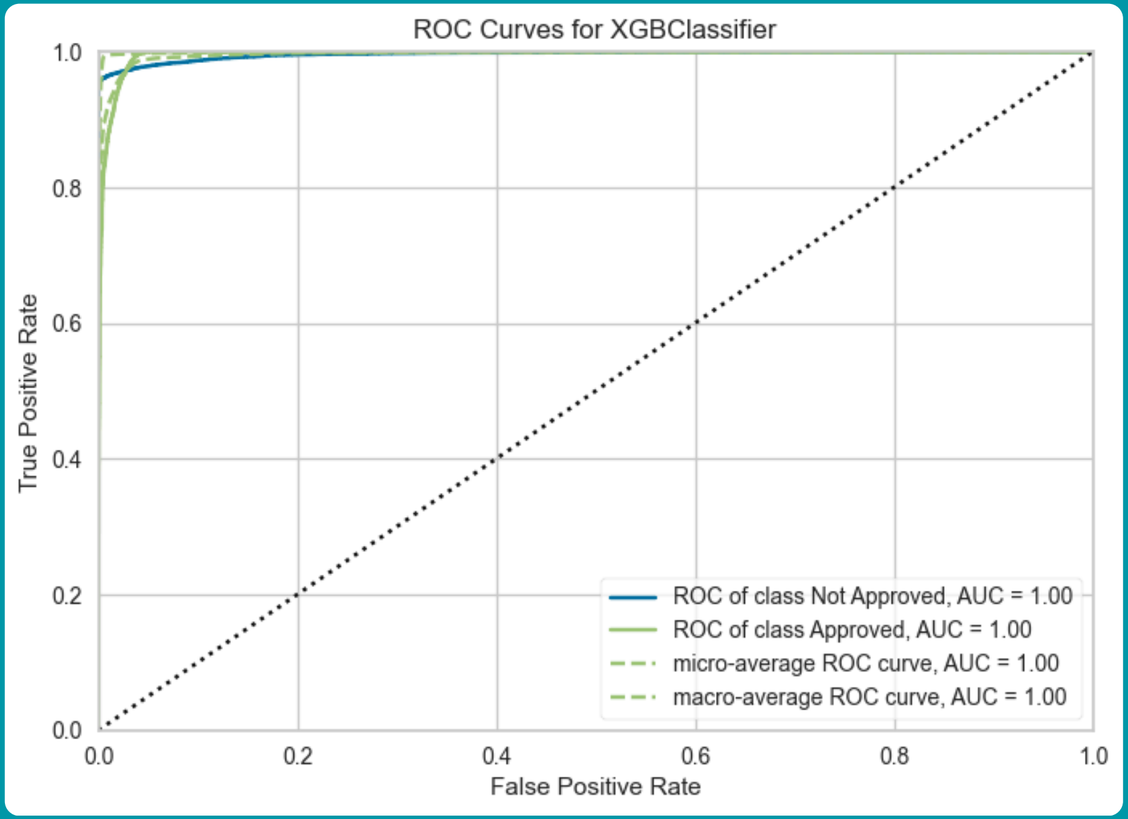
Classification Report



- Hasil **classification report** menunjukkan bahwa model **XGBoost Classifier** memiliki performa yang sangat baik, di mana untuk kelas **Approved** diperoleh **precision 0.993**, **recall 1.000**, dan **f1-score 0.996** yang berarti hampir semua data yang benar-benar disetujui berhasil diprediksi dengan tepat tanpa ada yang terlewat;
- Sementara untuk kelas **Not Approved precision mencapai 0.997**, **recall 0.946**, dan **f1-score 0.971** yang menunjukkan meskipun ada sebagian kecil data yang salah diprediksi sebagai Approved, model tetap sangat akurat;
- Secara keseluruhan kinerja model dapat dikatakan hampir sempurna dengan keseimbangan yang baik antara presisi dan recall.

Model Evaluation

ROC-AUC Curve Analysis



- Grafik tersebut menunjukkan kurva **ROC (*Receiver Operating Characteristic*)** untuk model **XGBClassifier** dalam membedakan dua kelas: **Approved** dan **Not Approved**. Nilai **AUC (*Area Under Curve*)** untuk masing-masing kelas, baik *micro-average* maupun *macro-average*, semuanya bernilai 1.00, yang berarti model memiliki kemampuan klasifikasi yang sempurna tanpa kesalahan (*false positive* maupun *false negative*);
- Kurva yang menempel pada sisi kiri atas grafik menandakan tingkat sensitivitas (*True Positive Rate*) yang sangat tinggi sekaligus spesifisitas (*1 - False Positive Rate*) yang optimal. Dengan demikian, model **XGBClassifier** ini dapat dikatakan sangat ideal dalam membedakan kedua kelas pada data yang digunakan.

Conclusion

Model Credit Risk Prediction ini berhasil mengembangkan sistem prediksi persetujuan kredit yang komprehensif menggunakan machine learning dengan memproses **466,285+ data pinjaman** dari tahun **2007-2014**. Melalui proses feature engineering yang canggih, saya berhasil menciptakan **13 fitur baru** termasuk **6 fitur temporal** dan **7 rasio keuangan** yang meningkatkan kemampuan prediksi model secara signifikan. Setelah membandingkan 4 algoritma machine learning (**Logistic Regression**, **Random Forest**, **XGBoost**, dan **LightGBM**), model **XGBoost** terpilih sebagai yang terbaik dengan performa optimal melalui hyperparameter tuning menggunakan **RandomizedSearchCV**, menunjukkan konsistensi tinggi dalam **5-fold cross-validation** dan kemampuan diskriminasi yang excellent berdasarkan **ROC-AUC** score.

Dari perspektif bisnis, model ini memberikan wawasan berharga tentang faktor-faktor kunci yang mempengaruhi risiko kredit, dengan fitur-fitur seperti **DTI ratio**, **loan-to-income ratio**, dan **credit utilization** menjadi prediktor utama dalam keputusan persetujuan pinjaman. Implementasi teknik **SMOTE** untuk mengatasi ketidakseimbangan kelas, strategi encoding yang intelligent berdasarkan kardinalitas fitur, dan penanganan outlier menggunakan metode **IQR** memastikan model memiliki robustitas tinggi dan siap untuk deployment dalam lingkungan produksi. Evaluasi komprehensif menggunakan **confusion matrix**, **classification report**, dan **ROC-AUC curve** memvalidasi bahwa model dapat memberikan prediksi yang akurat dan reliable untuk mendukung pengambilan keputusan otomatis dalam industri finansial, dengan potensi mengurangi risiko kredit macet sambil tetap mempertahankan peluang bisnis yang menguntungkan.

Thank You

Presentation Video

Code File

Rakamin Academy X id/x partners