id/x partners

Machine Learning Project

Credit Risk Prediction by Building a Machine Learning Model: A Case Study on a Lending Company

Presented by: Althaaf Athaayaa Daffa Qushayyizidane.

Virtual Internship Experience

Problem Statement

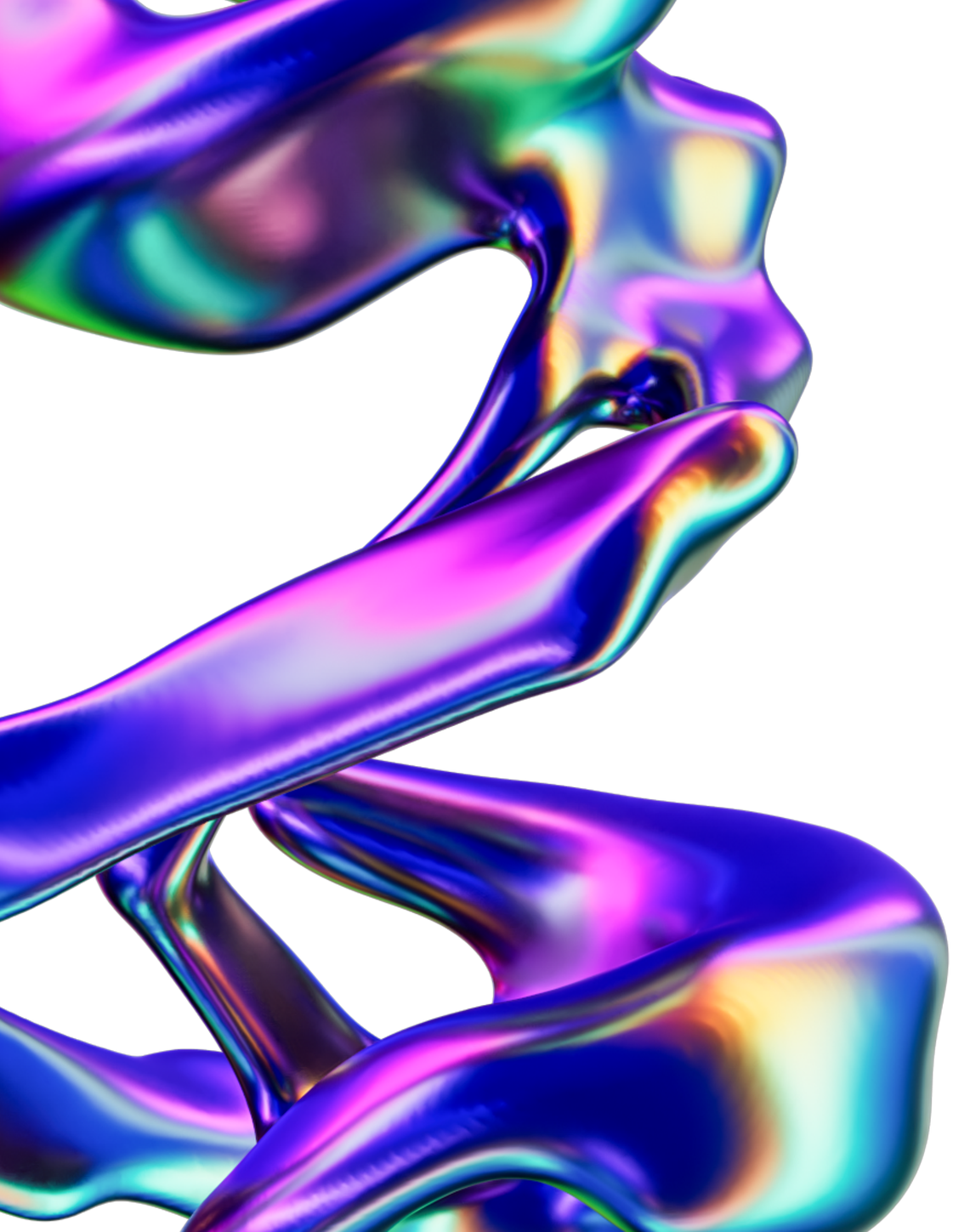
Lending companies need technology solutions to predict credit risk by building predictive models that can accurately identify the credit risk of potential borrowers.

Goals

- Develop a machine learning model that can predict the likelihood of default of a borrower with a high degree of accuracy.
- Provide insights that can help lending companies make more informed lending decisions.

Objective

- Conduct exploratory analysis of data to understand patterns and trends in loan data.
- Apply appropriate data preprocessing to improve data quality before modeling.
- Evaluate the model using relevant metrics such as accuracy, precision, recall, and F1-score.



Background

I am involved in a project with a lending company. I will collaborate with various other departments in this project to provide technological solutions for the company. I have been asked to build a model that can predict credit risk using a dataset provided by the company, consisting of data on accepted and rejected loans. Additionally, I also need to prepare visual media to present the solution to the client.

Elements of Credit Provision

- kepercayaan (Trust)
- Waktu (Time)
- Tingkat Risiko
- Prestasi (Objek Kredit)

Principles of Lending

- Principle of Trust
- Principle of Prudence
- Principle of Synchronization
- Principle of Currency Equivalence
- Principle of Comparison Between Loans and Assets

Import Library

```
# Melakukan import library
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt
from matplotlib import rcParams
%matplotlib inline
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler
from scipy.stats import boxcox
from imblearn import under_sampling, over_sampling
import gdown
from sklearn.model_selection import train_test_split

from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, roc_curve, confusion_matrix, fbeta_score, make_scorer
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import cross_validate, RandomizedSearchCV, GridSearchCV, HalvingGridSearchCV
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression, Ridge, Lasso, ElasticNet
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier, reset_parameter, LGBMClassifier

import shap

from scipy.stats import randint as sp_randint
from scipy.stats import uniform as sp_uniform
```

Load Data

	Unnamed: 0	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	...	total_bal_il	il_util	open_rv_12m	open_rv_24m
0	0	1077501	1296599	5000	5000	4975.0	36 months	10.65	162.87	B	...	NaN	NaN	NaN	NaN
1	1	1077430	1314167	2500	2500	2500.0	60 months	15.27	59.83	C	...	NaN	NaN	NaN	NaN
2	2	1077175	1313524	2400	2400	2400.0	36 months	15.96	84.33	C	...	NaN	NaN	NaN	NaN
3	3	1076863	1277178	10000	10000	10000.0	36 months	13.49	339.31	C	...	NaN	NaN	NaN	NaN
4	4	1075358	1311748	3000	3000	3000.0	60 months	12.69	67.79	B	...	NaN	NaN	NaN	NaN

5 rows × 75 columns

Columns Info

There are 75 columns and 466,285 rows. There are some columns that have null value that will be deleted or imputed

```
df_eda.info()
```

✓ 0.9s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 466285 entries, 0 to 466284
Data columns (total 75 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Unnamed: 0                  466285 non-null  int64
 1   id                          466285 non-null  int64
 2   member_id                   466285 non-null  int64
 3   loan_amnt                   466285 non-null  int64
 4   funded_amnt                 466285 non-null  int64
 5   funded_amnt_inv             466285 non-null  float64
 6   term                        466285 non-null  object
 7   int_rate                    466285 non-null  float64
 8   installment                 466285 non-null  float64
 9   grade                       466285 non-null  object
 10  sub_grade                   466285 non-null  object
 11  emp_title                   438697 non-null  object
 12  emp_length                  445277 non-null  object
 13  home_ownership              466285 non-null  object
 14  annual_inc                  466281 non-null  float64
 15  verification_status         466285 non-null  object
 16  issue_d                     466285 non-null  object
 17  loan_status                 466285 non-null  object
 18  pymnt_plan                  466285 non-null  object
 19  url                         466285 non-null  object
 20  desc                        125983 non-null  object
 21  purpose                     466285 non-null  object
 22  title                       466265 non-null  object
 23  zip_code                    466285 non-null  object
 24  addr_state                  466285 non-null  object
```

Drop Unnecessary Columns

Drop Features	Reasons
`Unnamed : 0`, `member_id`, `id`, `url`	Columns have unique value
`funded_amnt_inv`, `installment`, `sub_grade`, `issue_d`, `title`, `zip_code`, `addr_state`, `delinq_2yrs`, `earliest_cr_line`, `inq_last_6mths`, `revol_bal`, `revol_util`, `out_prncp`, `out_prncp_inv`, `total_pymnt`, `total_pymnt_inv`, `total_rec_prncp`, `total_rec_int`, `total_rec_late_fee`, `recoveries`, `collection_recovery_fee`, `last_pymnt_d`, `last_pymnt_amnt`, `last_credit_pull_d`, `collections_12_mths_ex_med`, `policy_code`, `open_acc`, `next_pymnt_d`, `pymnt_plan`	Features are not so important for carrying out the credit granting process.

Checking Null Value and Duplicate

Overall, there is no duplicate data in the dataset

```
missing_value = ((df_eda.isna().sum())/len(df_eda)*100).round(2).sort_values(ascending=False
missing_value.value_counts()
```

✓ 0.8s

```
0.00      37
100.00    17
0.01       8
15.07      3
86.57      1
78.77      1
72.98      1
53.69      1
48.73      1
5.92       1
4.51       1
0.08       1
0.07       1
0.03       1
dtype: int64
```

There is a lot of missing data in the dataset. There are **20 columns** in the dataset that **>70%** have no data in them and will be deleted. Then **deletion** or **imputation** will be performed on the columns that have missing value **<10%.**

Drop of columns that have missing value >70%

```
features = list(missing_value['feature'][missing_value['percentage'] > 70])
features


for feature in features:
  df_preprocess = df_preprocess.drop(columns=feature) #Penghapusan kolom yg memiliki missing value >70%
```

Columns Imputation

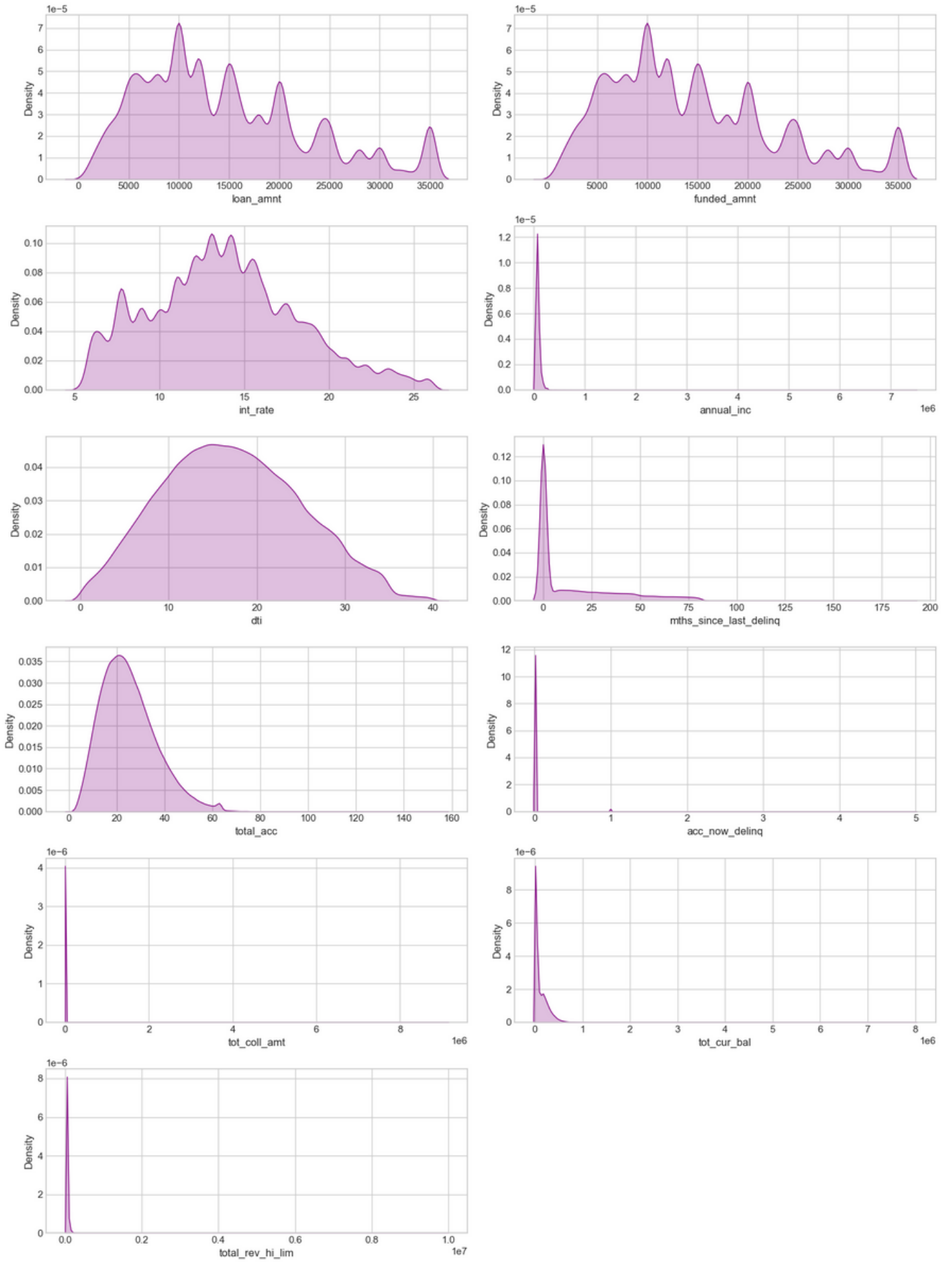
```
# Replace missing values with 0 in column: 'total_rev_hi_lim'
df_preprocess = df_preprocess.fillna({'total_rev_hi_lim': 0})
# Replace missing values with 0 in column: 'tot_cur_bal'
df_preprocess = df_preprocess.fillna({'tot_cur_bal': 0})
# Replace missing values with 0 in column: 'tot_coll_amt'
df_preprocess = df_preprocess.fillna({'tot_coll_amt': 0})
# Replace missing values with "Other" in column: 'next_pymnt_d'
df_preprocess = df_preprocess.fillna({'next_pymnt_d': "Other"})
# Replace missing values with 0 in column: 'mths_since_last_delinq'
df_preprocess = df_preprocess.fillna({'mths_since_last_delinq': 0})
# Drop rows with missing data in column: 'annual_inc'
df_preprocess = df_preprocess.dropna(subset=['annual_inc'])
df_preprocess = df_preprocess.dropna(subset=['emp_length'])
df_preprocess = df_preprocess.dropna(subset=['acc_now_delinq'])
df_preprocess = df_preprocess.dropna(subset=['pub_rec'])
df_preprocess = df_preprocess.dropna(subset=['total_acc'])
```

Feature Extraction

Extracting relevant information from raw data or complex datasets for use in data analysis or modeling. The goal is to reduce the dimensionality of the data by selecting the most important and relevant features, thereby improving model performance and computational efficiency.

```
for item in df_preprocess['grade'].unique():
    if item in ['A', 'B', 'C']:
        # No change needed for these specific values
        pass
    else:
        df_preprocess.loc[df_preprocess['grade'] == item, 'grade'] = "Other"
```

Univariate Analysis



loan_amnt : Number of loans applied for by borrowers (Has a ***Right Skewed*** distribution)

funded_amnt : The total amount committed to that loan at that time. (Has a ***Right Skewed*** distribution)

int_rate: The interest rate charged on loans. (***Normal*** distribution)

annual_inc: Annual income self-reported by the borrower at application. (Has a ***Right Skewed*** distribution)

dti : Ratio calculated by using the borrower's total monthly debt payments to total debt obligations (***Normal*** distribution)

mths_since_last_delinq : Number of months since the borrower's last arrears.

total_acc : The total number of credit lines currently in the borrower's credit file..

tot_coll_amt : Total amount of receivables that have been collected..

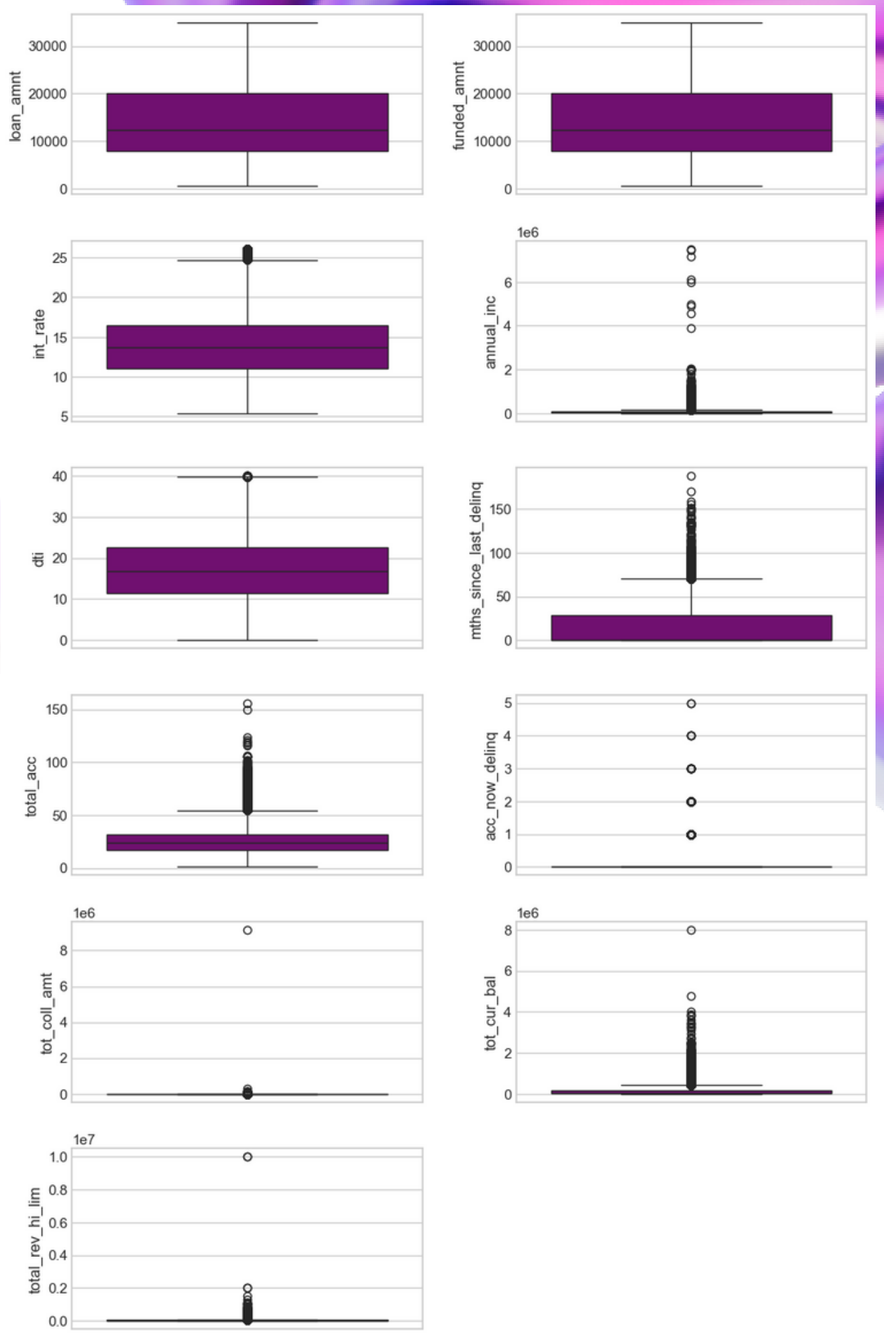
tot_cur_bal : Total current balance of all accounts..

total_rev_hi_lim : High total revolving credit/credit limit.

acc_now_delinq : Number of accounts in arrears. Many customers are not in arrears, but there are some who are..

Outliers Detection

Note : Previously, outliers were removed, but the performance of the model was not good enough, so outliers were removed.



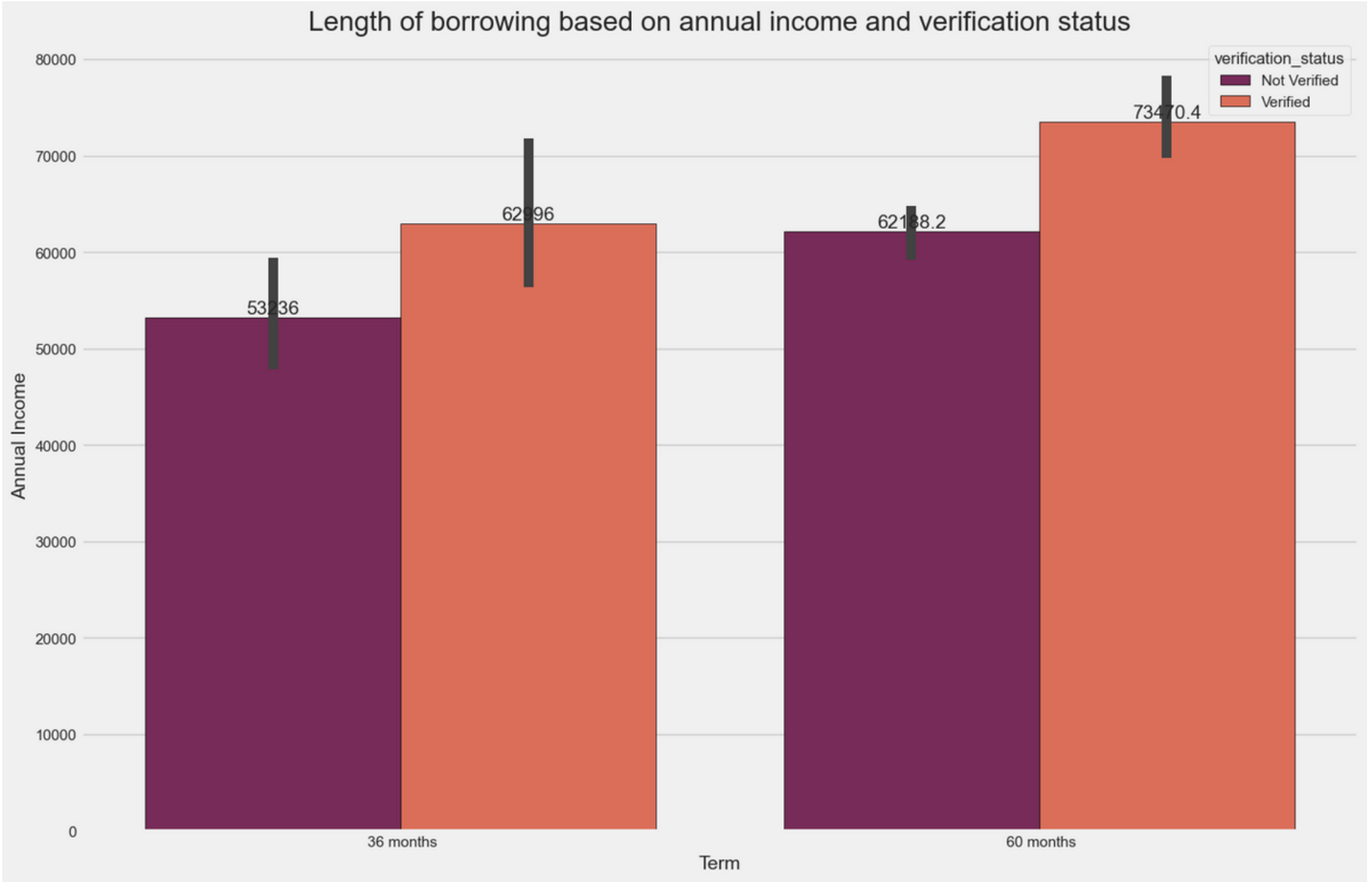
Multivariate Analysis

funded_amnt & **loan_amnt** have a very strong correlation. This means that customers are committed to the amount of loan they apply for. However, one of them will be deleted because it can cause redundancy.

annual_inc & **loan_amnt** also have a positive correlation, this means that annual income affects the amount of loan applied for

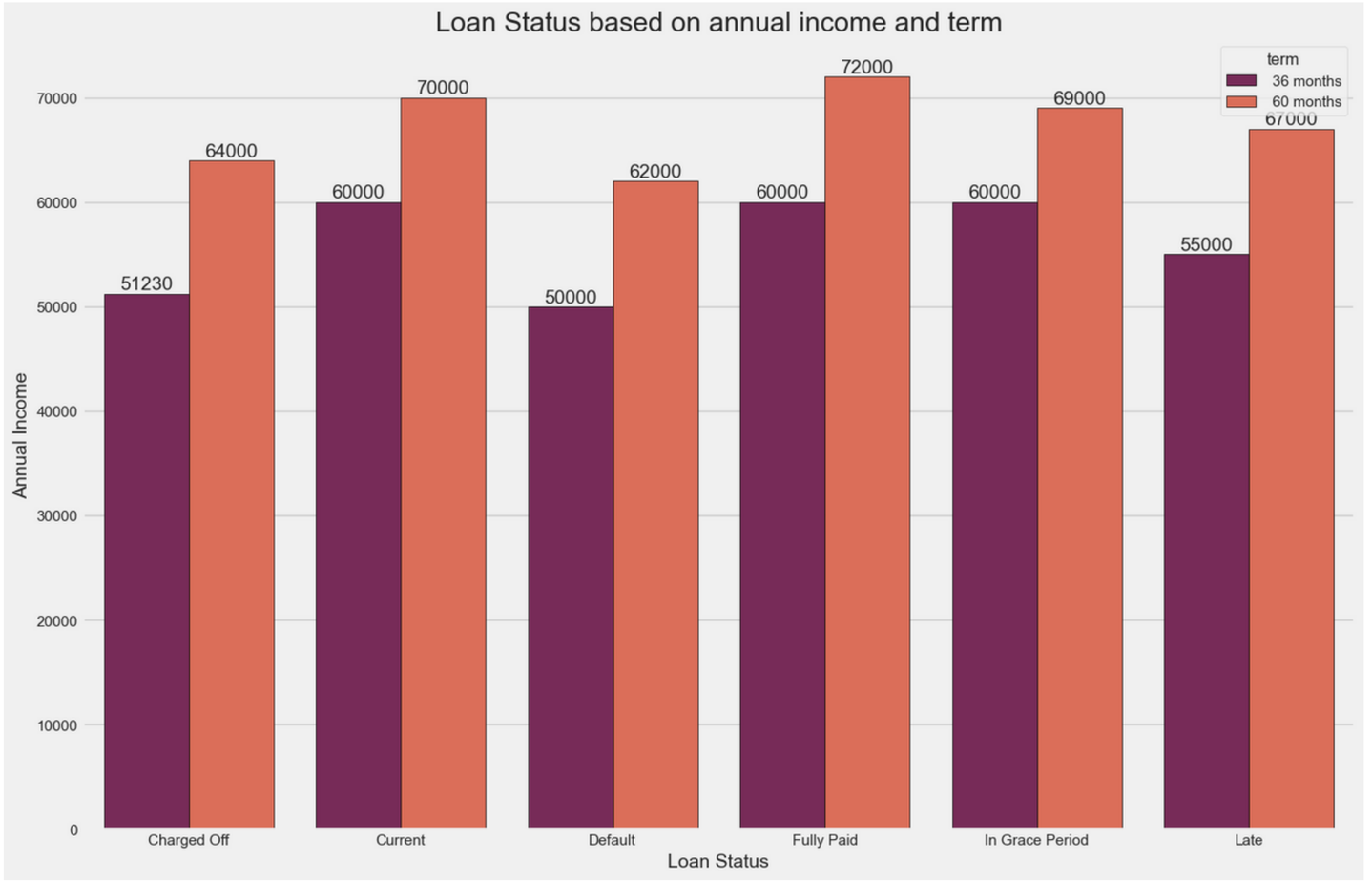
How long do customers borrow based on their annual income?

- In the element of lending, there is something called the risk level, the longer the loan is given, the higher the risk level.
- From the visualization above, customers who borrow for 60 months have an annual income above 70000, while customers who borrow for 36 months have an annual income below 70000.
- From here it can be concluded that verified customers who borrow for 60 months have a higher income, therefore customers who have an annual income below 70000 who want to borrow for 60 months are considered by the LC team because they can pose a high risk such as default.



Is it true that customers who want to borrow for 60 months but have an annual income below 70000 have a risk of default?

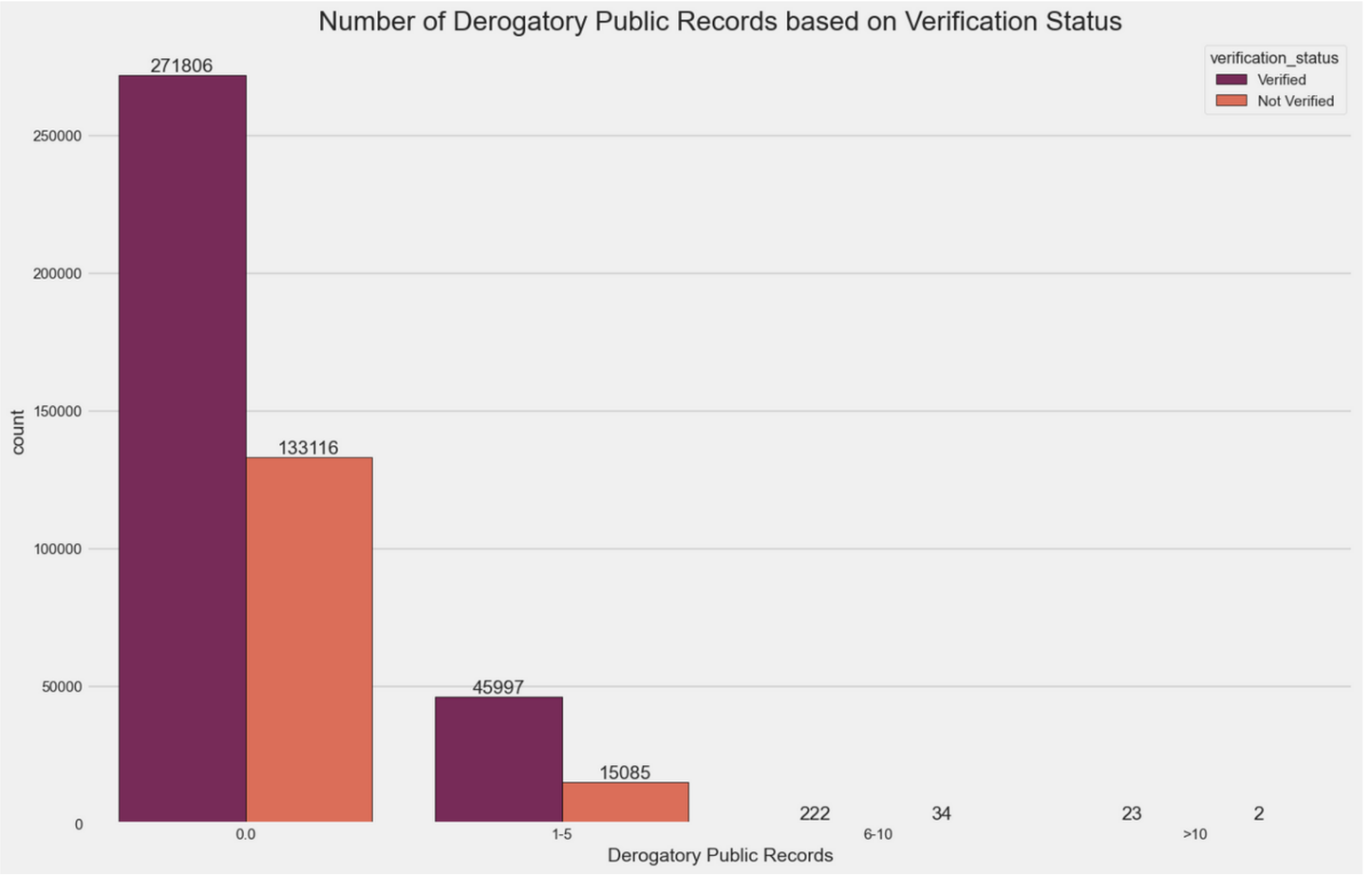
- Yes, that is true, it can be seen from the visualization above, customers who borrow for 60 months but **have an income below 70000 experience Default**.
- Therefore, for customers who have an income less than 70000, it is advisable to **take a short-term loan or for 36 months**.
- Not only default, customers who earn less than 70000 also **charged off, late, and in grace period**.



Do derogatory public records affect loan approval?

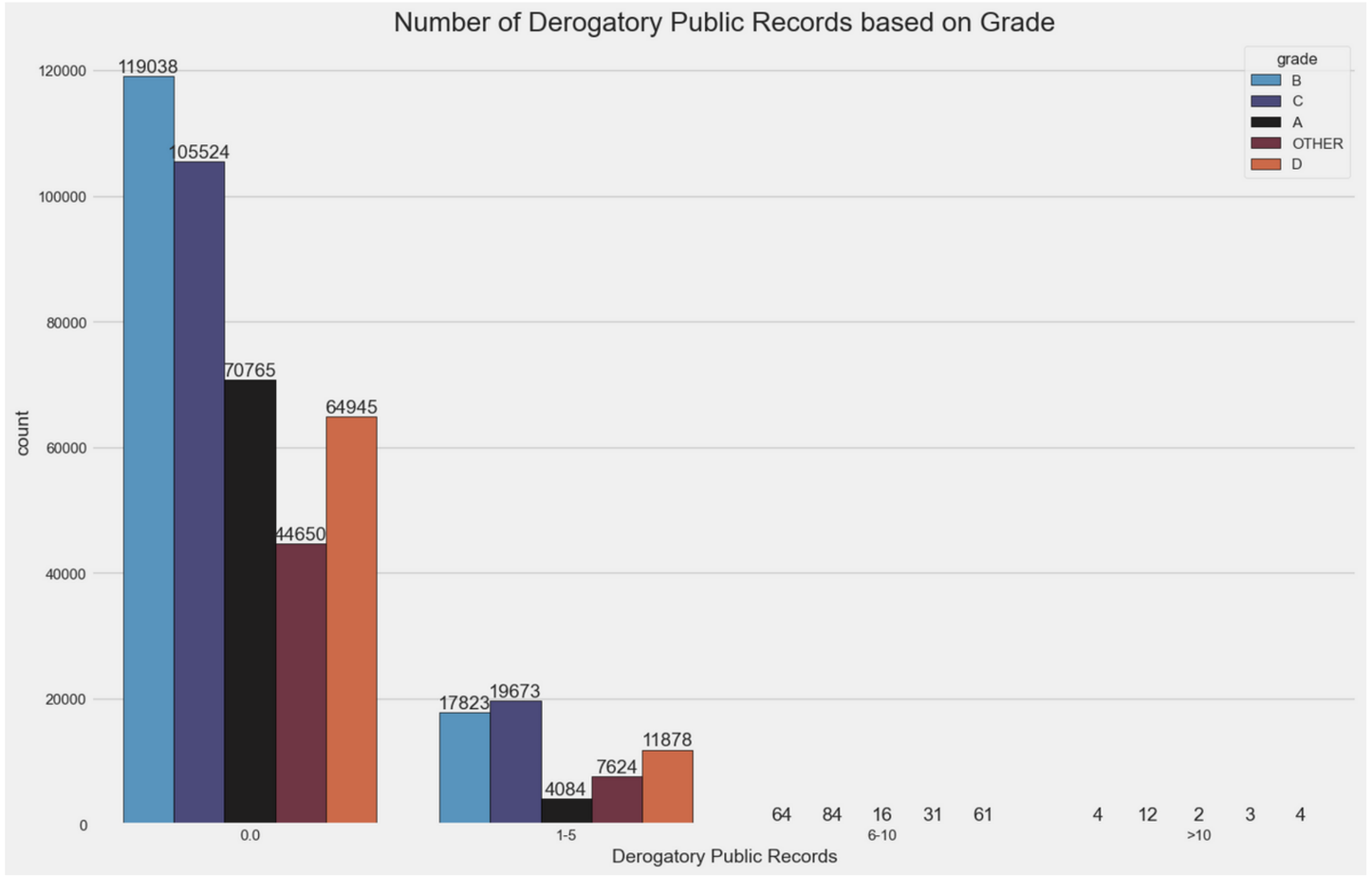
- Derogatory public records are negative records on your credit report that indicate that you have unpaid debts according to the agreement. Derogatory public records can affect your loan approval because they can lower your credit score and indicate a high risk for the lender.

- From the visualization above, it can be seen that customers who `do not have negative records` are more accepted than those who have bad records. But why are some customers who do not have bad records not accepted for their loans? Is it because of the customer grade?



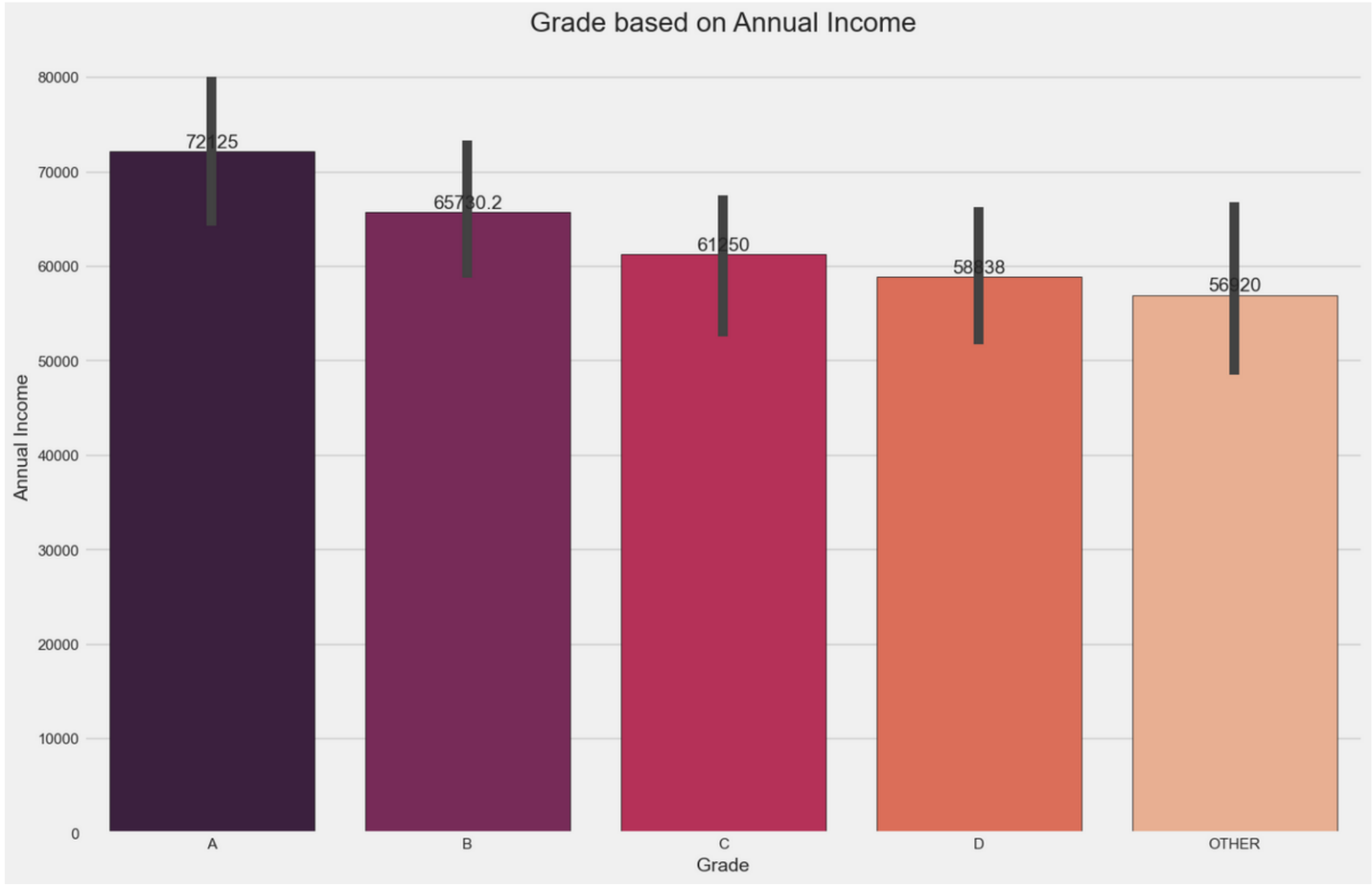
Customer Grade based on Derogatory Public Records

- It turns out that there are customers who have a low grade even though they do not have a bad record that can result in the rejection of the loan.



Do customers who have high income also have a good very grade (A)?

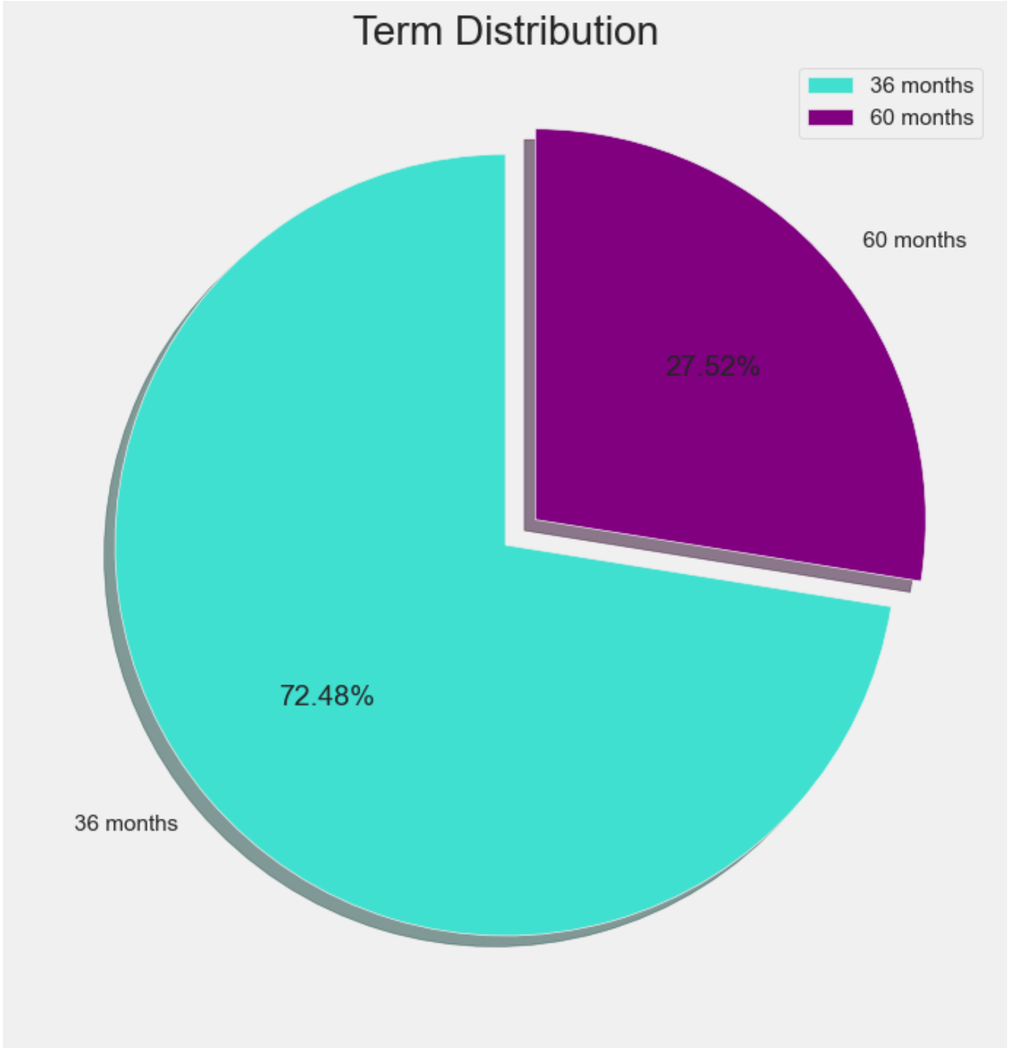
- Customers who have high income tend to have a very good grade (A), this shows that customers who have high income have a good history on credit score, loan history, and other factors.



Comparison of term distribution that customers choose

- It turns out that from the data I obtained, many customers took loans for 36 months rather than 60 months. The total value of the 36-month loans is USD **4 billion**, while the total value of the 60-month loans is only USD **2 billion**.

	term	loan_amnt
0	36 months	4126074275
1	60 months	2549857500



Rate The Amount of Loan Applied For From Annual Income

- From the data above, customers who have an annual income of thousands tend to borrow more than 10% of their annual income, but customers who have an income of millions, the loan rate from their annual income can be said to be very low, not up to 5%.

	annual_inc	verification_status	loan_amnt	loan_rate
0	1896.0	Not Verified	1800	94.94
1	2000.0	Not Verified	500	25.00
2	3000.0	Verified	1200	40.00
3	3300.0	Not Verified	500	15.15
4	3500.0	Not Verified	1600	45.71
...	...	...	...	...
36414	6000000.0	Verified	5000	0.08
36415	6100000.0	Verified	30000	0.49
36416	7141778.0	Verified	14825	0.21
36417	7446395.0	Verified	20000	0.27
36418	7500000.0	Verified	15000	0.20

36419 rows × 4 columns

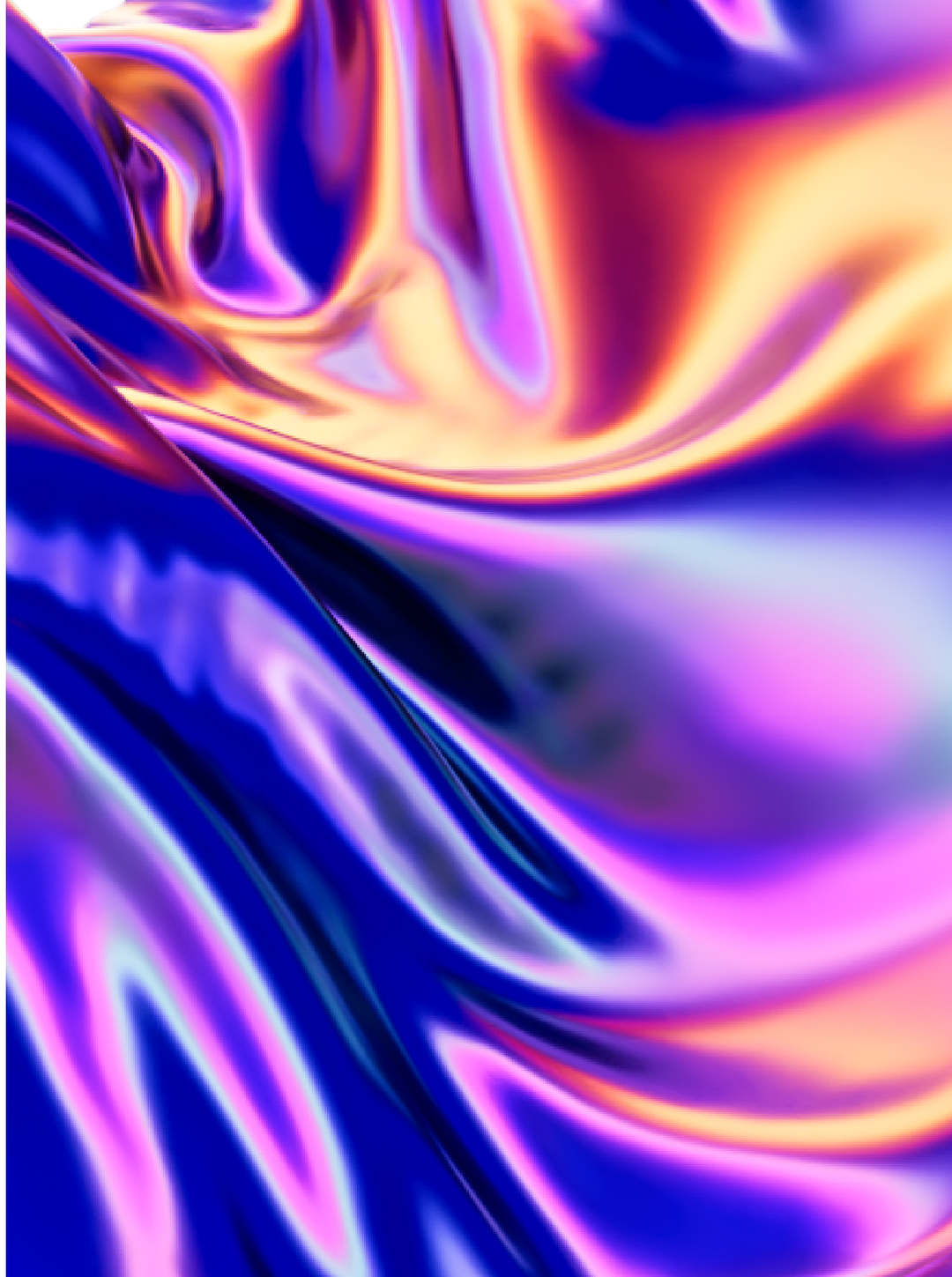
Feature Encoding

converts categorical data into numerical data that can be understood and processed by machine learning algorithms. This is important because most machine learning algorithms can only work with numerical values. There are several commonly used encoding techniques, such as One-Hot Encoding and Label Encoding.

- **grade** string -> integer (label encoding)
- **emp_length** string -> integer (label encoding)
- **pub_rec** string -> integer (label encoding)

The rest :

```
df_fe['verification_status'] = df_fe['verification_status'].astype('category').cat.codes
df_fe['home_ownership'] = df_fe['home_ownership'].astype('category').cat.codes
df_fe['loan_status'] = df_fe['loan_status'].astype('category').cat.codes
df_fe['purpose'] = df_fe['purpose'].astype('category').cat.codes
```



New Features

Create new features from existing features to improve the performance of machine learning models.

```
df_fe['monthly_payment'] = df_fe['loan_amnt'] / df_fe['term_month'] #Rasio Angsuran per Bulan
df_fe['total_cost'] = df_fe['loan_amnt'] * (1 + df_fe['int_rate']) #Total Biaya Pinjaman
df_fe['season'] = df_fe['term_month'] % 12 #Musim Pinjaman
df_fe['yearly_loan'] = df_fe['loan_amnt'] / (df_fe['term_month'] / 12)
#df_fe['risk_indicator'] = df_fe['monthly_payment'] / df_fe['annual_income'] / df_fe['credit_score']
```

monthly_payment : Calculate average monthly installments. This feature shows the borrower's ability to pay monthly installments.

total_cost : Calculate the total cost of the loan including interest.

season : Categorize loans by season (for example, Q1, Q2, Q3, Q4). This feature helps analyze loan patterns by season.

yearly_loan : Calculate the average loan per year. This feature helps analyze loan trends over time.

Feature Selection, Split Data, and Oversampling

```
df_model = df_fe.copy()
```

```
drop_columns2 = ['season','yearly_loan','acc_now_delinq','verification_status','grade','tot_coll_amt',
                 'emp_length','home_ownership','funded_amnt','initial_list_status','application_type']
```

Performing Oversampling using SMOTE, because there is an imbalance in the target

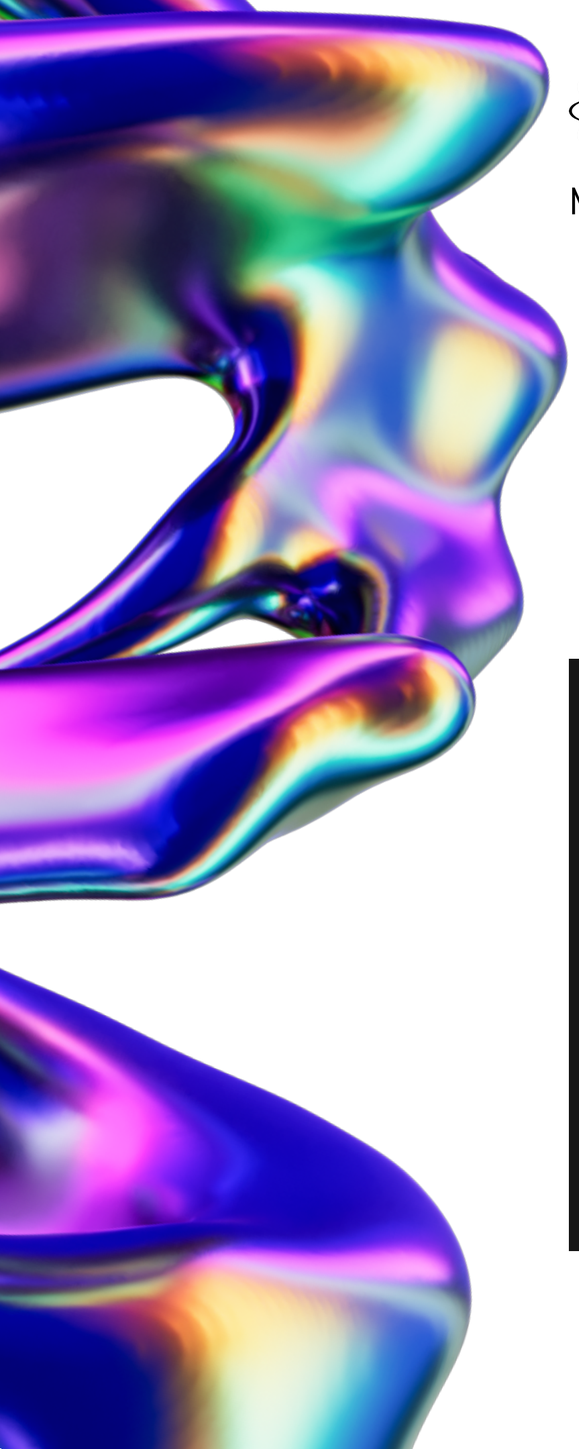
```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split

# Pisahkan fitur dan target
X = df_model.drop(drop_columns2, axis=1)
y = df_model['verification_status']

# Bagi data menjadi train dan test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=77)

# Terapkan SMOTE untuk oversampling pada data training
smote = SMOTE(random_state=77)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Proporsi data yang sudah seimbang
print(y_train_resampled.value_counts())
```



Modeling

Modeling uses 7 algorithms, namely :

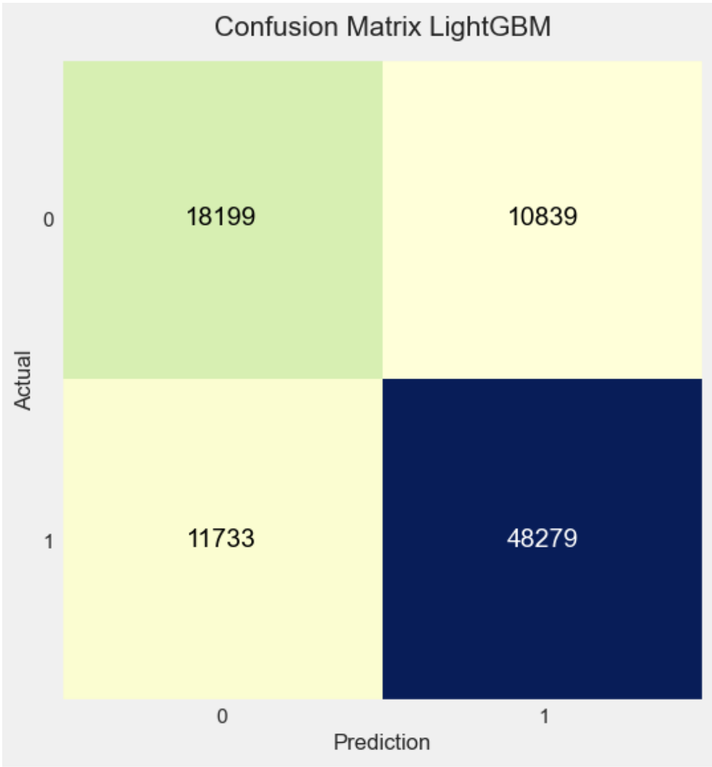
- Logistic Regression
- KNN
- Decision Tree
- XGBoost
- Random Forest
- Light GBM
- Gradient Boost

```
Accuracy (Test Set): 0.75
Accuracy (Train Set): 0.75
Precision (Test Set): 0.82
Precision (Train Set): 0.82
Recall (Test Set): 0.80
Recall (Train Set): 0.81
F1-Score (Test Set): 0.81
F1-Score (Train Set): 0.81
roc_auc (test-proba): 0.82
roc_auc (train-proba): 0.83
roc_auc (crossval train): 0.9999999999359055
roc_auc (crossval test): 0.794327513137959
```

Conclusion

- The model chosen is the LightGBM model which has not been parameter tuned. The main metrix used is **Precision** where we do not want high **False Positive** or reduce False Positive as small as possible.

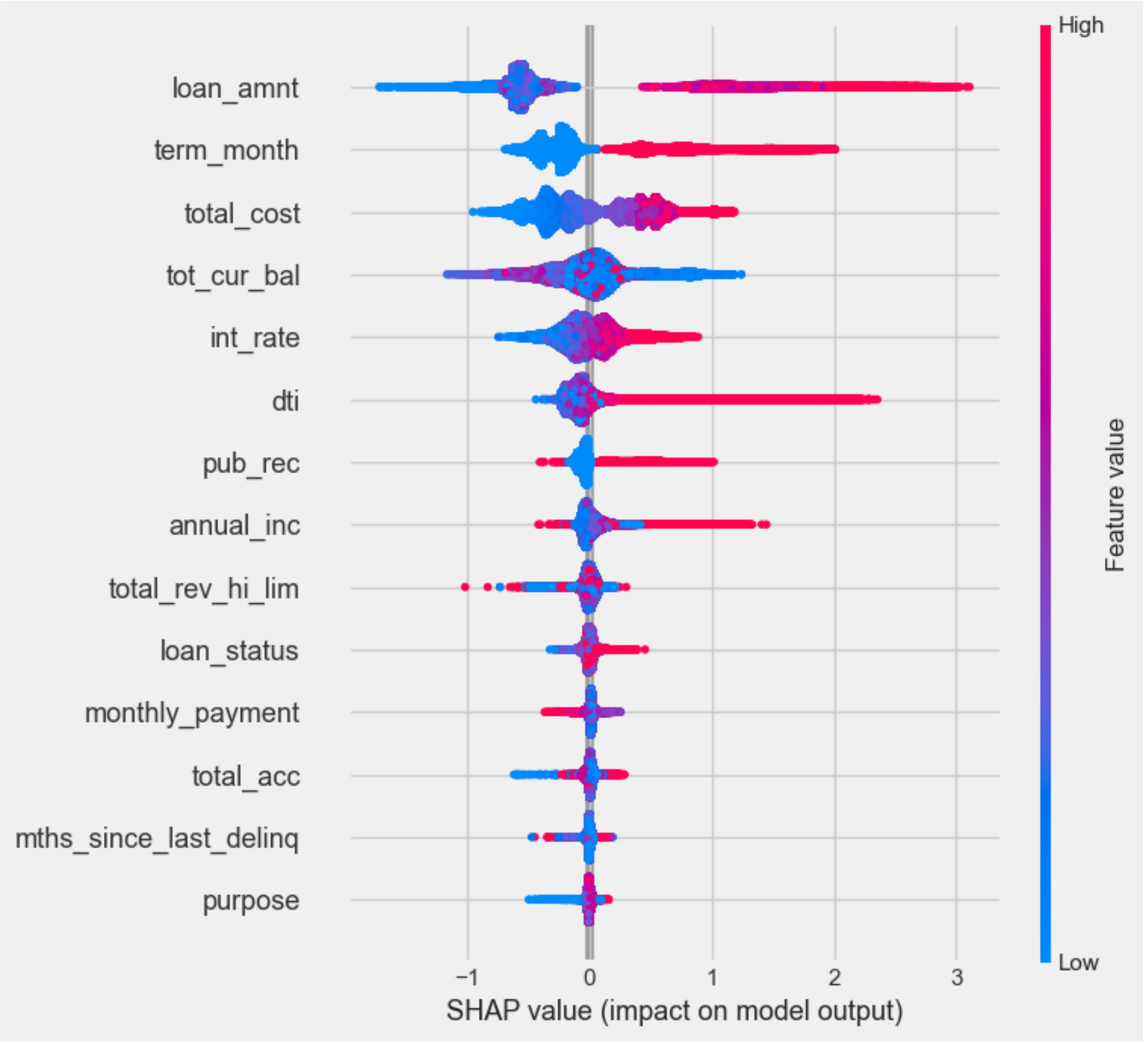
- In the context of a lending company, **False Positive** means approving a loan that should have been rejected, False Positive can cause financial loss to the Company.



The Best Fit Model

LightGBM Model

This model has a high precision score of **0.82** with a machine learning probability of **0.82**. The model is neither overfit nor underfit which can be referred to as the *best fit* model.



Business Recomendation

- Competitive Interest Rate Offer : Offer lower interest rates than competitors to attract customers with a good risk profile (low int_rate).
- Conduct regular monitoring and credit collection to minimize the risk of bad debts.
- Offer loan products with various ceilings to meet the needs of customers with different risk profiles.
- Targeting High Income Customers: Focus on marketing credit products to customers with high annual income levels.

THANK YOU

id/x partners