Final Task Data Analyst Id/x Partners - Prediction Model

Import Dataset & Library Python

In [30]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer, SimpleImputer
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score, roc_auc_score, balanced_accuracy_score,classi
from sklearn.metrics import RocCurveDisplay, roc_curve
```

In [3]:
```
df = pd.read_csv('D:/PBI/DA/loan_data_2007_2014.csv')
print("Dimensi Dataset : ",df.shape)
df.head()
```

```
C:\Users\Alvin\AppData\Local\Temp\ipykernel_10896\4223820869.py:1: DtypeWarning: Col
umns (20) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('D:/PBI/DA/loan_data_2007_2014.csv')
Dimensi Dataset :  (466285, 75)
```

Out[3]:

	Unnamed: 0	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	ir
0	0	1077501	1296599	5000	5000	4975.0	36 months	
1	1	1077430	1314167	2500	2500	2500.0	60 months	
2	2	1077175	1313524	2400	2400	2400.0	36 months	
3	3	1076863	1277178	10000	10000	10000.0	36 months	
4	4	1075358	1311748	3000	3000	3000.0	60 months	

5 rows × 75 columns

Preprocessing Data

In [4]:
```
print("Column Duplicate : ",df.duplicated().sum())
# Drop kolom NaN yang tidak dapat di analisis
df1 = df.dropna(axis=1, how='all')
df1 = df1.drop(['Unnamed: 0', 'url', 'desc', 'title', 'zip_code', 'id', 'member_id'
```

```
Column Duplicate :  0
```

In [5]:
```
var_cat = df1.select_dtypes(include = 'object').columns # Kolom kategorik

# cek unique values dari tiap kolom kategorik
for kolom in var_cat:
    unique = df1[kolom].unique()
    print(f"Kolom : {kolom} \n {unique}\n\n")
```

```
Kolom : term
 [' 36 months' ' 60 months']


Kolom : grade
 ['B' 'C' 'A' 'E' 'F' 'D' 'G']


Kolom : sub_grade
 ['B2' 'C4' 'C5' 'C1' 'B5' 'A4' 'E1' 'F2' 'C3' 'B1' 'D1' 'A1' 'B3' 'B4'
 'C2' 'D2' 'A3' 'A5' 'D5' 'A2' 'E4' 'D3' 'D4' 'F3' 'E3' 'F4' 'F1' 'E5'
 'G4' 'E2' 'G3' 'G2' 'G1' 'F5' 'G5']


Kolom : emp_title
 [nan 'Ryder' 'AIR RESOURCES BOARD' ... 'MecÃ¡nica'
 'Chief of Interpretation (Park Ranger)' 'Server Engineer Lead']


Kolom : emp_length
 ['10+ years' '< 1 year' '1 year' '3 years' '8 years' '9 years' '4 years'
 '5 years' '6 years' '2 years' '7 years' nan]


Kolom : home_ownership
 ['RENT' 'OWN' 'MORTGAGE' 'OTHER' 'NONE' 'ANY']


Kolom : verification_status
 ['Verified' 'Source Verified' 'Not Verified']


Kolom : issue_d
 ['Dec-11' 'Nov-11' 'Oct-11' 'Sep-11' 'Aug-11' 'Jul-11' 'Jun-11' 'May-11'
 'Apr-11' 'Mar-11' 'Feb-11' 'Jan-11' 'Dec-10' 'Nov-10' 'Oct-10' 'Sep-10'
 'Aug-10' 'Jul-10' 'Jun-10' 'May-10' 'Apr-10' 'Mar-10' 'Feb-10' 'Jan-10'
 'Dec-09' 'Nov-09' 'Oct-09' 'Sep-09' 'Aug-09' 'Jul-09' 'Jun-09' 'May-09'
 'Apr-09' 'Mar-09' 'Feb-09' 'Jan-09' 'Dec-08' 'Nov-08' 'Oct-08' 'Sep-08'
 'Aug-08' 'Jul-08' 'Jun-08' 'May-08' 'Apr-08' 'Mar-08' 'Feb-08' 'Jan-08'
 'Dec-07' 'Nov-07' 'Oct-07' 'Sep-07' 'Aug-07' 'Jul-07' 'Jun-07' 'Dec-13'
 'Nov-13' 'Oct-13' 'Sep-13' 'Aug-13' 'Jul-13' 'Jun-13' 'May-13' 'Apr-13'
 'Mar-13' 'Feb-13' 'Jan-13' 'Dec-12' 'Nov-12' 'Oct-12' 'Sep-12' 'Aug-12'
 'Jul-12' 'Jun-12' 'May-12' 'Apr-12' 'Mar-12' 'Feb-12' 'Jan-12' 'Dec-14'
 'Nov-14' 'Oct-14' 'Sep-14' 'Aug-14' 'Jul-14' 'Jun-14' 'May-14' 'Apr-14'
 'Mar-14' 'Feb-14' 'Jan-14']


Kolom : loan_status
 ['Fully Paid' 'Charged Off' 'Current' 'Default' 'Late (31-120 days)'
 'In Grace Period' 'Late (16-30 days)'
 'Does not meet the credit policy. Status:Fully Paid'
 'Does not meet the credit policy. Status:Charged Off']


Kolom : pymnt_plan
 ['n' 'y']
```

```
Kolom : purpose
 ['credit_card' 'car' 'small_business' 'other' 'wedding'
 'debt_consolidation' 'home_improvement' 'major_purchase' 'medical'
 'moving' 'vacation' 'house' 'renewable_energy' 'educational']


Kolom : addr_state
 ['AZ' 'GA' 'IL' 'CA' 'OR' 'NC' 'TX' 'VA' 'MO' 'CT' 'UT' 'FL' 'NY' 'PA'
 'MN' 'NJ' 'KY' 'OH' 'SC' 'RI' 'LA' 'MA' 'WA' 'WI' 'AL' 'CO' 'KS' 'NV'
 'AK' 'MD' 'WV' 'VT' 'MI' 'DC' 'SD' 'NH' 'AR' 'NM' 'MT' 'HI' 'WY' 'OK'
 'DE' 'MS' 'TN' 'IA' 'NE' 'ID' 'IN' 'ME']


Kolom : earliest_cr_line
 ['Jan-85' 'Apr-99' 'Nov-01' 'Feb-96' 'Jan-96' 'Nov-04' 'Jul-05' 'Jan-07'
 'Apr-04' 'Sep-04' 'Jan-98' 'Oct-89' 'Jul-03' 'May-91' 'Sep-07' 'Oct-98'
 'Aug-93' 'Oct-03' 'Jan-01' 'Nov-97' 'Feb-83' 'Jul-85' 'Apr-03' 'Jun-01'
 'Feb-02' 'Aug-84' 'Nov-06' 'Dec-87' 'Nov-81' 'Feb-97' 'Apr-05' 'Oct-07'
 'Dec-00' 'Apr-07' 'Dec-01' 'Jan-03' 'Mar-94' 'Sep-98' 'Jun-04' 'Nov-95'
 'Jul-99' 'Jun-95' 'Sep-92' 'Jan-02' 'Apr-92' 'Oct-06' 'May-00' 'Dec-98'
 'Dec-04' 'Oct-00' 'May-02' 'May-06' 'Jul-02' 'Jul-06' 'May-97' 'Oct-05'
 'Apr-95' 'Oct-02' 'Jan-00' 'Apr-00' 'Dec-94' 'Sep-05' 'Dec-84' 'Dec-99'
 'Nov-03' 'Jun-89' 'Jun-03' 'Oct-96' 'May-03' 'Jun-02' 'Jun-07' 'Dec-96'
 'Feb-84' 'Sep-02' 'Jan-86' 'May-98' 'Jan-97' 'Jun-05' 'Feb-90' 'Mar-04'
 'Jul-95' 'Aug-94' 'Jun-92' 'Mar-97' 'Apr-06' 'Apr-90' 'Aug-99' 'Sep-00'
 'Feb-01' 'Dec-88' 'Feb-99' 'Dec-91' 'Aug-00' 'Oct-04' 'Aug-04' 'Feb-05'
 'Nov-05' 'Nov-00' 'May-07' 'Jan-91' 'Jun-00' 'Aug-06' 'Dec-02' 'Jun-93'
 'Jun-06' 'Feb-04' 'Dec-90' 'Mar-00' 'Feb-95' 'Jul-01' 'Apr-02' 'Sep-06'
 'May-99' 'Aug-98' 'Dec-05' 'May-04' 'Oct-01' 'Jun-83' 'Mar-86' 'Apr-80'
 'Jul-04' 'Jul-08' 'May-96' 'Jan-04' 'Nov-02' 'Aug-02' 'Aug-01' 'Mar-91'
 'Sep-89' 'Sep-94' 'Sep-03' 'Sep-99' 'Aug-05' 'Dec-86' 'Nov-98' 'Feb-06'
 'May-94' 'Nov-07' 'Feb-93' 'Nov-91' 'May-05' 'Dec-73' 'May-01' 'Mar-90'
 'Mar-96' 'Oct-79' 'Jun-81' 'Mar-01' 'Apr-01' 'Jun-99' 'Nov-93' 'Jan-06'
 'Dec-97' 'Nov-94' 'Jul-97' 'Oct-91' 'Jun-94' 'Mar-06' 'Sep-96' 'Apr-91'
 'Jul-93' 'Jan-95' 'Sep-87' 'Mar-03' 'Oct-99' 'Jul-96' 'Dec-03' 'Aug-88'
 'Jan-92' 'Mar-98' 'Feb-07' 'Aug-82' 'Mar-95' 'Dec-92' 'Jul-98' 'Jul-89'
 'May-90' 'Jul-94' 'Sep-01' 'Mar-84' 'Aug-03' 'Nov-99' 'Mar-07' 'Mar-08'
 'Apr-94' 'Jan-05' 'Jul-86' 'Aug-90' 'May-92' 'Jul-00' 'Mar-88' 'May-83'
 'Apr-93' 'Jul-78' 'Feb-00' 'Dec-81' 'Mar-92' 'Jan-81' 'Sep-90' 'Jun-98'
 'May-93' 'Nov-96' 'Mar-02' 'Jan-88' 'Aug-97' 'Aug-87' 'Aug-08' 'Oct-94'
 'Oct-86' 'Feb-94' 'Jun-96' 'Feb-98' 'Nov-08' 'Apr-98' 'Jul-79' 'Jan-93'
 'May-87' 'Jul-71' 'Aug-07' 'Jun-97' 'Mar-80' 'Dec-06' 'Jul-07' 'Oct-95'
 'Jul-91' 'Jul-92' 'Dec-72' 'Dec-93' 'Jan-99' 'Feb-03' 'Apr-97' 'Dec-95'
 'Apr-96' 'Jul-90' 'Mar-70' 'Nov-84' 'Apr-84' 'Jul-84' 'Aug-95' 'Mar-99'
 'Sep-88' 'Mar-89' 'Mar-87' 'Oct-97' 'Dec-80' 'Jan-94' 'Sep-95' 'Mar-05'
 'Jan-89' 'Feb-92' 'Jan-90' 'Nov-90' 'Mar-69' 'Jun-75' 'Mar-85' 'Dec-07'
 'Oct-93' 'Dec-89' 'Sep-80' 'Jun-88' 'May-78' 'Apr-73' 'Aug-89' 'Oct-90'
 'Sep-91' 'Feb-82' 'Feb-87' 'Nov-85' 'Jan-84' 'Jul-88' 'May-08' 'Oct-85'
 'Mar-83' 'Aug-91' 'Sep-86' 'Jun-90' 'Feb-86' 'Sep-97' 'Jun-84' 'Sep-81'
 'Apr-86' 'Aug-79' 'Aug-80' 'Nov-92' 'Sep-93' 'Jun-87' 'Sep-82' 'Aug-92'
 'Aug-85' 'Jul-83' 'Jun-91' 'Dec-83' 'Jan-87' 'Nov-78' 'Oct-84' 'Aug-96'
 'Nov-89' 'Sep-76' 'Nov-86' 'Oct-87' 'Sep-08' 'May-77' 'May-86' 'Mar-81'
 'Jan-83' 'Nov-76' 'Sep-79' 'Oct-83' 'Sep-62' 'Jun-85' 'May-82' 'Feb-88'
 'Oct-92' 'Aug-83' 'Jun-73' 'Apr-85' 'Oct-88' 'Oct-81' 'Sep-68' 'Jul-74'
```

```
 'Nov-87' 'May-95' 'Feb-91' 'Nov-88' 'Mar-93' 'Jun-08' 'Jul-80' 'Dec-82'
 'Mar-75' 'Feb-80' 'Apr-88' 'Dec-79' 'Sep-85' 'Sep-71' 'Mar-78' 'Feb-08'
 'Aug-78' 'Nov-70' 'Jun-79' 'Jun-80' 'Apr-89' 'Sep-83' 'Feb-89' 'Nov-83'
 'Jun-86' 'Oct-82' 'Aug-86' 'Oct-80' 'May-88' 'Dec-85' 'Jan-82' 'Sep-77'
 'Dec-76' 'Apr-82' 'May-84' 'Apr-08' 'Feb-79' 'Jan-08' 'Sep-64' 'Jul-87'
 'Jan-78' 'May-89' 'Oct-77' 'Dec-75' 'Feb-85' 'Oct-08' 'Nov-82' 'May-75'
 'May-85' 'Feb-71' 'Jun-77' 'Apr-81' 'May-79' 'Jan-72' 'Sep-67' 'Apr-78'
 'Feb-65' 'Apr-83' 'Nov-75' 'Jun-67' 'Mar-74' 'Jul-72' 'Aug-67' 'Apr-71'
 'Sep-84' 'May-81' 'Dec-70' 'Oct-73' 'Jan-71' 'Dec-63' 'Apr-74' 'Jan-80'
 'Aug-69' 'Apr-75' 'Jul-77' 'Mar-77' 'Nov-69' 'Jan-76' 'Mar-82' 'Apr-87'
 'Dec-69' 'May-74' 'Aug-74' 'Jun-72' 'Mar-63' 'Nov-79' 'Aug-75' 'Sep-74'
 'Aug-81' 'May-73' 'Sep-73' 'Mar-73' 'Dec-77' 'Oct-76' 'Jan-74' 'Jan-70'
 'Aug-68' 'Feb-76' 'Jan-75' 'Oct-72' 'Dec-74' 'Feb-73' 'Nov-65' 'Mar-72'
 'Jun-82' 'Jun-74' 'May-65' 'Jun-71' 'Oct-70' 'Apr-76' 'Oct-71' 'Apr-77'
 'Sep-78' 'Oct-78' 'Oct-54' 'Feb-81' 'Jan-77' 'Aug-77' 'Dec-78' 'Aug-76'
 'Jun-68' 'Jun-78' 'Jun-69' 'May-80' 'Jan-79' 'Oct-65' 'Nov-74' 'Apr-66'
 'Jun-76' 'Feb-72' 'May-76' 'Mar-68' 'Mar-76' 'Jul-70' 'Mar-79' 'Jul-76'
 'Jul-82' 'Sep-65' 'Apr-67' 'Oct-63' 'Feb-70' 'Jul-73' 'Feb-78' 'Nov-71'
 'Aug-72' 'Jul-75' 'Sep-70' 'Jul-81' 'Oct-64' 'Sep-72' 'May-70' 'May-63'
 'Feb-69' 'Nov-80' 'Jul-67' 'Apr-70' 'Nov-77' 'Nov-66' 'May-71' 'Apr-79'
 'May-72' 'Feb-68' 'Jul-64' 'Nov-67' 'Apr-64' 'Feb-75' 'Jun-59' 'Sep-56'
 'Jun-66' 'Jan-46' 'Mar-66' 'Jan-63' 'Dec-50' 'Jul-69' 'Jan-68' 'Nov-73'
 'Jun-70' 'Feb-77' 'Feb-74' 'Jan-73' 'Feb-66' 'Dec-61' 'Aug-73' 'Aug-70'
 'Sep-69' 'Sep-75' 'Dec-68' 'Nov-54' 'Oct-69' 'Dec-65' 'Apr-72' 'Nov-72'
 'Sep-63' 'Apr-69' 'Nov-62' 'Oct-67' 'May-67' 'Nov-61' 'Feb-67' 'Nov-68'
 'Oct-75' 'Mar-71' 'Aug-71' 'Dec-66' 'Oct-68' 'Oct-74' 'Nov-63' 'Apr-68'
 'May-69' 'Nov-59' nan 'Oct-10' 'Dec-09' 'Nov-10' 'Jan-10' 'Dec-08'
 'Sep-09' 'Jul-10' 'Jan-09' 'May-09' 'May-10' 'Sep-10' 'Apr-09' 'Nov-09'
 'Apr-10' 'Mar-09' 'Feb-09' 'Aug-09' 'Oct-09' 'Jun-09' 'Jun-10' 'Jul-09'
 'Feb-10' 'Mar-10' 'Aug-10' 'Jan-61' 'Oct-62' 'Jun-64' 'Apr-62' 'Jan-60'
 'Sep-66' 'Mar-65' 'May-68' 'Jul-66' 'Dec-56' 'Dec-71' 'Apr-65' 'Jan-64'
 'Jul-65' 'Jan-65' 'Jan-62' 'Aug-64' 'Jan-69' 'Oct-66' 'Jun-62' 'Dec-67'
 'Oct-60' 'Jul-62' 'Dec-60' 'Jul-60' 'Jun-65' 'May-66' 'Oct-57' 'Aug-65'
 'May-60' 'Jan-67' 'Nov-53' 'Oct-59' 'Jun-63' 'Mar-67' 'Jul-63' 'Nov-60'
 'Mar-62' 'Mar-64' 'Jul-68' 'Feb-63' 'Jan-66' 'Dec-64' 'May-64' 'Oct-61'
 'Aug-60' 'Mar-60' 'Apr-55' 'Sep-59' 'Aug-66' 'Jul-55' 'Aug-61' 'Nov-64'
 'May-53' 'Nov-50' 'Nov-55' 'Mar-61' 'Feb-57' 'Aug-63' 'Feb-60' 'Sep-60'
 'Feb-64' 'Dec-51' 'Nov-58' 'Aug-62' 'Oct-58' 'May-62' 'Mar-11' 'Jun-11'
 'Dec-10' 'Oct-11' 'Sep-11' 'Jul-11' 'Apr-11' 'May-11' 'Jan-11' 'Feb-11'
 'Aug-11' 'Apr-63' 'Nov-11' 'Dec-58' 'Jan-58' 'Jul-59' 'Jan-55' 'May-59'
 'Aug-46' 'Apr-61' 'Jun-58' 'Jul-58' 'Jun-60' 'Jan-51' 'Dec-62' 'Mar-59'
 'Aug-58' 'Jan-59' 'Feb-61' 'Sep-57' 'Jun-49' 'Jun-61' 'Jan-56' 'Jan-54'
 'Jan-57' 'Jan-44' 'Oct-50' 'Jan-48' 'Jul-61' 'May-58' 'Jan-53' 'May-50'
 'Nov-56']


Kolom : initial_list_status
 ['f' 'w']


Kolom : last_pymnt_d
 ['Jan-15' 'Apr-13' 'Jun-14' 'Jan-16' 'Apr-12' 'Nov-12' 'Jun-13' 'Sep-13'
 'Jul-12' 'Oct-13' 'May-13' 'Feb-15' 'Aug-15' 'Oct-12' 'Sep-12' nan
 'Dec-12' 'Dec-14' 'Aug-13' 'Nov-13' 'Jan-14' 'Apr-14' 'Aug-14' 'Oct-14'
 'Aug-12' 'Jul-14' 'Jul-13' 'Apr-15' 'Feb-14' 'Sep-14' 'Jun-12' 'Feb-13'
 'Mar-13' 'May-14' 'Mar-15' 'Jan-13' 'Dec-13' 'Feb-12' 'Mar-14' 'Sep-15'
```

```
 'Nov-15' 'Dec-15' 'Jan-12' 'Oct-15' 'Nov-14' 'Mar-12' 'May-12' 'Jun-15'
 'May-15' 'Jul-15' 'Dec-11' 'Nov-11' 'Oct-11' 'Sep-11' 'Aug-11' 'Jul-11'
 'Jun-11' 'May-11' 'Apr-11' 'Mar-11' 'Feb-11' 'Jan-11' 'Dec-10' 'Nov-10'
 'Oct-10' 'Sep-10' 'Aug-10' 'Jul-10' 'Jun-10' 'May-10' 'Apr-10' 'Mar-10'
 'Feb-10' 'Jan-10' 'Dec-09' 'Nov-09' 'Oct-09' 'Sep-09' 'Aug-09' 'Jul-09'
 'Jun-09' 'May-09' 'Apr-09' 'Mar-09' 'Feb-09' 'Jan-09' 'Dec-08' 'Oct-08'
 'Aug-08' 'Jul-08' 'Sep-08' 'Jun-08' 'May-08' 'Nov-08' 'Apr-08' 'Mar-08'
 'Feb-08' 'Jan-08' 'Dec-07']


Kolom : next_pymnt_d
 [nan 'Feb-16' 'Jan-16' 'Sep-13' 'Feb-14' 'May-14' 'Jun-13' 'Mar-12'
 'Apr-12' 'May-13' 'Aug-12' 'Aug-13' 'Jun-12' 'Nov-13' 'Feb-12' 'Oct-11'
 'Jan-13' 'Jan-14' 'Jul-13' 'Jul-15' 'Jan-12' 'Dec-12' 'Jun-11' 'Feb-13'
 'Nov-11' 'Nov-12' 'Dec-11' 'Aug-11' 'Sep-11' 'Apr-11' 'Mar-14' 'Apr-13'
 'Mar-11' 'Jul-12' 'Aug-14' 'Oct-13' 'Sep-12' 'May-12' 'Apr-15' 'Jul-11'
 'Dec-15' 'Dec-13' 'Jan-11' 'Oct-12' 'Nov-14' 'Mar-13' 'Aug-15' 'Feb-15'
 'May-15' 'Jul-14' 'Nov-15' 'Sep-14' 'Oct-15' 'May-11' 'Feb-11' 'Dec-14'
 'Jun-15' 'Apr-14' 'Jan-15' 'Sep-15' 'Jun-14' 'Nov-10' 'Oct-10' 'Dec-10'
 'Mar-15' 'Oct-14' 'Jul-10' 'Sep-10' 'May-10' 'Aug-10' 'Mar-10' 'Jun-10'
 'Apr-10' 'Feb-10' 'Dec-09' 'Nov-09' 'Oct-09' 'Jan-10' 'Sep-09' 'Jun-09'
 'Aug-09' 'Jul-09' 'May-09' 'Apr-09' 'Jan-09' 'Oct-08' 'Feb-09' 'Nov-08'
 'Sep-08' 'Mar-09' 'Dec-08' 'Aug-08' 'Jun-08' 'Jul-08' 'Apr-08' 'May-08'
 'Feb-08' 'Jan-08' 'Mar-08' 'Dec-07' 'Mar-16']


Kolom : last_credit_pull_d
 ['Jan-16' 'Sep-13' 'Jan-15' 'Sep-15' 'Dec-14' 'Aug-12' 'Mar-13' 'Dec-15'
 'Aug-13' 'Nov-12' 'Mar-14' 'Apr-15' 'May-14' 'Jul-15' 'Jul-12' 'Sep-12'
 'May-13' 'Oct-15' 'Jun-12' 'Mar-15' 'Dec-12' 'Jul-14' 'Sep-14' 'Feb-14'
 'Jun-15' 'Oct-13' 'Apr-14' 'Oct-14' 'Feb-13' 'Nov-15' 'Oct-12' 'Nov-13'
 'Nov-14' 'Feb-12' 'Apr-12' 'Aug-15' 'Jun-14' 'Jan-12' 'Aug-14' 'Jun-13'
 'Dec-13' 'May-12' 'Jan-14' 'Jul-13' 'Apr-13' 'May-15' 'Feb-15' 'Mar-12'
 'Nov-11' 'Dec-11' 'Jan-13' 'Oct-11' 'Sep-11' 'Aug-11' 'Jul-11' 'Jun-11'
 'May-11' 'Apr-11' 'Mar-11' 'Feb-11' 'Jan-11' 'Dec-10' 'Nov-10' 'Oct-10'
 nan 'Sep-10' 'Aug-10' 'Jul-10' 'Jun-10' 'May-10' 'Apr-10' 'Feb-10'
 'Mar-10' 'Aug-07' 'Jan-10' 'Dec-09' 'Nov-09' 'Oct-09' 'Sep-09' 'Jul-09'
 'Aug-09' 'Jun-09' 'May-09' 'Apr-09' 'Mar-09' 'Feb-09' 'Jan-09' 'Dec-08'
 'Jun-08' 'Sep-08' 'May-08' 'Aug-08' 'Mar-08' 'Oct-08' 'Feb-08' 'Jan-08'
 'Dec-07' 'Jul-08' 'Oct-07' 'Sep-07' 'Jun-07' 'May-07' 'Jul-07' 'Nov-07']


Kolom : application_type
 ['INDIVIDUAL']
```

In [6]:
```python
df1 = df1.drop(['application_type'], axis=1) # Drop kolom application_type karena h
df2 = df1.copy()

# Format tanggal
date_format = ['issue_d', 'earliest_cr_line', 'last_pymnt_d', 'next_pymnt_d', 'last
df2[date_format] = df2[date_format].apply(lambda x: '01-' + x)
```

In [7]:
```python
# Convert to datetime
for col in date_format:
```

```
    df2[col] = pd.to_datetime(df2[col], format='%d-%b-%y')

df2[date_format].head()
```

Out[7]:

	issue_d	earliest_cr_line	last_pymnt_d	next_pymnt_d	last_credit_pull_d
0	2011-12-01	1985-01-01	2015-01-01	NaT	2016-01-01
1	2011-12-01	1999-04-01	2013-04-01	NaT	2013-09-01
2	2011-12-01	2001-11-01	2014-06-01	NaT	2016-01-01
3	2011-12-01	1996-02-01	2015-01-01	NaT	2015-01-01
4	2011-12-01	1996-01-01	2016-01-01	2016-02-01	2016-01-01

In [8]:
```
# Jumlah nilai unik untuk setiap kolom object
object_columns = df2.select_dtypes(include=['object']).columns
df2[object_columns].nunique()
```

Out[8]:
```
term                      2
grade                     7
sub_grade                35
emp_title            205475
emp_length               11
home_ownership            6
verification_status       3
loan_status               9
pymnt_plan                2
purpose                  14
addr_state               50
initial_list_status       2
dtype: int64
```

In [9]:
```
# Drop kolom emp_title karena memiliki terlalu banyak unique value
df2 = df2.drop(['emp_title'], axis = 1)

object_columns = df2.select_dtypes(include=['object']).columns
unik_cat = {}

for col in object_columns:
    unik_cat[col] = df2[col].value_counts()

for col, counts in unik_cat.items():
    print(f"Jumlah Unique Value Pada Kolom '{col}':")
    print(counts)
    print()
```

```
Jumlah Unique Value Pada Kolom 'term':
term
36 months    337953
60 months    128332
Name: count, dtype: int64

Jumlah Unique Value Pada Kolom 'grade':
grade
B    136929
C    125293
D     76888
A     74867
E     35757
F     13229
G      3322
Name: count, dtype: int64

Jumlah Unique Value Pada Kolom 'sub_grade':
sub_grade
B3    31686
B4    30505
C1    26953
C2    26740
B2    26610
C3    25317
B5    25252
C4    24105
B1    22876
C5    22178
A5    21757
D1    19261
A4    19045
D2    17046
D3    14916
D4    14099
A3    12568
D5    11566
A2    10956
A1    10541
E1     9033
E2     8669
E3     6976
E4     5992
E5     5087
F1     3940
F2     3001
F3     2708
F4     2067
F5     1513
G1     1109
G2      823
G3      583
G4      422
G5      385
Name: count, dtype: int64
```

```
Jumlah Unique Value Pada Kolom 'emp_length':
emp_length
10+ years    150049
2 years       41373
3 years       36596
< 1 year      36265
5 years       30774
1 year        29622
4 years       28023
7 years       26180
6 years       26112
8 years       22395
9 years       17888
Name: count, dtype: int64

Jumlah Unique Value Pada Kolom 'home_ownership':
home_ownership
MORTGAGE    235875
RENT        188473
OWN          41704
OTHER          182
NONE            50
ANY              1
Name: count, dtype: int64

Jumlah Unique Value Pada Kolom 'verification_status':
verification_status
Verified           168055
Source Verified    149993
Not Verified       148237
Name: count, dtype: int64

Jumlah Unique Value Pada Kolom 'loan_status':
loan_status
Current                                                224226
Fully Paid                                             184739
Charged Off                                             42475
Late (31-120 days)                                       6900
In Grace Period                                          3146
Does not meet the credit policy. Status:Fully Paid       1988
Late (16-30 days)                                        1218
Default                                                   832
Does not meet the credit policy. Status:Charged Off       761
Name: count, dtype: int64

Jumlah Unique Value Pada Kolom 'pymnt_plan':
pymnt_plan
n    466276
y         9
Name: count, dtype: int64

Jumlah Unique Value Pada Kolom 'purpose':
purpose
debt_consolidation    274195
credit_card           104157
home_improvement       26537
```

```
other                  23690
major_purchase          9828
small_business          7013
car                     5397
medical                 4602
moving                  2994
vacation                2487
wedding                 2343
house                   2269
educational              422
renewable_energy         351
Name: count, dtype: int64

Jumlah Unique Value Pada Kolom 'addr_state':
addr_state
CA    71450
NY    40242
TX    36439
FL    31637
IL    18612
NJ    18061
PA    16424
OH    15237
GA    14975
VA    14222
NC    12682
MI    11549
MA    11072
MD    10974
AZ    10712
WA    10517
CO     9739
MN     8158
MO     7508
CT     7204
IN     6525
NV     6519
TN     5984
OR     5949
WI     5911
AL     5853
SC     5583
LA     5489
KY     4438
KS     4190
OK     4117
AR     3488
UT     3428
NM     2588
HI     2487
WV     2412
NH     2232
RI     2050
DC     1426
MT     1396
DE     1272
```

```
AK     1251
MS     1226
WY     1128
SD      980
VT      905
IA       14
NE       14
ID       12
ME        4
Name: count, dtype: int64

Jumlah Unique Value Pada Kolom 'initial_list_status':
initial_list_status
f    303005
w    163280
Name: count, dtype: int64
```

In [10]:
```
# Memiliki nilai yang sama
df_p1 = df2.drop(['policy_code'], axis =1)
```

In [11]:
```
# Mengubah nilai loan_status menjadi 0 dan 1
df_p1['loan_status'] = df_p1['loan_status'].replace(['Current', 'Fully Paid', 'In G
df_p1['loan_status'].value_counts()
```

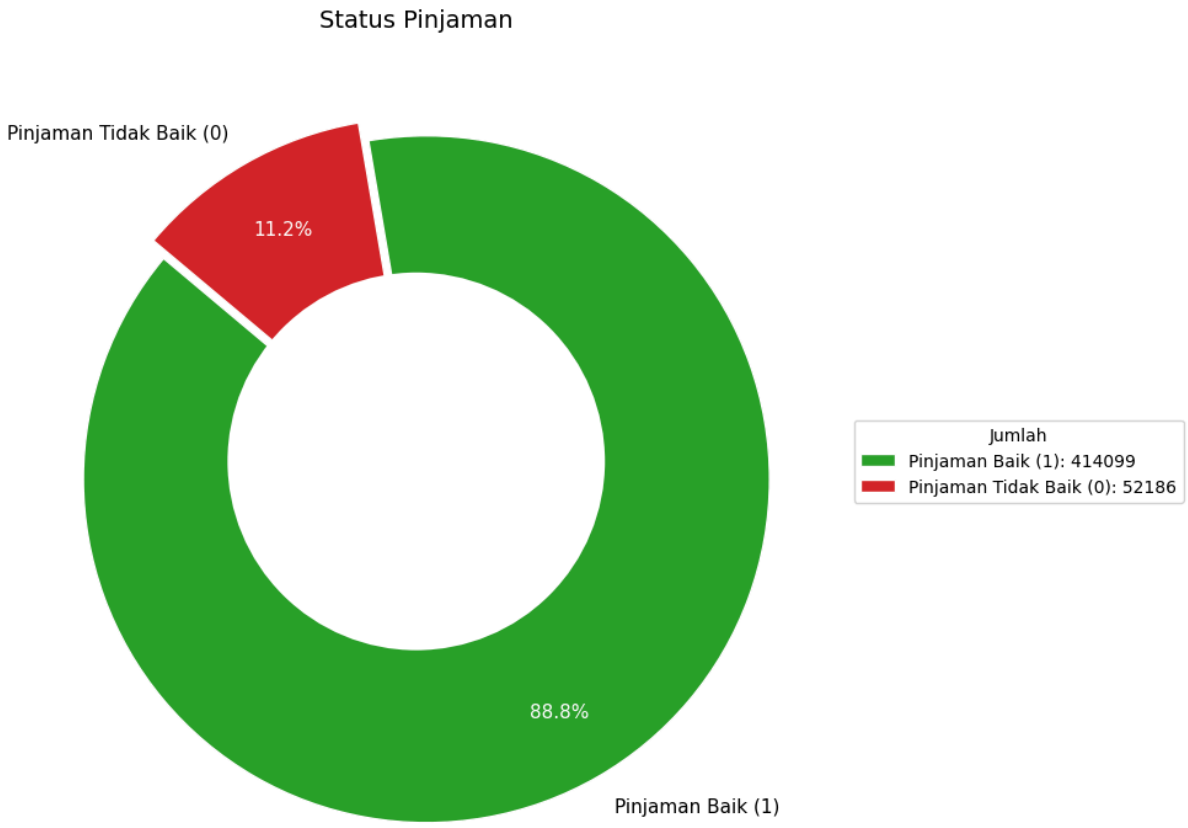
```
C:\Users\Alvin\AppData\Local\Temp\ipykernel_10896\2808815859.py:2: FutureWarning: Do
wncasting behavior in `replace` is deprecated and will be removed in a future versio
n. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. T
o opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting',
True)`
  df_p1['loan_status'] = df_p1['loan_status'].replace(['Current', 'Fully Paid', 'In
Grace Period', 'Does not meet the credit policy. Status:Fully Paid'], 1).replace(['C
harged Off', 'Late (31-120 days)', 'Late (16-30 days)', 'Default', 'Does not meet th
e credit policy. Status:Charged Off'], 0)
```

Out[11]:
```
loan_status
1    414099
0     52186
Name: count, dtype: int64
```

Exploratory Data Analysis (EDA)

In [12]:
```
counts = df_p1['loan_status'].value_counts()

# Fungsi heuristik untuk menentukan apakah sebuah status termasuk 'baik'
def is_good_status(lbl):
    s = str(lbl).strip().lower()
    if s in ('1', '1.0'):
        return True
    good_keywords = ('fully paid', 'paid', 'current', 'completed', 'active')
    if any(k in s for k in good_keywords):
        return True
    return False

# Agregasi ke dua kategori: baik vs tidak baik
```

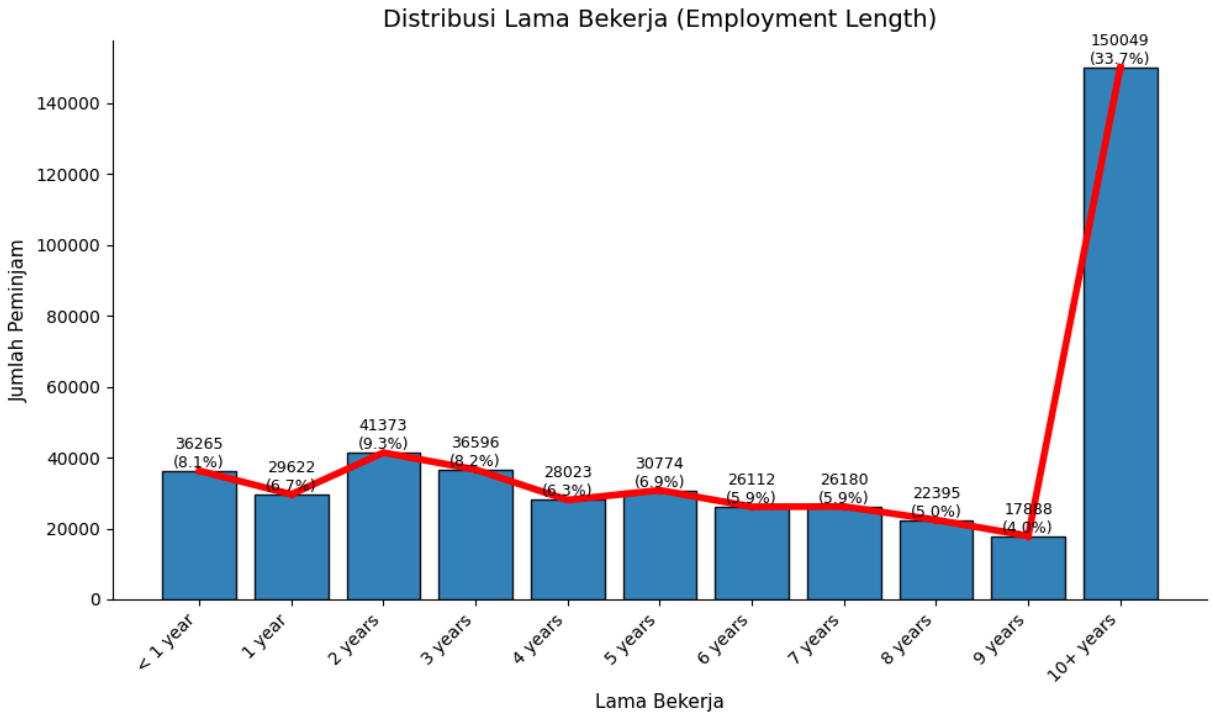
```
good_total = counts[[i for i in counts.index if is_good_status(i)]].sum() if any(is
bad_total = counts.sum() - good_total

sizes = [good_total, bad_total]
labels = ['Pinjaman Baik (1)', 'Pinjaman Tidak Baik (0)']
colors = ['#2ca02c', '#d62728']

fig, ax = plt.subplots(figsize=(10,10))
explode = (0.06, 0.0)
wedges, texts, autotexts = ax.pie(
    sizes,
    labels=labels,
    autopct='%1.1f%%',
    startangle=140,
    pctdistance=0.78,
    explode=explode,
    colors=colors,
    wedgeprops={'linewidth': 1, 'edgecolor': 'white'}
)

centre_circle = plt.Circle((0,0), 0.55, fc='white')
fig.gca().add_artist(centre_circle)

# Styling teks autopct dan judul
for t in autotexts:
    t.set_color('white')
    t.set_fontsize(11)
for t in texts:
    t.set_fontsize(11)

ax.set_title('Status Pinjaman', fontsize=14, pad=3)
ax.legend(wedges, [f'{labels[0]}: {good_total}', f'{labels[1]}: {bad_total}'],
          title='Jumlah', loc='center left', bbox_to_anchor=(1, 0.5), fontsize=10)

# ax.axis('equal')  # pastikan pie berbentuk lingkaran
plt.tight_layout()
plt.show()
```



In [13]:

```
counts = df_p1['emp_length'].value_counts()
order = [
    '< 1 year', '1 year', '2 years', '3 years', '4 years',
    '5 years', '6 years', '7 years', '8 years', '9 years',
    '10+ years'
]

counts = counts.reindex(order).dropna()

fig, ax = plt.subplots(figsize=(10, 6))

# Bar chart
bars = ax.bar(
    counts.index,
    counts.values,
    edgecolor='black',
    alpha=0.9
)

# Line chart
ax.plot(
    counts.index,
    counts.values,
    marker='.',
    color='red',
    linewidth=4
)
```

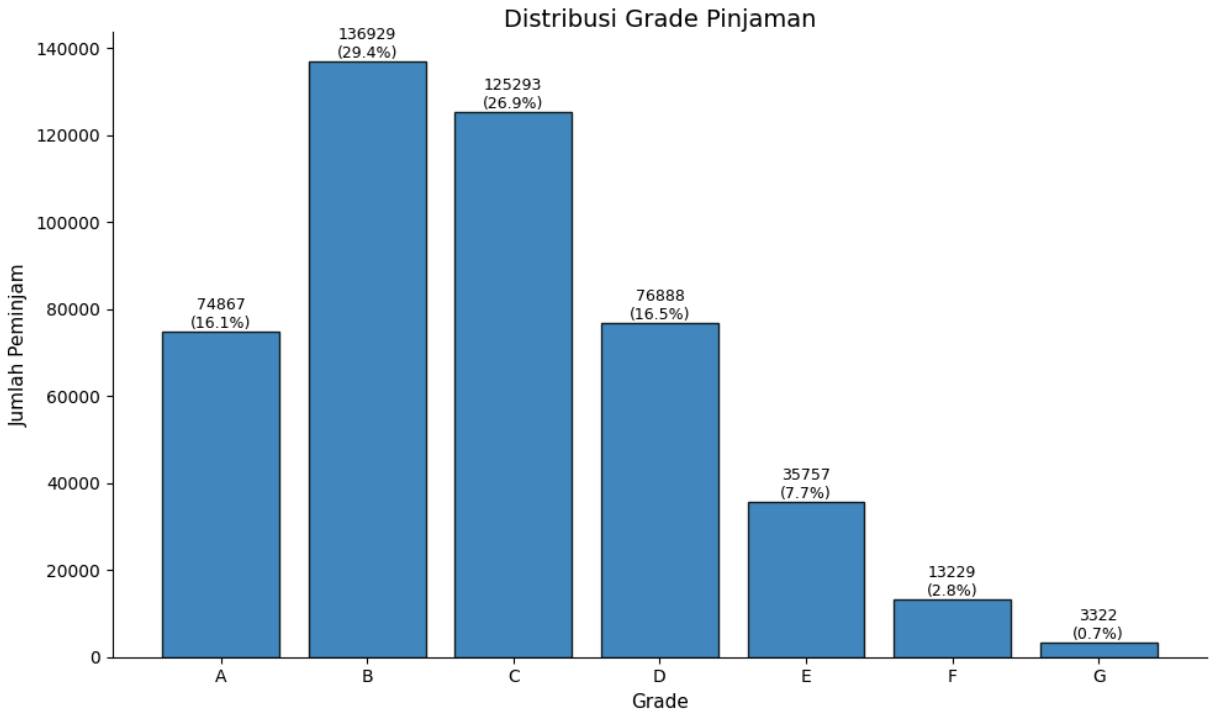
```
ax.set_title('Distribusi Lama Bekerja (Employment Length)', fontsize=14, pad=8)
ax.set_xlabel('Lama Bekerja', fontsize=11)
ax.set_ylabel('Jumlah Peminjam', fontsize=11)
plt.xticks(rotation=45, ha='right')

total = counts.sum()
for i, value in enumerate(counts.values):
    ax.text(
        i,
        value,
        f'{int(value)}\n({value/total:.1%})',
        ha='center',
        va='bottom',
        fontsize=9
    )

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout()
plt.show()
```



In [14]:

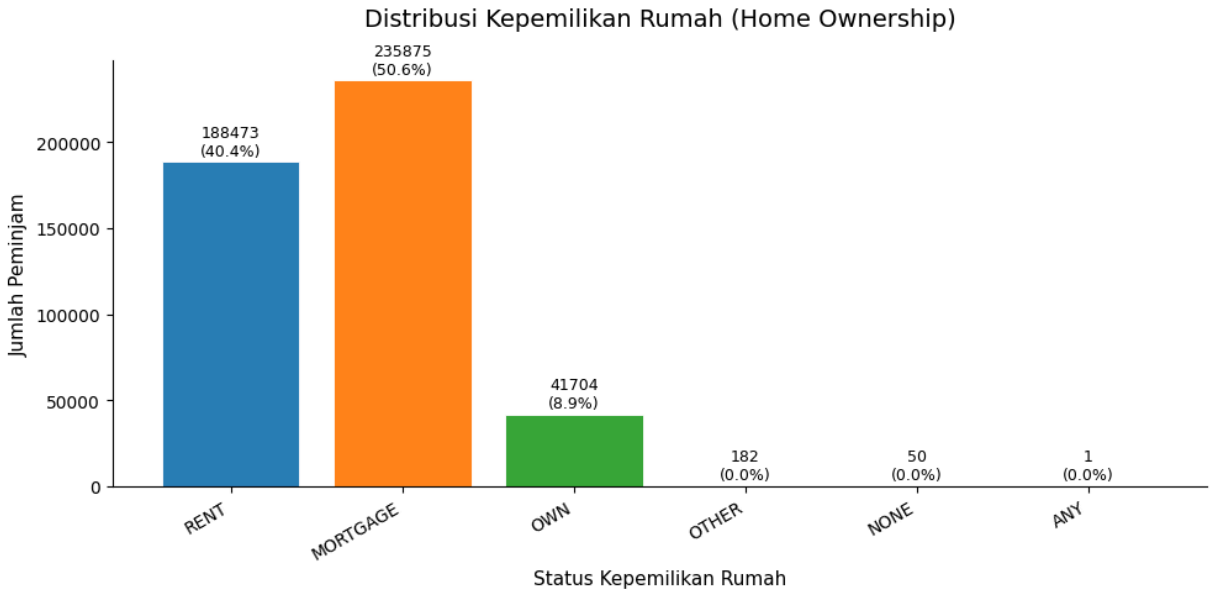
```
counts = df_p1['grade'].value_counts().sort_index()
fig, ax = plt.subplots(figsize=(10, 6))

# Bar chart
bars = ax.bar(
    counts.index,
    counts.values,
    edgecolor='black',
    alpha=0.85
)

ax.set_title('Distribusi Grade Pinjaman', fontsize=14, pad=2)
ax.set_xlabel('Grade', fontsize=11)
```

```
ax.set_ylabel('Jumlah Peminjam', fontsize=11)

total = counts.sum()
for bar in bars:
    height = bar.get_height()
    ax.text(
        bar.get_x() + bar.get_width() / 2,
        height,
        f'{int(height)}\n({height/total:.1%})',
        ha='center',
        va='bottom',
        fontsize=9
    )

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout()
plt.show()
```



In [15]:
```
counts = df_p1['home_ownership'].fillna('NA').astype(str).value_counts()
preferred_order = ['RENT', 'MORTGAGE', 'OWN', 'OTHER', 'NONE']

ordered = [c for c in preferred_order if c in counts.index]
others = [c for c in counts.index if c not in ordered]
counts = counts.reindex(ordered + others, fill_value=0)

fig, ax = plt.subplots(figsize=(10,5))
palette = plt.get_cmap('tab10').colors
bar_colors = palette[:len(counts)]
bars = ax.bar(counts.index, counts.values, color=bar_colors, edgecolor='white', lin

ax.set_title('Distribusi Kepemilikan Rumah (Home Ownership)', fontsize=14, pad=20)
ax.set_xlabel('Status Kepemilikan Rumah', fontsize=11)
ax.set_ylabel('Jumlah Peminjam', fontsize=11)
```

```
plt.xticks(rotation=30, ha='right')

total = counts.sum()
for rect in bars:
    h = rect.get_height()
    ax.text(
        rect.get_x() + rect.get_width() / 2,
        h + total * 0.004,
        f'{int(h)}\n({h/total:.1%})',
        ha='center',
        va='bottom',
        fontsize=9
    )

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout()
plt.show()
```



Modelling Data

In [16]:
```
data = df_p1.copy()
```

In [17]:
```
data['month_issue_since_crline'] = (data['issue_d'].dt.year - data['earliest_cr_lin

data['month_last_pymnt_since_issue'] = (data['last_pymnt_d'].dt.year - data['issue_

data[['month_issue_since_crline', 'month_last_pymnt_since_issue']].head()
```

Out[17]:

	month_issue_since_crline	month_last_pymnt_since_issue
0	323.0	37.0
1	152.0	16.0
2	121.0	30.0
3	190.0	37.0
4	191.0	49.0

In [18]:
```
# Split data menjadi fitur dan target
X = data.drop('loan_status', axis =1)
y = data['loan_status']

X_train, X_test, y_train, y_test = (train_test_split(X,y,test_size=0.2, stratify=y,
```

In [19]:
```
y_train.value_counts()
```

Out[19]:
```
loan_status
1    331279
0     41749
Name: count, dtype: int64
```

In [20]:
```
class NumImputer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        numeric = X.select_dtypes(include=['int64', 'float64']).columns
        X[numeric] = IterativeImputer().fit_transform(X[numeric])
        return X
class CatImputer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        cat = X.select_dtypes(include=['object']).columns
        X[cat] = SimpleImputer(strategy = 'most_frequent').fit_transform(X[cat])
        return X

class scaler(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        num = X.select_dtypes(include=['int64', 'float64']).columns
        X[num] = MinMaxScaler().fit_transform(X[num])
        return X
class OrdinalEnc(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
```

```
        oe = OrdinalEncoder(categories=[['< 1 year', '1 year', '2 years', '3 years'
        X['emp_length'] = oe.fit_transform(X[['emp_length']])
        return X

class Labelenc(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        categorical_cols = X.select_dtypes(include=['object', 'category']).columns

        # Inisialisasi LabelEncoder
        label_encoders = {}

        # Lakukan label encoding untuk setiap kolom kategorik
        for col in categorical_cols:
            le = LabelEncoder()
            X[col] = le.fit_transform(X[col])
            label_encoders[col] = le
        return X
```

In [21]:
```
pipe_prepo = Pipeline([
    ('num_imputer', NumImputer()),
    ('cat_imputer', CatImputer()),
    ('scaler', scaler()),
    ('ordinal encoder', OrdinalEnc()),
    ('label encoder', Labelenc())
])
```

In [22]:
```
X_train = pipe_prepo.fit_transform(X_train)
X_test = pipe_prepo.transform(X_test)
```

```
c:\Users\Alvin\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\imp
ute\_iterative.py:825: ConvergenceWarning: [IterativeImputer] Early stopping criteri
on not reached.
  warnings.warn(
```

In [23]:
```
display(X_train.head(), X_test.head())
```

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub
154804	0.923188	0.923188	0.924286	0	0.372578	0.771738	1	
420043	0.112319	0.112319	0.125000	0	0.120155	0.086946	0	
187234	0.492754	0.492754	0.500000	0	0.479167	0.425756	2	
1808	0.057971	0.057971	0.070714	0	0.253391	0.047170	1	
271265	0.275362	0.275362	0.285714	0	0.109981	0.212484	0	

5 rows × 49 columns

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub
451780	0.369565	0.369565	0.378571	0	0.438469	0.316136	2	
142708	0.507246	0.507246	0.514286	1	0.372578	0.283589	1	
397160	0.663768	0.663768	0.667143	0	0.398740	0.560240	2	
109361	0.889855	0.889855	0.891429	1	0.474806	0.524382	2	
304068	0.286957	0.286957	0.297143	0	0.181686	0.226839	1	

5 rows × 49 columns

In [24]:
```
X_train.isna().sum().sum()
X_test.isna().sum().sum()
```

Out[24]: 45484

In [ ]:
```
X_train = X_train.select_dtypes(exclude=['datetime64'])
X_test = X_test.select_dtypes(exclude=['datetime64'])

list_model = [
    LogisticRegression(random_state=0, max_iter=1000),
    MultinomialNB(),
    KNeighborsClassifier()
]

result = []

for model in list_model:
    pipeline = Pipeline([
        ('classifier', model)
    ])

    try:
        pipeline.fit(X_train, y_train)
        y_pred = pipeline.predict(X_test)

        bal_accuracy = balanced_accuracy_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred, average='binary')

        if hasattr(pipeline, "predict_proba"):
            roc_auc = roc_auc_score(y_test, pipeline.predict_proba(X_test)[:, 1])
        else:
            roc_auc = None

        hasil = {
            'Model': type(model).__name__,
            'Balanced Accuracy': bal_accuracy,
            'F1 Score': f1,
            'ROC AUC': roc_auc
        }
        result.append(hasil)
    except Exception as e:
```

```
        print(f"Gagal melatih model {type(model).__name__}: {e}")

result_all = pd.DataFrame(result)
print(result_all)
```

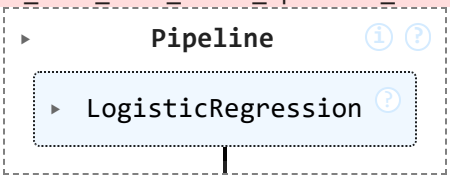
```
c:\Users\Alvin\AppData\Local\Programs\Python\Python312\Lib\site-packages\joblib\externals\loky\backend\context.py:136: UserWarning: Could not find the number of physical cores for the following reason:
found 0 physical cores < 1
Returning the number of logical cores instead. You can silence this warning by setting LOKY_MAX_CPU_COUNT to the number of cores you want to use.
  warnings.warn(
  File "c:\Users\Alvin\AppData\Local\Programs\Python\Python312\Lib\site-packages\joblib\externals\loky\backend\context.py", line 282, in _count_physical_cores
    raise ValueError(f"found {cpu_count_physical} physical cores < 1")
```

	Model	Balanced Accuracy	F1 Score	ROC AUC
0	LogisticRegression	0.892381	0.986307	0.968555
1	MultinomialNB	0.626491	0.910980	0.736351
2	KNeighborsClassifier	0.533860	0.937456	0.638805

In [26]:
```
best = Pipeline([
        ('classifier', LogisticRegression(random_state=0))
    ])
best.fit(X_train, y_train)
```

```
c:\Users\Alvin\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[26]:

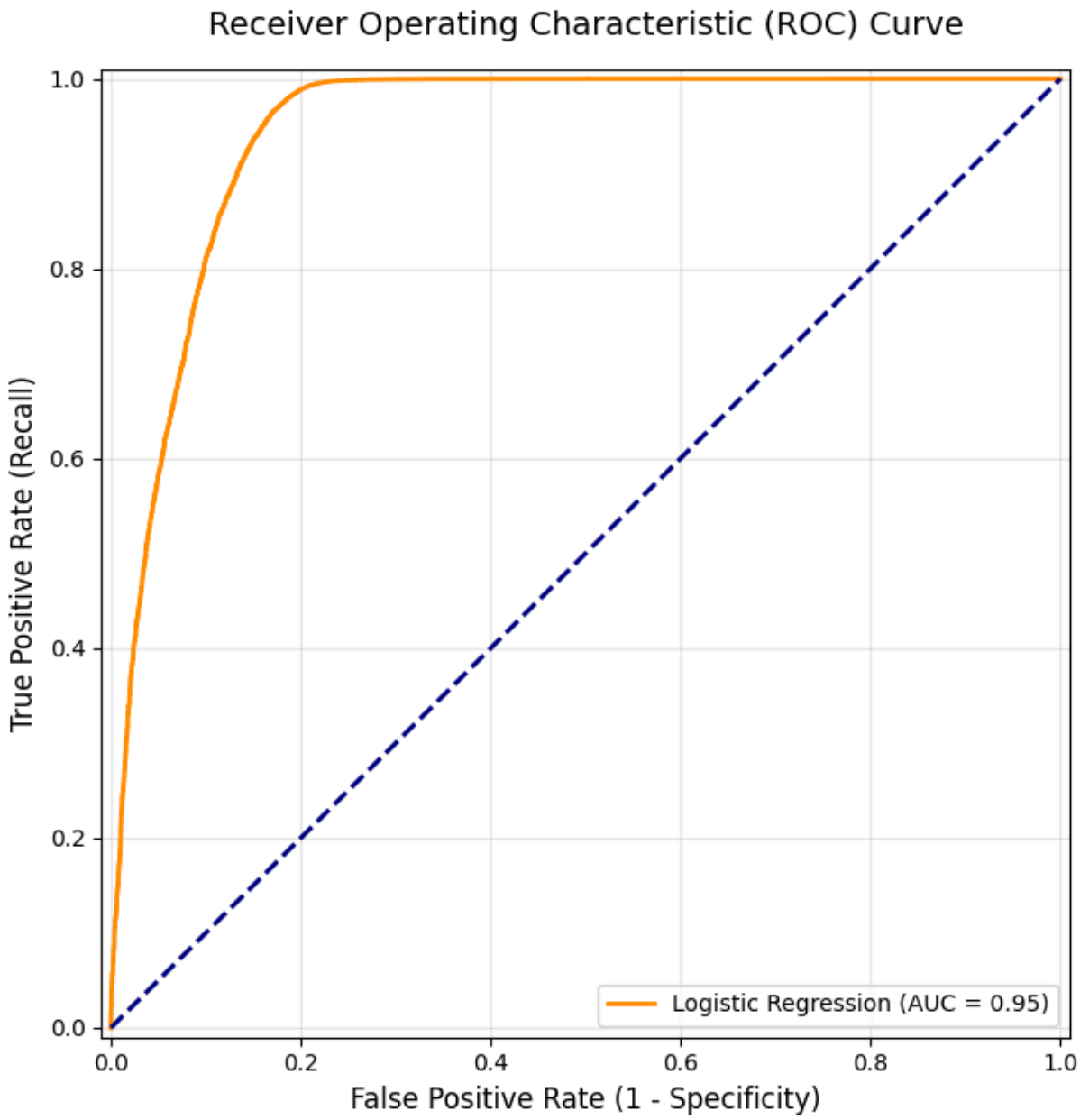


In [33]:
```
Y_pred = best.predict(X_test)

bal_accuracy = balanced_accuracy_score(y_test, Y_pred)
f1 = f1_score(y_test, Y_pred, average='binary')
roc_auc = roc_auc_score(y_test, best.predict_proba(X_test)[:, 1])

print(classification_report(y_test, Y_pred))
print(f'Balance Aquracy = {bal_accuracy}\nF1 score \t= {f1}\nROC AUC \t= {roc_auc}'
```

```
              precision    recall  f1-score   support

           0       0.99      0.74      0.85     10437
           1       0.97      1.00      0.98     82820

    accuracy                           0.97     93257
   macro avg       0.98      0.87      0.92     93257
weighted avg       0.97      0.97      0.97     93257

Balance Aquracy = 0.871222280845293
F1 score        = 0.9834790828186029
ROC AUC         = 0.9453035793908122
```

In [36]:

```
fig, ax = plt.subplots(figsize=(8, 7))

RocCurveDisplay.from_estimator(
    best,
    X_test,
    y_test,
    ax=ax,
    name='Logistic Regression',
    color='darkorange',
    linewidth=2
)

ax.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
ax.set_title('Receiver Operating Characteristic (ROC) Curve', fontsize=14, pad=15)
ax.set_xlabel('False Positive Rate (1 - Specificity)', fontsize=12)
ax.set_ylabel('True Positive Rate (Recall)', fontsize=12)
ax.legend(loc="lower right")
ax.grid(alpha=0.3)

plt.tight_layout()
plt.show()
```



In [ ]:
```python
# Save the best model
import joblib

filename = 'credit_risk_model_logistic.pkl'
joblib.dump(best, filename)
print(f"Model berhasil disimpan sebagai: {filename}")
```

```
Model berhasil disimpan sebagai: credit_risk_model_logistic.pkl
```