Credit Risk Modeling

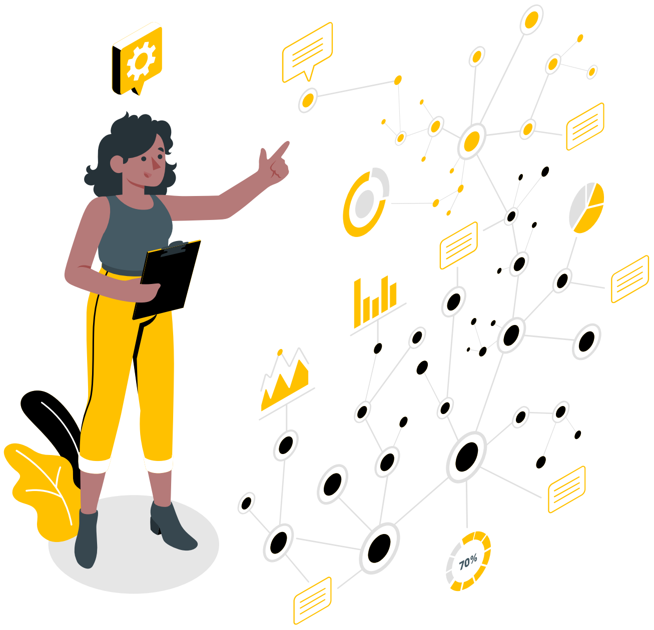
by Muhammad Noor Rizki Isya

Full Project:
Github/rizkyisya17/CreditRisk

Table of Contents

Data Understanding & EDA

01



data_raw.head()

```python
data_raw = pd.read_csv('loan_data_2007_2014.csv')
data_raw.head()
```

✓ 2.7s Python

```
C:\Users\rizky\AppData\Local\Temp\ipykernel_22848\4200974262.py:1: DtypeWarning: Columns (20) have mixed types. Specify dtype option on import or set low_memory=False.
  data_raw = pd.read_csv('loan_data_2007_2014.csv')
```

	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	max_bal_bc	all_util	total_rev_hi_
0	0	1077501	1296599	5000	5000	4975.0	36 months	10.65	162.87	B	...	NaN	NaN	NaN	NaN	NaN	NaN	N
1	1	1077430	1314167	2500	2500	2500.0	60 months	15.27	59.83	C	...	NaN	NaN	NaN	NaN	NaN	NaN	N
2	2	1077175	1313524	2400	2400	2400.0	36 months	15.96	84.33	C	...	NaN	NaN	NaN	NaN	NaN	NaN	N
3	3	1076863	1277178	10000	10000	10000.0	36 months	13.49	339.31	C	...	NaN	NaN	NaN	NaN	NaN	NaN	N
4	4	1075358	1311748	3000	3000	3000.0	60 months	12.69	67.79	B	...	NaN	NaN	NaN	NaN	NaN	NaN	N

5 rows × 75 columns

data_raw.info()

```
data_raw.info()
✓ 0.8s
<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
...
 73  total_cu_tl                  0 non-null       float64
 74  inq_last_12m                 0 non-null       float64
dtypes: float64(46), int64(7), object(22)
memory usage: 266.8+ MB
```

- Terdapat 76 kolom pada data degan 466285 baris
- Terdapat data object yang seharusnya int
- Terdapat data object yang seharusnya datetime

data_raw object

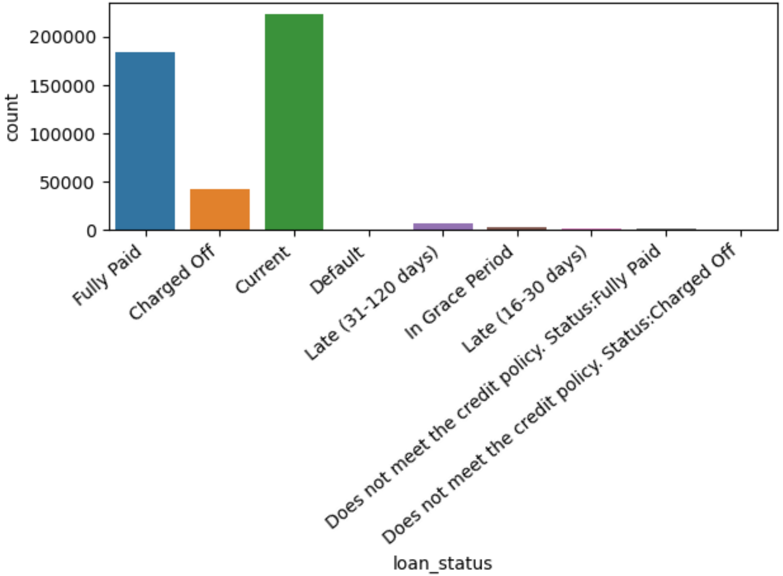
```python
data_raw.select_dtypes(include='object').columns.to_list()
```

✓ 0.0s

```
['term',
 'grade',
 'sub_grade',
 'emp_title',
 'emp_length',
 'home_ownership',
 'verification_status',
 'issue_d',
 'loan_status',
 'pymnt_plan',
 'url',
 'desc',
 'purpose',
 'title',
 'zip_code',
 'addr_state',
 'earliest_cr_line',
 'initial_list_status',
 'last_pymnt_d',
 'next_pymnt_d',
 'last_credit_pull_d',
 'application_type']
```

- Term, grade, emp_length, home ownership akan dirubah menjadi `int `
- issue_d, last_pymnt_d, next_pymnt_d akan dirubah menjadi `datetime`
- loan_status akan menjadi dasar sumber dari target model

Loan_status



- Dapat dipastikan Fully paid (pembayaran lunas) akan menjadi nilai positif atau good loan.
- Charged Off (pembayaran macet) dan Default (gagal bayar) akan menjadi nilai negatif atau bad loan.
- Current (pembayaran lancar) dan status yang lain akan dipertanyakan, umumnya akan menjadi good loan namun saya akan membuat threshold beberapa data menjadi bad loan karena tujuan dari model ini untuk memprediksi risk dari loan tersebut.

Duplicate records

```python
# Find duplicate records
duplicates = data_raw.duplicated()
# Print the number of duplicate records
print("Number of duplicate records:", duplicates.sum())
```

✓ 1.3s

```
Number of duplicate records: 0
```

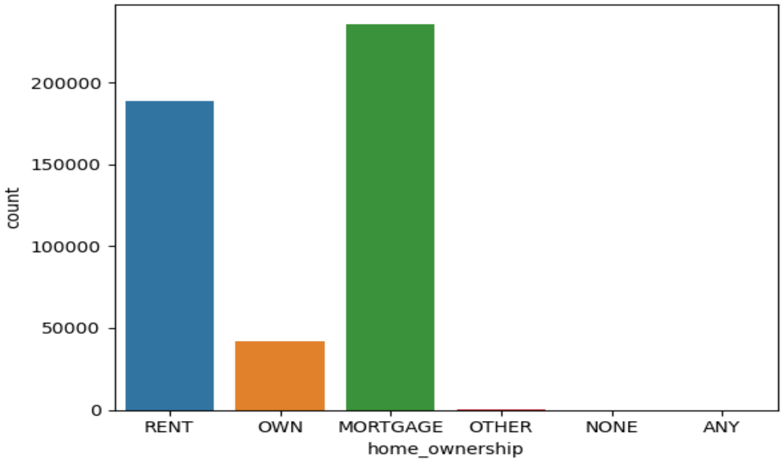
```python
print ("{:.0%} Data Unique".format(len(data_raw) / data_raw['member_id'].nunique()))
```

✓ 0.0s

```
100% Data Unique
```

Tidak ditemukan data duplicated dan dapat disimpulkan bahwa dataset benar merupakan data member loaners dan bukan data transaksi.

Home Ownership



- Home ownership tidak terlalu mempengaruhi loan_status
- Disamping itu karena data bad_loan sendiri juga tidak terlalu banyak dibandingkan good_loan dan status loan lainnya

```
data_raw[data_raw['home_ownership']=='RENT'].groupby('loan_status')
✓ 0.1s

loan_status
Current                                                85911
Fully Paid                                             76025
Charged Off                                            19906
Late (31-120 days)                                      3097
In Grace Period                                         1376
Does not meet the credit policy. Status:Fully Paid       911
Late (16-30 days)                                        483
Default                                                  412
Does not meet the credit policy. Status:Charged Off      352
Name: loan_status, dtype: int64
```

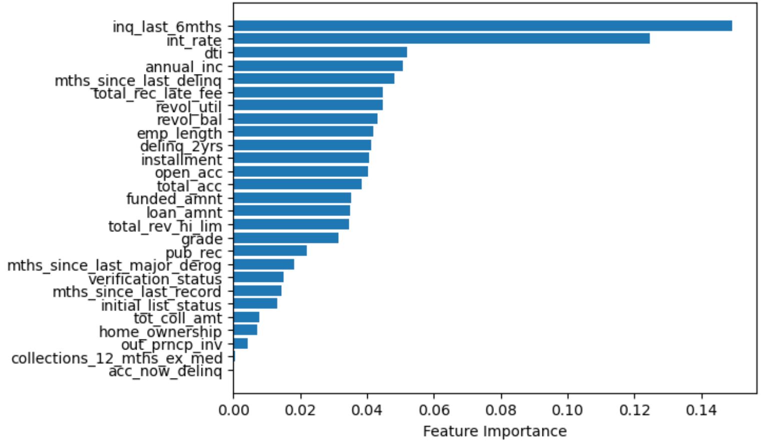
```
data_raw[data_raw['home_ownership']=='MORTGAGE'].groupby('loan_status')
✓ 0.1s

loan_status
Current                                               117038
Fully Paid                                             93221
Charged Off                                            18799
Late (31-120 days)                                      3142
In Grace Period                                         1461
Does not meet the credit policy. Status:Fully Paid       908
Late (16-30 days)                                        607
Default                                                  351
Does not meet the credit policy. Status:Charged Off      348
Name: loan_status, dtype: int64
```

```
data_raw[data_raw['home_ownership']=='OWN'].groupby('loan_status
✓ 0.0s

loan_status
Current                                                21272
Fully Paid                                             15342
Charged Off                                             3736
Late (31-120 days)                                       661
In Grace Period                                          309
Does not meet the credit policy. Status:Fully Paid       138
Late (16-30 days)                                        128
Default                                                   69
Does not meet the credit policy. Status:Charged Off       49
Name: loan_status, dtype: int64
```

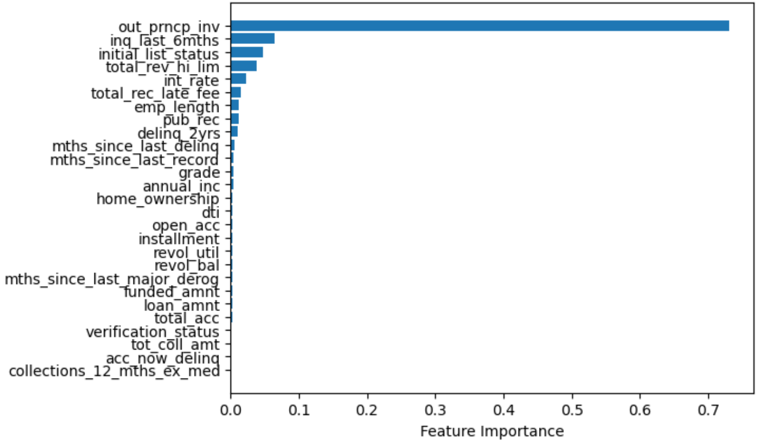
Data Preprocessing

02

Features Selection



- good loan: Fully Paid
- bad_loan: Charged Off dan Deafult



- good_loan: Current
- bad_loan: Charged Off dan Deafult

Features Selection

```
df = data_raw[['loan_amnt','funded_amnt', 'term', 'int_rate', 'installment', 'grade', 'emp_length', 'home_ownership', 'annual_inc', 'issue_d',
               'loan_status','dti','delinq_2yrs', 'inq_last_6mths', 'mths_since_last_delinq', 'mths_since_last_record', 'open_acc', 'pub_rec',
               'revol_bal', 'revol_util', 'total_acc', 'out_prncp_inv', 'total_rec_late_fee', 'last_pymnt_d', 'last_credit_pull_d']].copy()
```

Berdasarkan feature importance dan asumsi maka dipilih feature-feature berikut untuk membuat model.

Feature Engineering

mths_check

```
df_train['last_credit_pull_d'] = pd.to_datetime(df_train['last_credit_pull_d'], format='%b-%y')
df_train['last_pymnt_d'] = pd.to_datetime(df_train['last_pymnt_d'], format='%b-%y')
df_train['mths_check'] = round((df_train['last_credit_pull_d'] - df_train['last_pymnt_d'])/np.timedelta64(1, 'M'))
```

```
print(df_train[:1]['last_credit_pull_d'])
print(df_train[:1]['last_pymnt_d'])
print(df_train[:1]['mths_check'])
```

```
0   2016-01-01
Name: last_credit_pull_d, dtype: datetime64[ns]
0   2015-01-01
Name: last_pymnt_d, dtype: datetime64[ns]
0    12.0
Name: mths_check, dtype: float64
```

Perbedaan jumlah bulan dari tgl pengecekan terakhir credit history dan tgl pembayaran terakhir.

Feature Engineering

finish_d

```python
df_train['term'] = df_train['term'].str.replace(' months', '')
df_train['term'] = df_train['term'].astype('int')
```

```python
df_train['issue_d'] = pd.to_datetime(df_train['issue_d'], format='%b-%y')
df_train['finish_d'] = ((df_train['issue_d'].dt.to_period('M')) + df_train['term']).dt.to_timestamp()
```

Tanggal dimana pembayaran seharusnya selesai sesuai term.

mths_remain

```python
df_train['mths_remain'] = round((df_train['finish_d'] - df_train['last_pymnt_d'])/np.timedelta64(1, 'M'))
```

```python
print(df_train[:1]['finish_d'])
print(df_train[:1]['last_pymnt_d'])
print(df_train[:1]['mths_remain'])
```

```
0   2014-12-01
Name: finish_d, dtype: datetime64[ns]
0   2015-01-01
Name: last_pymnt_d, dtype: datetime64[ns]
0   -1.0
```

Jumlah bulan yang tersisa untuk menyelesaikan pembayaran sesuai finish_d.

Feature Engineering

paid_potention

```
df_train['paid_potention'] = round(((df_train['annual_inc']/12) * df_train['mths_remain']) - df_train['out_prncp_inv'])
✓ 0.0s

len(df_train[(df_train['loan_status']=='Current') &
             (df_train['paid_potention']<0)])
✓ 0.1s
370
```

Kemampuan loaners untuk membayar sisa pembayaran sampai tanggal penyelesaian sesuai dengan income.

emp_length

```
df_train = df_train.replace({'emp_length' : { '< 1 year' : '0 years', '1 year' : '1 years', '10+ years' : '10 years'}}
df_train['emp_length'] = df_train['emp_length'].fillna('0 years')
df_train['emp_length'] = df_train['emp_length'].replace(' years', '', regex=True)
df_train['emp_length'] = df_train['emp_length'].astype('int')
df_train['emp_length'].unique()

array([10,  0,  1,  3,  8,  9,  4,  5,  6,  2,  7])
```

Cleaning feature emp_length menjadi dtype int.

Feature Engineering

home_ownership

```
df_train = df_train.replace({'home_ownership' : { 'MORTGAGE' : 0, 'RENT' : 0, 'OWN' : 1, 'NONE': 1, 'ANY':1, 'OTHER':1}})
```
✓ 0.2s

Cleaning feature home_ownership:
0 (MORTGAGE, RENT)
1 (OWN, NONE, ANY, OTHER)

grade

```
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

df_train['grade'] = le.fit_transform(df_train['grade'])
#df_train['verification_status']=le.fit_transform(df_train['verification_status'])
#df_train['initial_list_status']=le.fit_transform(df_train['initial_list_status'])
```

Cleaning feature grade menjadi dtype int dengan label encoder:

A,B,C,D,E,F,G ->
0,1,2,3,4,5,6

Feature Engineering

risk

```
df_train['loan_status'].unique()
```

```
array(['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'],
      dtype=object)
```

```
bad = [
    'Charged Off',
    'Default',
    'Does not meet the credit policy. Status:Charged Off'
]

df_train['risk'] = np.where(df_train['loan_status'].isin(bad), 1, 0)
```

```
df_train['risk'].value_counts(normalize=True)*100
```

```
0    90.622646
1     9.377354
Name: risk, dtype: float64
```

- bad loan (Charged Off, Default, Does not meet the credit policy. Status: Charged Off) = 1
- good loan (Fully Paid, Current, In Grace Period, Late (16-30 days), Late (31-120 days), Does not meet the credit policy. Status: Fully Paid) = 0

Feature Engineering

risk

```
df_train.loc[((df_train['loan_status']=='Current') & ((df_train['mths_check']>=3) | (df_train['paid_potention']<0))), 'risk'] = 1
df_train.loc[((df_train['loan_status']=='Late (31-120 days)') & ((df_train['mths_check']>=3) | (df_train['paid_potention']<0))), 'risk'] = 1
df_train.loc[((df_train['loan_status']=='Late (16-30 days)') & ((df_train['mths_check']>=3) | (df_train['paid_potention']<0))), 'risk'] = 1
df_train.loc[((df_train['loan_status']=='In Grace Period') & ((df_train['mths_check']>=3) | (df_train['paid_potention']<0))), 'risk'] = 1
```

```
df_train['risk'].value_counts(normalize=True)*100
```

```
0    89.784638
1    10.215362
Name: risk, dtype: float64
```

Threshold good loan ke bad loan untuk status Current, Late, dan In Grace Period dengan parameter:

mths_check >= 3
paid_potention <0

Feature Transformation

Split Dataset

```
X = df_train[['loan_amnt','funded_amnt', 'int_rate', 'installment', 'grade', 'emp_length', 'home_ownership', 'dti','delinq_2yrs',
             'inq_last_6mths', 'mths_since_last_delinq', 'open_acc', 'pub_rec','revol_bal', 'revol_util', 'total_acc', 'out_prncp_inv',
             'total_rec_late_fee', 'mths_check', 'mths_remain', 'paid_potention']].copy()
y = df_train['risk'].copy()
```

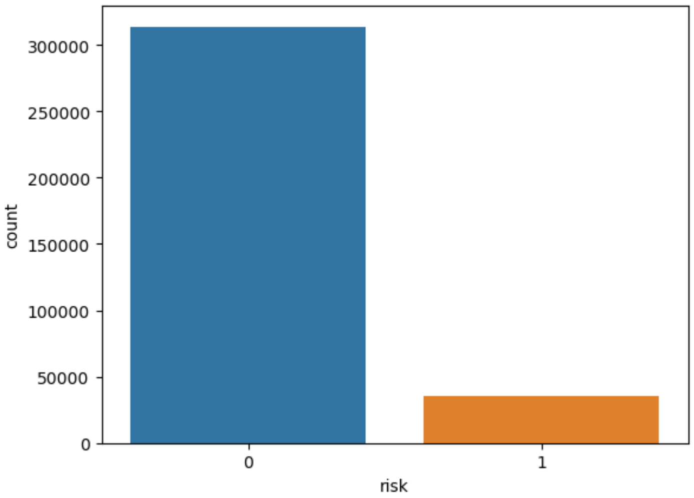
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
                                                    random_state=42)
print(len(X_train))
print(len(X_test))
```

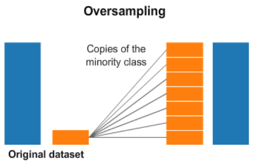
```
349400
116467
```

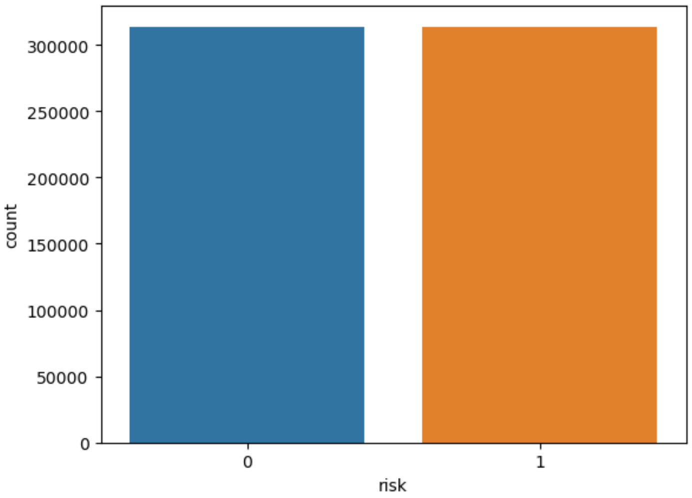
Split dataset untuk keperluan modelling dengan 75% menjadi data training dan 25% menjadi data test

Feature Transformation

SMOTE Imbalanced Dataset

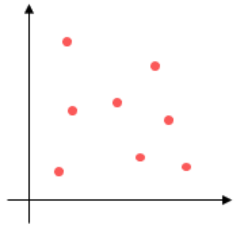






Feature Transformation

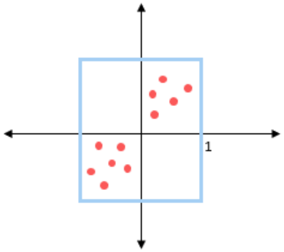
Standarization



Actual Data

	loan_amnt	funded_amnt	int_rate	installment	grade	emp_length	home_ownership	dti	delinq_2yrs	inq_last_6mths	...
0	8400	8400	14.33	288.45	2	1	0	19.28	0.0	0.0	...
1	7200	7200	19.52	265.83	4	0	0	27.19	1.0	0.0	...
2	17000	17000	7.69	530.30	0	5	0	16.38	0.0	0.0	...
3	18825	18825	16.59	463.71	3	8	0	10.20	0.0	1.0	...
4	14000	14000	16.99	347.87	3	2	0	22.43	1.0	3.0	...

```
scaler = StandardScaler()
scaler.fit(X_smote[:])
X_smote[:] = scaler.transform(X_smote[:])
X_test[:] = scaler.transform(X_test[:])
✓ 0.1s
```



After standardization

	loan_amnt	funded_amnt	int_rate	installment	grade	emp_length	home_ownership	dti	delinq_2yrs	inq_last_6mths	...
0	-0.725166	-0.723100	-0.106093	-0.608735	0.065672	-1.263566	-0.237779	0.218804	-0.381715	-0.840884	...
1	-0.869809	-0.867986	1.136133	-0.701992	1.669180	-1.548559	-0.237779	1.297247	0.982888	-0.840884	...
2	0.311445	0.315246	-1.695377	0.388361	-1.537836	-0.123595	-0.237779	-0.176580	-0.381715	-0.840884	...
3	0.531423	0.535593	0.434838	0.113825	0.867426	0.731383	-0.237779	-1.019156	-0.381715	0.084014	...
4	-0.050163	-0.046968	0.530578	-0.363759	0.867426	-0.978574	-0.237779	0.648272	0.982888	1.933809	...

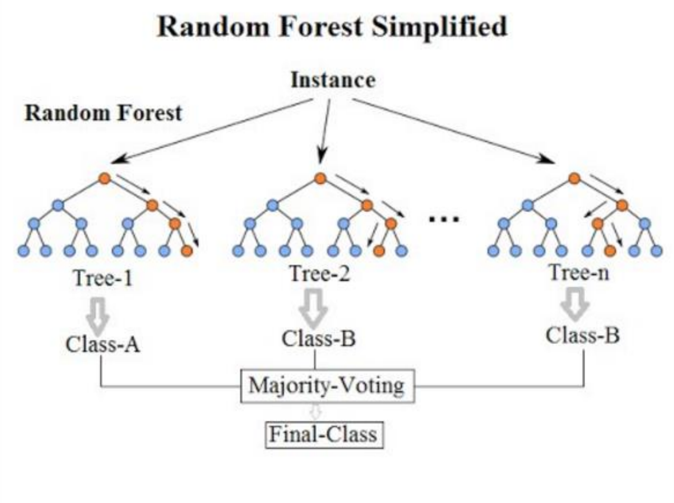
Data Modeling

03

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

rf = RandomForestClassifier()
rf.fit(X_smote, y_smote)
y_pred_rf = rf.predict(X_test)
print(classification_report(y_test, y_pred_rf))
```

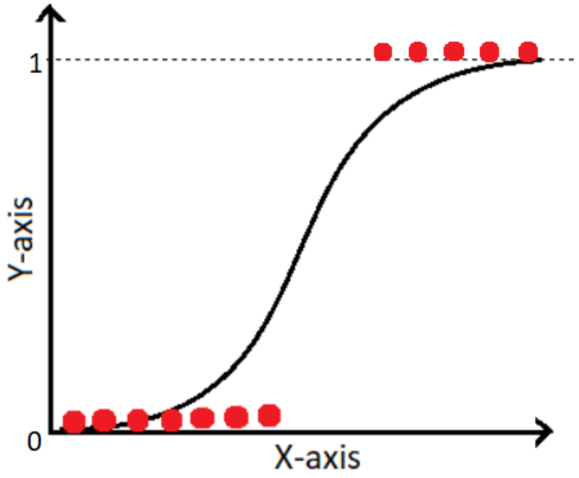
```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96    104476
           1       0.67      0.65      0.66     11991

    accuracy                           0.93    116467
   macro avg       0.82      0.81      0.81    116467
weighted avg       0.93      0.93      0.93    116467
```



Logistic Regression

```
from sklearn.linear_model import LogisticRegression

log = LogisticRegression()
log.fit(X_smote, y_smote)
y_pred_log = log.predict(X_test)
print(classification_report(y_test, y_pred_log))
```

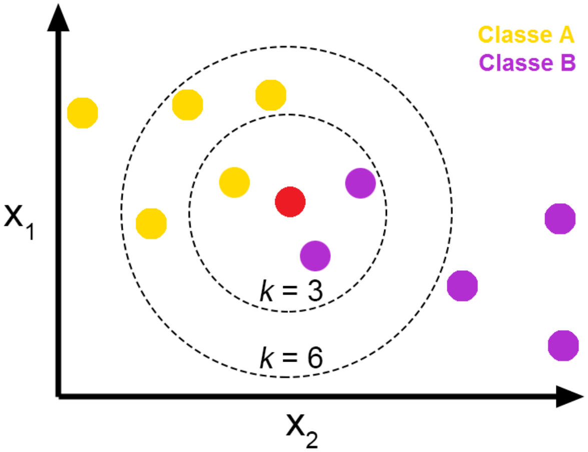
```
              precision    recall  f1-score   support

           0       0.95      0.81      0.88    104476
           1       0.29      0.65      0.40     11991

    accuracy                           0.80    116467
   macro avg       0.62      0.73      0.64    116467
weighted avg       0.88      0.80      0.83    116467
```



KNN

```python
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(X_smote, y_smote)
y_pred_knn = knn.predict(X_test)
print(classification_report(y_test, y_pred_knn))
```

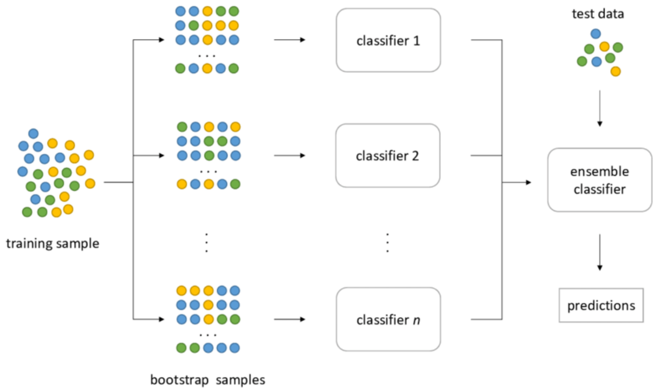
	precision	recall	f1-score	support
0	0.95	0.80	0.87	104476
1	0.27	0.65	0.38	11991
accuracy			0.79	116467
macro avg	0.61	0.72	0.63	116467
weighted avg	0.88	0.79	0.82	116467



XGBoost

```
from xgboost import XGBClassifier

xgb = XGBClassifier()
xgb.fit(X_smote, y_smote)
y_pred_xgb = xgb.predict(X_test)
print(classification_report(y_test, y_pred_xgb))
```

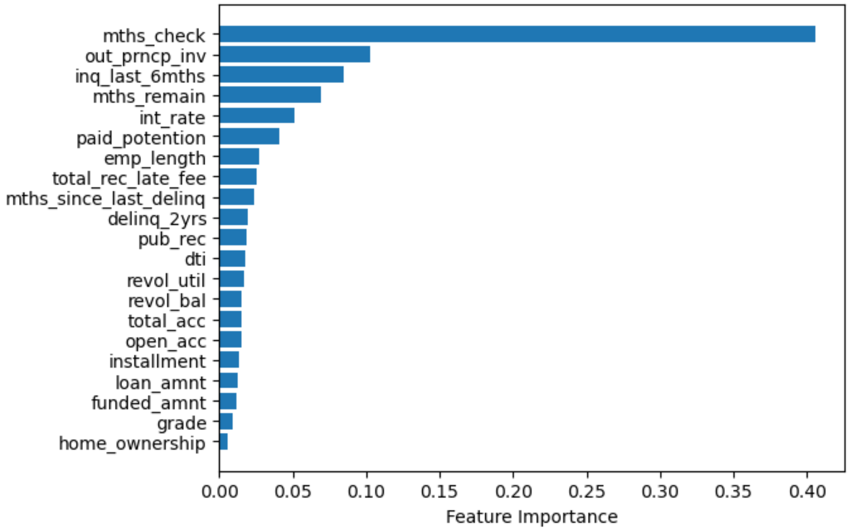
```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97    104476
           1       0.77      0.60      0.67     11991

    accuracy                           0.94    116467
   macro avg       0.86      0.79      0.82    116467
weighted avg       0.94      0.94      0.94    116467
```



Feature Importance & Remodeling

04

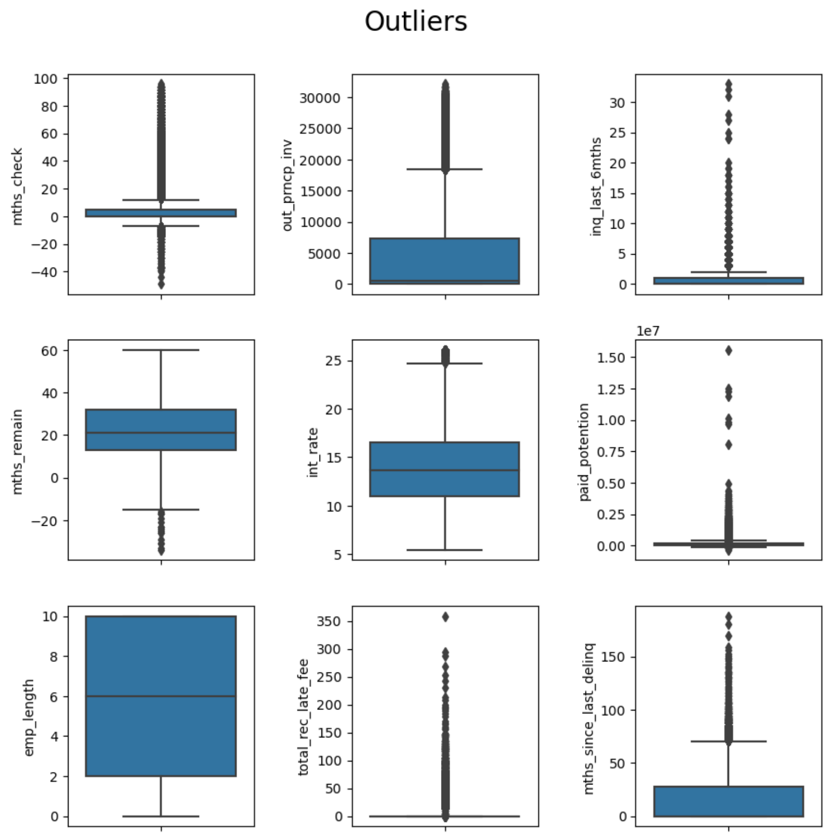
Features Importance



Model akan direbuild dengan feature berikut:

'mths_check',
'out_prncp_inv',
'inq_last_6mths',
'mths_remain',
'int_rate',
'paid_potention',
'emp_length',
'total_rec_late_fee',
'mths_since_last_delinq'

New Features: Outlier



Terdapat banyak sekali outlier, namun apabila semua outlier dihilangkan akan membuat kehilangan banyak data. Sehingga saya hanya mencoba menghilangkan outlier dengan threshold IQR biasa.

```
for x in X_importance:
    q1 = np.percentile(df_train_importance[x], 25)
    q3 = np.percentile(df_train_importance[x], 75)
    iqr = q3-q1
    lower_bound = q1 - (1.5*iqr)
    upper_bound = q3 + (1.5*iqr)
    df_train_new = df_train_importance[(df_train_importance[x] >= lower_bound) & (df_train_importance[x] <= upper_bound)]
```
✓ 0.2s

```
print(len(df_train_importance))
print(len(df_train_new))
```
✓ 0.0s

```
465867
449104
```

New Features: Correlation

	mths_check	out_prncp_inv	inq_last_6mths	mths_remain	int_rate	paid_potention	emp_length	total_rec_late_fee	mths_since_last_delinq	risk
mths_check	1.000000	-0.305449	0.106134	0.056424	-0.011240	0.031033	-0.036183	0.016731	-0.013785	0.217281
out_prncp_inv	-0.305449	1.000000	-0.070845	0.343532	0.137767	0.299820	0.087130	-0.006573	0.010787	-0.184983
inq_last_6mths	0.106134	-0.070845	1.000000	0.022975	0.205419	0.056684	-0.012327	0.030573	0.036821	0.073142
mths_remain	0.056424	0.343532	0.022975	1.000000	0.364467	0.633069	0.074722	-0.019401	0.022110	0.177357
int_rate	-0.011240	0.137767	0.205419	0.364467	1.000000	0.191212	0.025642	0.057762	0.074740	0.164652
paid_potention	0.031033	0.299820	0.056684	0.633069	0.191212	1.000000	0.103011	0.003874	0.029211	0.067302
emp_length	-0.036183	0.087130	-0.012327	0.074722	0.025642	0.103011	1.000000	-0.008753	0.030589	-0.020060
total_rec_late_fee	0.016731	-0.006573	0.030573	-0.019401	0.057762	0.003874	-0.008753	1.000000	0.004350	0.123114
mths_since_last_delinq	-0.013785	0.010787	0.036821	0.022110	0.074740	0.029211	0.030589	0.004350	1.000000	-0.007297
risk	0.217281	-0.184983	0.073142	0.177357	0.164652	0.067302	-0.020060	0.123114	-0.007297	1.000000

Correlation feature dengan target tidak terlalu besar, namun terdapat correlation antar feature yang bersifat multicollinearity sehingga column ‘mths_remain’ akan di drop pada modeling selanjutnya.

Remodeling + New Parameter

```
from sklearn.preprocessing import MinMaxScaler

Xnew = df_train[['mths_check', 'out_prncp_inv', 'inq_last_6mths', 'mths_remain', 'int_rate', 'paid_potention',  'emp_length',
                 'total_rec_late_fee', 'mths_since_last_delinq']].copy()
ynew = df_train['risk'].copy()

Xnew_train, Xnew_test, ynew_train, ynew_test = train_test_split(Xnew, ynew, test_size=0.25,
                                                                random_state=42)

Xnew_smote, ynew_smote = oversample.fit_resample(Xnew_train, ynew_train)

scaler_new = MinMaxScaler()
scaler_new.fit(Xnew_smote[:])
Xnew_smote[:] = scaler_new.transform(Xnew_smote[:])
Xnew_test[:] = scaler_new.transform(Xnew_test[:])
```

✓ 1.0s

```
rf_new = RandomForestClassifier(n_estimators=400, max_depth=10,min_samples_leaf=1,max_features='sqrt',
                                min_samples_split=5, criterion='gini')
rf_new.fit(Xnew_smote, ynew_smote)
y_pred_rf_new = rf_new.predict(Xnew_test)
print(classification_report(ynew_test, y_pred_rf_new))
```

✓ 6m 1.1s

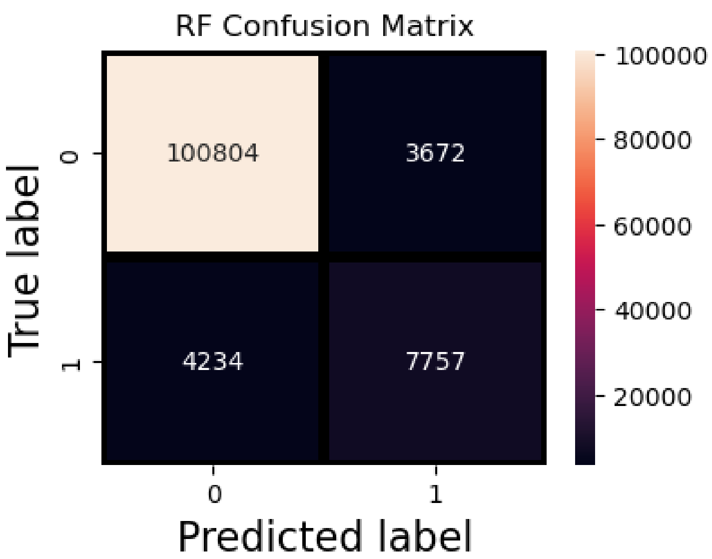
```
              precision    recall  f1-score   support

           0       0.98      0.88      0.93    104476
           1       0.44      0.86      0.59     11991

    accuracy                           0.87    116467
   macro avg       0.71      0.87      0.76    116467
weighted avg       0.93      0.87      0.89    116467
```

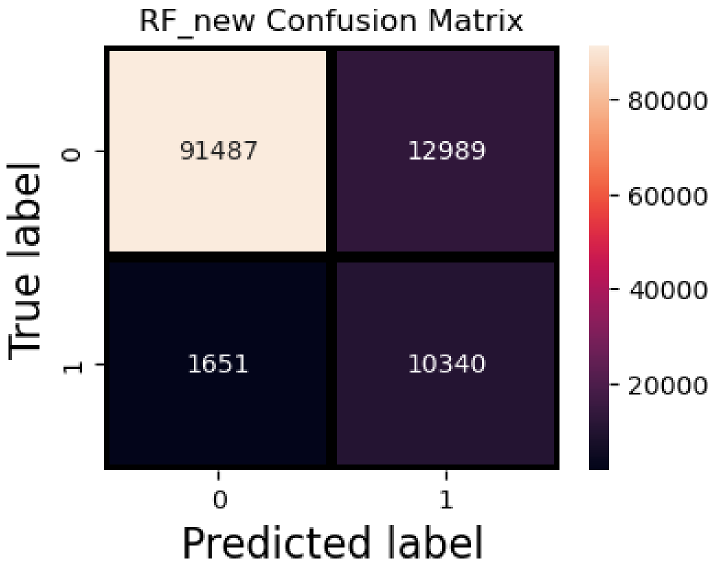
Model Evaluation

05

Evaluation: Confusion Matrix

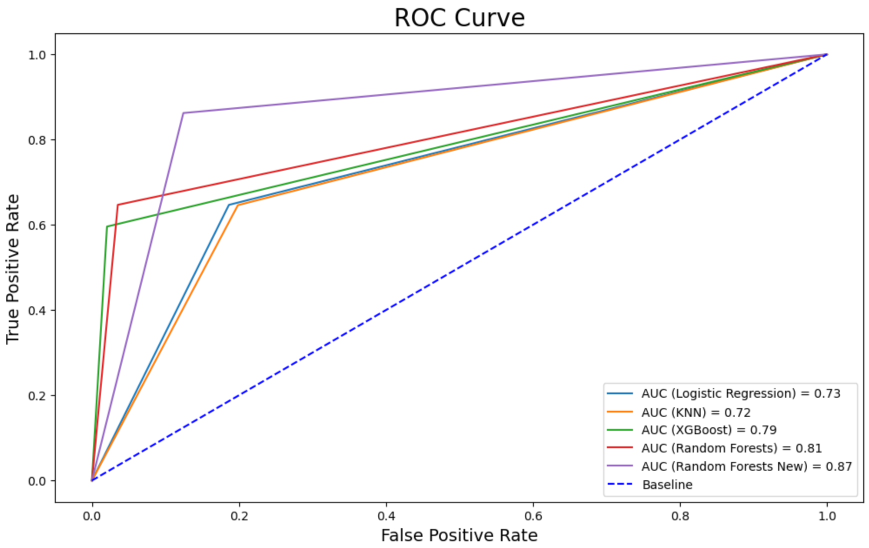
What is the difference?





Model RF yang baru memiliki nilai recall yang lebih baik dimana lebih banyak loaners yang diprediksi Bad Loan, namun dengan mengorbankan beberapa True Good Loan menjadi Bad Loan. Tetapi bagi saya lebih baik memprediksi lebih banyak Bad Loan untuk menghindari kerugian yang lebih banyak sehingga saya lebih prefer ke model yang baru.

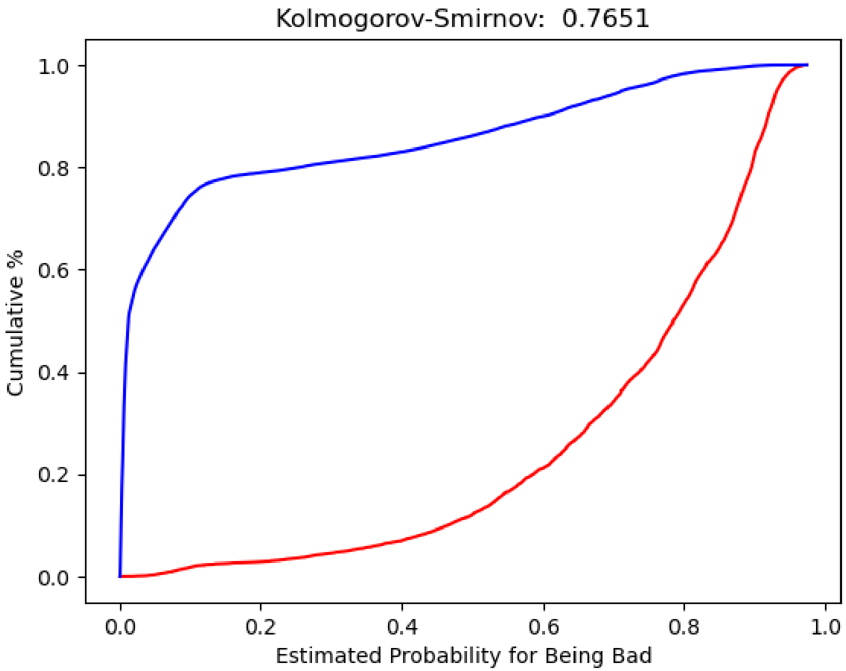
Evaluation: AUC



Berdasarkan nilai AUC, model Random Forests yang baru merupakan model dengan skor AUC tertinggi sebesar 0.87

Umumnya nilai AUC diatas 0.7 model dapat dikatakan memiliki performa yang cukup baik

Evaluation: KS

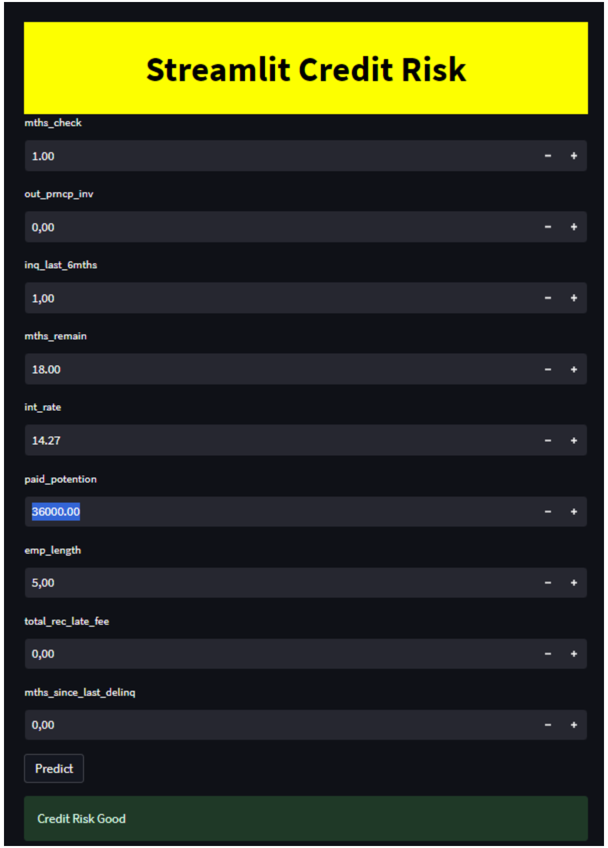


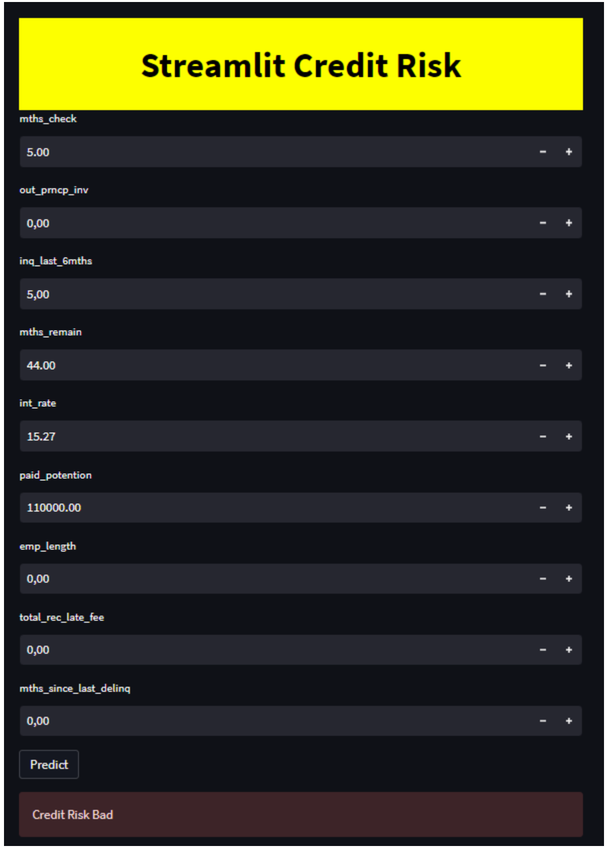
Evaluasi KS menjelaskan bagaimana berbedanya dua distribusi data satu sama lain. Pada RF new dengan evaluasi KS didapatkan nilai 0.7651

Model Deployment

06

Deployment on Streamlit





Thanks

Notebook details and project:
[Github/rizkyisya17/CreditRisk]

Reach me on LinkedIn:
Muhammad Noor Rizki Isya
or Email:
rizkyisya@gmail.com