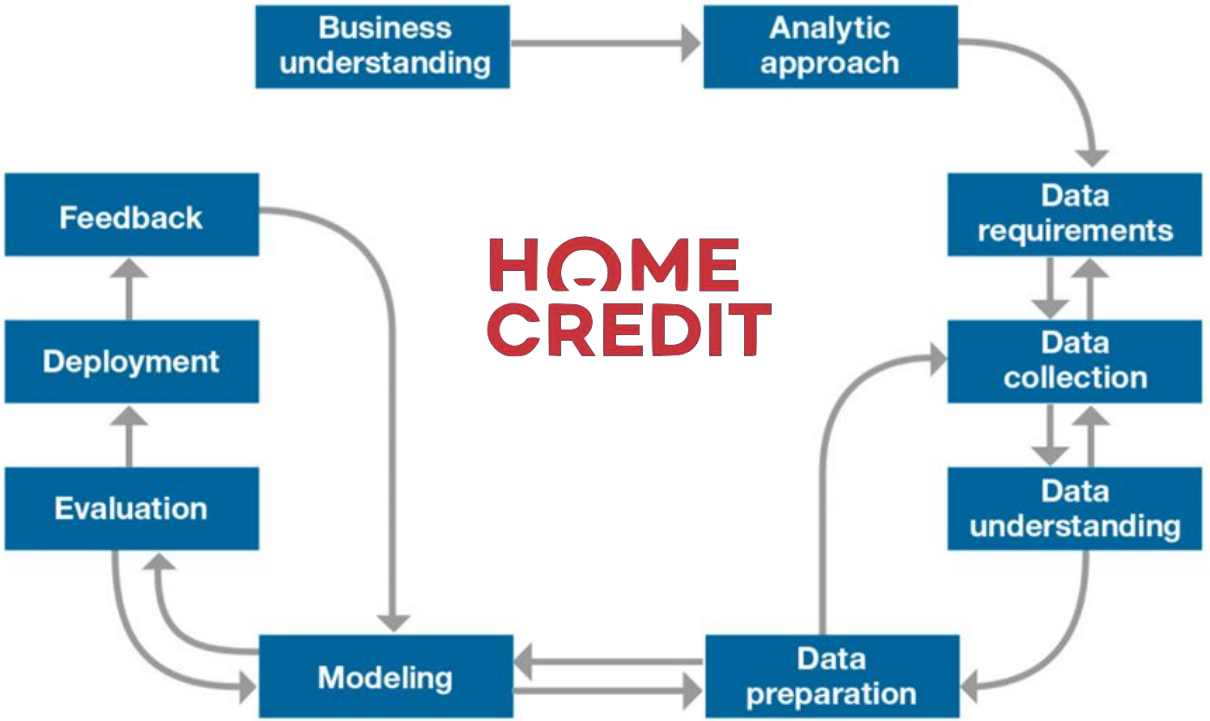
Dataset & Modelling Project

Home Credit Default Risk
Java Team

Data Science Methodology



01

Business Understanding

Home Credit:
Business Context

Menyediakan pembiayaan yang bertanggung jawab, khususnya bagi masyarakat yang tidak memiliki catatan kredit atau memiliki catatan kredit yang terbatas.

Menyediakan pembiayaan di toko (pembiayaan non-tunai langsung di tempat) untuk konsumen yang ingin membeli produk-produk seperti alat rumah tangga, alat-alat elektronik, handphone, dan furniture.

Business understanding

Permasalahan bisnis yang akan dipecahkan:
Pemohon manakah yang mampu membayar pinjaman?

Why

- Mengurangi kesalahan dalam menyetujui permohonan pinjaman dengan cepat secara terautomasi.
- Pertumbuhan perusahaan Home Credit kebanyakan datang dari cross-selling produk terhadap customer yang sudah memiliki credit history baik/ mampu membayar pinjaman.

  (meningkatkan growth empat atau lima kali original amount melalui peningkatan volume dan memperpanjang repayment period).

Goals

- Membuat sistem untuk membantu dalam persetujuan permohonan pinjaman secara otomatis
- Membuat model untuk meningkatkan kecepatan pengajuan kredit tanpa menambah cost karyawan

02

Exploratory Data Analysis

2.1 Feature Correlation

Top 5 feature yang paling berpengaruh terhadap target.

Negative Correlation:
Semakin ***tinggi*** nilainya, semakin ***tinggi*** kemungkinan orang membayar tepat waktu.

```
Most Negative Correlations:
 EXT_SOURCE_3                -0.178919
EXT_SOURCE_2                 -0.160472
EXT_SOURCE_1                 -0.155317
YEAR_BIRTH                   -0.078263
DAYS_BIRTH                   -0.078239
DAYS_EMPLOYED                -0.074958
DAYS_EMPLOYED_ANOM           -0.045987
FLOORSMAX_AVG                -0.044003
FLOORSMAX_MEDI               -0.043768
FLOORSMAX_MODE               -0.043226
AMT_GOODS_PRICE              -0.039645
REGION_POPULATION_RELATIVE   -0.037227
ELEVATORS_AVG                -0.034199
ELEVATORS_MEDI               -0.033863
FLOORSMIN_AVG                -0.033614
FLOORSMIN_MEDI               -0.033394
LIVINGAREA_AVG               -0.032997
LIVINGAREA_MEDI              -0.032739
FLOORSMIN_MODE               -0.032698
TOTALAREA_MODE               -0.032596
Name: TARGET, dtype: float64
```

Positive Correlation:
Semakin ***tinggi*** nilainya, semakin ***rendah*** kemungkinan orang membayar tepat waktu.

```
Most Positive Correlations:
 OBS_60_CNT_SOCIAL_CIRCLE     0.009022
OBS_30_CNT_SOCIAL_CIRCLE     0.009131
CNT_FAM_MEMBERS              0.009308
CNT_CHILDREN                 0.019187
AMT_REQ_CREDIT_BUREAU_YEAR   0.019930
FLAG_WORK_PHONE              0.028524
DEF_60_CNT_SOCIAL_CIRCLE     0.031276
DEF_30_CNT_SOCIAL_CIRCLE     0.032248
LIVE_CITY_NOT_WORK_CITY      0.032518
OWN_CAR_AGE                  0.037612
DAYS_REGISTRATION            0.041975
FLAG_DOCUMENT_3              0.044346
REG_CITY_NOT_LIVE_CITY       0.044395
FLAG_EMP_PHONE               0.045982
REG_CITY_NOT_WORK_CITY       0.050994
DAYS_ID_PUBLISH              0.051457
DAYS_LAST_PHONE_CHANGE       0.055218
REGION_RATING_CLIENT         0.058899
REGION_RATING_CLIENT_W_CITY  0.060893
TARGET                       1.000000
Name: TARGET, dtype: float64
```

2.1.1. Days of Birth (1/2)

```
df['DAYS_BIRTH'].head()
```

```
0    -9461
1   -16765
2   -19046
3   -19005
4   -19932
Name: DAYS_BIRTH, dtype: int64
```

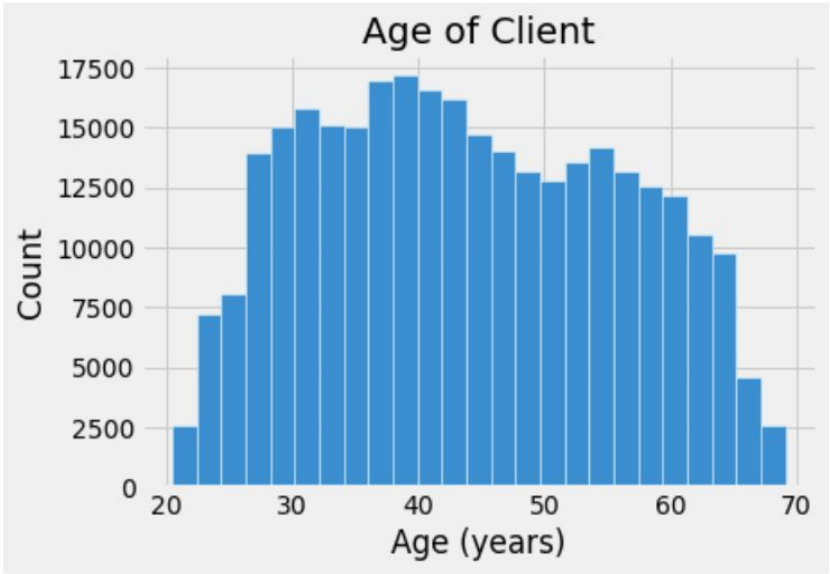
```
#no anomalies
(df['DAYS_BIRTH'] / -365).describe()
```

```
count    307511.000000
mean         43.936973
std          11.956133
min          20.517808
25%          34.008219
50%          43.150685
75%          53.923288
max          69.120548
Name: DAYS_BIRTH, dtype: float64
```

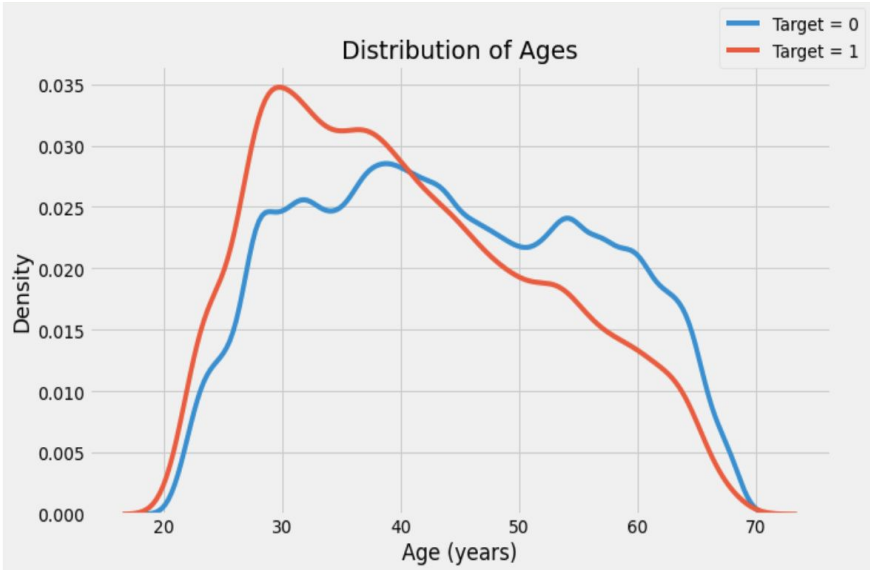
Data hari yang minus dikonversi untuk melihat apakah terdapat anomali.

Tidak terdapat anomali jika melihat dari descriptive statistics.

2.1.1. Days of Birth (2/2)



- Data umur tersebar secara normal



- Target = 1 atau telat membayar skew kepada orang - orang yang lebih muda. Berusia sekitar late 20s - 40.

2.1.2. Days of Employment

Terdapat anomali data karena nilai min & max sangatlah jauh.

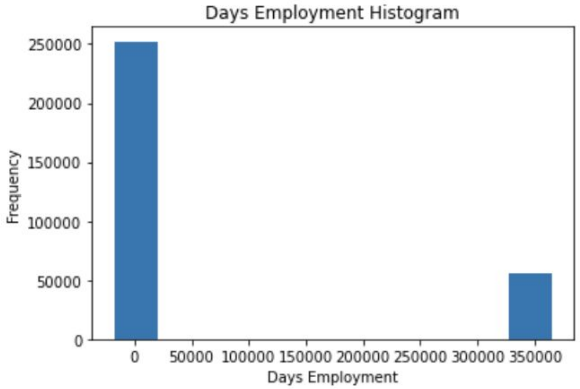
```
df['DAYS_EMPLOYED'].head()
```

```
0    -637
1   -1188
2    -225
3   -3039
4   -3038
Name: DAYS_EMPLOYED, dtype: int64
```

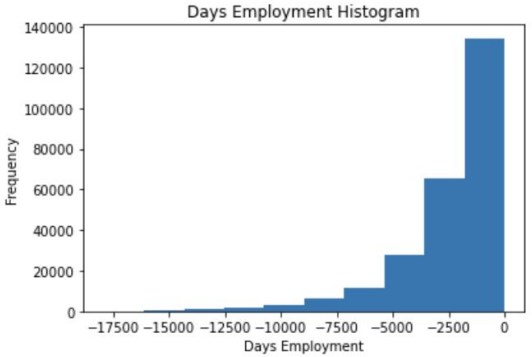
```
df['DAYS_EMPLOYED'].describe()
```

```
count    307511.000000
mean      63815.045904
std      141275.766519
min      -17912.000000
25%       -2760.000000
50%       -1213.000000
75%        -289.000000
max      365243.000000
Name: DAYS_EMPLOYED, dtype: float64
```

Untuk handle anomali set jadi "null" pada imputasi.



Sebaran data setelah anomali di takeout.



2.1.3. Amount Income Total

```
df['AMT_INCOME_TOTAL'].describe()
```

```
count    3.075110e+05
mean     1.687979e+05
std      2.371231e+05
min      2.565000e+04
25%      1.125000e+05
50%      1.471500e+05
75%      2.025000e+05
max      1.170000e+08
Name: AMT_INCOME_TOTAL, dtype: float64
```

```
df['AMT_INCOME_TOTAL'].head()
```

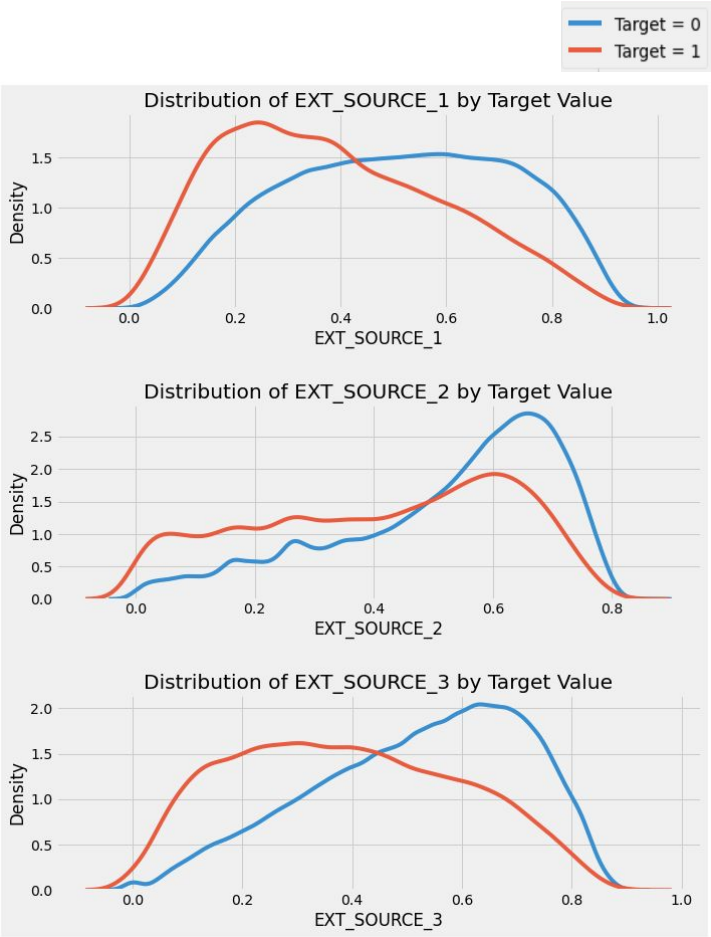
```
0    202500.0
1    270000.0
2     67500.0
3    135000.0
4    121500.0
Name: AMT_INCOME_TOTAL, dtype: float64
```

Tidak terdapat anomali, data normal.

2.1.4. Feature Exterior (1/2)

EXT_SOURCE adalah nilai kumulatif credit scoring yang terbentuk dari beberapa sumber data.

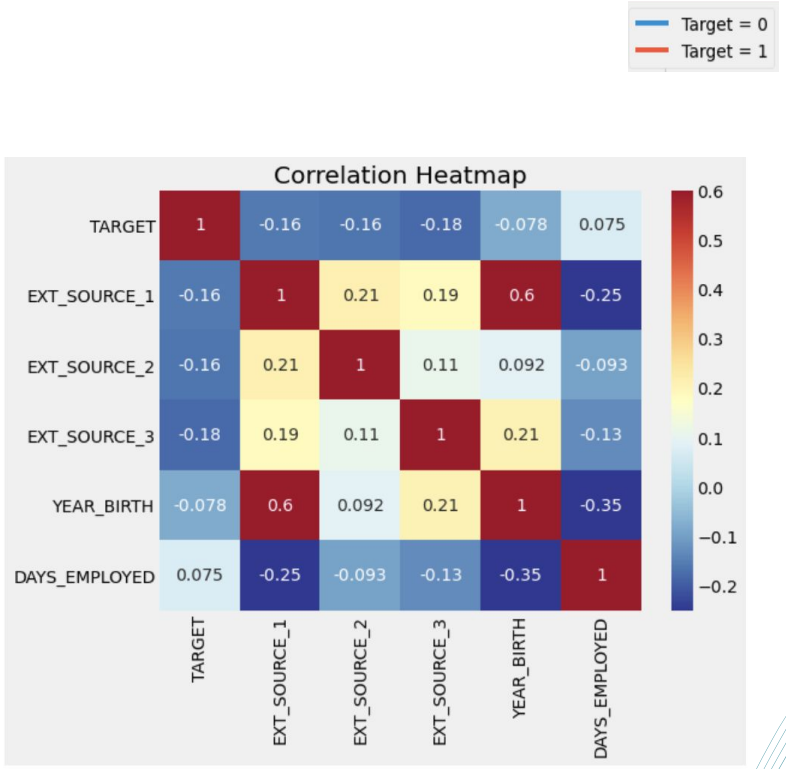
Ketiga fitur EXT_SOURCE memiliki korelasi negatif dengan target. *(semakin tinggi nilai EXT_SOURCE, semakin tinggi kemungkinan client akan membayar kredit)*



2.1.4. Feature Exterior (1/2)

EXT_SOURCE adalah nilai kumulatif credit scoring yang terbentuk dari beberapa sumber data.

Ketiga fitur EXT_SOURCE memiliki korelasi negatif dengan target. *(semakin tinggi nilai EXT_SOURCE, semakin tinggi kemungkinan client akan membayar kredit)*



2.1.5. Duplication, Inconsistency, Missing Values

1. Tidak ada duplikasi

```
df.duplicated().sum()
```

```
0
```

2. Terdapat value redundant pada _MEDI, _MODE terhadap _AVG

3. Dari 121 feature, terdapat 17 feature yang memiliki missing values >60%

```
df_persen.loc[df_persen >60]
```

```
OWN_CAR_AGE                 65.990810
YEARS_BUILD_AVG             66.497784
COMMONAREA_AVG              69.872297
FLOORSMIN_AVG               67.848630
LIVINGAPARTMENTS_AVG        68.354953
NONLIVINGAPARTMENTS_AVG     69.432963
YEARS_BUILD_MODE            66.497784
COMMONAREA_MODE             69.872297
FLOORSMIN_MODE              67.848630
LIVINGAPARTMENTS_MODE       68.354953
NONLIVINGAPARTMENTS_MODE    69.432963
YEARS_BUILD_MEDI            66.497784
COMMONAREA_MEDI             69.872297
FLOORSMIN_MEDI              67.848630
LIVINGAPARTMENTS_MEDI       68.354953
NONLIVINGAPARTMENTS_MEDI    69.432963
FONDKAPREMONT_MODE          68.386172
dtype: float64
```

03

Data Preprocessing

3.1. Handling Missing Values & Inconsistency

Drop kolom dengan missing value > 60%

```
cols = ['OWN_CAR_AGE','YEARS_BUILD_AVG','COMMONAREA_AVG','FLOORSMIN_AVG','LIVINGAPARTMENTS_AVG','NONLIVINGAPARTMENTS_AV
        'YEARS_BUILD_MODE','COMMONAREA_MODE','FLOORSMIN_MODE','FLOORSMIN_MODE','LIVINGAPARTMENTS_MODE','NONLIVINGAPARTME
        'YEARS_BUILD_MEDI','COMMONAREA_MEDI','FLOORSMIN_MEDI','LIVINGAPARTMENTS_MEDI','NONLIVINGAPARTMENTS_MEDI','FONDKA
```

```
df2 = df.drop(cols,axis=1)
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 106 entries, SK_ID_CURR to YEAR_BIRTH
dtypes: bool(1), float64(51), int64(39), object(15)
memory usage: 246.6+ MB
```

Hapus data dengan keterangan _MODE dan _MEDI karena redundant dengan _AVG

```
cols_medmod = ['APARTMENTS_MODE','BASEMENTAREA_MODE','YEARS_BEGINEXPLUATATION_MODE','ELEVATORS_MODE','ENTRANCES_MODE','
               'LANDAREA_MODE','LIVINGAREA_MODE','NONLIVINGAREA_MODE','APARTMENTS_MEDI','BASEMENTAREA_MEDI','YEARS_BEGI
               'ELEVATORS_MEDI','ENTRANCES_MEDI','FLOORSMAX_MEDI','LANDAREA_MEDI','LIVINGAREA_MEDI','NONLIVINGAREA_MEDI
```

```
df3 = df2.drop(cols_medmod,axis=1)
```

3.1. Handling Missing Values & Inconsistency

Imputasi Numerik: mengisi missing values dengan rata - rata

```
df3['AMT_ANNUITY'].fillna(df['AMT_ANNUITY'].mean(), inplace=True)
df3['AMT_GOODS_PRICE'].fillna(df['AMT_GOODS_PRICE'].mean(), inplace=True)
df3['CNT_FAM_MEMBERS'].fillna(df['CNT_FAM_MEMBERS'].mean(), inplace=True)
df3['CNT_FAM_MEMBERS'].fillna(df['CNT_FAM_MEMBERS'].mean(), inplace=True)
df3['EXT_SOURCE_1'].fillna(df['EXT_SOURCE_1'].mean(), inplace=True)
df3['EXT_SOURCE_2'].fillna(df['EXT_SOURCE_2'].mean(), inplace=True)
df3['EXT_SOURCE_3'].fillna(df['EXT_SOURCE_3'].mean(), inplace=True)
df3['APARTMENTS_AVG'].fillna(df['APARTMENTS_AVG'].mean(), inplace=True)
df3['BASEMENTAREA_AVG'].fillna(df['BASEMENTAREA_AVG'].mean(), inplace=True)
df3['YEARS_BEGINEXPLUATATION_AVG'].fillna(df['YEARS_BEGINEXPLUATATION_AVG'].mean(), inplace=True)
df3['ELEVATORS_AVG'].fillna(df['ELEVATORS_AVG'].mean(), inplace=True)
df3['ENTRANCES_AVG'].fillna(df['ENTRANCES_AVG'].mean(), inplace=True)
df3['FLOORSMAX_AVG'].fillna(df['FLOORSMAX_AVG'].mean(), inplace=True)
df3['LANDAREA_AVG'].fillna(df['LANDAREA_AVG'].mean(), inplace=True)
df3['LIVINGAREA_AVG'].fillna(df['LIVINGAREA_AVG'].mean(), inplace=True)
df3['NONLIVINGAREA_AVG'].fillna(df['NONLIVINGAREA_AVG'].mean(), inplace=True)
df3['OBS_30_CNT_SOCIAL_CIRCLE'].fillna(df['OBS_30_CNT_SOCIAL_CIRCLE'].mean(), inplace=True)
df3['DEF_30_CNT_SOCIAL_CIRCLE'].fillna(df['DEF_30_CNT_SOCIAL_CIRCLE'].mean(), inplace=True)
df3['OBS_60_CNT_SOCIAL_CIRCLE'].fillna(df['OBS_60_CNT_SOCIAL_CIRCLE'].mean(), inplace=True)
df3['DEF_60_CNT_SOCIAL_CIRCLE'].fillna(df['DEF_60_CNT_SOCIAL_CIRCLE'].mean(), inplace=True)
df3['DAYS_LAST_PHONE_CHANGE'].fillna(df['DAYS_LAST_PHONE_CHANGE'].mean(), inplace=True)
df3['AMT_REQ_CREDIT_BUREAU_HOUR'].fillna(df['AMT_REQ_CREDIT_BUREAU_HOUR'].mean(), inplace=True)
df3['AMT_REQ_CREDIT_BUREAU_DAY'].fillna(df['AMT_REQ_CREDIT_BUREAU_DAY'].mean(), inplace=True)
df3['AMT_REQ_CREDIT_BUREAU_WEEK'].fillna(df['AMT_REQ_CREDIT_BUREAU_WEEK'].mean(), inplace=True)
df3['AMT_REQ_CREDIT_BUREAU_MON'].fillna(df['AMT_REQ_CREDIT_BUREAU_MON'].mean(), inplace=True)
df3['AMT_REQ_CREDIT_BUREAU_QRT'].fillna(df['AMT_REQ_CREDIT_BUREAU_QRT'].mean(), inplace=True)
df3['AMT_REQ_CREDIT_BUREAU_YEAR'].fillna(df['AMT_REQ_CREDIT_BUREAU_YEAR'].mean(), inplace=True)
```

3.1. Handling Missing Values & Inconsistency

Imputasi Kategorikal: mengisi dengan "others"

```
df3.select_dtypes('object').info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 15 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   NAME_CONTRACT_TYPE           307511 non-null  object
 1   CODE_GENDER                  307511 non-null  object
 2   FLAG_OWN_CAR                 307511 non-null  object
 3   FLAG_OWN_REALTY              307511 non-null  object
 4   NAME_TYPE_SUITE              306219 non-null  object
 5   NAME_INCOME_TYPE             307511 non-null  object
 6   NAME_EDUCATION_TYPE          307511 non-null  object
 7   NAME_FAMILY_STATUS           307511 non-null  object
 8   NAME_HOUSING_TYPE            307511 non-null  object
 9   OCCUPATION_TYPE              211120 non-null  object
 10  WEEKDAY_APPR_PROCESS_START   307511 non-null  object
 11  ORGANIZATION_TYPE            307511 non-null  object
 12  HOUSETYPE_MODE               153214 non-null  object
 13  WALLSMATERIAL_MODE           151170 non-null  object
 14  EMERGENCYSTATE_MODE          161756 non-null  object
dtypes: object(15)
memory usage: 35.2+ MB
```

```
df3['NAME_TYPE_SUITE'].fillna('Others', inplace=True)
df3['OCCUPATION_TYPE'].fillna('Others', inplace=True)
df3['HOUSETYPE_MODE'].fillna('Others', inplace=True)
df3['WALLSMATERIAL_MODE'].fillna('Others', inplace=True)
df3['EMERGENCYSTATE_MODE'].fillna('Others', inplace=True)
```

3.2. Final Feature

DAYS_EMPLOYED	0.072360
REGION_RATING_CLIENT_W_CITY	0.063890
REGION_RATING_CLIENT	0.062404
REG_CITY_NOT_WORK_CITY	0.055244
DAYS_LAST_PHONE_CHANGE	0.054977
DAYS_ID_PUBLISH	0.049844
REG_CITY_NOT_LIVE_CITY	0.045482
FLAG_EMP_PHONE	0.045470
FLAG_DOCUMENT_3	0.040174
LIVE_CITY_NOT_WORK_CITY	0.038056
DEF_30_CNT_SOCIAL_CIRCLE	0.031639
DAYS_REGISTRATION	0.031207
DEF_60_CNT_SOCIAL_CIRCLE	0.027150
FLAG_WORK_PHONE	0.026316
CNT_CHILDREN	0.021099
AMT_REQ_CREDIT_BUREAU_YEAR	0.019101
AMT_INCOME_TOTAL	0.015718
CNT_FAM_MEMBERS	0.012400
FLAG_DOCUMENT_7	0.009161
REG_REGION_NOT_WORK_REGION	0.008321
OBS_60_CNT_SOCIAL_CIRCLE	0.006994
LIVE_REGION_NOT_WORK_REGION	0.006665
OBS_30_CNT_SOCIAL_CIRCLE	0.006561
FLAG_CONT_MOBILE	0.004066
YEARS_BEGINEXPLUATATION_AVG	0.003297
AMT_REQ_CREDIT_BUREAU_DAY	0.002108
REG_REGION_NOT_LIVE_REGION	0.002104
AMT_REQ_CREDIT_BUREAU_HOUR	0.000582

AMT_REQ_CREDIT_BUREAU_WEEK	-0.000687
FLAG_DOCUMENT_10	-0.001690
FLAG_DOCUMENT_2	-0.001690
AMT_REQ_CREDIT_BUREAU_QRT	-0.002223
FLAG_DOCUMENT_8	-0.002804
FLAG_DOCUMENT_4	-0.003380
FLAG_DOCUMENT_20	-0.004263
FLAG_DOCUMENT_5	-0.004581
FLAG_DOCUMENT_17	-0.004780
FLAG_DOCUMENT_21	-0.005070
FLAG_DOCUMENT_19	-0.007368
FLAG_DOCUMENT_15	-0.007381
FLAG_EMAIL	-0.007699
FLAG_DOCUMENT_14	-0.008778
BASEMENTAREA_AVG	-0.010543
FLAG_DOCUMENT_11	-0.010996
NONLIVINGAREA_AVG	-0.011355
FLAG_DOCUMENT_16	-0.011429
LANDAREA_AVG	-0.012507
FLAG_DOCUMENT_18	-0.012858
AMT_ANNUITY	-0.013666
FLAG_DOCUMENT_9	-0.016631
ENTRANCES_AVG	-0.016906
FLAG_DOCUMENT_13	-0.016970
AMT_REQ_CREDIT_BUREAU_MON	-0.018175
HOUR_APPR_PROCESS_START	-0.020828
FLAG_DOCUMENT_6	-0.022487
LIVINGAREA_AVG	-0.023101
ELEVATORS_AVG	-0.024932
APARTMENTS_AVG	-0.025324
FLOORSMAX_AVG	-0.029719
FLAG_PHONE	-0.029757
AMT_CREDIT	-0.032073
REGION_POPULATION_RELATIVE	-0.041233
AMT_GOODS_PRICE	-0.041700
YEAR_BIRTH	-0.077653
EXT_SOURCE_1	-0.099855
EXT_SOURCE_2	-0.147440
EXT_SOURCE_3	-0.158636

Menyisakan 67 feature yang akan digunakan untuk modelling

3.3. Sampling

Menggunakan stratified random sampling dengan 10% sample dari dataset.

```
df_sample = df3.groupby(['TARGET']).apply(lambda x: x.sample(frac=0.1,random_state=123))
```

```
df_sample
```

		SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTA
TARGET									
	242916	381209	0	Cash loans	F	N	Y	0	77850
	112259	230229	0	Cash loans	F	Y	Y	1	270000
0	**71881**	183353	0	Cash loans	F	N	Y	0	189000
	28988	133681	0	Cash loans	M	N	Y	0	157500
	185409	314926	0	Cash loans	F	N	Y	0	180000
...	...	...	...	...	...	...	...	...	
	287571	433111	1	Cash loans	M	Y	Y	1	270000
	219938	354794	1	Cash loans	F	Y	N	0	135000
1	**43267**	150080	1	Cash loans	F	N	N	1	90000
	262574	404014	1	Cash loans	F	N	N	2	112500
	408	100472	1	Cash loans	M	Y	Y	1	135000

30751 rows × 88 columns

```
df_sample['TARGET'].value_counts()
```

```
0    28269
1     2482
Name: TARGET, dtype: int64
```

3.4. Cek Outlier

Menggunakan Z-score

```
print(f'Jumlah baris sebelum memfilter outlier: {len(df_tescor3)}')

filtered_entries = np.array([True] * len(df_tescor3))
for col in df_tescor3:
    zscore = abs(stats.zscore(df_tescor3[col]))
    filtered_entries = (zscore < 3) & filtered_entries

df3_outlier = df_tescor3[filtered_entries]

print(f'Jumlah baris setelah memfilter outlier: {len(df_tescor3)}')
```

```
Jumlah baris sebelum memfilter outlier: 30751
Jumlah baris setelah memfilter outlier: 30751
```

Menggunakan IQR

```
print(f'Jumlah baris sebelum memfilter outlier: {len(df_tescor3)}')

filtered_entries = np.array([True] * len(df_tescor3))
for col in df_tescor3:
    Q1 = df_tescor3[col].quantile(0.25)
    Q3 = df_tescor3[col].quantile(0.75)
    IQR = Q3 - Q1
    low_limit = Q1 - (IQR * 1.5)
    high_limit = Q3 + (IQR * 1.5)

    filtered_entries = ((df_tescor3[col] >= low_limit) & (df_tescor3[col] <= high_limit)) & filtered_entries

df_tescor3 = df_tescor3[filtered_entries]

print(f'Jumlah baris setelah memfilter outlier: {len(df_tescor3)}')
```

```
Jumlah baris sebelum memfilter outlier: 30751
Jumlah baris setelah memfilter outlier: 621
```

3.4. Sandarization & Train Test Split

```
X = df_dummy.drop(['TARGET'],axis = 1)
y = df_dummy['TARGET']
```

Standarization

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaler = scaler.fit_transform(X)
```

```
X_scaler = pd.DataFrame(X_scaler,columns = X.columns)
```

```
# baseline
y.value_counts(normalize=True)*100
```

```
0    91.928718
1     8.071282
Name: TARGET, dtype: float64
```

Train-Test Split

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_scaler,y,test_size = 0.3,stratify = y,random_state = 123)
```

```
y_train.value_counts()
```

```
0    19788
1     1737
Name: TARGET, dtype: int64
```

3.5. Balancing Dataset

```
from imblearn.under_sampling import RandomUnderSampler
undersampling = RandomUnderSampler()
X_under, y_under = undersampling.fit_resample(X_train,y_train)
```

```
from imblearn.over_sampling import RandomOverSampler
oversampling = RandomOverSampler()
X_over, y_over = oversampling.fit_resample(X_train,y_train)
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_smote, y_smote = smote.fit_resample(X_train,y_train)
```

```
y_under.value_counts()
```

```
1    1737
0    1737
Name: TARGET, dtype: int64
```

```
y_over.value_counts()
```

```
1    19788
0    19788
Name: TARGET, dtype: int64
```

```
y_smote.value_counts()
```

```
1    19788
0    19788
Name: TARGET, dtype: int64
```

```
# new baseline
y_smote.value_counts(normalize=True)*100
```

```
1    50.0
0    50.0
Name: TARGET, dtype: float64
```

Undersampling menggunakan SMOTE

Why SMOTE?

Sample yang on time bayar lebih banyak dari yang telat bayar.

Kita ingin memprediksi orang yang cenderung untuk telat bayar. Sehingga yang telat bayar diperbanyak, yang on time diseimbangkan.

Dengan SMOTE, model akan mencondongkan applicant telat bayar → tujuannya agar bisa diproses lebih lanjut secara manual.

04

Modelling

Model Accuracy

K-Nearest Neighbor
Accuracy : 77.08649%

Decision Tree
Accuracy : 86.18102%

Logistic Regression
Accuracy : 71.38923%

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_smote,y_smote)
```

```
LogisticRegression()
```

```
y_pred = model.predict(X_smote)
y_pred
```

```
array([0, 0, 1, ..., 1, 0, 1], dtype=int64)
```

```
evaluasi(logreg,X_test,y_test)
```

```
69.27162367223065
```

```
params ={'C':[0.01,0.1,1,2,3,5]}
```

```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(
            estimator=logreg, # model yang akan digunakan
            param_grid=params, # hyperparameter yang dipilih
            scoring = 'accuracy', # metrics evaluation
            cv = 3 # 3-fold cross validation (artinya kita melakukan iterasi model sebanyak 3 kali)
           )
```

```
grid.fit(X_smote,y_smote)
```

```
GridSearchCV(cv=3, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1, 2, 3, 5]}, scoring='accuracy')
```

```
grid.best_params_
```

```
{'C': 1}
```

```
grid.best_score_
```

```
0.7138922579341015
```

```
print('accuracy : {0:.5f}%'.format(grid.best_score_*100))
```

```
accuracy : 71.38923%
```

1. Logistic Regression

2. KNN

Modeling k-Nearest Neighbor

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_smote,y_smote)
```

```
KNeighborsClassifier()
```

```
accuracy = evaluasi(knn,X_test,y_test)
```

```
print('accuracy : {0:.5f}%'.format(accuracy))
```

```
accuracy : 61.22914%
```

Tuning Hyperparameter

```
params = {'n_neighbors':[1,2,3,4,5]}
```

```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(
             estimator=knn,
             param_grid=params,
             scoring = 'accuracy',
             cv = 3 # 3-fold cross validation (artinya kita melakukan iterasi model sebanyak 3 kali)
            )
```

```
grid.fit(X_smote,y_smote)
```

```
GridSearchCV(cv=3, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': [1, 2, 3, 4, 5]}, scoring='accuracy')
```

```
grid.best_params_
```

```
{'n_neighbors': 2}
```

```
accuracy = evaluasi(grid,X_test,y_test)
```

```
print('accuracy : {0:.5f}%'.format(accuracy))
```

```
accuracy : 77.72599%
```

3. Decision Tree

Modeling Decicion Tree

```
from sklearn.tree import DecisionTreeClassifier
model_tree = DecisionTreeClassifier(max_depth=4)
```

```
model_tree.fit(X_smote,y_smote)
```

```
DecisionTreeClassifier(max_depth=4)
```

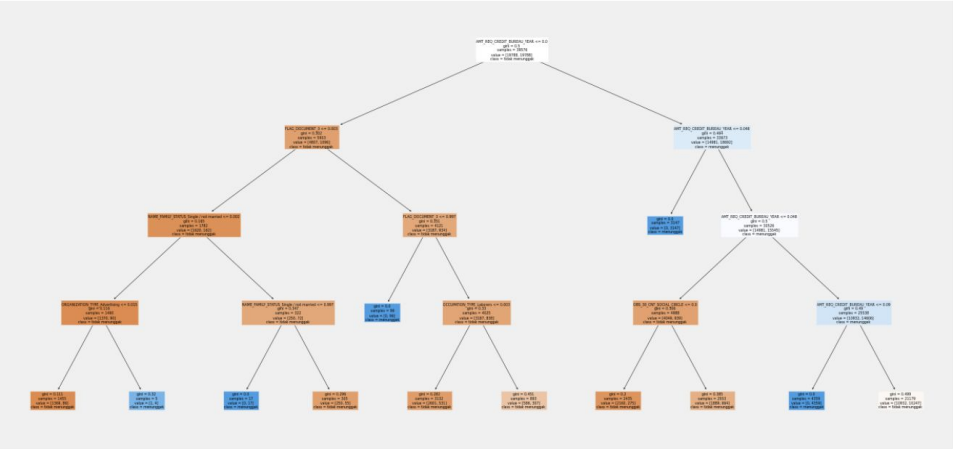
```
accuracy = evaluasi(model_tree,X_test,y_test)
```

```
print('accuracy : {0:.5f}%'.format(accuracy))
```

```
accuracy : 91.92499%
```

```
from sklearn import tree
```

```
plt.figure(figsize = (20,10))
tree.plot_tree(model_tree,
               feature_names = X_train.columns,
               class_names = ['tidak menunggak','menunggak'],filled=True)
plt.show()
```



```
params ={'max_depth':[1,2,3,4,5,6,7,8,'max']}
```

```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(
            estimator=model_tree, # model yang akan digunakan
            param_grid=params, # hyperparameter yang dipilih
            scoring = 'accuracy', # metrics evaluation
            cv = 3 # 3-fold cross validation (artinya kita melakukan iterasi model sebanyak 3 kali)
           )
```

```
grid.fit(X_smote,y_smote)
```

```
GridSearchCV(cv=3, estimator=DecisionTreeClassifier(max_depth=4),
             param_grid={'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 'max']},
             scoring='accuracy')
```

```
grid.best_params_
```

```
{'max_depth': 8}
```

```
grid.best_score_
```

```
0.8616080452799677
```

```
print('accuracy : {0:.5f}%'.format(grid.best_score_*100))
```

```
accuracy : 86.16080%
```

05

Deployment

Flowchart Deployment