The Revolutionaries

Model Klasifikasi untuk Memprediksi Loan Repayment Pelanggan di Home Credit Group

Khikmafaris Y.
Supervisor

Jasmine
Facilitator

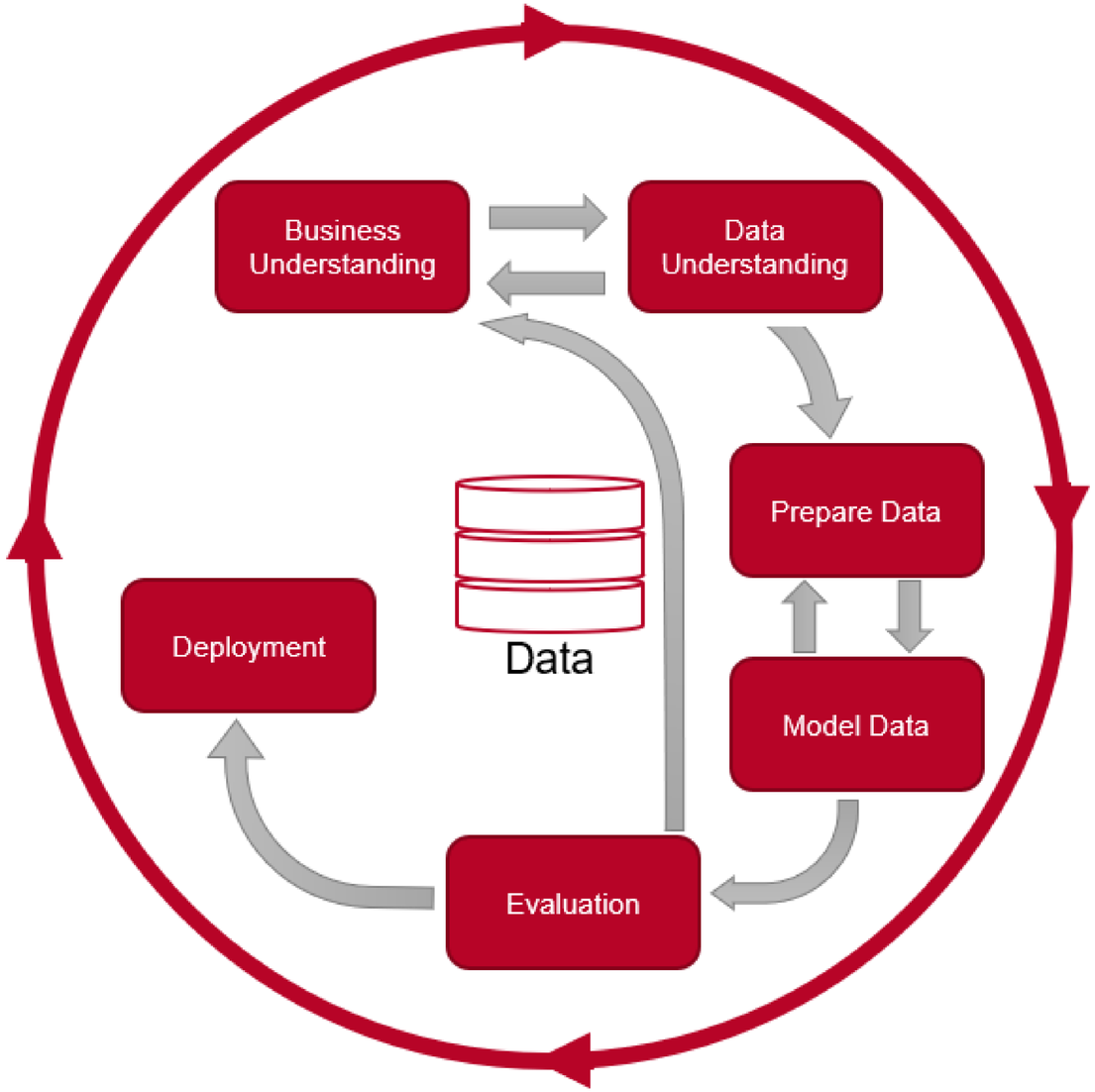
Suprianto Dharma
Institut Teknologi Del

Sri Indrayani
Universitas Brawijaya

Winona Timothea
Institut Teknologi Bandung

Zalfa Nurjihan
Universitas Pendidikan Indonesia

Workflow during Project



Cross-Industry Standard Process for Data Mining

Business Understanding

Home Credit Group ingin memperluas penawaran ke lebih banyak pelanggan dengan tujuan untuk meningkatkan pendapatannya. Namun, semakin banyak pinjaman yang ditawarkan perusahaan, semakin besar pula resiko yang terjadi. Jika terjadi gagal bayar di antara pelanggan meningkat, maka perusahaan akan mengalami kerugian dalam ekspansinya.

Goal

Meminimalkan jumlah pemohon yang pinjaman kreditnya disetujui tetapi sebenarnya tidak mampu membayar kredit.

Objective

Membuat model machine learning untuk memprediksi dan mengklasifikasikan apakah pemohon dapat membayar pinjaman atau tidak, dengan menggunakan data historis aplikasi pinjaman.

Dataset

Dataset	Deskripsi Singkat
application_train	Informasi tentang data statis untuk semua pemohon pinjaman (train samples).
bureau	Semua kredit klien sebelumnya yang dilaporkan ke Credit Bureau.
previous_application	Semua aplikasi sebelumnya untuk pinjaman klien di Home Credit.
POS_CASH_balance	Saldo bulanan dari POS sebelumnya dan pinjaman tunai yang dimiliki pemohon.
credit_card_balance	Saldo bulanan kartu kredit sebelumnya yang dimiliki pemohon dengan Home Credit.
installments_payments	Riwayat kredit yang dicairkan sebelumnya di Home Credit terkait dengan pinjaman.

Feature Selection

```
[ ]  import pandas as pd
     from sklearn.feature_selection import mutual_info_classif

     # Check the number of samples
     print("Number of samples in X:", len(X))
     print("Number of samples in y:", len(y))

     # Ensure X and y have the same number of samples
     min_samples = min(len(X), len(y))
     X = X[:min_samples]
     y = y[:min_samples]

     # calculate mutual info score
     mi = mutual_info_classif(X, y)

     # create a dataframe to store the scores
     df_mi = pd.DataFrame({'Feature': X.columns, 'Mutual_Info_Score': mi}
```

```
Number of samples in X: 265151
Number of samples in y: 307511
```

Dilakukan feature selection secara manual dan dengan menggunakan **mutual information** untuk setiap dataset.

```
print(df_mi)
```

	Feature	Mutual_Info_Score
0	SK_ID_CURR	0.000000
1	EXT_SOURCE_2	0.000609
2	EXT_SOURCE_3	0.000298
3	EXT_SOURCE_1	0.011859
4	DAYS_BIRTH	0.000242
5	REGION_RATING_CLIENT_W_CITY	0.042164
6	REGION_RATING_CLIENT_x	0.021680
7	NAME_EDUCATION_TYPE	0.031118
8	CODE_GENDER	0.016082
9	DAYS_EMPLOYED	0.004844
10	NAME_INCOME_TYPE	0.022985
11	FLAG_DOCUMENT_3	0.048580
12	DAYS_REGISTRATION	0.000000
13	NAME_HOUSING_TYPE	0.044766
14	DEF_60_CNT_SOCIAL_CIRCLE	0.001376
15	NAME_CONTRACT_TYPE_x	0.001379
16	AMT_CREDIT_x	0.000592
17	TOTALAREA_MODE	0.007849
18	AMT_INCOME_TOTAL	0.001460
19	AMT_CREDIT_MAX_OVERDUE	0.000000
20	AMT_CREDIT_SUM	0.000000
21	AMT_CREDIT_SUM_DEBT	0.000000
22	SK_DPD_DEF_x	0.000294
23	SK_DPD	0.000000
24	AMT_BALANCE	0.000299
25	AMT_CREDIT_LIMIT_ACTUAL	0.003991

Feature Engineering: data aggregation

Dilakukan data merging untuk menggabungkan semua dataset, dengan jumlah baris dari merged data yang selalu sama yaitu 307.511 baris.

Total fitur yang digunakan adalah 41 fitur prediktor dan 1 fitur target.

```
[ ]  # Define Variable for Merge Dataset (Train and Bureau Dataset)
     df1 = pd.merge(left = feature_train, right = fitur_bureau, how = 'left', left_on = 'SK_ID_CURR', right_on = 'SK_ID_CURR')
     df1.shape

     (307511, 32)
```

```
[ ]  # Define Variable for Merge Data (df1 and Pos Cash Dataset)
     df2 = pd.merge(left = df1, right = fitur_poshcash, how = 'left', left_on = 'SK_ID_CURR', right_on = 'SK_ID_CURR')
     df2.shape

     (307511, 36)
```

```
[ ]  feature_final.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 307511 entries, 0 to 307510
     Data columns (total 42 columns):
```

Feature Engineering: create new feature

Pada fitur FLAG_DOCUMENT di lakukan feature engineering untuk membuat kolom baru dan semua kolom flag document disatukan menjadi satu kolom.

```
# CREATE NEW COLUMN : FLAG_DOCS
list_col_new_flagDoc = [
    'FLAG_DOCUMENT_2',
    'FLAG_DOCUMENT_3',
    'FLAG_DOCUMENT_4',
    'FLAG_DOCUMENT_5',
    'FLAG_DOCUMENT_6',
    'FLAG_DOCUMENT_7',
    'FLAG_DOCUMENT_8',
    'FLAG_DOCUMENT_9',
    'FLAG_DOCUMENT_10',
    'FLAG_DOCUMENT_11',
    'FLAG_DOCUMENT_12',
    'FLAG_DOCUMENT_13',
    'FLAG_DOCUMENT_14',
    'FLAG_DOCUMENT_15',
    'FLAG_DOCUMENT_16',
    'FLAG_DOCUMENT_17',
    'FLAG_DOCUMENT_18',
    'FLAG_DOCUMENT_19',
    'FLAG_DOCUMENT_20',
    'FLAG_DOCUMENT_21']
```

```
[ ]  def create_sum_cols(df,ls_cols,newcol):
         df[newcol] = np.nan
         ls_sum_value = df[ls_cols].sum(axis=1)
         df[newcol] = ls_sum_value
         return df
```

```
[ ]  # Create kolom baru dengan nama kolom FLAG_DOCS
     create_sum_cols(data_train,list_col_new_flagDoc,'FLAG_DOCS');
```

	FLAG_DOCS
0	1
1	1
2	0
3	1
4	1

Feature Engineering: create new feature

Dilakukan feature engineering terhadap fitur FLAG_OWN_CAR dan FLAG_OWN_REALTY.

```
[ ]  # CREATE NEW COLUMN : 0 - none, 1 - with car no realty, 2 - no car with realty, 3 - with car with realty
     list_col_new_asset = ['FLAG_OWN_CAR','FLAG_OWN_REALTY']
```

```
[ ]  def create_asset(df):
         df['FLAG_ASSET'] = np.nan
         filter_0 = (df.FLAG_OWN_CAR=='N')&(df.FLAG_OWN_REALTY=='N')
         filter_1 = (df.FLAG_OWN_CAR=='Y')&(df.FLAG_OWN_REALTY=='N')
         filter_2 = (df.FLAG_OWN_CAR=='N')&(df.FLAG_OWN_REALTY=='Y')
         filter_3 = (df.FLAG_OWN_CAR=='Y')&(df.FLAG_OWN_REALTY=='Y')

         df['FLAG_ASSET'][filter_0] = 0
         df['FLAG_ASSET'][filter_1] = 1
         df['FLAG_ASSET'][filter_2] = 2
         df['FLAG_ASSET'][filter_3] = 3
         return df


     create_asset(data_train);
```

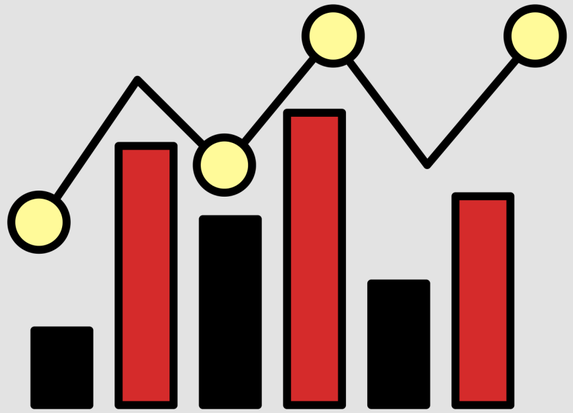
```
[ ]  data_train[['FLAG_OWN_CAR','FLAG_OWN_REALTY','FLAG_ASSET']].head()
```

	FLAG_OWN_CAR	FLAG_OWN_REALTY	FLAG_ASSET
0	N	Y	2.0
1	N	N	0.0
2	Y	Y	3.0
3	N	Y	2.0
4	N	Y	2.0

Feature Engineering: create new feature

Fitur Days_Birth yang semula menghitung usia klien dalam satuan hari dikonversi ke dalam bentuk tahun.

Kemudian dibuat kategori berdasarkan usia yaitu 'child', 'young', 'adult', and 'old'.



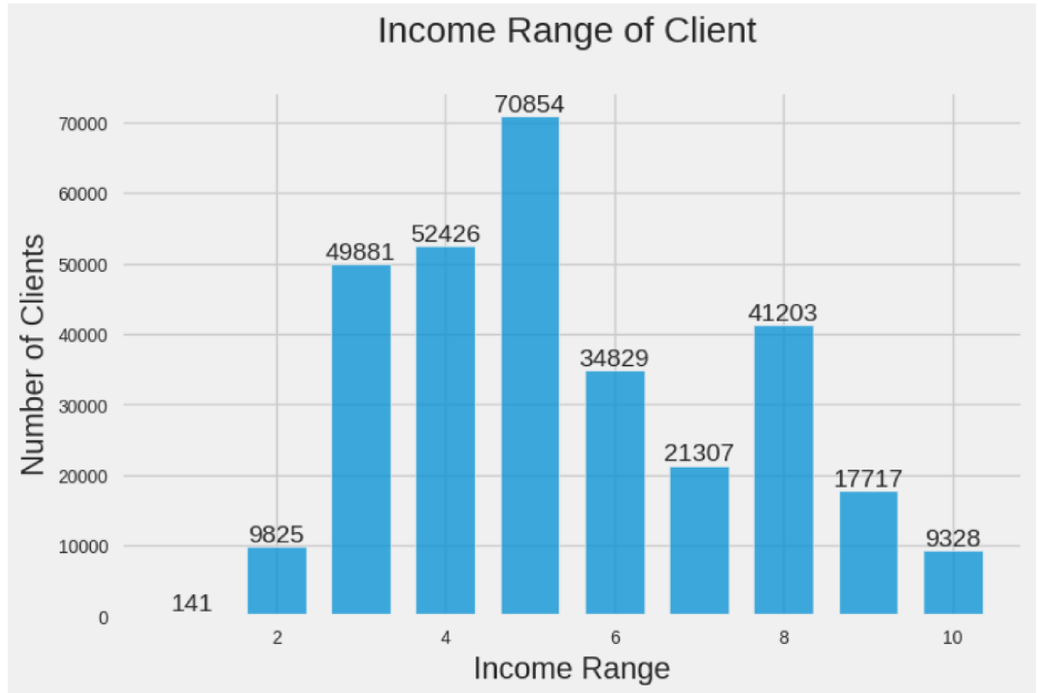
```python
# Konversi Days_Birth ke dalam bentuk tahun
def konversi_hari_ke_tahun(hari):
    tahun = hari / -365  # atau 365.25
    return tahun

# Membuat kolom baru dalam tahun
feature_final['AGE'] = feature_final['DAYS_BIRTH'].apply(konversi_hari_ke_tahun)
```

```python
[ ]  #Create categorical column 'age_status' to make it model easier to process
     def age_status(x):
         if (x >= 0) & (x<=16):
             return 'child'
         elif (x>=17) & (x<=30):
             return 'young'
         elif (x>=31) & (x<=45):
             return 'adult'
         else:
             return 'old'
```

Feature Engineering: binning data

Dilakukan proses pengelompokan data pada fitur AMT_INCOME_TOTAL.

```
[ ]  # Menentukan batas binning
     bins_total = [0, 30000, 65000, 95000, 130000, 160000, 190000, 220000, 275000, 375000, feature_final['AMT_INCOME_TOTAL'].max()]
     labels_total = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

     # Tambahkan kolom baru dengan nilai binning AMT_INCOME_TOTAL
     feature_final['INCOME_BINS'] = pd.cut(feature_final['AMT_INCOME_TOTAL'], bins=bins_total, labels=labels_total)
```



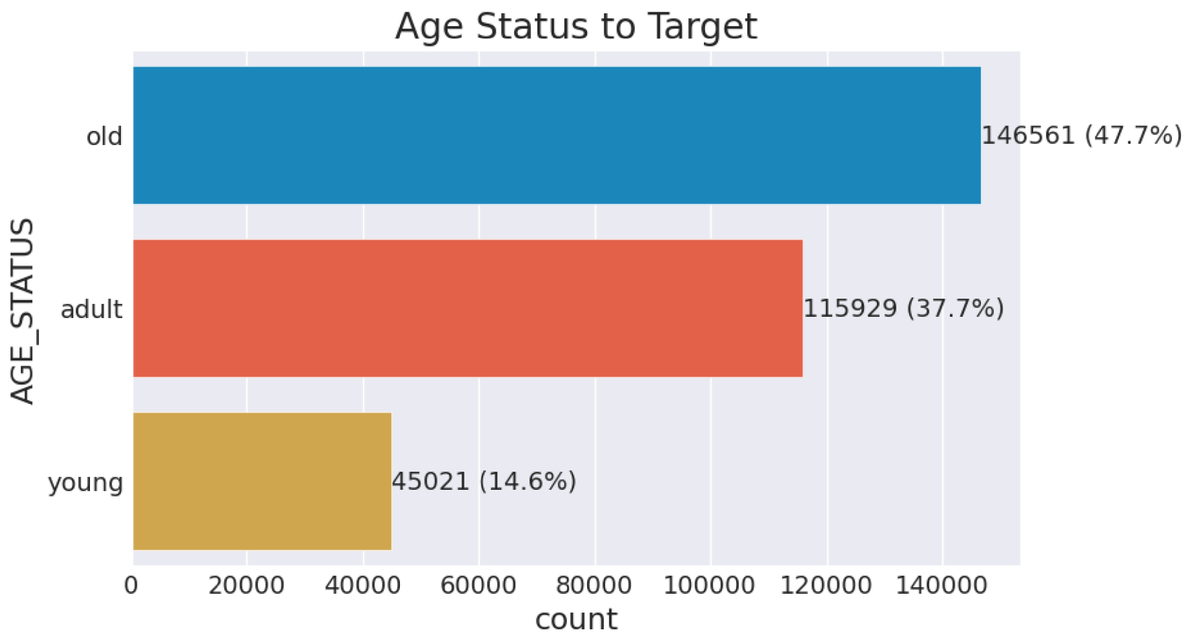
Exploratory Data Analysis

[Active]	[Active, Completed]
337034	337208

[Active, Completed] tanpa [Signed]	[Active, Signed]
337208	337077

Insight:
terdapat jumlah pelanggan yang masih aktif dan Signed (status kontrak yang telah ditandatangani oleh peminjam dan lembaga keuangan yang memberikan kredit) sehingga ini dapat menjadi pertimbangan untuk perusahaan dalam memberi pinjaman ke masing-masing individu.

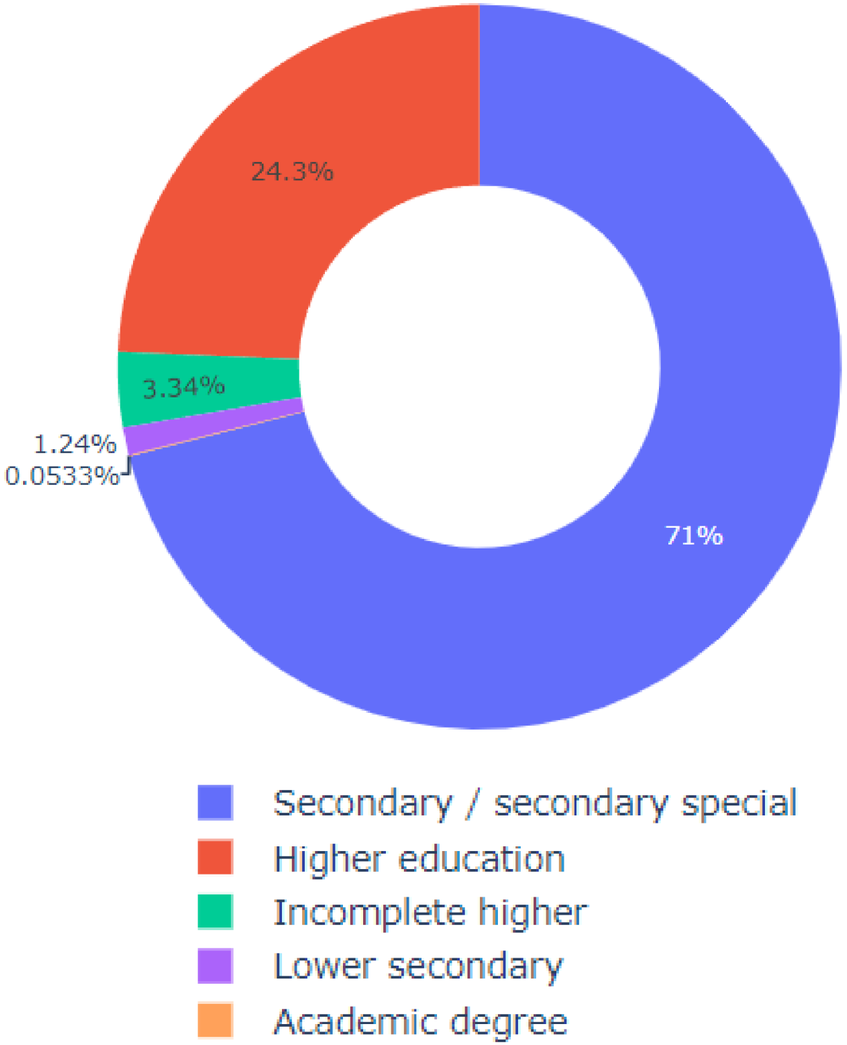
Exploratory Data Analysis



Insight:

Persentase tertinggi adalah old (>45 tahun) sebanyak 14.6561 customer atau 47,7%. Dari persentase tersebut, sebesar 44,6% merupakan customer dengan label 0 dan 3,1% berlabel 1.

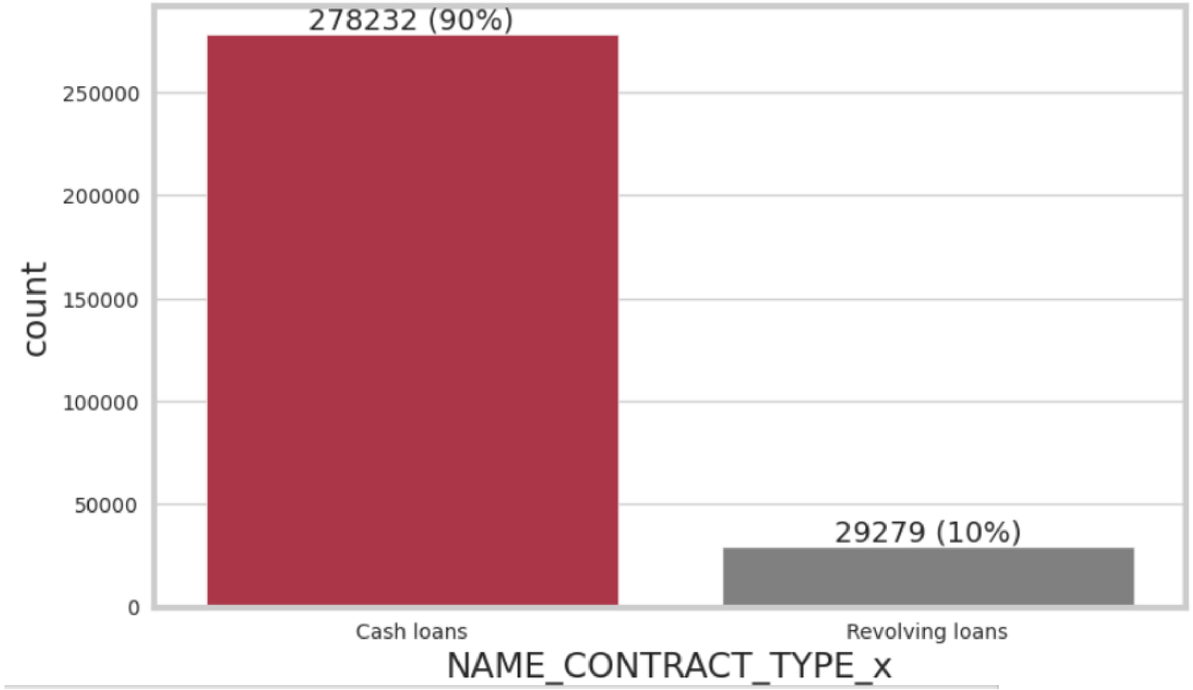
Exploratory Data Analysis



Insight:

71% pemohon pinjaman berpendidikan SMP dan 24,3% berpendidikan SMA.

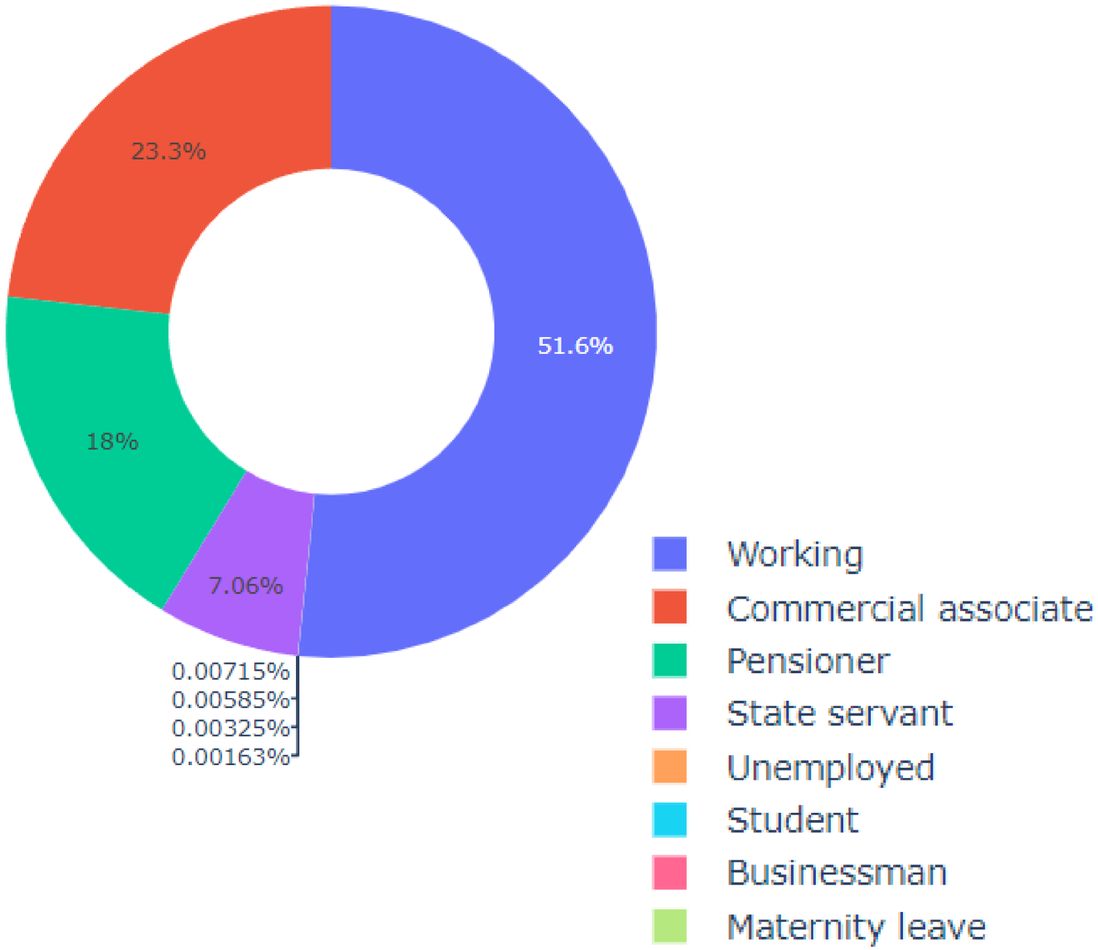
Exploratory Data Analysis



Insight:

Mayoritas pemohon pinjaman lebih memilih cash loans dibandingkan revolving loans.

Exploratory Data Analysis



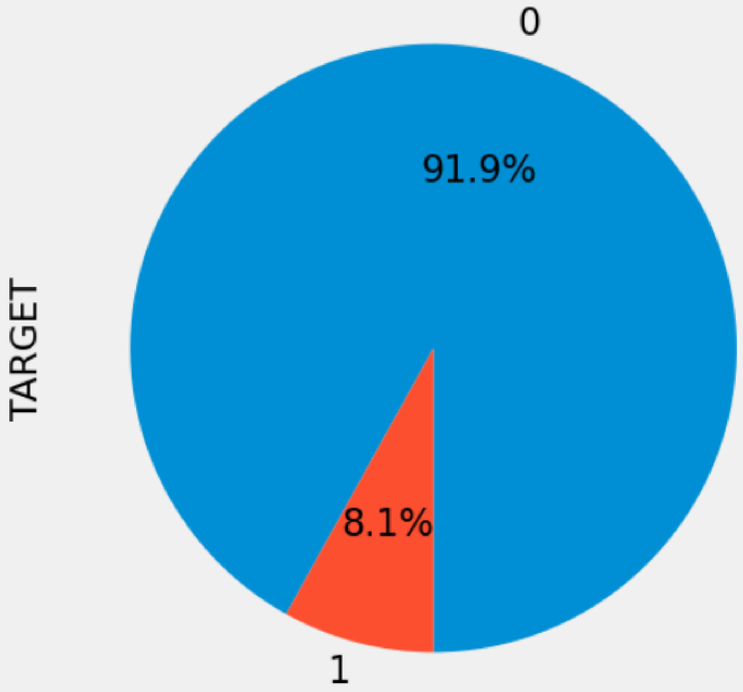
Insight:

Peminjam yang bekerja memiliki persentase tertinggi yaitu 51,6% diikuti dengan peminjam dengan jenis pendapatan bisnis sebesar 23,3% dan pensiunan sebesar 18%.

Modeling Imbalanced Data

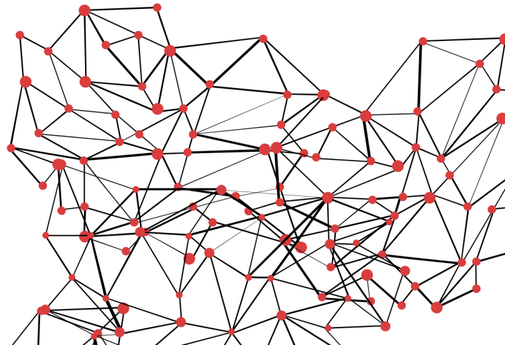
Model	Precision	Recall	F1-Score
Logisctic Regression	0,828070	0,831258	0,828905
Gradient Boosting	0,899490	0,904343	0,899639
Random Forest	0,878195	0,881871	0,879227
LightGBM	0,879488	0,884098	0,879566
Decission Tree	0,852971	0.856008	0,853945

Balancing Dataset



Dari grafik terlihat bahwa istribusi data target tidak seimbang. Ketidakseimbangan kelas ini dapat mempengaruhi model saat pelatihan.

Untuk mengatasi masalah ini, dilakukan LabelEncoder karena Catboost akan bekerja efektif dengan kolom kategorikal. Kemudian, dilakukan penyeimbangan data dengan KMeansSmotr.



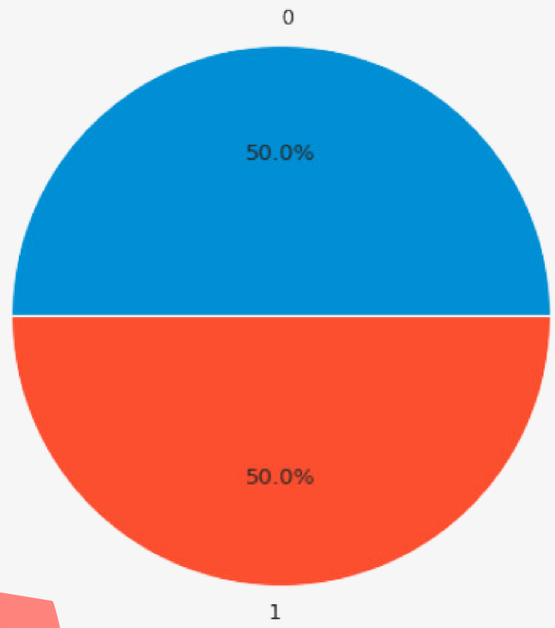
Balancing Dataset

```
[ ]  # over sampling
     from imblearn.over_sampling import SMOTEN
```

```
[ ]  # As there is imbalance in the classes so to overcome this issue using oversampling through KMeansSMOTE to generate synthetic samples
     cc = SMOTEN(random_state = 42)
     X_smote, y_smote = cc.fit_resample(X, y)

     # Randomly pick a point from the minority class.
     # Compute the k-nearest neighbors (for some pre-specified k) for this point.
     # Add k new points somewhere between the chosen point and each of its neighbors

     #Splitting the data
     X_train_smote, X_test_smote, y_train_smote, y_test_smote = train_test_split(X_smote, y_smote, test_size=0.2, random_state=1, stratify=y_smote)
```

```
[ ]  #After oversampling
     unique_values, counts = np.unique(y_smote, return_counts=True)

     # Menampilkan nilai unik dan frekuensinya
     for value, count in zip(unique_values, counts):
         print("Nilai:", value, "- Jumlah:", count)
```

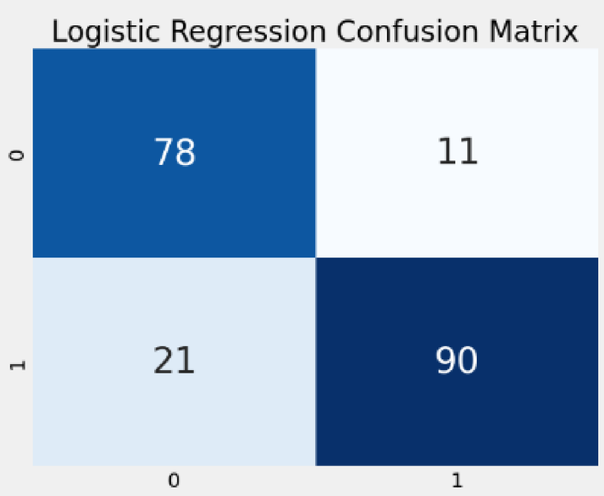
```
Nilai: 0 - Jumlah: 501
Nilai: 1 - Jumlah: 501
```

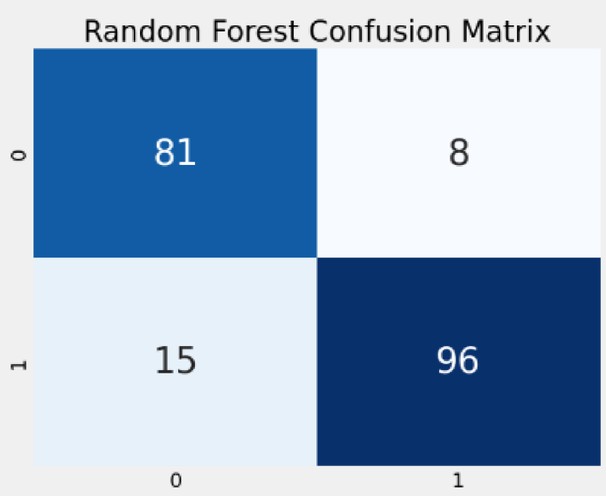


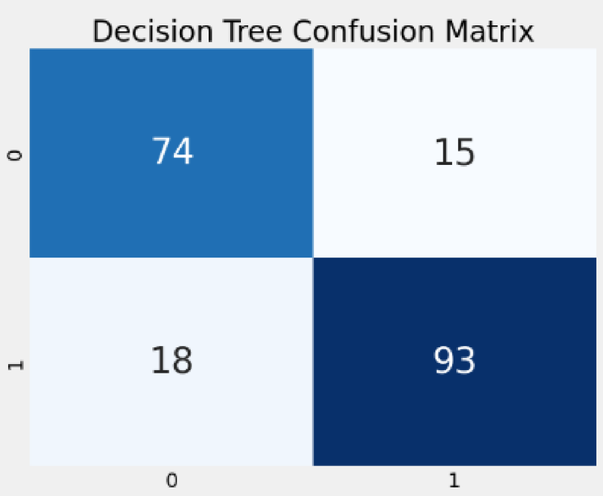
Modeling Balanced Data

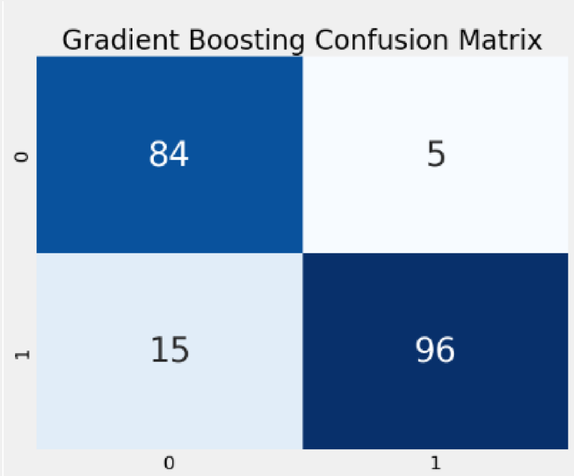
Model	Precision	Recall	F1-Score
Logisctic Regression	0,849462	0,782178	0,814433
Gradient Boosting	0,955556	0,851485	0,900524
Random Forest	0,916667	0,871287	0,893401
LightGBM	0,936170	0,871287	0,902564
Decission Tree	0,887755	0,861386	0,874372

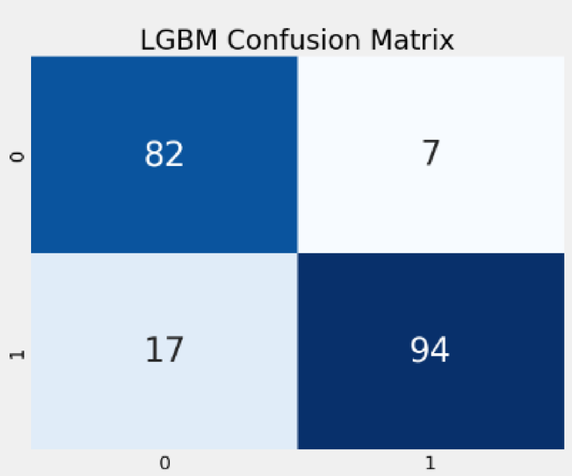
Model Evaluation Metrics



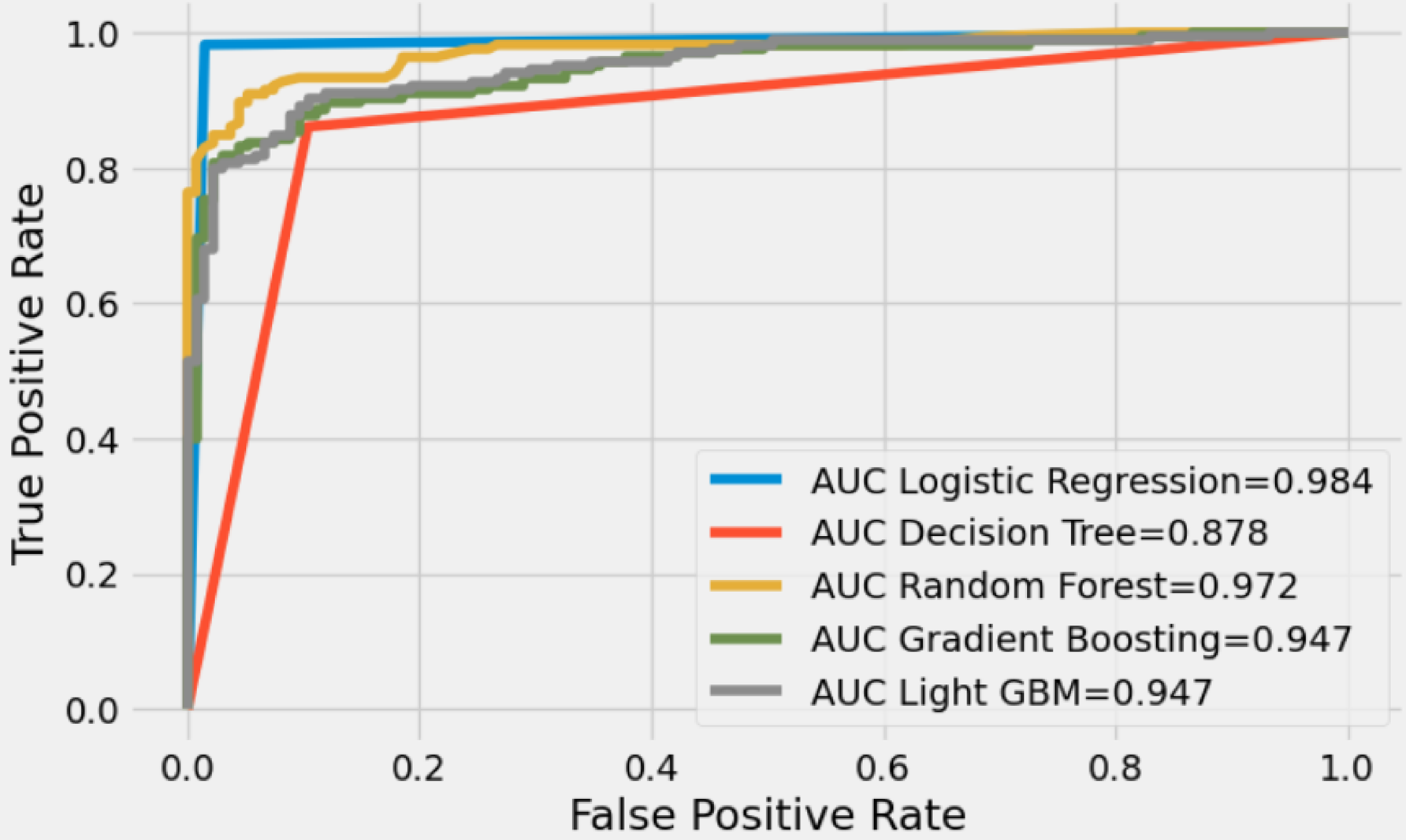








ROC-AUC Curve



Threshold Validation

```python
from sklearn.metrics import confusion_matrix

def find_rates_for_thresholds(y_test, y_pred, thresholds):
    fpr_list = []
    tpr_list = []
    for threshold in thresholds:
        y_pred_binary = (y_pred > threshold).astype(int)
        tn, fp, fn, tp = confusion_matrix(y_test, y_pred_binary).ravel()
        fpr = fp / (fp + tn)
        tpr = tp / (tp + fn)
        fpr_list.append(fpr)
        tpr_list.append(tpr)
    return fpr_list, tpr_list

thresholds = np.arange(0, 1.1, 0.1)

# fpr_logreg, tpr_logreg = find_rates_for_thresholds(y_test, y_pred_logreg, thresholds
fpr_rf, tpr_rf = find_rates_for_thresholds(y_test, proba_rf, thresholds)
fpr_gb, tpr_gb = find_rates_for_thresholds(y_test, proba_lgbm, thresholds)
fpr_lg, tpr_lg = find_rates_for_thresholds(y_test, y_pred_proba_lr, thresholds)
# fpr_svc, tpr_svc = find_rates_for_thresholds(y_test, y_pred_svc, thresholds)

summary_df = pd.DataFrame({
    'Threshold': thresholds,
    'FPR_RF': fpr_rf,
    'TPR_RF': tpr_rf,
    'FPR_Light GBM': fpr_gb,
    'TPR_Light GBM' : tpr_gb,
    'FPR, LG' : fpr_lg,
    'TPR_LG' : tpr_lg,

})

print(summary_df)
```

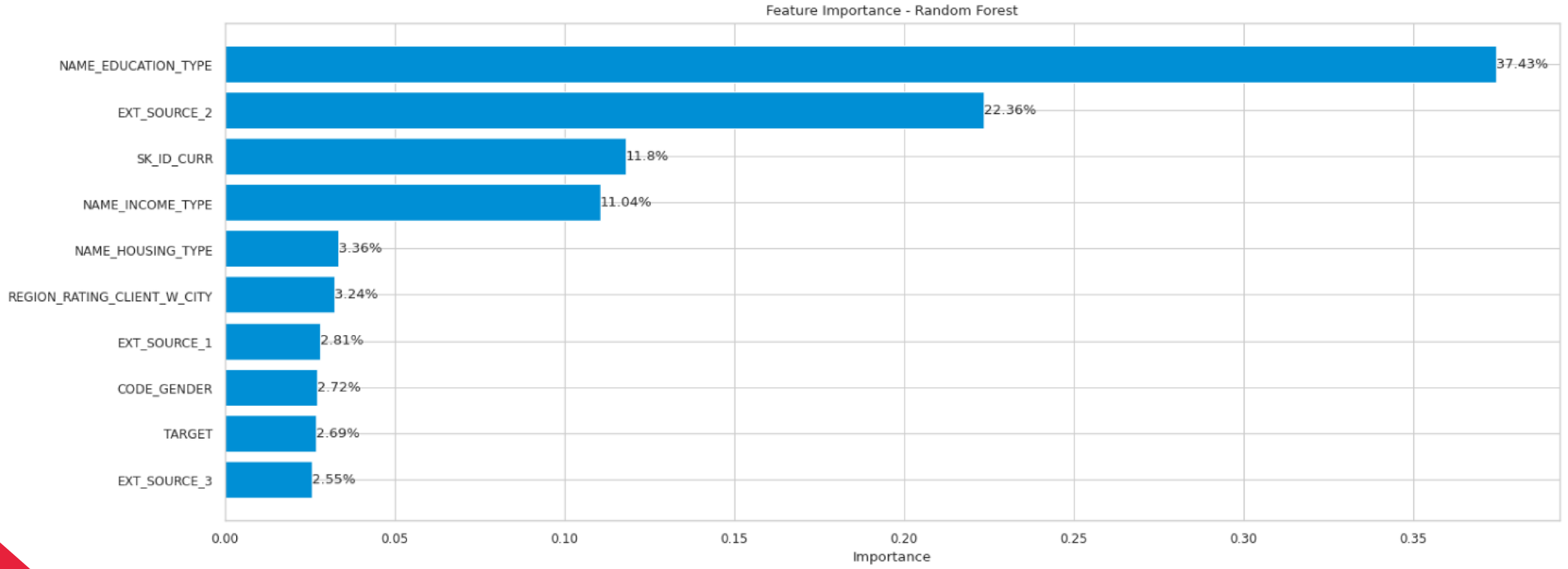
```
    Threshold    FPR_RF    TPR_RF  FPR_Light GBM  TPR_Light GBM   FPR, LG  \
0         0.0  0.932584  1.000000       1.000000       1.000000  0.011236
1         0.1  0.382022  0.963964       0.168539       0.927928  0.011236
2         0.2  0.269663  0.936937       0.123596       0.909910  0.011236
3         0.3  0.157303  0.900901       0.089888       0.891892  0.011236
4         0.4  0.112360  0.882883       0.078652       0.882883  0.011236
5         0.5  0.089888  0.864865       0.078652       0.846847  0.011236
6         0.6  0.067416  0.837838       0.067416       0.837838  0.011236
7         0.7  0.044944  0.810811       0.044944       0.828829  0.011236
8         0.8  0.022472  0.729730       0.022472       0.801802  0.011236
9         0.9  0.000000  0.594595       0.022472       0.792793  0.011236
10        1.0  0.000000  0.000000       0.000000       0.000000  0.000000

      TPR_LG
0   0.981982
1   0.981982
2   0.981982
3   0.981982
4   0.981982
5   0.981982
6   0.981982
7   0.981982
8   0.981982
9   0.981982
10  0.000000
```

Model Selection

Jadi dapat disimpulkan, bahwa model yang peformanya lebih bagus ialah model **Random Forest** dengan ditandai dengan model Random Forest yang memiliki memiliki recall, precission, dan F1-Score yang lebih baik.

Berdasarkan uji ROC-AUC dan setelah divalidasi dengan treshold random forest tetap memiliki kinerja yang lebih baik dibandingkan model lain.

Feature Importance



Suggestion

- Perusahaan Home Credit Group dapat memberikan penawaran baru berupa paket pinjaman pelajar dengan bunga yang rendah.
- Fokus pada segmen adult dan old karena berdasarkan data historis kedua rentang usia ini berpotensi tinggi churn.
- Jika perusahaan ingin menyaring peminjam mereka harus lebih memperhatikan bidang-bidang seperti jenjang pendidikan, sumber pendapatan, dan tempat tinggal.

Thank You