FINAL TASK

"Score Card Model & Loan Default Prediction"

Home Credit - Data Scientist

Presented by : Farras Fadhilah

PROBLEM STATEMENT

Home Credit is currently using various statistical and machine learning methods to make credit score predictions. So the data from this company has the maximum potential. By doing so, we can ensure that customers who are capable of making repayments are not rejected when applying for a loan, and loans can be made with a principal, maturity, and repayment calendar that will motivate customers to succeed.

Dataset

Contains loan information and the loan applicant at the time of application.

https://rakamin-lms.s3.ap-southeast-1.amazonaws.com/vix-assets/home-credit-indonesia/home-credit-default-risk.zip

Tools

Google Colaboratory

DATA COLLECTION & UNDERSTANDING

Based on the dataset provided by the company, I used two files: **application_train** and **application_test.**

- The training dataset contains 307,511 loan records with 122 features, including the target label.
- The testing dataset contains 48,744 loan records with 121 features, excluding the target label.

The target column (**TARGET**) is a binary variable indicating loan repayment status:

- **0** = Loan repaid
- **1** = Loan default

Other columns include numerical (float, integer) and categorical features that serve as predictors. Additionally, there are identifier columns with unique values that are not used for modeling.

	count
float64	65
int64	41
object	16

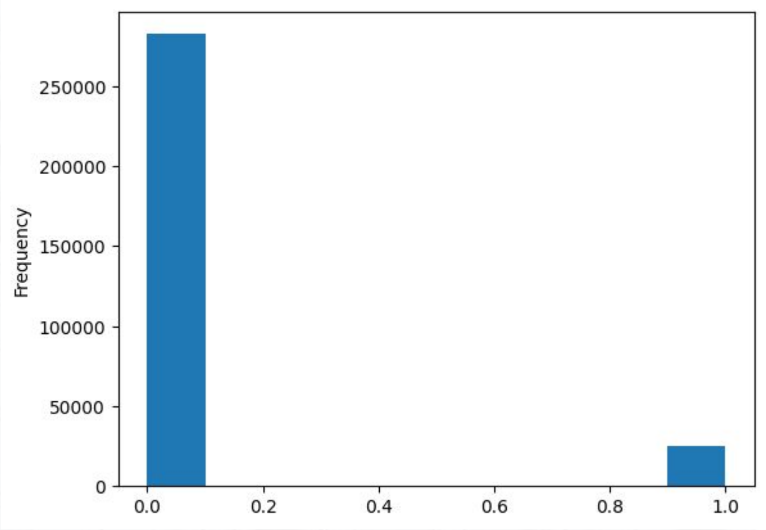
	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL
0	100002	1	Cash loans	M	N	Y	0	202500.0
1	100003	0	Cash loans	F	N	N	0	270000.0
2	100004	0	Revolving loans	M	Y	Y	0	67500.0
3	100006	0	Cash loans	F	N	Y	0	135000.0
4	100007	0	Cash loans	M	N	Y	0	121500.0

GOALS

- Build a classification model that predicts the credit repayment behavior of a customer.
- Help minimize loan defaults and support safe lending decisions.
- Provide insights and recommendations to improve credit approval policies.

METRICS

The metric that i used to evaluate the model performance is **ROC-AUC**. This metric was chosen because it evaluates how well the model separates two classes: customers who repay their loans versus those who default, also because the dataset has **imbalance class**. Unlike simple accuracy, ROC AUC is not affected by class imbalance and provides a fair measure of how reliably the model distinguishes between good and risky borrowers.



EXPLORATORY DATA ANALYSIS (EDA)

Structure and Types :

```
app_train = pd.read_csv('application_train.csv')
print("shape:", app_train.shape)

shape: (307511, 122)
```

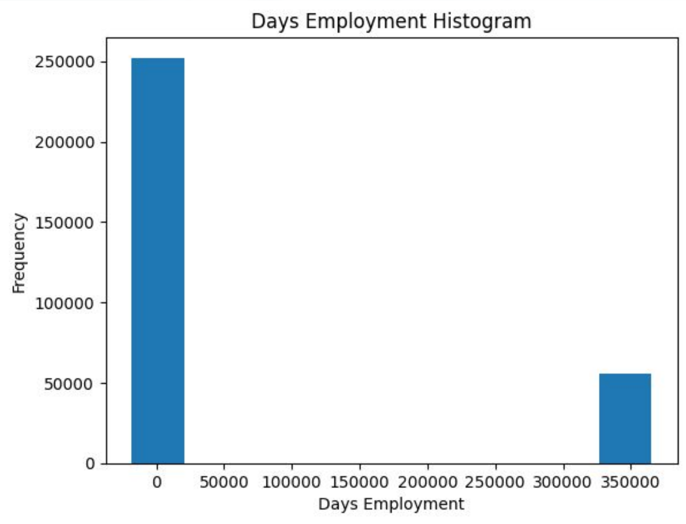
	count
float64	65
int64	41
object	16

Identify Missing Values :

```
Your selected dataframe has 122 columns.
There are 67 columns that have missing values.
```

	Missing Values	% of Total Values
COMMONAREA_MEDI	214865	69.9
COMMONAREA_MODE	214865	69.9
COMMONAREA_AVG	214865	69.9
NONLIVINGAPARTMENTS_MODE	213514	69.4
NONLIVINGAPARTMENTS_MEDI	213514	69.4
NONLIVINGAPARTMENTS_AVG	213514	69.4
FONDKAPREMONT_MODE	210295	68.4
LIVINGAPARTMENTS_AVG	210199	68.4
LIVINGAPARTMENTS_MEDI	210199	68.4
LIVINGAPARTMENTS_MODE	210199	68.4

Univariate Analysis:

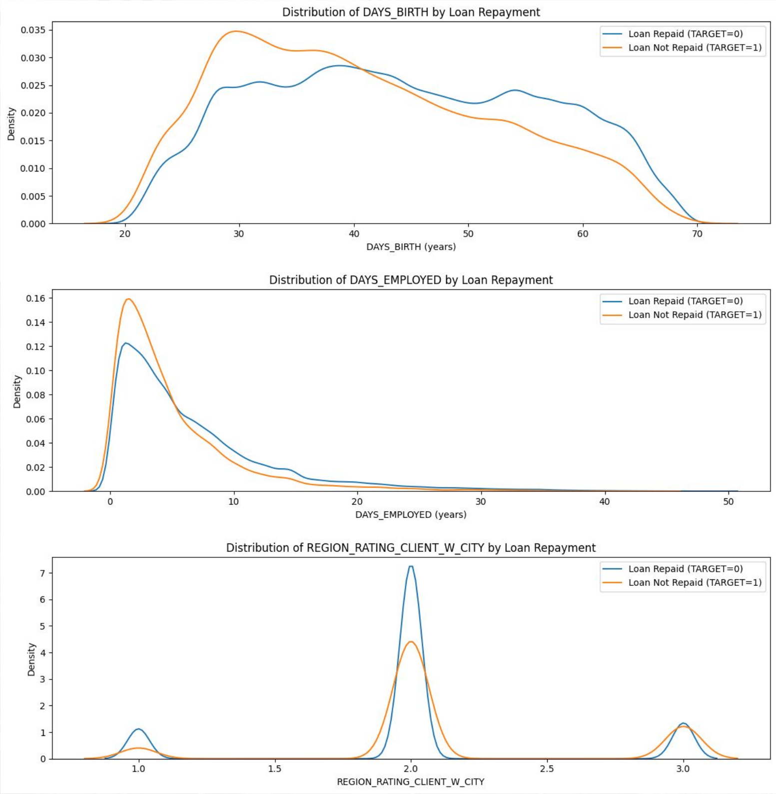


	DAYS_EMPLOYED
count	307511.000000
mean	63815.045904
std	141275.766519
min	-17912.000000
25%	-2760.000000
50%	-1213.000000
75%	-289.000000
max	365243.000000

Based on the distribution of days employment, the max value looks weird, because it's positive and if we converted to Years (/365) it is around 1000 years.

EXPLORATORY DATA ANALYSIS (EDA)

Bivariate Analysis :



Correlation :

```
EXT_SOURCE_3                             -0.178919
EXT_SOURCE_2                             -0.160472
EXT_SOURCE_1                             -0.155317
NAME_EDUCATION_TYPE_Higher education     -0.056593
CODE_GENDER_F                            -0.054704
NAME_INCOME_TYPE_Pensioner               -0.046209
ORGANIZATION_TYPE_XNA                    -0.045987
FLOORSMAX_AVG                            -0.044003
FLOORSMAX_MEDI                           -0.043768
FLOORSMAX_MODE                           -0.043226
Name: TARGET, dtype: float64
REG_CITY_NOT_WORK_CITY          0.050994
DAYS_ID_PUBLISH                 0.051457
CODE_GENDER_M                   0.054713
DAYS_LAST_PHONE_CHANGE          0.055218
NAME_INCOME_TYPE_Working        0.057481
REGION_RATING_CLIENT            0.058899
REGION_RATING_CLIENT_W_CITY     0.060893
DAYS_EMPLOYED                   0.074958
DAYS_BIRTH                      0.078239
TARGET                          1.000000
Name: TARGET, dtype: float64
```

Explanation :

- Values close to +1 or -1 mean a strong linear relationship.
- Values close to 0 mean weak or no linear relationship.
- Negative correlation → higher values = less chance of default (safer borrower)
- Positive correlation → higher values = more chance of default

DATA PREPROCESSING

- Data Cleaning: Removed duplicates and handled missing values

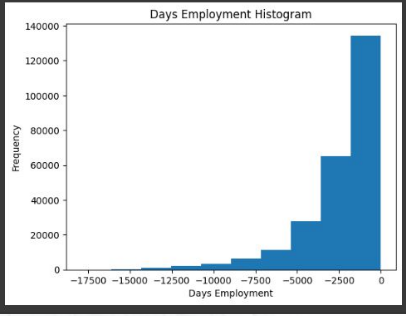
```
# Features (X) and target (y)
X = app_train.drop(columns=['TARGET'])
y = app_train['TARGET']

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Median imputation (Handling missing values)
imputer = SimpleImputer(strategy='median')
x_train = imputer.fit_transform(x_train)
x_test = imputer.transform(x_test)
```

- Data Encoding: Converted categorical features into numerical format

```
[ ] # One-hot Encoding
    app_train = pd.get_dummies(app_train)
    app_test = pd.get_dummies(app_test)

    print("Training Features shape:", app_train.shape)
    print("Testing Features shape:", app_test.shape)

    Training Features shape: (307511, 242)
    Testing Features shape: (48744, 238)
```

- Outlier Treatment: Detected and managed unusual/extreme values

```
[ ] # NaN values for DAYS_EMPLOYED: 365.243 -> nan
    app_train['DAYS_EMPLOYED'].replace(365243, np.nan, inplace= True)

    app_train['DAYS_EMPLOYED'].plot.hist(title = 'Days Employment Histogram');
    plt.xlabel('Days Employment');
```



- Feature Engineering: Created and transformed features to improve model performance

  - `CREDIT_INCOME_PERCENT`: the percentage of the credit amount relative to a client's income
  - `ANNUITY_INCOME_PERCENT`: the percentage of the loan annuity relative to a client's income
  - `CREDIT_TERM`: the length of the payment in months (since the annuity is the monthly amount due
  - `DAYS_EMPLOYED_PERCENT`: the percentage of the days employed relative to the client's age

```
[ ] app_train_new = app_train.copy()
    app_test_new = app_test.copy()

[ ] # Training Data New features (percentages)
    app_train_new['DAYS_EMPLOYED_PERC'] = app_train_new['DAYS_EMPLOYED'] / app_train_new['DAYS_BIRTH']
    app_train_new['INCOME_CREDIT_PERC'] = app_train_new['AMT_INCOME_TOTAL'] / app_train_new['AMT_CREDIT']
    app_train_new['INCOME_PER_PERSON'] = app_train_new['AMT_INCOME_TOTAL'] / app_train_new['CNT_FAM_MEMBERS']
    app_train_new['ANNUITY_INCOME_PERC'] = app_train_new['AMT_ANNUITY'] / app_train_new['AMT_INCOME_TOTAL']
    app_train_new['PAYMENT_RATE'] = app_train_new['AMT_ANNUITY'] / app_train_new['AMT_CREDIT']
```

MODEL DEVELOPMENT & EVALUATION

The selection of the best learning method (ML Algorithm) is necessary. There is no universal model that is suitable for all data and purposes, so we need to try several possible algorithms to obtain the best performance (evaluation). Hyperparameter tuning is also carried out to obtain the best performance from the selected model.

```python
# Features (X) and target (y)
X = app_train.drop(columns=['TARGET'])
y = app_train['TARGET']

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Median imputation (Handling missing values)
imputer = SimpleImputer(strategy='median')
x_train = imputer.fit_transform(x_train)
x_test = imputer.transform(x_test)

# Scaling
scaler = MinMaxScaler(feature_range=(0, 1))
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

print('Training shape:', x_train.shape)
print('Validation shape:', x_test.shape)

### Logistic Regression Model (Training)
log_reg = LogisticRegression(class_weight="balanced", C = 0.0001)
log_reg.fit(x_train, y_train)
```

Show hidden output

```python
# Predict probabilities
log_reg_test_pred = log_reg.predict_proba(x_test)[:, 1]

# ROC AUC
test_auc = roc_auc_score(y_test, log_reg_test_pred)
print("Logistic Regression ROC AUC (Test):", test_auc)
```

```
Logistic Regression ROC AUC (Test): 0.7158422637214258
```

```python
### RF MODEL ###
# Features (X) and target (y)
X = app_train.drop(columns=['TARGET'])
y = app_train['TARGET']

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Median imputation
imputer = SimpleImputer(strategy='median')
x_train = imputer.fit_transform(x_train)
x_test = imputer.transform(x_test)

# Train Random Forest
rf_model = RandomForestClassifier(
    n_estimators=200,
    max_depth=10,
    class_weight="balanced",
    random_state=42,
    n_jobs=-1
)
rf_model.fit(x_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(class_weight='balanced', max_depth=10, n_estimators=200,
                       n_jobs=-1, random_state=42)
```

```python
# Predict probabilities
rf_test_pred = rf_model.predict_proba(x_test)[:, 1]

# ROC AUC
test_auc = roc_auc_score(y_test, rf_test_pred)
print("Random Forest ROC AUC (Test):", test_auc)
```

```
Random Forest ROC AUC (Test): 0.731138794756629
```

```python
# Features (X) and target (y)
X = app_train.drop(columns=['TARGET']).copy()
y = app_train['TARGET']

# Replace spaces and special characters with underscores
X.columns = X.columns.str.replace('[^A-Za-z0-9_]+', '_', regex=True)

# Train-test split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# LightGBM Model
lgb_model = lgb.LGBMClassifier(
    n_estimators=500,
    learning_rate=0.05,
    num_leaves=31,
    random_state=42,
    n_jobs=-1
)

lgb_model.fit(x_train, y_train)
```

Show hidden output

```python
# Prediction Probability
lgb_test_pred = lgb_model.predict_proba(x_test)[:, 1]

# ROC AUC
test_auc = roc_auc_score(y_test, lgb_test_pred)
print("LightGBM ROC AUC (Test):", test_auc)
```

```
LightGBM ROC AUC (Test): 0.7599026497047834
```

Logistic Regression	Random Forest	LightGBM
ROC AUC = 0.715	ROC AUC = 0.731	ROC AUC = 0.759

BUSINESS INSIGHT & RECOMMENDATION

Insights	Recommendations
Younger clients, especially under 30 years old, have **higher default rates**, while older clients tend to repay more reliably.	For young clients, require a co-signer or limit the loan amount, while offering safer loan products and better conditions for older clients.
Clients with short or **unstable employment history** show **higher default risk**, while long-term employed clients are more stable.	Ask clients with short employment history for extra proof of income stability, and reward long-term employed clients with better loan terms.
Clients with **high credit amounts** compared to their **income** have a higher chance of **default**.	Apply a maximum credit-to-income ratio before approving loans, and flag or decline applications where the credit request is too high compared to income.
Clients with a history of **multiple previous loans** often show patterns: those who repay **on time** can become **loyal**, profitable customers, while those with many **past defaults** carry higher risk.	Offer better terms or larger credit limits to repeat customers with good repayment history. Also apply stricter approval rules or smaller loans to customers with a record of past defaults.

Thank You

Rakamin Academy X HOME CREDIT Kamu Bisa!