DTSA5509 FINAL PROJECT

CREDIT SCORING FOR CREDIT CARD APPLICATION

AGENDA

Problem and Solution

Model Development Process

Result and Conclusion

GitHub: https://github.com/Toon6115/Introduction-to-Machine-Learning-Supervised-Learning

PROBLEM AND SOLUTION

Problems:

- Bank needs a tool to access the customer creditworthiness to prevent the loss from the person who cannot reply in the future.
- In banking industry especially for the retail portfolio like credit card, the bank will facing the large amount of application. This cannot be reviewed one-by-one by credit analyst.

Solution:

Credit scoring is a tool that can help back process the credit assessment quickly based on customer application/behavior data.

In this final project, I will perform credit scoring for credit card application as a supervised learning problem. The objective to help the bank to classify credit card customer card customers based on their likelihood of default. This will help the bank to access creditworthiness from the large volume of the applicants, help to prevent and reduce potential loss from high risk customer.

THE DATASET CONSISTS OF 2 CSV FILES. THERE'RE TWO TABLES COULD BE MERGED BY ID.

Application_record, it is the customer information which contains.

- ID: Client identification number
- CODE_GEN: Client gender
- FLAG_OWN_CAR: Is there a car?
- FLAG_OWN_REALTY: Is there a property?
- CNT_CHILDREN: Number of children
- AMT_INCOME_TOTAL: Total annual income
- NAME_INCOME_TYPE: Income category
- NAME_EDUCATION_TYPE: Education level of the client
- NAME_FAMILY_STATUS: Marital status
- NAME_HOUSING_TYPE: Type of living
- DAYS_BIRTH: Count backwards from current day (0), -1 means yesterday
- DAYS_EMPLOYED: Count backwards from current day(0). If positive, it means the person currently unemployed.
- FLAG_MOBIL: Is there a mobile phone?
- FLAG_WORK_PHONE: Is there a work phone?
- FLAG_PHONE: Is there a phone?
- FLAG_EMAIL: Is there any email?
- OCCUPATION_TYPE: Occupation of the client
- CNT_FAM_MEMBERS: Size of the Family

THE DATASET CONSISTS OF 2 CSV FILES. THERE'RE TWO TABLES COULD BE MERGED BY ID.

Credit_records, It is credit performance information of the customer based on day past due. The table contains;

- ID: Client identification number
- MONTH_BALANCE: The month of the extracted data is the starting point, backwards, 0 is the current month, -1 is the previous month, and so on
- STATUS: 0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overdue 5: Overdue or bad debts, write-offs for more than 150 days C: paid off that month X: No loan for the month

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C

HERE IS MY OUTLINE OF THE PROCESS THAT I PLAN TO DEVELOP IN THE NOTEBOOK.

Step 0: Load required libraries

Step 1: Load Data and Exploratory Data Analysis (EDA)

Step 2: Perform Factor Analysis, Transformation and Reduction

Step 3: Model Development

Step 4: Logistic Regression Evaluate the model -

Step 5: Develop the challenger Model

Step 6: Random Forest model Evaluation

Step 7: Discussion/Conclusion: -

STEP 0: LOAD REQUIRED LIBRARIES

Step 0: Load required libaries

```
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.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, roc_curve, roc_curve, accuracy_score, auc
import statsmodels.api as sm
import scipy.stats as stats
import scorecardpy as sc
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.1 Load the Application data and perform basic checking

1.1 Load the Application data and perform basic checking

[2]:
```
## Load and Exploration Application Data
application_record = pd.read_csv('application_record.csv')
application_record.head()
```

[2]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil
1	5008805	M	Y	Y	0	427500.0	Working	Higher education	Civil
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not

[3]:
```
## Check the number of record
application_record.shape
```

[3]: (438557, 18)

[4]:
```
## Check basic info of the Application Data
application_record.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 438557 entries, 0 to 438556
Data columns (total 18 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   ID                   438557 non-null  int64
 1   CODE_GENDER          438557 non-null  object
 2   FLAG_OWN_CAR         438557 non-null  object
 3   FLAG_OWN_REALTY      438557 non-null  object
 4   CNT_CHILDREN         438557 non-null  int64
 5   AMT_INCOME_TOTAL     438557 non-null  float64
 6   NAME_INCOME_TYPE     438557 non-null  object
 7   NAME_EDUCATION_TYPE  438557 non-null  object
 8   NAME_FAMILY_STATUS   438557 non-null  object
 9   NAME_HOUSING_TYPE    438557 non-null  object
 10  DAYS_BIRTH           438557 non-null  int64
 11  DAYS_EMPLOYED        438557 non-null  int64
 12  FLAG_MOBIL           438557 non-null  int64
 13  FLAG_WORK_PHONE      438557 non-null  int64
 14  FLAG_PHONE           438557 non-null  int64
 15  FLAG_EMAIL           438557 non-null  int64
 16  OCCUPATION_TYPE      304354 non-null  object
 17  CNT_FAM_MEMBERS      438557 non-null  int64
dtypes: float64(1), int64(9), object(8)
memory usage: 60.2+ MB
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.2 Remove dupplicate record of the Application Data

1.2 Remove dupplicate record of the Application Data

Based on the nature of application record, it should be unique because it refers to the inividual's customer information based on thier application. So, in this step we will check weather the data contains the duplication and then we will remove it.

```
[5]: ## Check uniqueness for the ID column and remove dupplicate record of the Application Data
     application_record['ID'].duplicated().sum()
```

```
[5]: 47
```

```
[6]: application_record = application_record.drop_duplicates(subset='ID',keep='first')
```

```
[7]: ## RE-Check the number of record
     application_record.shape
```

```
[7]: (438510, 18)
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.3 Load the credit_card data and perform basic checking

1.3 Load the credit_card data and perform basic checking

[8]:
```
## Load and Exploration Credit Record Data
credit_record = pd.read_csv('credit_record.csv')
credit_record.head()
```

[8]:

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C

[9]:
```
# Check for Credit Record Table Data Size
credit_record.shape
```

[9]: (1048575, 3)

[10]:
```
## Check basic info of the Application Data
credit_record.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 3 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   ID              1048575 non-null  int64
 1   MONTHS_BALANCE  1048575 non-null  int64
 2   STATUS          1048575 non-null  object
dtypes: int64(2), object(1)
memory usage: 24.0+ MB
```

[11]:
```
## Check the data duplication
credit_record.duplicated().sum()
```

[11]: 0

[12]:
```
## Check number of ID Uniquesness
credit_record['ID'].nunique()
```

[12]: 45985

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.4 Transform Target data based on defualt definetion

1.4 Transform Target data based on defualt definetion

Tranform credit record based on defualt definetion. I would define the default defination based on normal banking practice here;If the customer is past due more than 90 days (3 months delinquent). Please do note that if it's already paid off or on loan at the month. It will be considered as good.

Refer to data dicctionary of the STATUS:

- 0: 1-29 days past due
- 1: 30-59 days past due
- 2: 60-89 days overdue
- 3: 90-119 days overdue
- 4: 120-149 days overdue
- 5: Overdue or bad debts, write-offs for more than 150 days
- C: paid off that month
- X: No loan for the month.

Therfore, I would tranform STATUS - 0,1,2,C and X as a performing loan (0), and 4-5 as non-performing loan (1).

[13]:
```
credit_record['target']=credit_record['STATUS']
credit_record['target'].replace('1', 0, inplace=True)
credit_record['target'].replace('2', 0, inplace=True)
credit_record['target'].replace('X', 0, inplace=True)
credit_record['target'].replace('C', 0, inplace=True)
credit_record['target']=credit_record['target'].astype(int)
credit_record.loc[credit_record['target']>=1,'target']=1
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.4 Transform Target data based on default definition

1.4 Transform Target data based on defualt definetion

Tranform credit record based on defualt definetion. I would define the default defination based on normal banking practice here;If the customer is past due more than 90 days (3 months delinquent). Please do note that if it's already paid off or on loan at the month. It will be considered as good.

Refer to data dicctionary of the STATUS:

- 0: 1-29 days past due
- 1: 30-59 days past due
- 2: 60-89 days overdue
- 3: 90-119 days overdue
- 4: 120-149 days overdue
- 5: Overdue or bad debts, write-offs for more than 150 days
- C: paid off that month
- X: No loan for the month.

Therfore, I would tranform STATUS - 0,1,2,C and X as a performing loan (0), and 4-5 as non-performing loan (1).

[13]:
```
credit_record['target']=credit_record['STATUS']
credit_record['target'].replace('1', 0, inplace=True)
credit_record['target'].replace('2', 0, inplace=True)
credit_record['target'].replace('X', 0, inplace=True)
credit_record['target'].replace('C', 0, inplace=True)
credit_record['target']=credit_record['target'].astype(int)
credit_record.loc[credit_record['target']>=1,'target']=1
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.5 Perform data aggregation

1.5 Perform data aggregation

Group the record based on customer ID. If there are any default records under the cutomer ID, it considers as a default customer.

[15]:
```
## Check uniqueness for the ID column and remove dupplicate record of the Application Data
credit_record=pd.DataFrame(credit_record.groupby(['ID'])['target'].agg("max")).reset_index()
```

[16]:
```
credit_record["target"].value_counts()
```

[16]:
```
target
0    45654
1      331
Name: count, dtype: int64
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.6 Merge Application data and credit information

1.6 Merge Application data and credit information

```
[17]: df = pd.merge(application_record, credit_record, how='inner', on=['ID'])
```

```
[19]: df.head(5)
```

[19]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATIO
0	5008804	M	Y	Y	0	427500.0	Working	Higher ec
1	5008805	M	Y	Y	0	427500.0	Working	Higher ec
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / se
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / se
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / se

```
[20]: # Perform data exploration
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36457 entries, 0 to 36456
Data columns (total 19 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   36457 non-null  int64
 1   CODE_GENDER          36457 non-null  object
 2   FLAG_OWN_CAR         36457 non-null  object
 3   FLAG_OWN_REALTY      36457 non-null  object
 4   CNT_CHILDREN         36457 non-null  int64
 5   AMT_INCOME_TOTAL     36457 non-null  float64
 6   NAME_INCOME_TYPE     36457 non-null  object
 7   NAME_EDUCATION_TYPE  36457 non-null  object
 8   NAME_FAMILY_STATUS   36457 non-null  object
 9   NAME_HOUSING_TYPE    36457 non-null  object
 10  DAYS_BIRTH           36457 non-null  int64
 11  DAYS_EMPLOYED        36457 non-null  int64
 12  FLAG_MOBIL           36457 non-null  int64
 13  FLAG_WORK_PHONE      36457 non-null  int64
 14  FLAG_PHONE           36457 non-null  int64
 15  FLAG_EMAIL           36457 non-null  int64
 16  OCCUPATION_TYPE      25134 non-null  object
 17  CNT_FAM_MEMBERS      36457 non-null  int64
 18  target               36457 non-null  int32
dtypes: float64(1), int32(1), int64(9), object(8)
memory usage: 5.1+ MB
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.7 Checking Missing Value and hendle it

1.6 Merge Application data and credit information

```
[17]: df = pd.merge(application_record, credit_record, how='inner', on=['ID'])
```

```
[19]: df.head(5)
```

[19]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATIO
0	5008804	M	Y	Y	0	427500.0	Working	Higher ec
1	5008805	M	Y	Y	0	427500.0	Working	Higher ec
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / se
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / se
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / se

```
[20]: # Perform data exploration
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36457 entries, 0 to 36456
Data columns (total 19 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   36457 non-null  int64
 1   CODE_GENDER          36457 non-null  object
 2   FLAG_OWN_CAR         36457 non-null  object
 3   FLAG_OWN_REALTY      36457 non-null  object
 4   CNT_CHILDREN         36457 non-null  int64
 5   AMT_INCOME_TOTAL     36457 non-null  float64
 6   NAME_INCOME_TYPE     36457 non-null  object
 7   NAME_EDUCATION_TYPE  36457 non-null  object
 8   NAME_FAMILY_STATUS   36457 non-null  object
 9   NAME_HOUSING_TYPE    36457 non-null  object
 10  DAYS_BIRTH           36457 non-null  int64
 11  DAYS_EMPLOYED        36457 non-null  int64
 12  FLAG_MOBIL           36457 non-null  int64
 13  FLAG_WORK_PHONE      36457 non-null  int64
 14  FLAG_PHONE           36457 non-null  int64
 15  FLAG_EMAIL           36457 non-null  int64
 16  OCCUPATION_TYPE      25134 non-null  object
 17  CNT_FAM_MEMBERS      36457 non-null  int64
 18  target               36457 non-null  int32
dtypes: float64(1), int32(1), int64(9), object(8)
memory usage: 5.1+ MB
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.7 Checking Missing Value and handle it

1.7 Checking Missing Value and hendle it

[22]:
```
# Check missing value
print(df.isnull().sum())
```

```
ID                        0
CODE_GENDER               0
FLAG_OWN_CAR              0
FLAG_OWN_REALTY           0
CNT_CHILDREN              0
AMT_INCOME_TOTAL          0
NAME_INCOME_TYPE          0
NAME_EDUCATION_TYPE       0
NAME_FAMILY_STATUS        0
NAME_HOUSING_TYPE         0
DAYS_BIRTH                0
DAYS_EMPLOYED             0
FLAG_MOBIL                0
FLAG_WORK_PHONE           0
FLAG_PHONE                0
FLAG_EMAIL                0
OCCUPATION_TYPE       11323
CNT_FAM_MEMBERS           0
target                    0
dtype: int64
```

Handle Missing Value by replacement

There are 11,323 null records in OCCUPATION_TYPE, so I will replace it with "non-specified".

[23]:
```
df['OCCUPATION_TYPE'].fillna('non-specified',inplace=True)
```

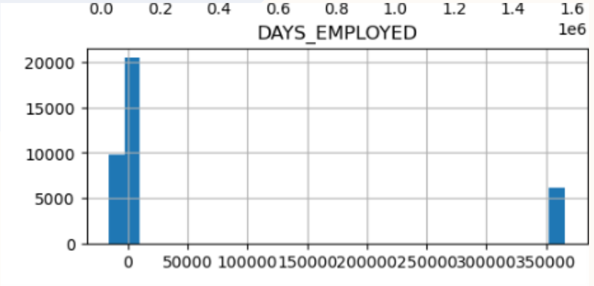
```
C:\Users\t_car\AppData\Local\Temp\ipykernel_3188\3518268457.py:1: FutureWarning: A value is trying
chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediat
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, i
ead, to perform the operation inplace on the original object.


  df['OCCUPATION_TYPE'].fillna('non-specified',inplace=True)
```

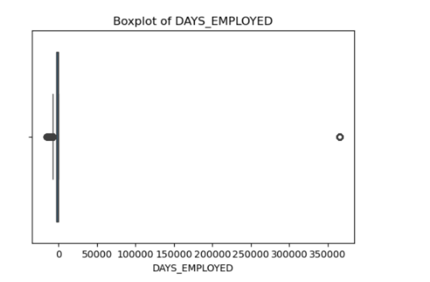
[24]:
```
# Re-Check missing value
print(df.isnull().sum())
```

```
ID                     0
CODE_GENDER            0
FLAG_OWN_CAR           0
FLAG_OWN_REALTY        0
CNT_CHILDREN           0
AMT_INCOME_TOTAL       0
NAME_INCOME_TYPE       0
NAME_EDUCATION_TYPE    0
NAME_FAMILY_STATUS     0
NAME_HOUSING_TYPE      0
DAYS_BIRTH             0
DAYS_EMPLOYED          0
FLAG_MOBIL             0
FLAG_WORK_PHONE        0
FLAG_PHONE             0
FLAG_EMAIL             0
OCCUPATION_TYPE        0
CNT_FAM_MEMBERS        0
target                 0
dtype: int64
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.8 Examine numerical data and handle it





The outlier of DAYS_EMPLOYED can be detected from the plot. I checked the data dict for DAYS_EMPLOYED defnetion agian and I found that DAYS_EMPLOYED: Count backwards from current day(0). If positive, it means the person currently unemployed. SO, we will find the outlier and find the way to handle it.

[28]:
```
## Identify the outlier and count it.
df[df['DAYS_EMPLOYED']>=0]['DAYS_EMPLOYED'].value_counts()
```

[28]:
```
DAYS_EMPLOYED
365243    6135
Name: count, dtype: int64
```

Handle deteccted outlier Value by replacing with proper value.

As checking, there are 6135 records fo 365243. Per the data dict,if DAYS_EMPLOYED is negative, refer to unemploy. So, I will convert those value to be 0.

[29]:
```
df['DAYS_EMPLOYED'].replace(365243,0,inplace=True)
```

```
C:\Users\t_car\AppData\Local\Temp\ipykernel_3188\1132558553.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['DAYS_EMPLOYED'].replace(365243,0,inplace=True)
```

[32]:
```
#df.head(10)
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

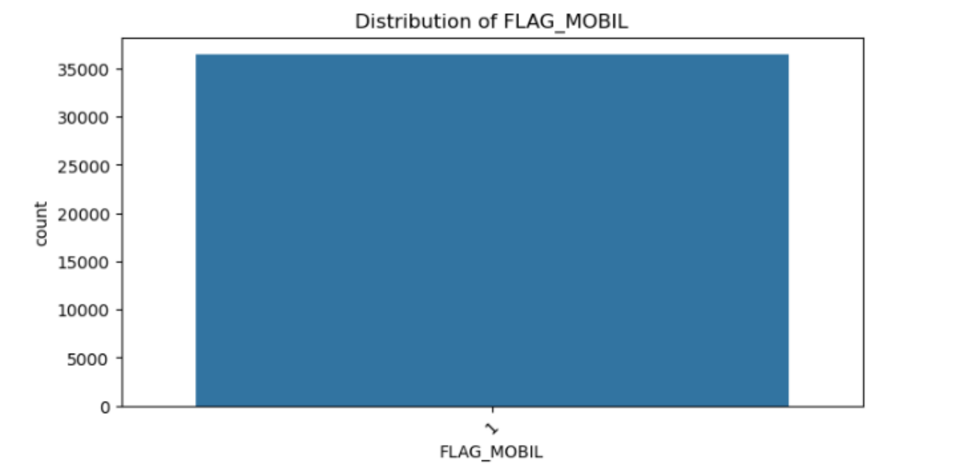
Convert DAYS_EMPLOYED and DAYS_BIRTH to Year

Convert DAYS_EMPLOYED and DAYS_BIRTH to Year

[33]:
```
df['AGE']=round(-df['DAYS_BIRTH']/365,0)
df['YEARS_EMPLOYED']=round(-df['DAYS_EMPLOYED']/365)
df.loc[df['YEARS_EMPLOYED']<0,'YEARS_EMPLOYED']=0
df.drop(columns=["DAYS_BIRTH","DAYS_EMPLOYED"],inplace=True)
```

STEP 1: LOAD AND EXPLORATORY DATA ANALYSIS (EDA)

1.9 Examine Categorical data and handle it



After examine the catagorical data, we found that every customer has mobile as all record FLAG_MOBIL = 1. So, I will drop this feature because there are no different from each other which no contribution to the model.

```
[39]: df = df.drop('FLAG_MOBIL', axis=1)
```

STEP 2: PERFORM FACTOR ANALYSIS, TRANSFORMATION AND REDUCTION

2.1: Perform Information Value Analysis (IV

Step 2: Perform Factor Analysis, Transformation and Reduction

2.1: Perform Information Value Analysis (IV)

[43]:
```
# Split data into features and target
features = df.columns[df.columns != 'target']
target = 'target'
```

[44]:
```
# Calculate WoE and binning information
bins = sc.woebin(df, y=target)
```

```
[INFO] creating woe binning ...
C:\Users\t_car\anaconda3\envs\study\Lib\site-packages\scorecardpy\condition_fun.py:131: FutureWarning: Setting an item of inc
ated and will raise in a future error of pandas. Value '[0 0 0 ... 1 0 0]' has dtype incompatible with int32, please explicit
type first.
  dat.loc[:,y] = dat[y].apply(lambda x: x if pd.isnull(x) else int(x)) #dat[y].astype(int)
C:\Users\t_car\anaconda3\envs\study\Lib\site-packages\scorecardpy\condition_fun.py:40: FutureWarning: errors='ignore' is depr
```

[45]:
```
# Extracting IV for each variable from the bins
# bins is a dictionary with keys as variable names and values as DataFrames containing binning results
iv_dict = {}
for key, dataframe in bins.items():
    # The 'total_iv' from the last row of each binning DataFrame contains the IV for the variable
    iv_dict[key] = dataframe['total_iv'].values[-1]  # Ensure this key exists in your DataFrame
```

[46]:
```
# Create a DataFrame from the dictionary to view IV values
iv_df = pd.DataFrame.from_dict(iv_dict, orient='index', columns=['IV']).reset_index()
iv_df.rename(columns={'index': 'Variable'}, inplace=True)
```

[48]:
```
iv_df
```

[48]:

	Variable	IV
0	CNT_FAM_MEMBERS	0.017515
1	NAME_HOUSING_TYPE	0.039364
2	AMT_INCOME_TOTAL	0.077314
3	NAME_FAMILY_STATUS	0.092021
4	NAME_INCOME_TYPE	0.051452
5	FLAG_OWN_CAR	0.002825
6	FLAG_WORK_PHONE	0.001490
7	CODE_GENDER	0.013040
8	NAME_EDUCATION_TYPE	0.022593
9	AGE	0.104023
10	ID	0.121705
11	FLAG_PHONE	0.007413
12	CNT_CHILDREN	0.002878
13	FLAG_OWN_REALTY	0.025502
14	FLAG_EMAIL	0.001100
15	OCCUPATION_TYPE	0.087996
16	YEARS_EMPLOYED	0.087195

STEP 2: PERFORM FACTOR ANALYSIS, TRANSFORMATION AND REDUCTION

2.1: Perform Information Value Analysis (IV)

Information Value (IV) is used to evaluate the predictive power of a categorical or binned continuous variable. It can be intepreted as follows;

- IV < 0.02: Predictive power is considered weak.
- 0.02 ≤ IV < 0.1: Predictive power is considered medium.
- 0.1 ≤ IV < 0.3: Predictive power is considered strong.
- IV ≥ 0.3: Predictive power is considered very strong.

[49]:
```
# Filter out features with IV less than 0.02
low_iv_features = iv_df[iv_df['IV'] < 0.02]['Variable'].tolist()
#print("Features to remove due to low IV (< 0.01):", low_iv_features)
```

[50]:
```
data_cleaned = df.drop(columns=low_iv_features)
```

[51]:
```
data_cleaned.head(10)
```

STEP 2: PERFORM FACTOR ANALYSIS, TRANSFORMATION AND REDUCTION

2.2 Perform classing (Binning)

2.2 Perform classing (Binning)

[52]:
```python
# Perform initial Fine Classing
bins_fine = sc.woebin(df, y='target', max_num_bin=10)  # start with a higher number of bins

# Review the fine classing results
for var in bins_fine:
    print(f"Binning for {var}:")
    print(bins_fine[var])
    print("\n")
```

```
[INFO] creating woe binning ...

C:\Users\t_car\anaconda3\envs\study\Lib\site-packages\scorecardpy\condition_fun.py:131: FutureWarning: Setting an item of incompatible dtype is deprec
ated and will raise in a future error of pandas. Value '[0 0 0 ... 1 0 0]' has dtype incompatible with int32, please explicitly cast to a compatible d
type first.
  dat.loc[:,y] = dat[y].apply(lambda x: x if pd.isnull(x) else int(x)) #dat[y].astype(int)
```

Based on the result above, I decided to perform manually binning adjustment with two factors;

- AGE: There is a variability across bins and WoE varies significantly from bin to bin.
- YEARS_EMPLOYED: WoE increasing trend.

[53]:
```python
### Manually Classing
breaks = {
    'AGE':[-float('inf'), 30, 40, 50, 60, float('inf')],
    'YEARS_EMPLOYED':  [-float('inf'), 5, 15, 25, float('inf')]
}
bins = sc.woebin(df, y='target', breaks_list=breaks)
```

```
C:\Users\t_car\anaconda3\envs\study\Lib\site-packages\scorecardpy\condition_fun.py:131: FutureWarning: Setting an item of incompatible dtype is deprec
ated and will raise in a future error of pandas. Value '[0 0 0 ... 1 0 0]' has dtype incompatible with int32, please explicitly cast to a compatible d
type first.
```

STEP 2: PERFORM FACTOR ANALYSIS, TRANSFORMATION AND REDUCTION

2.3 Apply WoE Tranformation

[54]:
```
X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply WoE transformation
train_woe = sc.woebin_ply(X_train, bins)
test_woe = sc.woebin_ply(X_test, bins)
```

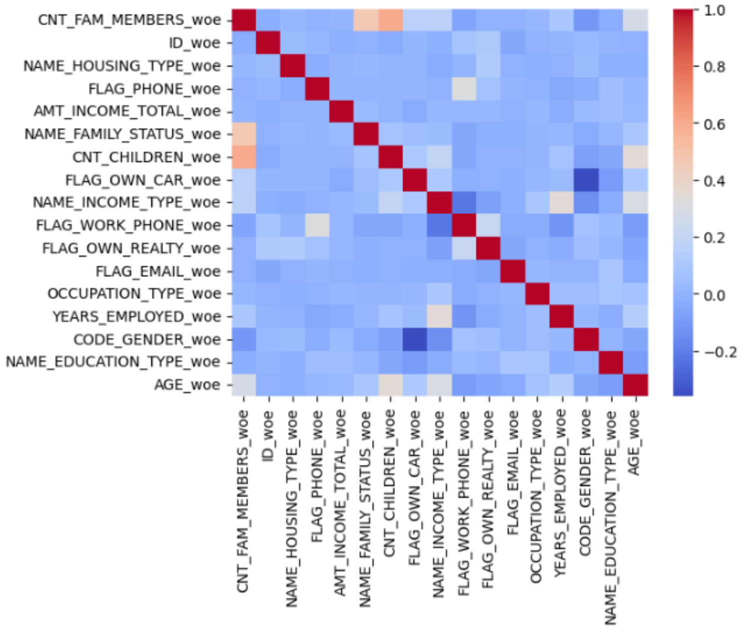
```
[INFO] converting into woe values ...
[INFO] converting into woe values ...
```

STEP 2: PERFORM FACTOR ANALYSIS, TRANSFORMATION AND REDUCTION

2.4 Perform correlation check to prevent multicollinearity

2.4 Perform correlation check to prevent multicollinearity

[55]:
```
# Check correlations
corr_matrix = train_woe.corr()
sns.heatmap(corr_matrix, annot=False, cmap='coolwarm')
plt.show()
```



[56]:
```
# Identify highly correlated variables (greater than 0.5)
high_corr_vars = {corr_matrix.columns[x]: corr_matrix.columns[y] for x in range(len(corr_matrix.columns)) for y
high_corr_vars
```

[56]: {'CNT_FAM_MEMBERS_woe': 'CNT_CHILDREN_woe'}

[57]:
```
# Remove highly correlated variables
train_woe = train_woe.drop(columns=high_corr_vars.keys())
test_woe = test_woe.drop(columns=high_corr_vars.keys())
```

STEP 2: PERFORM FACTOR ANALYSIS, TRANSFORMATION AND REDUCTION

2.5 Apply SMOTE for the imbalance class

2.5 Apply SMOTE for the imbalance class

```
[ ]: # Apply SMOTE to oversample the minority class
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(train_woe, y_train)
```

STEP 3: MODEL DEVELOPMENT

3.1 Develop the first model Logistic Regression

Step 3: Model Development

3.1 Develop the first model Logistic Regression

[ ]:

```
model = LogisticRegression(max_iter=1000)
model.fit(X_train_resampled, y_train_resampled)
```

STEP 4: MODEL EVALUATION

4.1 Evaluate with the matric Accuracy, ROC-AUC and F1

4.1 Evaluate with the matric Accuracy, ROC-AUC and F1

[61]:
```
# Evaluate the model
from sklearn.metrics import accuracy_score, roc_auc_score, f1_score
accuracy = accuracy_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)
f1_score = f1_score(y_test, y_pred)
```

[62]:
```
print(f"Accuracy: {accuracy}")
print(f"ROC-AUC: {roc_auc}")
print(f"F1-Score: {f1_score}")
```

```
Accuracy: 0.6515359297860669
ROC-AUC: 0.6176605926743972
F1-Score: 0.0215633423180593
```

STEP 4: MODEL EVALUATION

4.2 Confusion Matrix

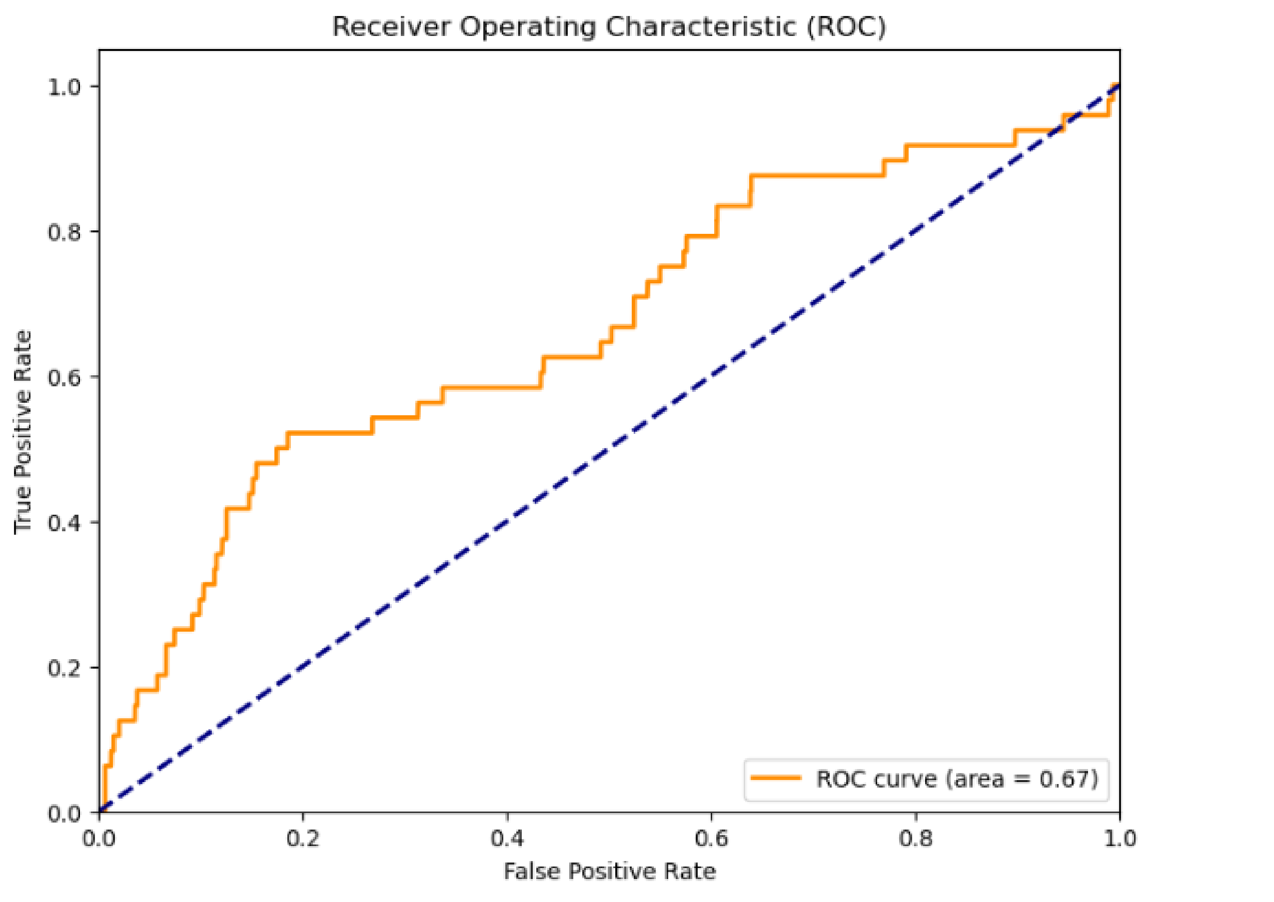
4.2 Confusion Matrix

```
[63]: conf_matrix = confusion_matrix(y_test, y_pred)
      conf_matrix
```

```
[63]: array([[4723, 2521],
             [  20,   28]], dtype=int64)
```

STEP 4: MODEL EVALUATION

4.3 Plot the ROC Curve



STEP 5: DEVELOP THE CHALLENGER MODEL

5.1 Develop the challenger model Random Forest

Step 5: Develop the challenger Model

5.1 Develop the the challenger model Random Forest

[65]:
```
# Train Random Forest model
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train_resampled, y_train_resampled)
```

[65]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

STEP 6: RANDOM FOREST MODEL EVALUATION

6.1 Evaluate with the matric Accuracy, ROC-AUC and F1

Step 6: Random Forest model Evaluation

6.1 Evaluate with the matric Accuracy, ROC-AUC and F1

[66]:
```
# Drop CNT_CHILDREN_woe from the test set
X_test_woe = sc.woebin_ply(X_test, bins)
X_test_woe = X_test_woe.drop(columns=['CNT_FAM_MEMBERS_woe'])

# Predict on the test set
y_pred_rf = rf_classifier.predict(X_test_woe)
```

```
[INFO] converting into woe values ...
```

[67]:
```
# Evaluate the model
from sklearn.metrics import accuracy_score, roc_auc_score, f1_score
accuracy_rf = accuracy_score(y_test, y_pred_rf)
roc_auc_rf = roc_auc_score(y_test, y_pred_rf)
f1_score_rf = f1_score(y_test, y_pred_rf)

print(f"Random Forest Accuracy: {accuracy_rf}")
print(f"Random Forest ROC-AUC: {roc_auc_rf}")
print(f"Random Forest F1 Score: {f1_score_rf}")
```

```
Random Forest Accuracy: 0.9912232583653319
Random Forest ROC-AUC: 0.6541102981778023
Random Forest F1 Score: 0.3191489361702128
```

STEP 6: RANDOM FOREST MODEL EVALUATION

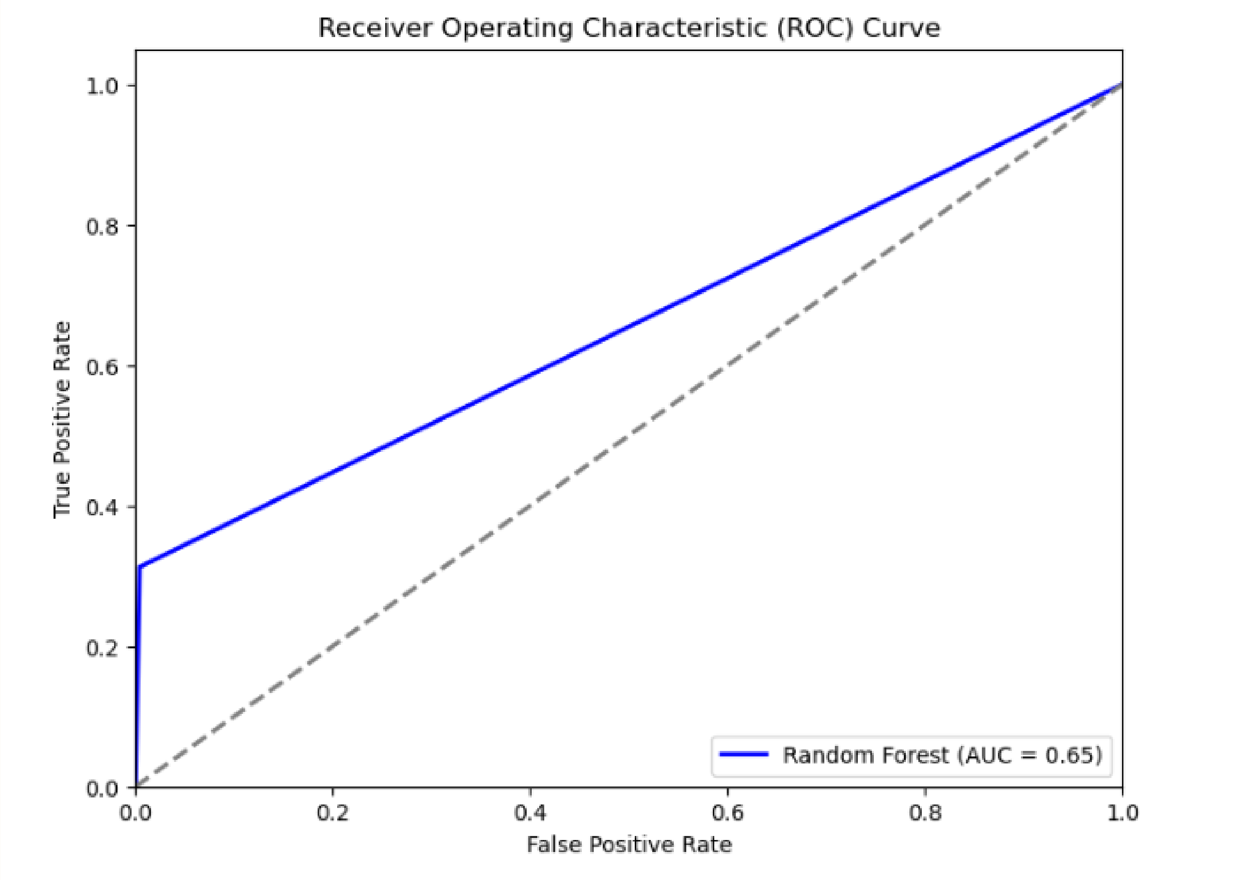
6.2 Confusion Matrix

6.2 Confusion Matrix

```
[68]: conf_matrix = confusion_matrix(y_test, y_pred_rf)
      conf_matrix

[68]: array([[7213,   31],
             [  33,   15]], dtype=int64)
```

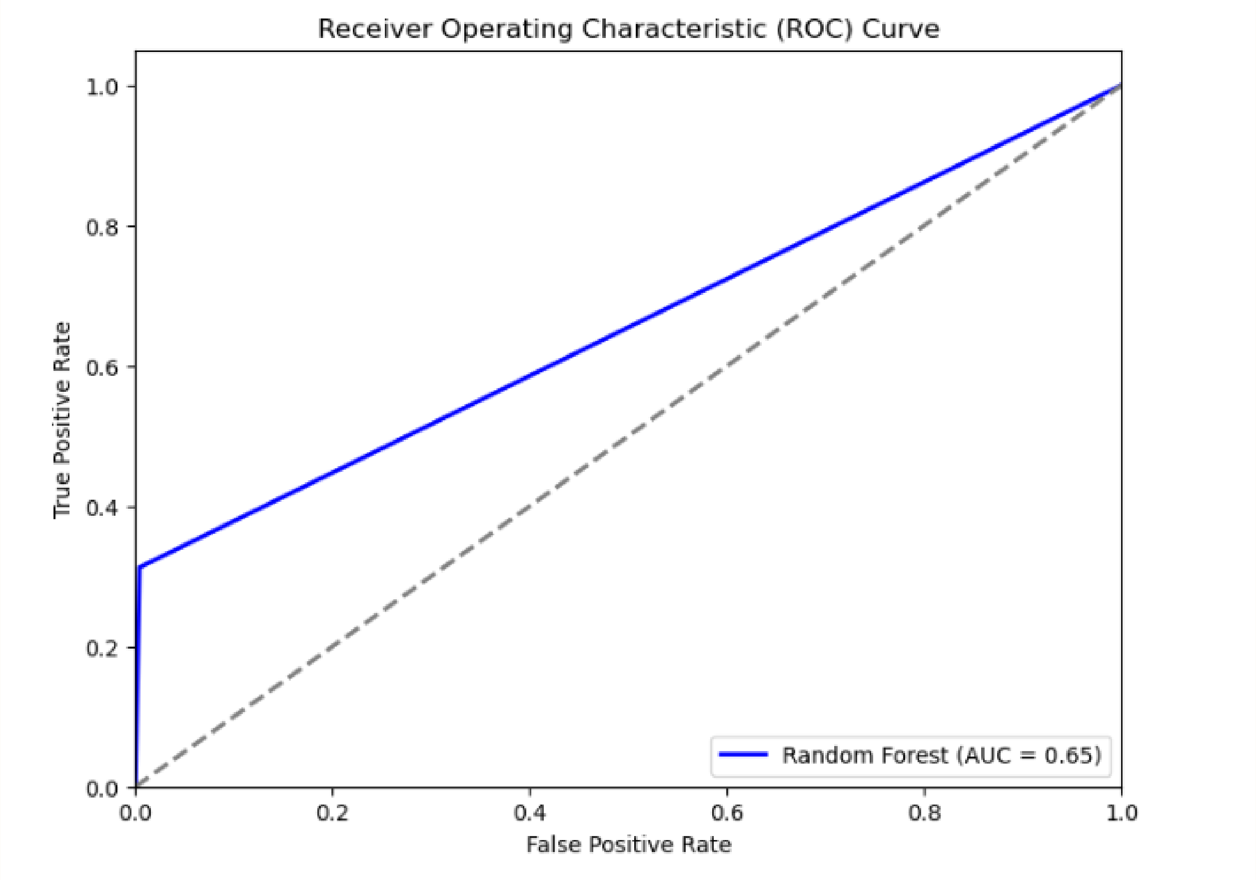
STEP 6: RANDOM FOREST MODEL EVALUATION

6.3 Plot the ROC Curve



STEP 6: RANDOM FOREST MODEL EVALUATION

6.3 Plot the ROC Curve



STEP 7: DISCUSSION/CONCLUSION

7.1 Summarize key findings.

- There are two data application data and credit data. The application data contains the features that we can use to predict the outcome and credit data contain the target. However, I have to perform some data processing such as deduplicate or data aggregation before merging these two data sets.
- I have to transform original target feature to be binary (Default/non-default) based on definition.
- The data contain missing value and outlier that I need to clean it before process.
- I have perform factor analysis to select the features the process include IV (information Value) to check which feature has more power of prediction, perform correlation check to prevent multicollinearity, and classing to group the data.
- I found that the data is very imbalance. So, I try to handle it by using SMOTE method to synthetic oversample data.
- Then, I train two model Logistic Regression and Random Forest and compare the performance.
- Based on performance comparison, the random forest perform better than logistic regression due to Accuracy, ROC-AUC and F1. This is an imbalance case, so I more focus on F1 score. Actually, both model is not perform well. There is a room for improvement.

STEP 7: DISCUSSION/CONCLUSION

7.2 Discuss the model's limitations and assumptions.

- Based on the results, the performance of both model are poor which might cause from the imbalance data. Even I try to overcome by using SMOTE method and look into F1 score.

STEP 7: DISCUSSION/CONCLUSION

7.3 Propose future work or improvements.

- To improve model in the future, I will try to check the data leakage and remove it from model.-
- Try other method to overcome imbalance class.
- Try another model to improve F1 score.

THANK YOU

Teerarat Siwapathomchai (Toon)