Credit Default Risk Predictor

A CLASSIFICATION PROBLEM

By Erensu Akdogan

The PROBLEM

- Credit Default
- Credit Risk
- Decision to lend and pricing
- Use the data of historical customers to predict if a prospective client will be ‘good’ or ‘bad’

Data Set

Customer Payment Record

```
<class 'pandas.core.frame.DataFrame'>
Index: 839345 entries, 1 to 1048574
Data columns (total 4 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   ID              839345 non-null  int64
 1   MONTHS_BALANCE  839345 non-null  int64
 2   STATUS          839345 non-null  int64
```

- Kaggle
- ~43k # of unique IDs in Customer Payment Record

Customer Attributes

```
<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  float64
dtypes: float64(2), int64(8), object(8)
memory usage: 60.2+ MB
```

Feature engineering - creating customer labels

	ID	MONTHS_BALANCE	STATUS
1	5001711	-1	1
2	5001711	-2	1
3	5001711	-3	1
4	5001712	0	0
5	5001712	-1	0
...	...	...	...
1048570	5150487	-25	0
1048571	5150487	-26	0
1048572	5150487	-27	0
1048573	5150487	-28	0
1048574	5150487	-29	0

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

- Duration of the loan is an important attribute.
- Better to have an average loan status metric as dependent var.
- Assigned penalty points per month. Sum them. Divide my duration.
- Average Days Past Due (DPD).
- Average DPD > 45 days. I consider risky.

Data Cleaning

- ID should be unique. For some reason there are duplicated IDs in the customer attributes table.
- Redundant variable: CNT_CHILDREN
- 'OCCUPATION_TYPE': this had missing values, better to get rid of.
- 'FLAG_MOBIL': this was 1 for all. Dropped.

Preprocessing

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33110 entries, 0 to 33109
Data columns (total 29 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Gender_F                            33110 non-null  int32
 1   Gender_M                            33110 non-null  int32
 2   Income_Type_Commercial associate    33110 non-null  int32
 3   Income_Type_Pensioner               33110 non-null  int32
 4   Income_Type_State servant           33110 non-null  int32
 5   Income_Type_Student                 33110 non-null  int32
 6   Income_Type_Working                 33110 non-null  int32
 7   Family_Status_Civil marriage        33110 non-null  int32
 8   Family_Status_Married               33110 non-null  int32
 9   Family_Status_Separated             33110 non-null  int32
 10  Family_Status_Single / not married  33110 non-null  int32
 11  Family_Status_Widow                 33110 non-null  int32
 12  Housing_Type_Co-op apartment        33110 non-null  int32
 13  Housing_Type_House / apartment      33110 non-null  int32
 14  Housing_Type_Municipal apartment    33110 non-null  int32
 15  Housing_Type_Office apartment       33110 non-null  int32
 16  Housing_Type_Rented apartment       33110 non-null  int32
 17  Housing_Type_With parents           33110 non-null  int32
 18  Education_Type                      33110 non-null  float64
 19  Months_on_Book                      33110 non-null  int64
 20  Owns_Car                            33110 non-null  int64
 21  Owns_Realty                         33110 non-null  int64
 22  Annual_Income                       33110 non-null  float64
 23  Days_Since_Birth                    33110 non-null  int64
 24  Days_Employed                       33110 non-null  int64
 25  Has_Work_Phone                      33110 non-null  int64
 26  Has_Phone                           33110 non-null  int64
 27  Has_Email                           33110 non-null  int64
 28  Family_Size                         33110 non-null  float64
dtypes: float64(3), int32(18), int64(8)
memory usage: 5.1 MB
```

- **Binary Encoding:** Convert 'Owns_Car' and 'Owns_Realty' columns to binary format (0 or 1) based on whether the client owns a car or real estate.
- **Numerical and Categorical Separation:** Separate the dataset into numerical and categorical features.
- **Ordinal Encoding:** Encode ordinal categorical feature 'Education_Type' into numerical values using Ordinal Encoder.
- **One-Hot Encoding:** Encode nominal categorical features into binary format using one-hot encoding.
- **Concatenation:**Combine the processed nominal, ordinal, and numerical features into a single dataframe 'ndf'.

Getting Data Ready for the models

- X - Y Split (0.3)
- Test-Train Split
- Tomek links
- Standart Scaler
- SMOTE

```
Train distibution:  Loan_Status
0            24865
1             1301
Name: count, dtype: int64
Test distibution:  Loan_Status
0             6226
1              316
Name: count, dtype: int64
```

Error metrics

- **Precision**: Precision measures the accuracy of the positive predictions. What percentage of the predicted defaults are actual defaults.
- **Recall:** Recall measures the proportion of actual positives that were correctly identified by the model. So how good model is to predict defaults. Relatively more important.
- **F1 Score:** Harmonic mean of precision and recall and provides a single measure of a model's accuracy.
- **ROC AUC**: The area under the Receiver Operating Characteristic (ROC) curve, provides an aggregate measure of the model's performance across different classification thresholds.. i.e evaluates the model's ability to discriminate between default and non-default cases across all possible thresholds.

Model Comparison

Log. Regression			Log. w. Best Params			Decision Tree w. Best Params		
Error_metric	Train	Test	Error_metric	Train	Test	Error_metric	Train	Test
Accuracy	0.86	0.80	***Accuracy***	0.87	0.96	***Accuracy***	0.98	0.96
Precision	0.82	0.18	***Precision***	0.96	0.53	***Precision***	0.98	0.55
Recall	0.92	0.91	***Recall***	0.77	0.71	***Recall***	0.97	0.74
F1 Score	0.87	0.30	***F1 Score***	0.85	0.61	***F1 Score***	0.98	0.64
ROC AUC	0.86	0.85	***ROC AUC***	0.94	0.93	***ROC AUC***	0.98	0.86

Model COMPARISON - Ensembled Methods

eXtreme Gradient Boosting		
Error_metric	Train	Test
Accuracy	0.9960	0.9755
Precision	0.9961	0.7453
Recall	0.9959	0.7500
F1 Score	0.9960	0.7476
ROC AUC	0.9960	0.8685

Random Forest		
Error_metric	Train	Test
Accuracy	0.9993	0.9751
Precision	0.9991	0.7297
Recall	0.9994	0.7690
F1 Score	0.9993	0.7488
ROC AUC	0.9993	0.8773

- Both XGBoost and Random Forest exhibit high accuracy, precision, recall, and F1 score on both training and test sets.

- XGBoost slightly outperforms Random Forest in precision, while Random Forest shows a slightly higher recall.

- Both models demonstrate strong discriminative power, as indicated by their high ROC AUC scores on the test set.

- Both models demonstrate robust F1 scores, highlighting their balanced performance in terms of both false positives and false negatives.

THANKS FOR LISTENING!