

Final Project

PBI Data Scientist VIX with ID/X Partners

Presented by
Revito Pradipa


Revito Pradipa

<https://linkedin.com/in/revitopradipa>

About You

Undergraduate Informatics Engineering
Student that always passionate to learn
especially on Data and Machine Learning

Insert Your Experience

- 
- Head of AI Development GDSC UMS
 - Head of Legislation and Advocacy DPM FKl UMS
 - BDT 2022 - Data Analytics Program

Business Understanding

Sebuah perusahaan penyedia jasa kredit sering dihadapkan dengan masalah terkait risiko kredit. Risiko kredit adalah suatu risiko kerugian yang disebabkan oleh ketidakmampuan dari debitur atas kewajiban pembayaran utangnya baik utang pokok maupun bunganya ataupun keduanya.

Untuk mengurangi kerugian yang ada, perusahaan dapat membuat tindakan pencegahan dimana digunakan supaya dapat mengambil tindakan berdasarkan kemungkinan adanya risiko kredit

Analytical approach

Sebagai seorang Data Scientist, terdapat berbagai cara untuk menanggulangi masalah tersebut. Salah satu caranya dengan membuat model prediktif yang digunakan untuk mendeteksi kemungkinan debitur yang akan mengalami kesulitan dalam pembayaran kredit atau hutang.

Solusi yang sesuai adalah membuat model klasifikasi yang dapat mengklasifikasikan debitur yang terindikasi akan mengalami kesulitan pembayaran dan debitur yang dapat membayar secara lancar, dengan menggunakan data debitur yang telah ada sebelumnya.

Import



```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

Tahap pertama yaitu
mengimpor beberapa library
Data Analysis

Read CSV File

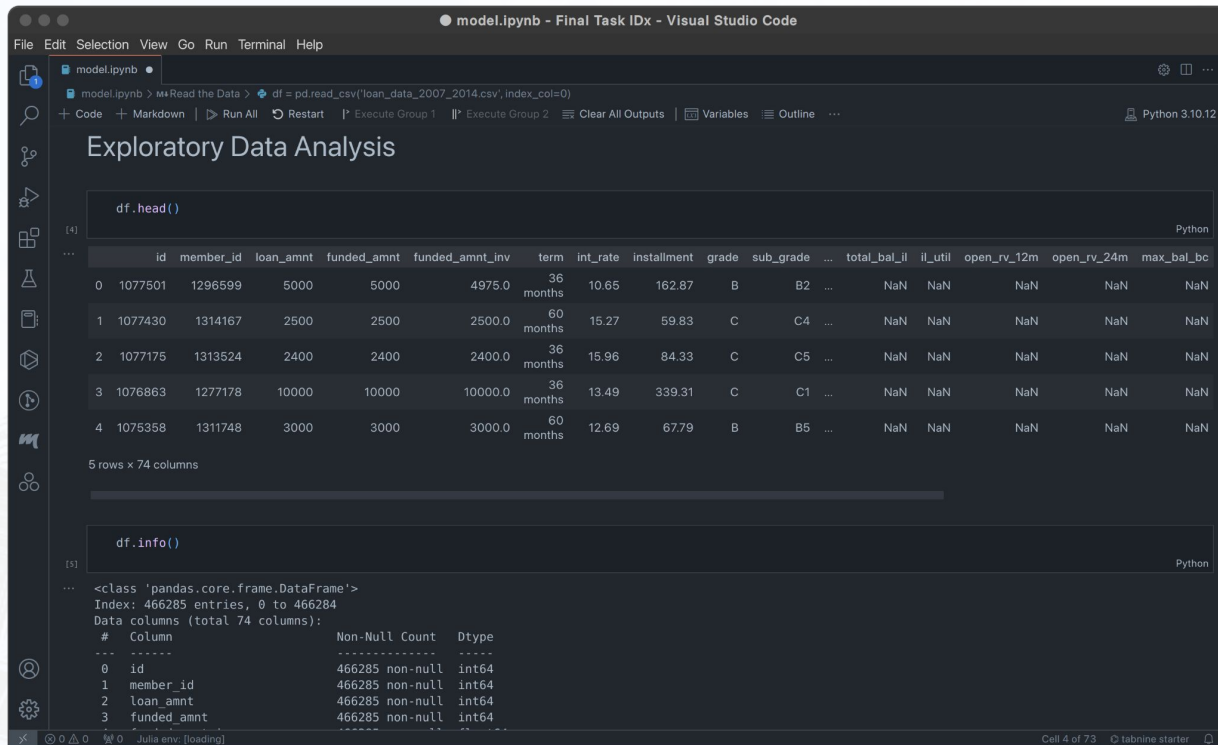


```
1 df = pd.read_csv('loan_data_2007_2014.csv', index_col=0)
2
3 df.shape
4
```

Kemudian dilanjutkan dengan membaca file dataset yang ada

EDA

Lalu melakukan
Exploratory Data Analysis



The screenshot shows a Jupyter Notebook titled "model.ipynb" in Visual Studio Code. The notebook is in "Edit" mode. The first cell contains the code `df = pd.read_csv('loan_data_2007_2014.csv', index_col=0)`. The second cell contains the code `df.head()`, which has been executed, resulting in a preview of the first 5 rows of the dataset. The third cell contains the code `df.info()`, which has also been executed, showing the data structure and statistics.

```
df.head()
```

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	total_bal_il	il_util	open_rv_12m	open_rv_24m	max_bal_bc
0	1077501	1296599	5000	5000	4975.0	36 months	10.65	162.87	B	B2	...	NaN	NaN	NaN	NaN	NaN
1	1077430	1314167	2500	2500	2500.0	60 months	15.27	59.83	C	C4	...	NaN	NaN	NaN	NaN	NaN
2	1077175	1313524	2400	2400	2400.0	36 months	15.96	84.33	C	C5	...	NaN	NaN	NaN	NaN	NaN
3	1076863	1277178	10000	10000	10000.0	36 months	13.49	339.31	C	C1	...	NaN	NaN	NaN	NaN	NaN
4	1075358	1311748	3000	3000	3000.0	60 months	12.69	67.79	B	B5	...	NaN	NaN	NaN	NaN	NaN

5 rows x 74 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 466285 entries, 0 to 466284
Data columns (total 74 columns):
#   Column              Non-Null Count  Dtype
---  -
0   id                   466285 non-null int64
1   member_id            466285 non-null int64
2   loan_amnt            466285 non-null int64
3   funded_amnt          466285 non-null int64
```

Data Preprocessing

```
1 """## Delete columns in high missing data columns"""
2
3 nan_df_percent[nan_df_percent > 40]
4
5 # Drop columns with many missing values (above 40%)
6 df.drop(nan_df_percent[nan_df_percent > 40].index, axis=1, inplace=True)
7
8 nan_df_percent = df.isnull().sum() / df.shape[0] * 100
9
10 """## Delete rows in small missing data columns"""
11
12 nan_df_small_percent = nan_df_percent[(nan_df_percent > 0) & (nan_df_percent < 1)].sort_values(ascending=False)
13 nan_df_small_percent
14
15 def del_missing_row(df, list_col):
16     for col in list_col.index:
17         df.drop(df[df[col].isnull()].index, axis=0, inplace=True)
18     nan_df_percent = df.isnull().sum() / df.shape[0] * 100
19     nan_df_percent = nan_df_percent[nan_df_percent > 0]
20     return nan_df_percent
21
22 """## Impute data with mean or mode for the rest of missing data columns"""
23
24 nan_df_percent = del_missing_row(df, nan_df_small_percent)
25 nan_df_percent
26
27 df[nan_df_percent.index].info()
28
29 df[nan_df_percent.index].describe()
30
31 def impute_missing(df, cat_col, num_col):
32     for col in cat_col.index:
33         df[col].fillna(df[col].mode()[0], inplace=True)
34     for col in num_col.index:
35         df[col].fillna(df[col].median(skipna=True), inplace=True)
36     nan_df_percent = df.isnull().sum() / df.shape[0] * 100
37     nan_df_percent = nan_df_percent[nan_df_percent > 0]
38     return nan_df_percent
39
```

Pada tahap EDA, dilakukan pula data preprocessing seperti membersihkan data dan juga menghapus data duplikat

Pada tahap cleaning, dilakukan 3 cara yaitu menghapus kolom, baris, dan juga mengisi nilai kosong dengan teknik tertentu.

Target Variable

Kemudian membuat variabel dependent, yang menjadi indikator yang akan diprediksi kedepannya.

```
model.ipynb - Final Task IDx - Visual Studio Code
model.ipynb • final_task.py
model.ipynb > Read the Data > df = pd.read_csv('loan_data_2007_2014.csv', index_col=0)
+ Code + Markdown + Run All + Restart + Execute Group 1 + Execute Group 2 + Clear All Outputs + Variables + Outline + Python 3.10.12

Make the target columns

df['loan_status'].value_counts()

[21] Python

loan_status
Current                224096
Fully Paid             184523
Charged Off            42056
Late (31-120 days)     6895
In Grace Period        3144
Does not meet the credit policy. Status:Fully Paid  1913
Late (16-30 days)     1217
Default                832
Does not meet the credit policy. Status:Charged Off  725
Name: count, dtype: int64

# The risk of current status is unknown, so we won't need that
df.drop(df[['loan_status']][df['loan_status'] == 'Current'].index, axis=0, inplace=True)

[22] Python

potential_val = ['Charged Off', 'Default', 'Does not meet the credit policy. Status:Charged Off', 'Late (31-120 days)']
df['Target'] = np.where(df['loan_status'].isin(potential_val), 1, 0)

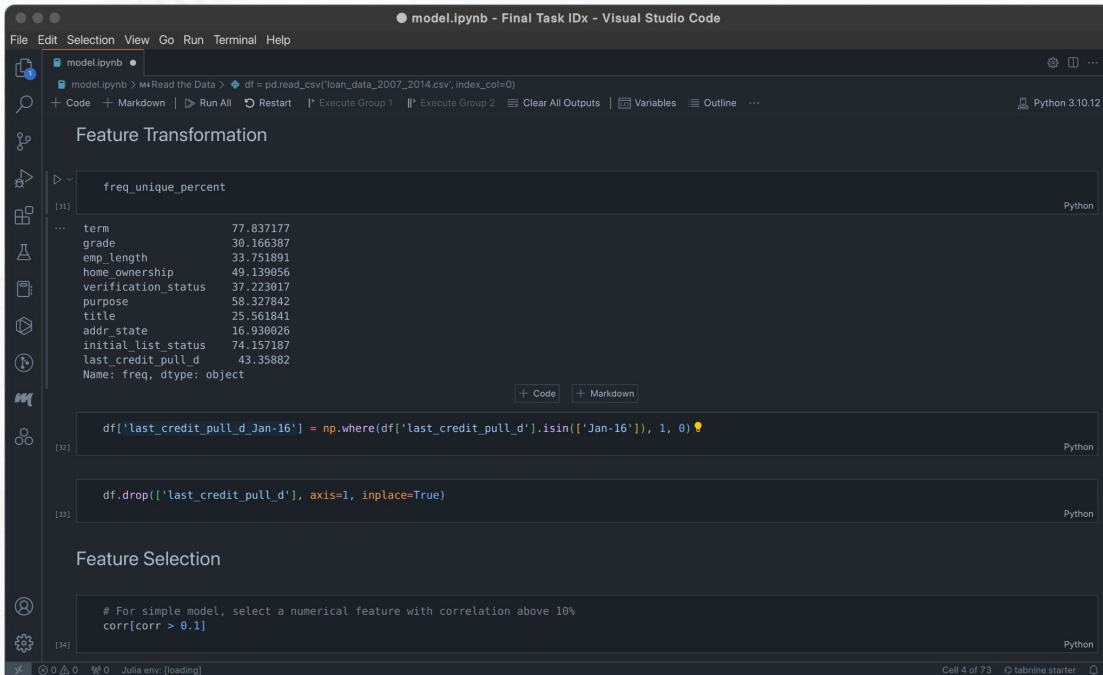
[23] Python

df['Target'].value_counts(normalize=True)

[24] Python

Target
```

Feature Engineering



The screenshot shows a Jupyter Notebook titled "model.ipynb" in Visual Studio Code. The notebook is divided into three sections: "Feature Transformation", "Feature Selection", and "Feature Selection".

Feature Transformation

```
[31]: freq_unique_percent
```

term	77.837177
grade	30.166387
emp_length	33.751891
home_ownership	49.139056
verification_status	37.223017
purpose	58.327842
title	25.561841
addr_state	16.930026
initial_list_status	74.157187
last_credit_pull_d	43.35882

Name: freq, dtype: object

```
[32]: df['last_credit_pull_d_Jan-16'] = np.where(df['last_credit_pull_d'].isin(['Jan-16']), 1, 0)
```

```
[33]: df.drop(['last_credit_pull_d'], axis=1, inplace=True)
```

Feature Selection

```
[34]: # For simple model, select a numerical feature with correlation above 10%
corr[corr > 0.1]
```

Kemudian melakukan feature engineering yang berupa transformasi fitur dan pemilihan fitur

Model Building & Tuning

Membuat model sekaligus mencari parameter terbaik menggunakan Grid Search

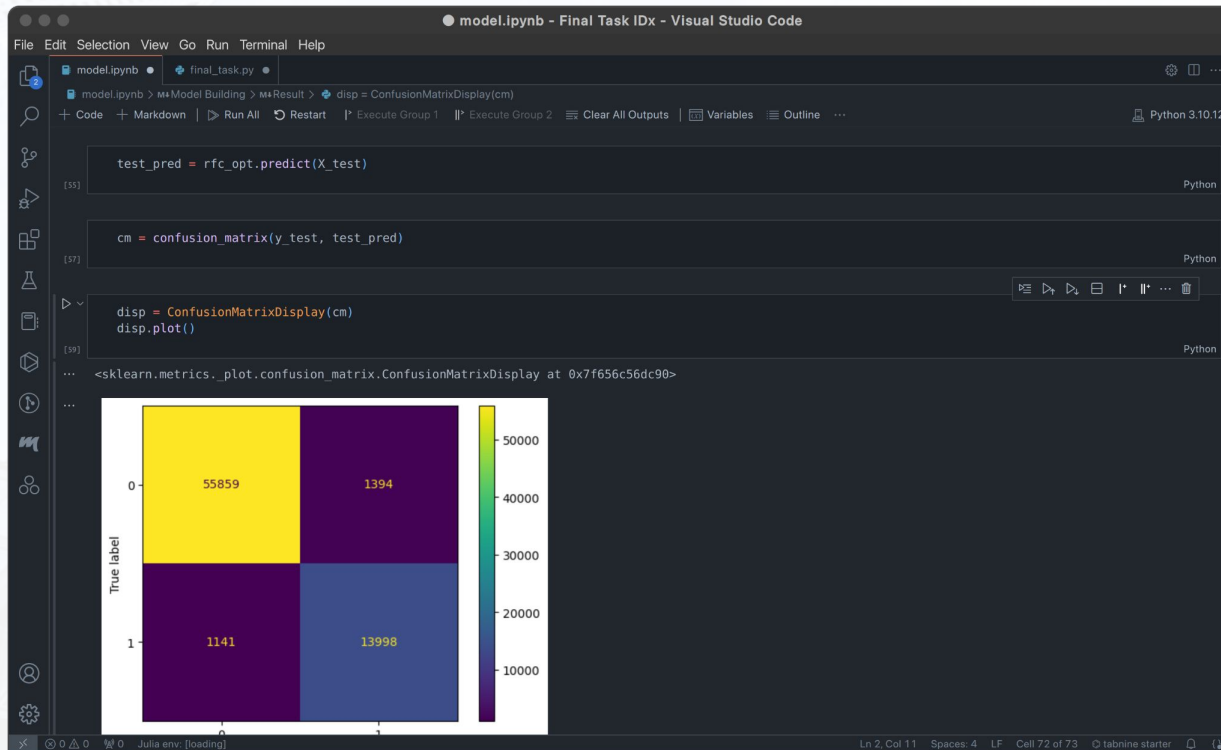
```
1 from sklearn.model_selection import train_test_split, GridSearchCV
2
3 X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size=0.30, random_state=101)
4
5 X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size=0.30, random_state=101)
6
7 cv_params = {'n_estimators' : [50,100],
8              'max_depth' : [10,50],
9              'min_samples_leaf' : [0.5,1],
10             'min_samples_split' : [0.001, 0.01],
11             'max_features' : ["sqrt"],
12             'max_samples' : [.5,.9]}
13
14 from sklearn.ensemble import RandomForestClassifier
15
16 rfc = RandomForestClassifier(random_state=101)
17
18 rfc_val = GridSearchCV(rfc, cv_params, cv=5, refit='f1', n_jobs = -1, verbose = 1)
19
20 rfc_val.fit(X_train, y_train)
21
22 rfc_val.best_params_
23
24 y_vpred = rfc_val.predict(X_val)
```

Final Model

```
1 from sklearn.metrics import confusion_matrix, classification_report, ConfusionMatrixDisplay
2
3 confusion_matrix(y_val, y_vpred)
4
5 disp = ConfusionMatrixDisplay(confusion_matrix(y_val, y_vpred))
6 disp.plot()
7
8 print(classification_report(y_val, y_vpred))
9
10 """### Final Model"""
11
12 rfc_opt = RandomForestClassifier(n_estimators = 100, max_depth = 50,
13                                min_samples_leaf = 1, min_samples_split = 0.001,
14                                max_features="sqrt", max_samples = 0.9, random_state = 101)
15
16 rfc_opt.fit(X_temp, y_temp)
```

Finalisasi model sesuai parameter terbaik yang telah dicari menggunakan Grid search

Result



Model Performance

```
print(classification_report(y_test, test_pred))
```

[62]

```
...           precision    recall  f1-score   support

      0       0.98        0.98        0.98       57253
      1       0.91        0.92        0.92       15139

 accuracy          0.96       72392
 macro avg         0.94        0.95        0.95       72392
 weighted avg      0.97        0.96        0.97       72392
```

Performa akhir model cukup baik,
dengan metrics f1-score 92% dan
juga recall 91%

Github

<https://github.com/revitotan/idx-partners>

Thank You



Rakamin
Academy



id/x partners