CREDIT RISK MODELLING USING PYTHON

BY:RIFQI ARRAYAN

BACKGROUD

In the financial industry, lending to individuals or companies is one of the main sources of income for financial institutions, both banks and financial technology (fintech) companies. However, every lending decision carries risks, one of which is credit risk, where the borrower is unable to fulfill their obligations in accordance with the agreed terms.

To manage this risk, financial institutions need to implement a data-driven approach to assess the creditworthiness of potential borrowers before granting loans. One effective way is to use credit risk modeling, which builds a predictive model to assess the likelihood of someone defaulting based on historical data and borrower characteristics.

With the development of technology and the increasing availability of data, credit risk analysis can be performed more accurately using machine learning techniques in the Python programming language. The utilization of these techniques enables the processing of large amounts of data, the creation of more precise predictive models, and automation in the credit decision-making process.

Tools

GOOGLE COLAB

Helps to create and visualize interactive dashboards to generate useful insights.

PYTHON

to view the overall data easily

METADATA ON COLUMNS USED

TERM :Indicates how long it takes someone to borrow until they pay off their loan (in months).

emp_lenght :Describes the duration a person works in a particular company or field.

issue_d :The date on which the loan is granted to the borrower.

earliest_cr_line :The date on which a person first had credit or loan history.

last_pymt_d :The date when a person last made a payment against their loan.

next_pymt_d :Next scheduled payment date.

las_credit_pull_d:The date on which a financial institution last evaluated a person's credit status.

TABLE OF CONTENT

INTRODUCE TO DATASET

The dataset is taken from kaggle with detail dataset:

The LendingClub Issued Loans dataset contains data on loans granted by LendingClub, a peer-to-peer (P2P) lending platform. The dataset includes information on loan characteristics such as loan amount, interest rate, loan duration, purpose of use, as well as borrower data such as credit score, annual income, and employment status.

Link dataset below:

Link dataset

Rows

759.338

Columns

72

- **Target variable:**

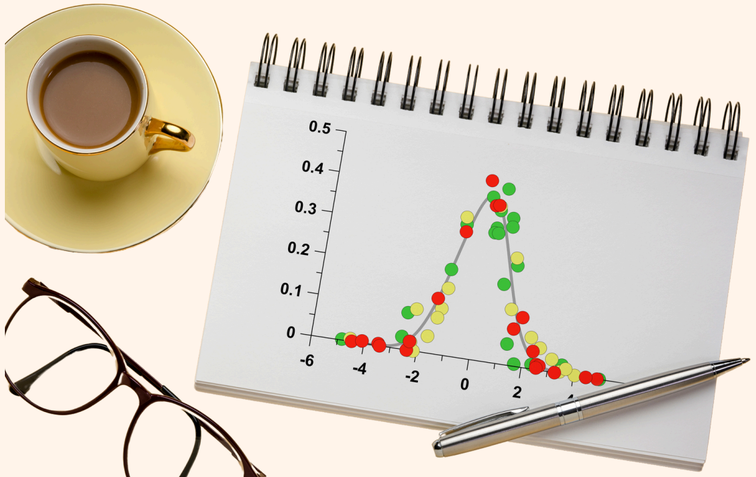
Current, Fully Paid, In Grace Period, Late (31-120 days), Late (16-30 days), Charged Off, Default

- **Label classification**

If the loan_status is 'Charged Off', 'Default', 'Late (31-120 days)', 'Late (16-30 days)' will be considered as bad_loan or (1) and other values will be considered as good loan or (0).

- **Distribute 0 and 1**

Label 0 is 702875 and label 1 is 56463



HANDLING MISSING VALUE

	0
member_id	1.000000
desc	0.999978
dti_joint	0.955223
annual_inc_joint	0.955220
verification_status_joint	0.955220
mths_since_last_record	0.814076
mths_since_last_major_derog	0.716720

```
missing_values = pd.DataFrame(loan_data.isnull().sum()/loan_data.shape[0])
missing_values = missing_values[missing_values.iloc[:,0] > 0.50]
missing_values.sort_values([0], ascending=False)
```

```
loan_data.dropna(thresh = loan_data.shape[0]*0.5, how='all', axis=1, inplace=True)
```

The picture on the side is data that has missing values above 50% and will be deleted because if you want to fill it with other values such as median or mean, the error will be very high. It is better to drop it so as not to make the model even more inaccurate.

DATA SPLITTING

```
X = loan_data.drop('good_bad', axis=1)
y = loan_data['good_bad']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, stratify= y, random_state=42)
```

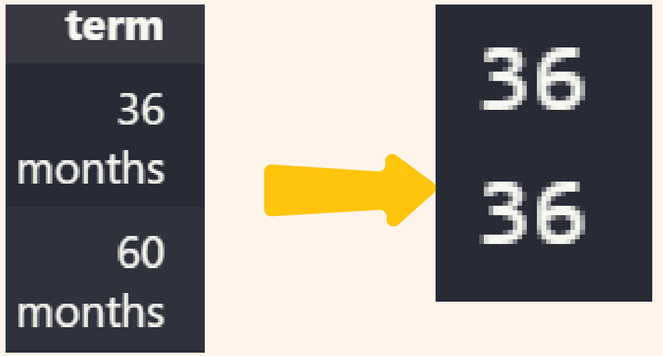
```
y_train.value_counts(normalize=True)
```

In the picture above, I divided the data into 80% data for training and 20% for testing.
And also counted the number of occurrences of each unique value in y_train and converted the absolute number into a proportion (on a scale of 0 to 1).

DATA CLEANING

- **Colum Term**

Eliminate or remove the month and change data type to numeric



- **Emp_lenght,issue_d,earliest_cr_line,last_pymnt_d, next_pymnt_d,last_credit_pull_d**

Remove the year sentence and change the data type to numeric

FEATURE ENGINEERING

```python
# feature engineering untuk date columns
def date_columns(df, column):
    today_date = pd.to_datetime(date.today().strftime('%Y-%m-%d'))
    df[column] = pd.to_datetime(df[column], format = "%b-%y")
    df['mths_since_' + column] = round(pd.to_numeric((today_date - df[column]) / np.timedelta64(1, 'M')))
    df.drop(columns = [column], inplace=True)

# apply to X_train
date_columns(X_train, 'earliest_cr_line')
date_columns(X_train, 'issue_d')
date_columns(X_train, 'last_pymnt_d')
date_columns(X_train, 'last_credit_pull_d')
```

```python
# apply to X_test
date_columns(X_test, 'earliest_cr_line')
date_columns(X_test, 'issue_d')
date_columns(X_test, 'last_pymnt_d')
date_columns(X_test, 'last_credit_pull_d')
```

Columns that will be feature engineered
term, emp_length, issue_d, earliest_cr_line, last_pymnt_d, next_pymnt_d, last_credit_pull_d

- Date_columns(df, column) function:

Converts a date column to datetime format (‘%b-%y’, for example ‘Jan-22’).
Calculates the number of months from that date to today.
Removes the original column and replaces it with a new feature (mths_since_<column>).

- Applied to X_train and X_test:

The earliest_cr_line, issue_d, last_pymnt_d, and last_credit_pull_d columns are converted to the number of months since the event.
This helps the model understand time information in numerical form which is more useful for analysis and prediction.

- Function:

Improve the quality of the features so that the model can capture patterns based on credit age, last payment time, and last credit withdrawal time.

MODELLING

This model uses Logistic Regression to predict credit risk based on training (y_train) and testing (y_test) data. Model evaluation is done using accuracy and confusion matrix

MODEL 1

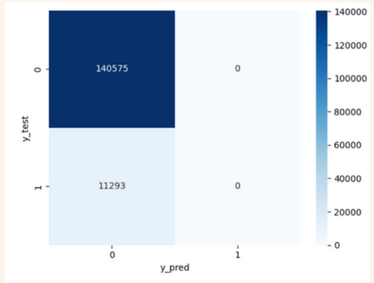
- **Model and predict**

	y_pred	y_test
0	0	1
1	0	0
2	0	0
3	0	0
4	0	0

Most predictions are 0 (meaning the credit is considered safe), although there are some cases that are actually 1 (high credit risk).

- **Model evaluation and confusion matriks**

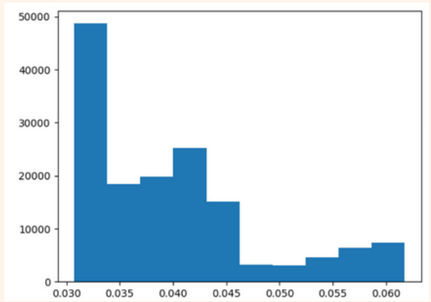
Accuracy = 92.56% → The model has high performance



- The model never predicts class 1 (Bad Loan), as it only predicts class 0.
- False Negative (FN) and True Positive (TP) are zero, which means the model does not recognize any Bad Loan at all.
- The number of False Negative (0,1) of 11,293 indicates that there are 11,293 cases that should be categorized as Bad Loan but are predicted as Good Loan.
- The True Negative (TN) of 140,575 means that the model is quite good at recognizing Good Loan, but ignores Bad Loan.

MODEL 2

- **Predicted Probability & Threshold Default (0.5)**



- The next figure is the probability distribution of the model's prediction of a variable while still using the default threshold of 0.5.
- The model tends to give a very low probability for the "Bad Loan" class.
- This is an indication that the model is not sensitive to "Bad Loan", probably due to the imbalance dataset and to solve this I will use ROC Curve and Youden's J-Statistic.

- **Threshold Optimisation with Youden's J-Statistic**

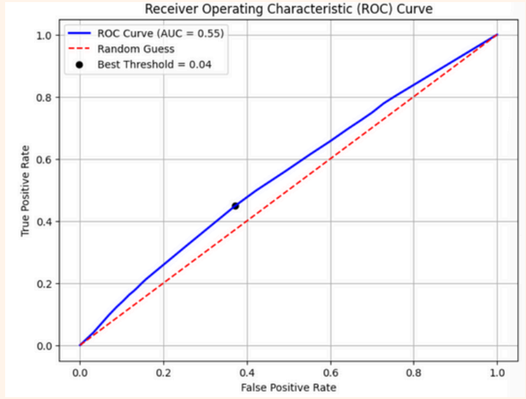
```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, y_pred)
```

```
# youden j-statistic
j = tpr - fpr

ix = np.argmax(j)

best_thresh = thresholds[ix]
best_thresh
```

The ROC Curve was used to evaluate the performance of the model at various thresholds.
Youden's J-Statistic (TPR - FPR) was used to find the optimal threshold, where the balance between TPR and FPR is maximised.
Best Threshold = 0.041 (no longer 0.5).

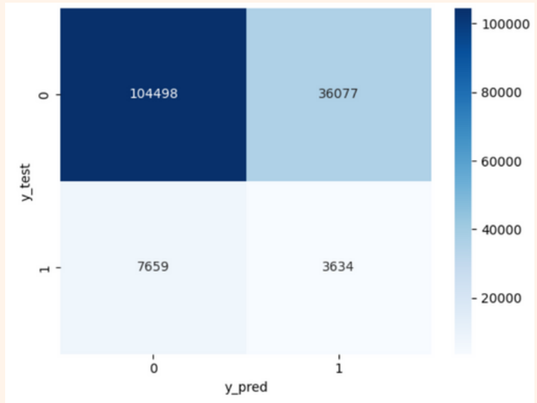
MODELLING

Receiver Operating Characteristic (ROC) Curve



- The ROC Curve provides a clear picture of how well the model distinguishes between borrowers who default and those who do not default at various threshold values.
- A threshold of 0.04 increases the True Positive Rate (TPR), but also increases the False Positive Rate (FPR), which means the model predicts non-defaults as defaults more often. This leads to a trade-off between recall and precision, and the decision to choose the best threshold depends on how much risk the financial institution is prepared to take in terms of misprediction.

Model Evaluation with New Threshold (0.042)



- True Negatives (TN) dropped from 140,575 to 104,498.
- False Positives (FP) increased significantly, from 0 to 36,077.
- True Positives (TP) appeared, which is 3,634 correct predictions for class 1.
- False Negatives (FN) reduced drastically, from 11,293 to 7,659.

Comparison with Previous Models

Metric	Tanpa Threshold ROC	Setelah Threshold 0.042
True Negatives (TN)	140,575	104,498
False Positives (FP)	0	36,077
True Positives (TP)	0	3,634
False Negatives (FN)	11,293	7,659

- True Negatives (TN):

Decreased after the threshold was changed, as the model now predicts more examples as 1 (positive), even though they are actually 0 (negative).

- False Positives (FP):

Increased significantly. This indicates that the model is starting to predict more data as 1, even though they should be 0.

- True Positives (TP):

Appeared for the first time. With an ROC threshold of 0.042, the model started to successfully classify some positive examples correctly (3,634 TPs).

- False Negatives (FN):

Decreased, meaning less data that should have been positive (1) but was predicted as negative (0).

Conclusion modelling

- The use of an ROC threshold of 0.042 reduced the number of False Negatives, which means the model started to correctly identify some risky loans (True Positives).
- However, the number of False Positives increased significantly, meaning the model became more sensitive and predicted more data as risky (class 1), even though they were not actually risky (class 0).
- True Negatives decreased, which is a result of the lower threshold, which made the model predict more positive classes.

CONCLUSION

	term	emp_length	mths_since_earliest_cr_line	mths_since_issue_d	mths_since_last_pymnt_d	mths_since_last_credit_pull_d
0	0.017271	-0.031557	-0.001035	0.045839	0.234735	-0.15285

TERM (0.017271):
The positive coefficient (0.017271) indicates that the longer the loan term (the higher the TERM value), the higher the probability of default.
EMP_LENGTH (-0.031557):
The negative coefficient (-0.031557) indicates that the longer one works (the higher the emp_length value), the lower the probability of default.
MTHS_SINCE_EARLIEST_CR_LINE (-0.001035):
The negative coefficient (-0.001035) indicates that the longer the age of the first credit line opened, the lower the probability of default.
MTHS_SINCE_ISSUE_D (0.045839):
The positive coefficient (0.045839) indicates that the longer the time since the loan was issued, the higher the probability of default.
MTHS_SINCE_LAST_PYMT_D (0.234735):
A positive coefficient (0.234735) indicates that the longer the time since the last payment was made, the higher the probability of default.
Conclusion: If payments are often late or stopped, the probability of default is higher.
MTHS_SINCE_LAST_CREDIT_PULL_D (-0.15285):
The negative coefficient (-0.15285) indicates that the longer the time since the last credit pull, the lower the probability of default.
This could suggest that the more frequently credit is checked or updated, the more likely it is to detect financial problems early, thus reducing the risk of default.


VIEW IN GITHUB

THANK YOU