Id/x Partners - Rakamin Academy

CREDIT RISK MODELING

By : Muhamad Albani Syahril

TABLE OF CONTENT

BUSINESS UNDERSTANDING

Loans are one of the key elements in the financial system that allows individuals, businesses, and institutions to obtain the necessary funds for investment, development, or immediate needs.

In the modern economy, banks are one of the main institutions that provide lending facilities for various purposes.

Objective :

Build a machine learning model to predict the customer's ability to repay a loans.

Benefit :

- Help company in making decisions to provide loans to customers or not based on the output model.
- Minimize losses due to bad credit or failure to pay.

ABOUT DATASET

- Columns : 75
- Rows : 466285
- Numerical Columns : 53
- Non-Numerical Columns : 22
- Missing Value : 9776224
- Duplicate Value : 0

DATA PREPROCESSING

Drop irrelevant data and has no effect on the training and inference model.

Handling Missing Values

Fixing The Incorrect Data Types

Data Normalization

Encoding Categorical Columns

DROP IRRELEVANT DATA AND HAS NO EFFECT ON THE TRAINING AND INFERENCE MODEL

There are many columns that irrelevant and doesn't have effect for modeling (training and inference) i must drop, for example : Unnamed: 0, id, member_id, Unnamed: 0, id, member_id, sub_grade, issue_d, pymnt_plan, emp_title, url, desc, etc.

```
[9]  df.drop(columns=['Unnamed: 0', 'id', 'member_id', 'sub_grade', 'issue_d', 'pymnt_plan', 'emp_title', 'url'
```

HANDLING MISSING VALUES

DROP

Removing columns that have more than 50% missing values.

annual_inc

This column just have 4 missing value and i think the best way to fill it are with the median value of the column.

revol_util

This column has quite a lot of missing values, and since this column has a little relationship with the revol_bal column,so i think the best way are fill revol_util with 0 if the value in revol_bal is 0, and fill revol_util with its median value if revol_bal is more than 0.

emp_length

This column has a lot of missing value, because this column contains employment length in years, and it will be weird if i fill it with mean or median or mode, so the best way are fill it with 0.

DROP

delinq_2yrs, earliest_cr_line, inq_last_6mths, open_acc, pub_rec, the missing value in these column are in the same rows, so i drop these rows.

HANDLING MISSING VALUES

collections_12_mths_ex_med

This column has quite a lot of missing values, and since this column dominated by the values of 0, so i think the best way is to fill it with 0.

DROP

tot_coll_amt, tot_cur_bal, total_rev_hi_lim, these columns has ~15% missing value, and the distribution data of these column are balance except for tot_coll_amt, so i think the best way is just drop these columns, given the large number of missing values.

No more missing values

```
loan_amnt                   0
funded_amnt                 0
funded_amnt_inv             0
term                        0
int_rate                    0
installment                 0
grade                       0
emp_length                  0
home_ownership              0
annual_inc                  0
verification_status         0
purpose                     0
dti                         0
delinq_2yrs                 0
earliest_cr_line            0
inq_last_6mths              0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                  0
total_acc                   0
initial_list_status         0
out_prncp                   0
out_prncp_inv               0
collections_12_mths_ex_med  0
acc_now_delinq              0
loan_category               0
dtype: int64
```

FIXING THE INCORRECT DATA TYPES

term

Since this column contains value of month, i decided to change the data type from object to int and remove word ‘months’ after the number of months

emp_length

Since this column contains employment length in years, i decided to change the data type from object to int and remove every string except the number

DATA NORMALIZATION

Filtered the columns that contain value more than 1000 and it turns out there are loan_amnt, annual_inc, revol_bal, out_prncp columns that the value are more than 1000, then normalize it using MinMaxScaler() method from sklearn.

The goals of normalize the data is so the model can better to process the data when training and enhancing model performance.

loan_amnt	annual_inc	revol_bal	out_prncp
5000.0	24000.0	13648.0	NaN
2500.0	30000.0	1687.0	NaN
2400.0	12252.0	2956.0	NaN
10000.0	49200.0	5598.0	NaN
3000.0	80000.0	27783.0	NaN
...	...	...	...
18400.0	110000.0	23208.0	12574.00
22000.0	78000.0	18238.0	NaN
20700.0	46000.0	6688.0	14428.31
2000.0	83000.0	11404.0	NaN
10000.0	46000.0	11325.0	3984.38

loan_amnt	annual_inc	revol_bal	out_prncp
0.130435	0.002948	0.005313	0.000000
0.057971	0.003748	0.000657	0.000000
0.055072	0.001381	0.001151	0.000000
0.275362	0.006309	0.002179	0.000000
0.072464	0.010416	0.010815	0.023846
...	...	...	...
0.518841	0.014418	0.009034	0.390978
0.623188	0.010150	0.007099	0.000000
0.585507	0.005882	0.002603	0.448636
0.043478	0.010817	0.004439	0.000000
0.275362	0.005882	0.004408	0.123891

ENCODING CATEGORICAL COLUMNS

Good Loan :

- Fully Paid
- Does not meet the credit policy. Status:Fully Paid
- Current
- In Grace Period

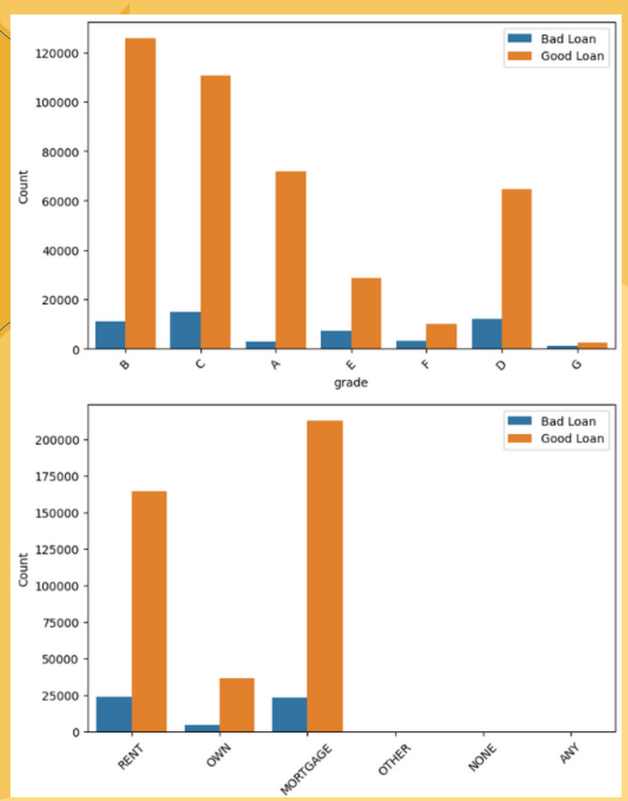
Bad Loan :

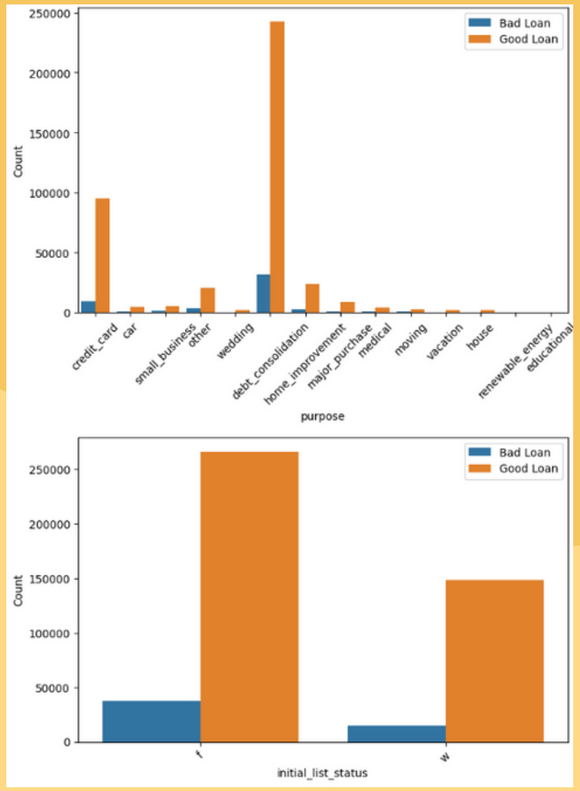
- Charged Off
- Late (31-120 days)
- Late (16-30 days)
- Default
- Does not meet the credit policy. Status:Charged Off

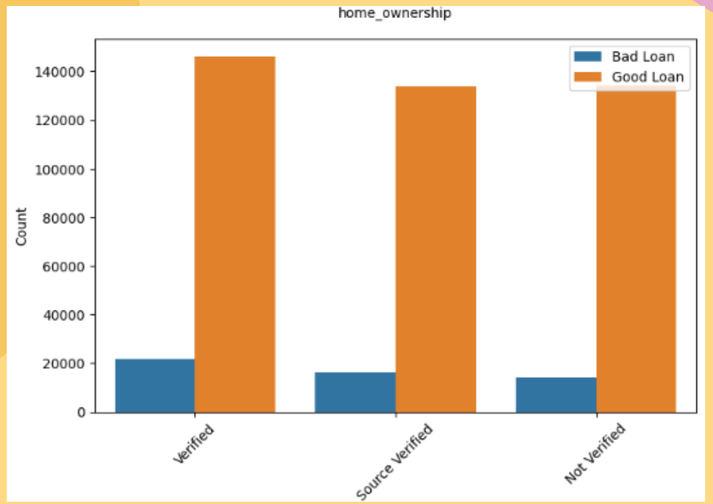
	loan_status	loan_category
0	Fully Paid	1
1	Charged Off	0
2	Fully Paid	1
3	Fully Paid	1
4	Current	1

ENCODING CATEGORICAL COLUMNS

loan_category based on categorical columns :







ENCODING CATEGORICAL COLUMNS

One Hot Encoding and feature selection :

Example :

```
Value dari kolom Grade : ['A' 'B' 'C' 'D' 'E' 'F' 'G']
```

grade_A	grade_B	grade_C	grade_D	grade_E	grade_F	grade_G
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0

Final result :

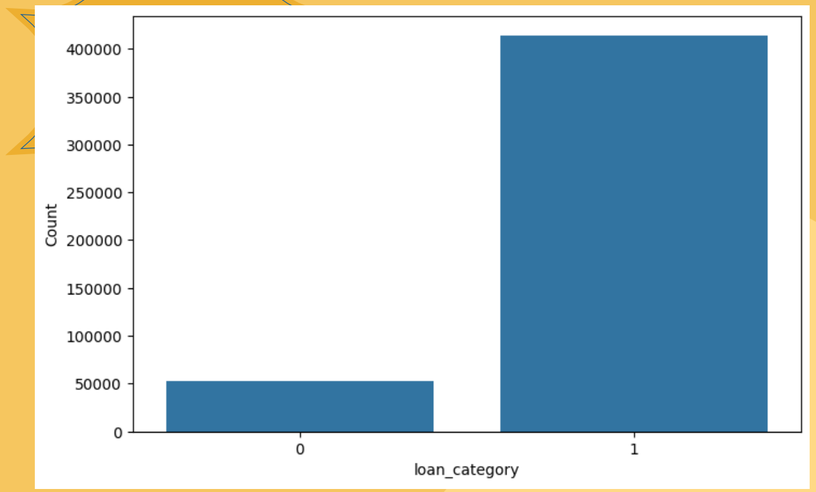
```
0   loan_amnt
1   funded_amnt
2   funded_amnt_inv
3   term
4   int_rate
5   installment
6   grade
7   emp_length
8   home_ownership
9   annual_inc
10  verification_status
11  purpose
12  dti
13  delinq_2yrs
14  earliest_cr_line
15  inq_last_6mths
16  open_acc
17  pub_rec
18  revol_bal
19  revol_util
20  total_acc
21  initial_list_status
22  out_prncp
23  out_prncp_inv
24  collections_12_mths_ex_med
25  acc_now_delinq
26  loan_category
```

```
0   loan_amnt
1   annual_inc
2   revol_bal
3   out_prncp
4   term
5   int_rate
6   installment
7   emp_length
8   dti
9   delinq_2yrs
10  earliest_cr_line
11  inq_last_6mths
12  open_acc
13  pub_rec
14  revol_util
15  total_acc
16  collections_12_mths_ex_med
17  acc_now_delinq
18  loan_category
19  grade_A
20  grade_B
21  grade_C
22  grade_D
23  grade_E
24  grade_F
25  grade_G
26  home_ownership_ANY
27  home_ownership_MORTGAGE
28  home_ownership_NONE
29  home_ownership_OTHER
30  home_ownership_OWN
31  home_ownership_RENT
32  verification_status_Not Verified
33  verification_status_Source Verified
34  verification_status_Verified
35  purpose_car
36  purpose_credit_card
37  purpose_debt_consolidation
38  purpose_educational
39  purpose_home_improvement
40  purpose_house
41  purpose_major_purchase
42  purpose_medical
43  purpose_moving
44  purpose_other
45  purpose_renewable_energy
46  purpose_small_business
47  purpose_vacation
48  purpose_wedding
49  initial_list_status_f
50  initial_list_status_w
```

EXPLORATORY DATA ANALYSIS

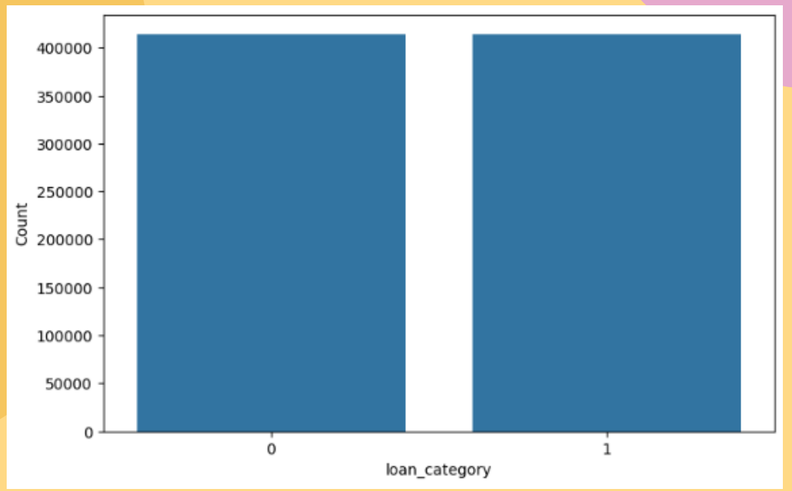
Imbalance Dataset :

Label



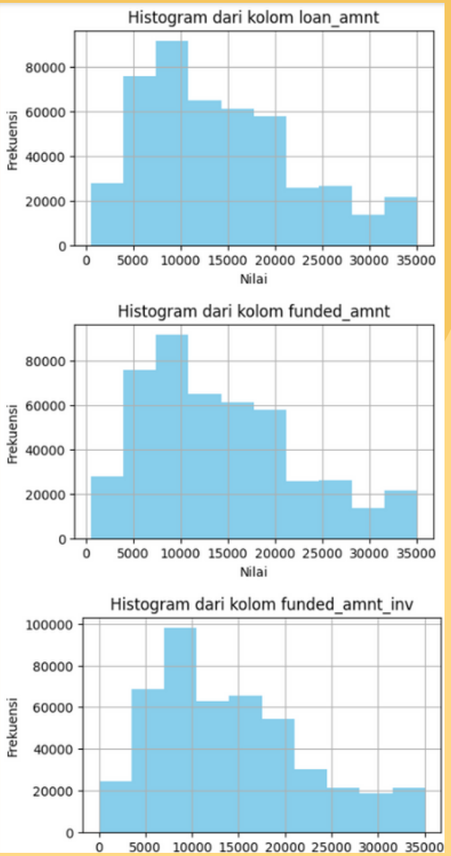
Use SMOTE method oversampling

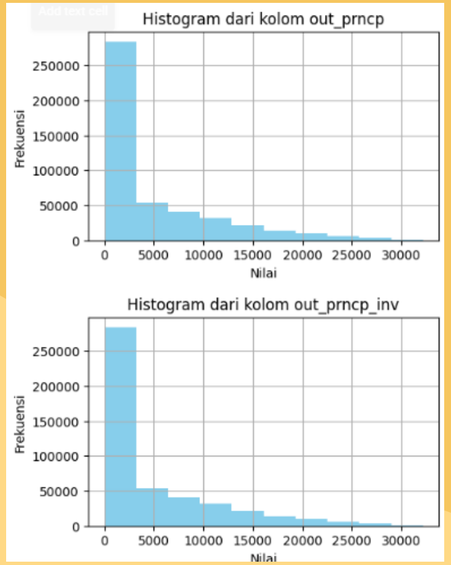
Final label



EXPLORATORY DATA ANALYSIS

Same frequency value columns :



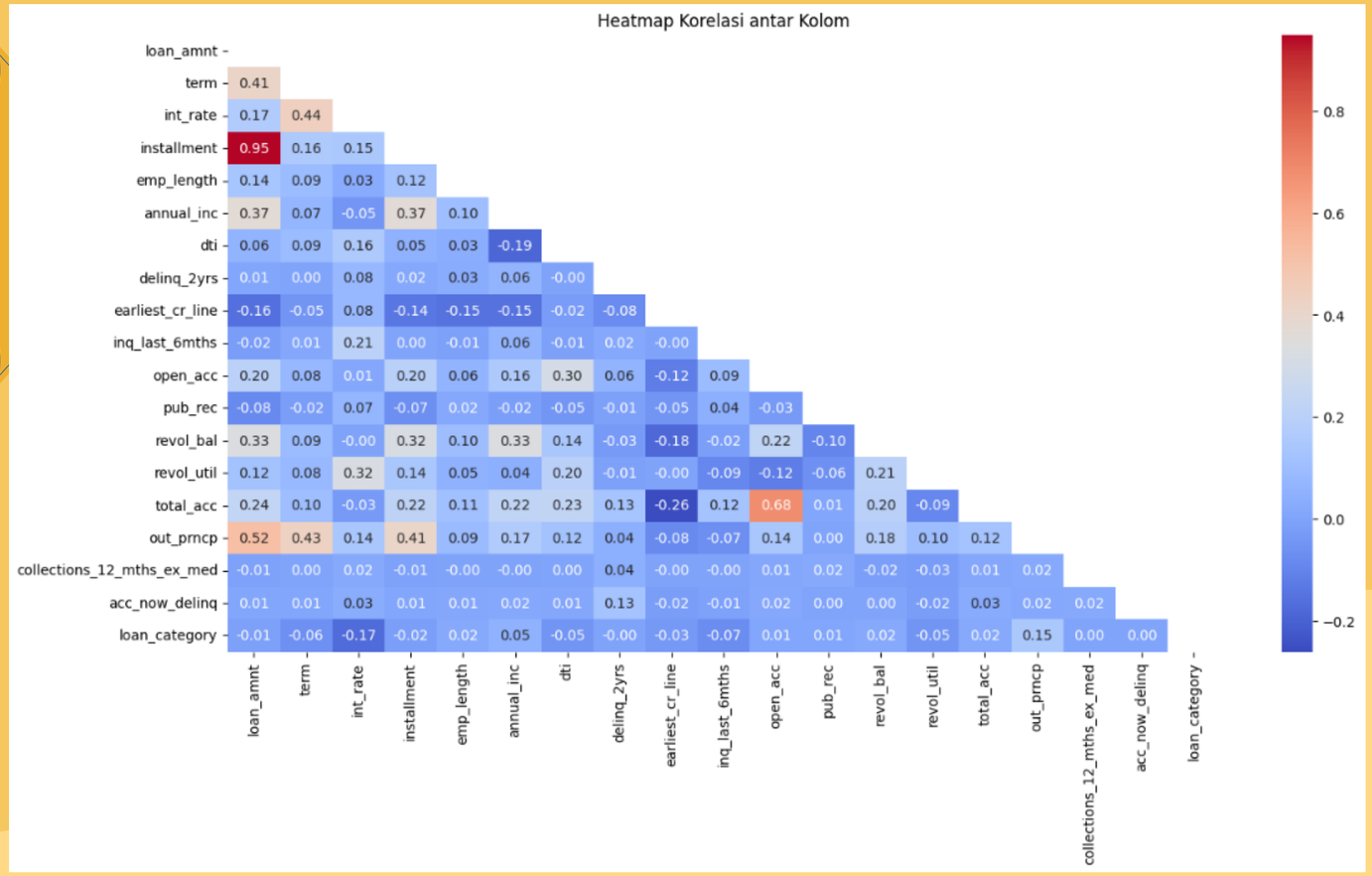


Drop the same frequency value columns and leave just one

```
df.drop(columns=['funded_amnt','funded_amnt_inv', 'out_prncp_inv'], inplace=True)
```

EXPLORATORY DATA ANALYSIS

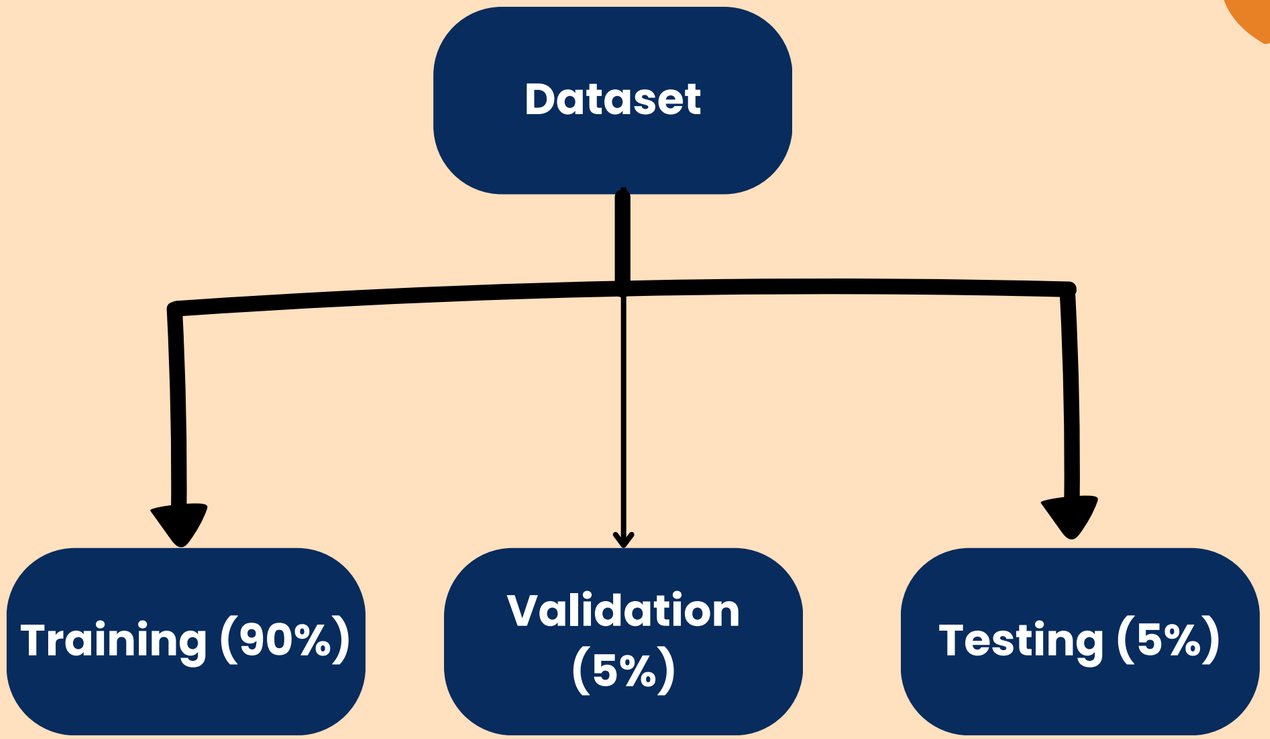
correlation matrix on numerical columns :



installment, loan_amnt and open_acc, total_acc columns have a strong correlation each other, and the rest of the columns have a low correlation each other.

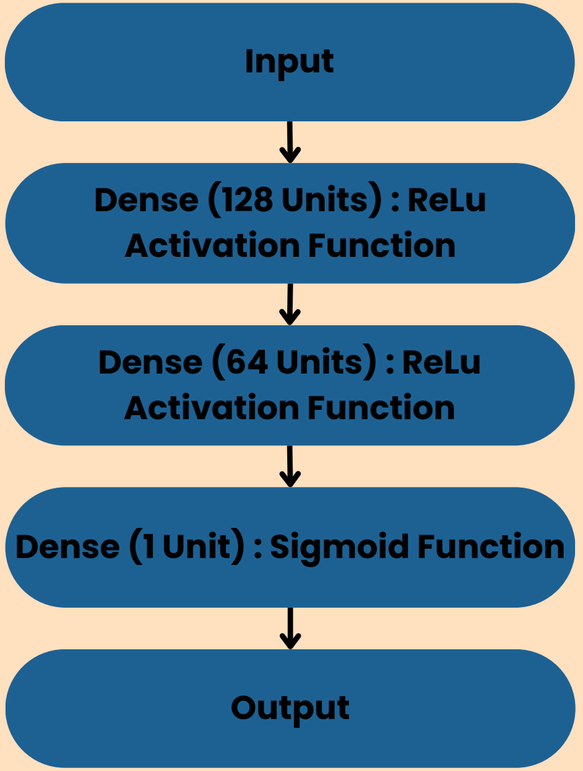
MODELING

Split dataset for training, validation and testing



MODELING

Building deep learning model architecture for binary classification using TensorFlow

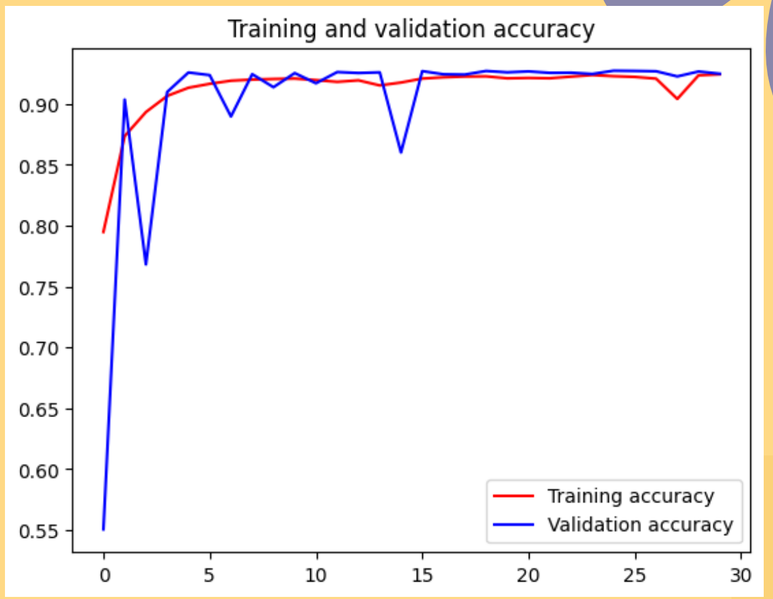


Compiler :

- optimizer = adam
- loss = binary_crossentropy
- metrics = accuracy

After finish build the model architecture, time to feed the training and validation set to the model

EVALUATION

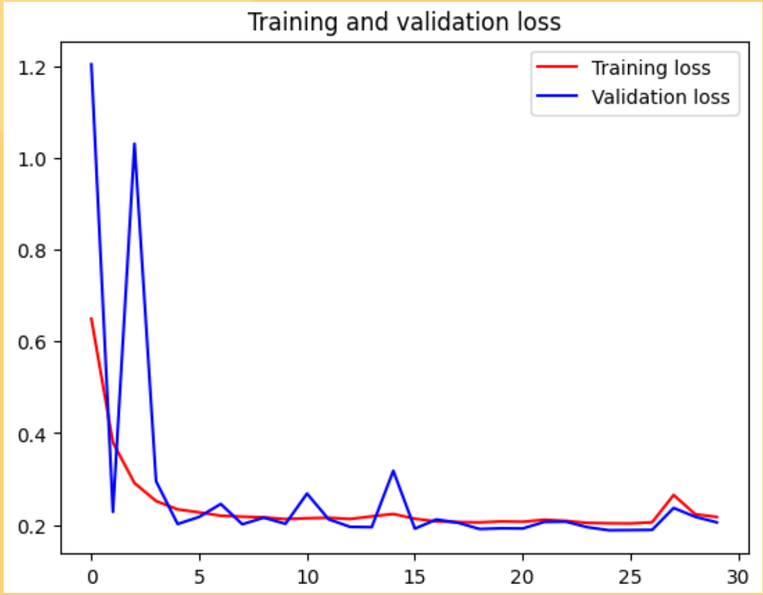


As we can see in the line chart, the loss and accuracy of model are very good for just 30 epochs, but in my observation based on the line chart, we can get the optimal model just for 20 epochs.

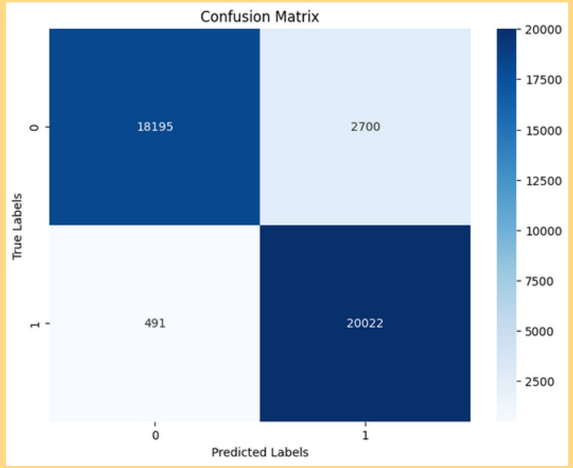
The result is very good when evaluate the model using testing data that hasn't seen by the model.

Evaluate using testing data

```
Loss: 0.20919576287269592
Accuracy: 0.9229375720024109
```



EVALUATION



Based on the evaluation metrics, the model is better at classifying positive data than negative data.

	precision	recall	f1-score	support
0	0.97	0.87	0.92	20895
1	0.88	0.98	0.93	20513
accuracy			0.92	41408
macro avg	0.93	0.92	0.92	41408
weighted avg	0.93	0.92	0.92	41408

CONCLUSION

1

Model accuracy has reach 92% when training and 92% accuracy when evaluate it using testing data, it’s indicates the model can generalize data that hasn’t been seen properly.

2

Deep learning are the powerful method for training machine learning model, but the drawback is that it requires a very large datasets.

THANK YOU