TITAN BAGUS BRAMANTYO

CREDIT RISK CLASSIFICATION MODEL

Final Task - Data Scientist Virtual Internship Experience at ID/X Partners

RAKAMIN ACADEMY

Table of Content

Outline of this presentation

Background

brief presentation on the topic of analysis

Problem & Solution

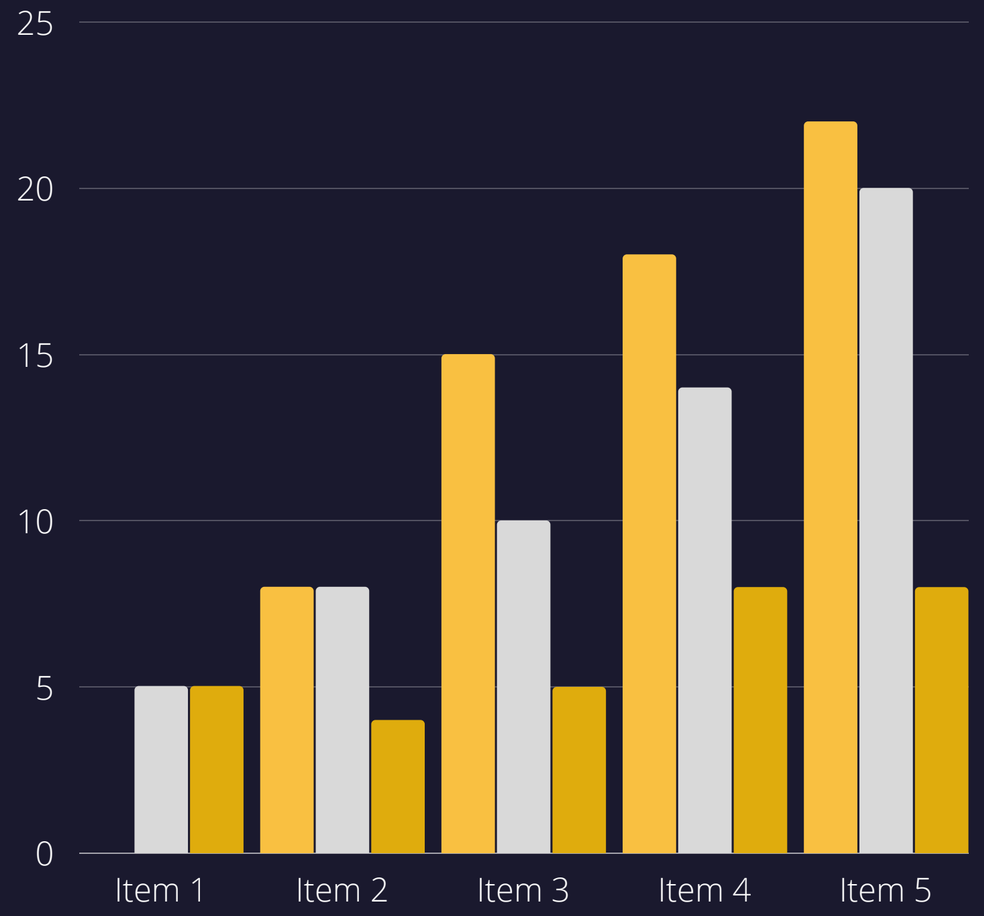
overview of the analyzed problem and proposed solution

Technical Explanation

technical process such as data preparation, data understanding, data preprocessing, modeling, etc.

Conclusion

Final explanation about the model result, etc.



Background

Determining the eligibility of loan receivers is essential for banks or other financial services. If a financial institution misleadingly examines the feasibility of a prospective loan receiver, the business can suffer. A mechanism which could quickly and massively determine feasibility is also necessary in order to create effectiveness and efficiency.

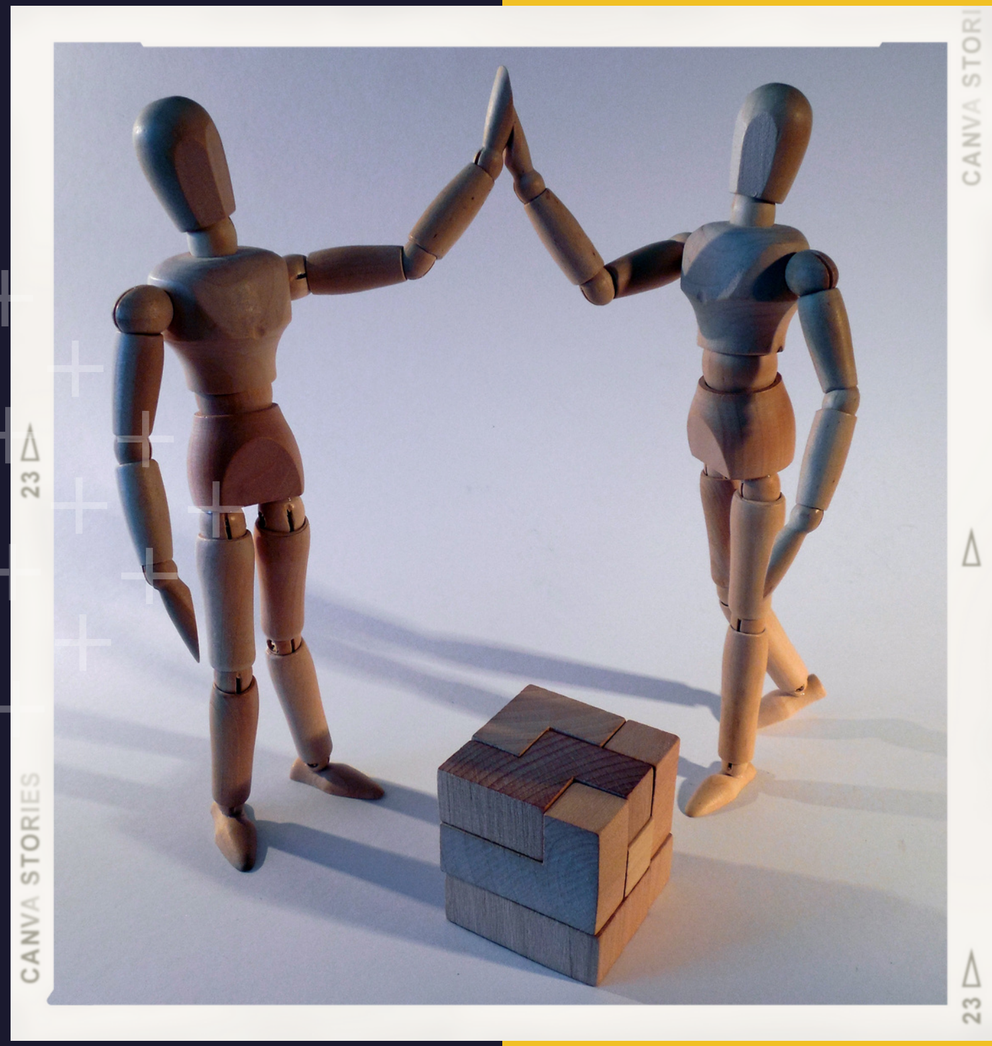
READ MORE

PROBLEM

The number of loan applications is extremely high. while the financial institution should evaluate the applicant's eligibility first before determining loan approval. On the other hand, financial institutions must be able to respond quickly to loan applications.

SOLUTION

With the evolution of technology, machine learning is possible to handle these problems. The model that will be used is the classification



~

FEATURE / COLUMN

The dataset contains 74 features or columns with various data type.

~

DATA RECORD

The dataset contains 466.285 records. And the dataset still contains many missing value

ABOUT DATASET

The dataset that I use is the dataset provided by Rakamin Academy on its platform. The dataset title is Lending Credit Club from 2007 - 2014.

Selected Features

loan_amount

The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value

funded_amount

The total amount committed to that loan at that point in time

emp_length

Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years

last_pymnt_amount

Last total payment amount received

loan_status

Current status of the loan

home_ownership

The home ownership status provided by the borrower during registration. Our values are: RENT, OWN, MORTGAGE, OTHER

annual_income

The self-reported annual income provided by the borrower during registration

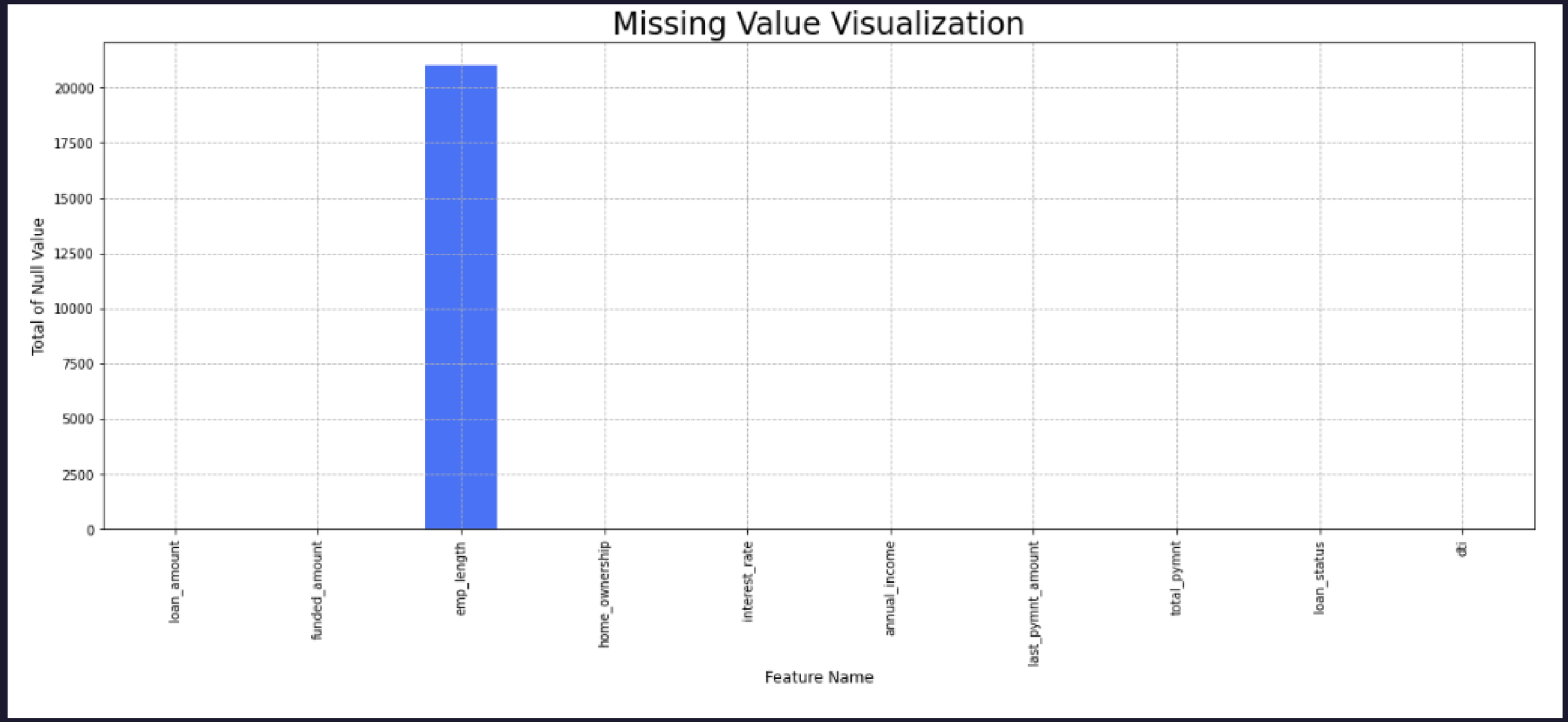
total_pymnt

Payments received to date for total amount funded

dti

A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income

MISSING VALUE CHECK



CHECK THE MISSING VALUE PERCENTAGE

```
# check the percentage between missing value and total data in 'emp_length'
propo = df['emp_length'].isnull().sum()/df.shape[0]

print("'emp_length' missing value percentage to total data :", propo)

```

The result is 0.04505, it means not too significant. So I decide to delete those rows

TRANSFORM

Some features still in non-numerical data type. ML only can process numerical data, so I'll transfrom those features

emp_length

contains data like "10 years, < 3 years, etc"

home_ownership

still contains 'RENT', 'OWN', 'MORTGAGE', 'OTHER', 'NONE', 'ANY'

loan_status

still contains "Charged Off", "Default", "Does not meet the credit policy. Status:Charged Off" " etc

EMP_LENGTH TRANSFORM

```
# 'emp_length' conversion to int

employment_length = ['10+ years', '< 1 year', '1 year', '3 years', '8 years', '9 years',
                     '4 years', '5 years', '6 years', '2 years', '7 years', 'n/a']

# Create a new column and convert emp_length to integers.

lst = [df]
df['emp_length_int'] = np.nan

for col in lst:
    col.loc[col['emp_length'] == '10+ years', "emp_length_int"] = 10
    col.loc[col['emp_length'] == '9 years', "emp_length_int"] = 9
    col.loc[col['emp_length'] == '8 years', "emp_length_int"] = 8
    col.loc[col['emp_length'] == '7 years', "emp_length_int"] = 7
    col.loc[col['emp_length'] == '6 years', "emp_length_int"] = 6
    col.loc[col['emp_length'] == '5 years', "emp_length_int"] = 5
    col.loc[col['emp_length'] == '4 years', "emp_length_int"] = 4
    col.loc[col['emp_length'] == '3 years', "emp_length_int"] = 3
    col.loc[col['emp_length'] == '2 years', "emp_length_int"] = 2
    col.loc[col['emp_length'] == '1 year', "emp_length_int"] = 1
    col.loc[col['emp_length'] == '< 1 year', "emp_length_int"] = 0.5
    col.loc[col['emp_length'] == 'n/a', "emp_length_int"] = 0
```

HOME_OWNERSHIP TRANSFORM

```
# conversion to int
mapping = {'OWN' : 'OWNED', 'MORTGAGE' : 'OWNED', 'RENT' : 'NONE', 'OTHER' : 'NONE',
            'ANY' : 'NONE'}
df['home_ownership'] = df['home_ownership'].map(mapping).factorize()[0]
```

LOAN_STATUS TRANSFORM

```python
# Set the status loan to 'good' or 'bad' category
bad_loans = ["Charged Off", "Default", "Does not meet the credit policy. Status:Charged Off", "In Grace Period",
             "Late (16-30 days)", "Late (31-120 days)"]
good_loans = [status for status in df['loan_status'].unique() if status not in bad_loans]

# transform to --> bad:1, good:0
mapping = dict(zip(bad_loans + good_loans, [1]*len(bad_loans) + [0]*len(good_loans)))

df['loan_condition'] = df['loan_status'].map(mapping)
```

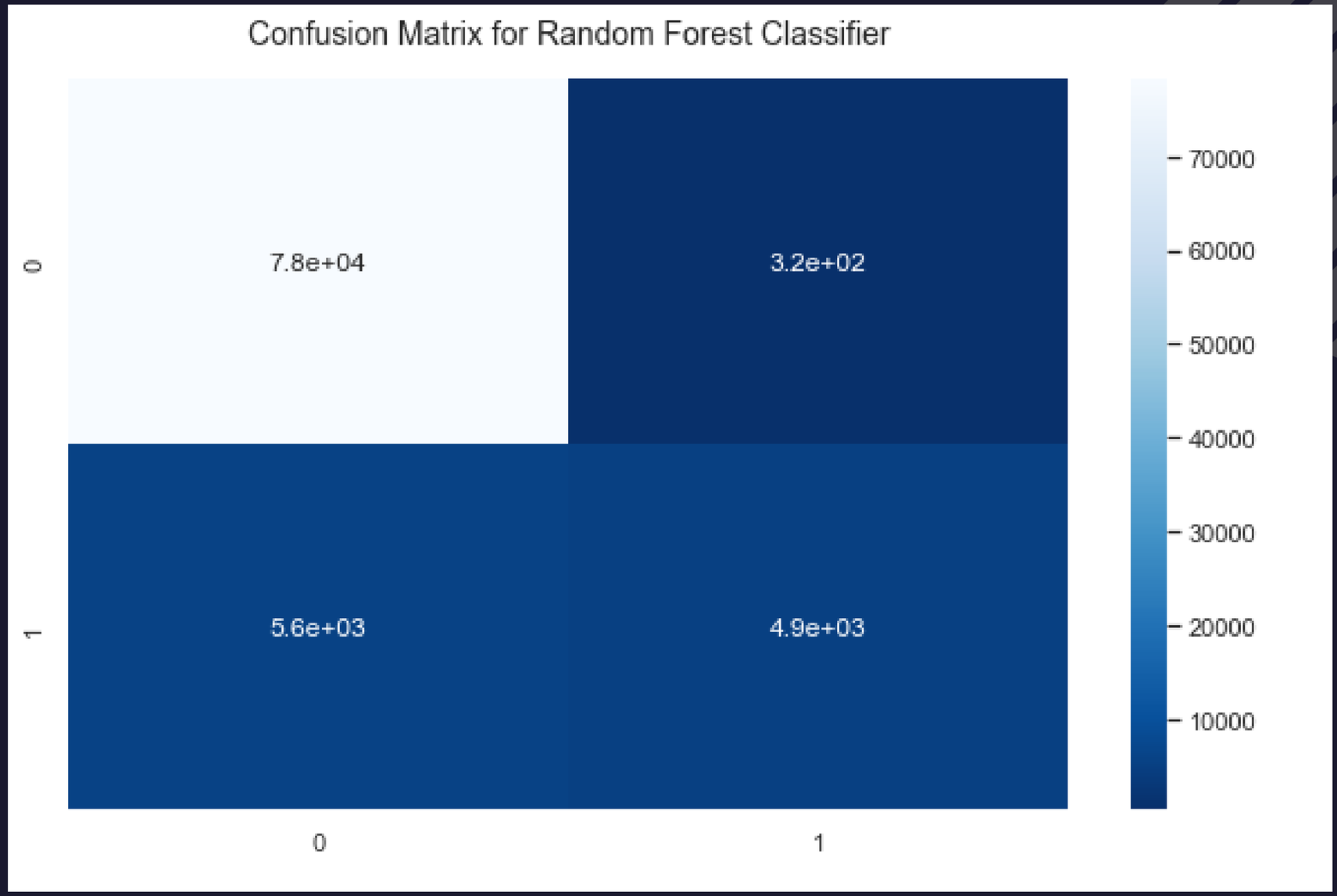
Classification Model

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import *

import warnings
warnings.filterwarnings('ignore')

X = df.drop('loan_condition', axis=1)
y = df['loan_condition']

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

Random Forest Clf.

```
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)

RandomForestClassifierScore = rfc.score(X_test, y_test)
print("Accuracy obtained by Random Forest Classifier :",
RandomForestClassifierScore*100)

y_pred_rfc = rfc.predict(X_test)
cf_matrix = confusion_matrix(y_test, y_pred_rfc)
sns.set(rc={"figure.figsize":(10, 6)})
sns.heatmap(cf_matrix, annot=True, cmap="Blues_r")

plt.title("Confusion Matrix for Random Forest Classifier", fontsize=14,
y=1.03)

from sklearn import metrics
print(metrics.classification_report(y_test, y_pred_rfc))
```

Evaluation Matrix



Evaluation Matrix

	precision	recall	f1-score	support
0	0.93	1.00	0.96	78580
1	0.94	0.47	0.62	10475
accuracy			0.93	89055
macro avg	0.94	0.73	0.79	89055
weighted avg	0.93	0.93	0.92	89055

Closing

Conclusion

With the help of machine learning, loan receiver eligibility assessment problems can be solved using a classification learning model. In this project I used the Random Forest algorithm which achieve an accuracy rate of 93%. In the future, other algorithms can be used for comparison.