Rakamin Academy

id/x partners

Project-Based Internship Id/x Partners & Rakamin Academy

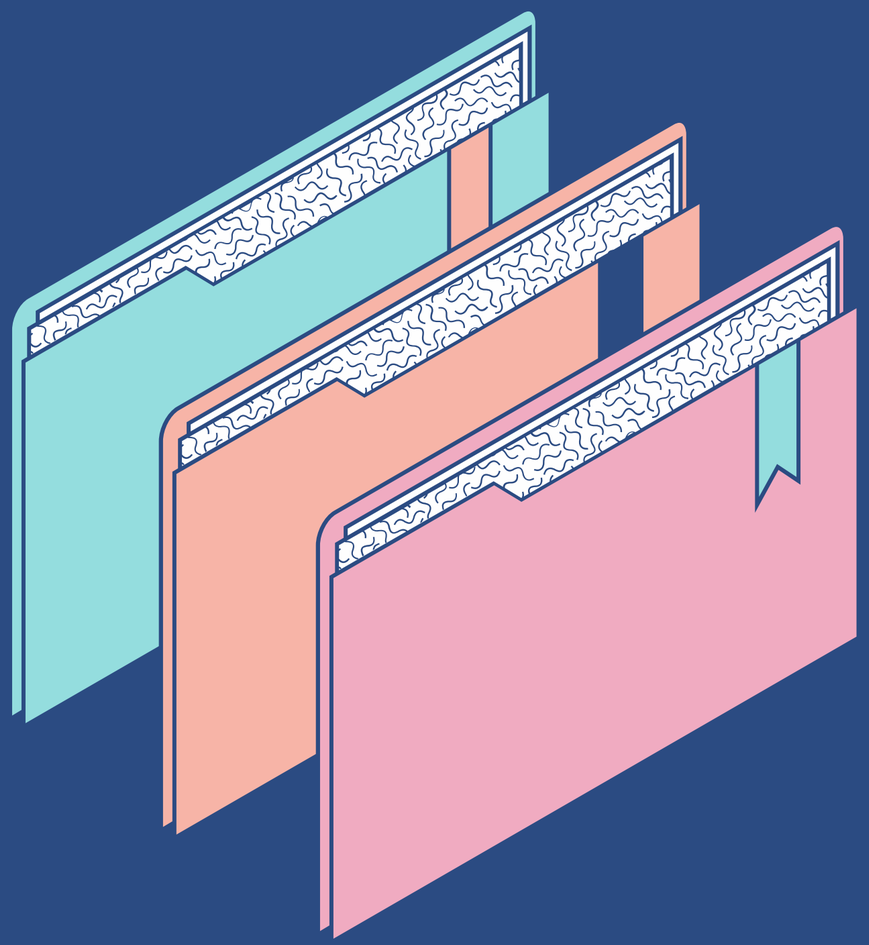
Final Task Credit Risk Assessment

NAUFAL FAUZAN

ABOUT ME

As a recent graduate from Gunadarma University with a degree in Information System, I'm enthusiastic in data related role such as Data Analytic, Data Science, and Machine Learning. Utilizing various tools such as Spreadsheet, SQL, and Python. Skilled in Exploratory Analysis, Data Cleansing, Data Visualization, Machine Learning and Evaluation.

MY LINKEDIN

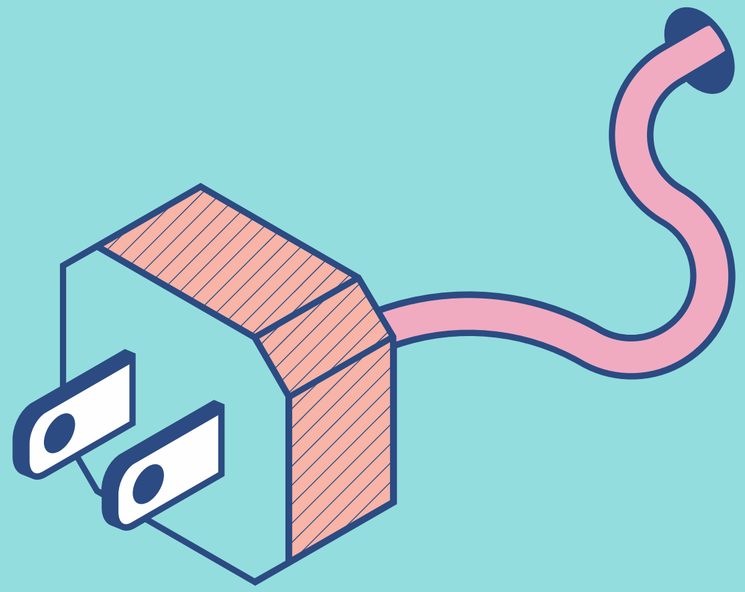


TASK BACKGROUND

A lending institution is currently grappling with the challenge of enhancing efficiency and expediting the loan application process for every customer. In my capacity as a Data Science Intern at ID/X Partners, our objective is to leverage data processing techniques and develop models capable of forecasting and evaluating ideal credit applications while predicting potential risks.

OBJECTIVE

Build an accurate prediction model to assess the credit risk of a company's customers. This model will provide information on whether a customer is likely to experience delays in loan payments or not. Hopefully, the company can reduce the number of customers who have the potential to experience payment failure, so that the risk of company losses can be minimized.



contents

1 Tools

The tools to be used are:

- Python
- Pandas
- Matplotlib
- Seaborn
- Google Colab

More

1 Dataset

```
EXPLORING DATA

[ ]  df_create.shape

     (466285, 75)
```

This dataset has 466.285 rows and 75 columns.

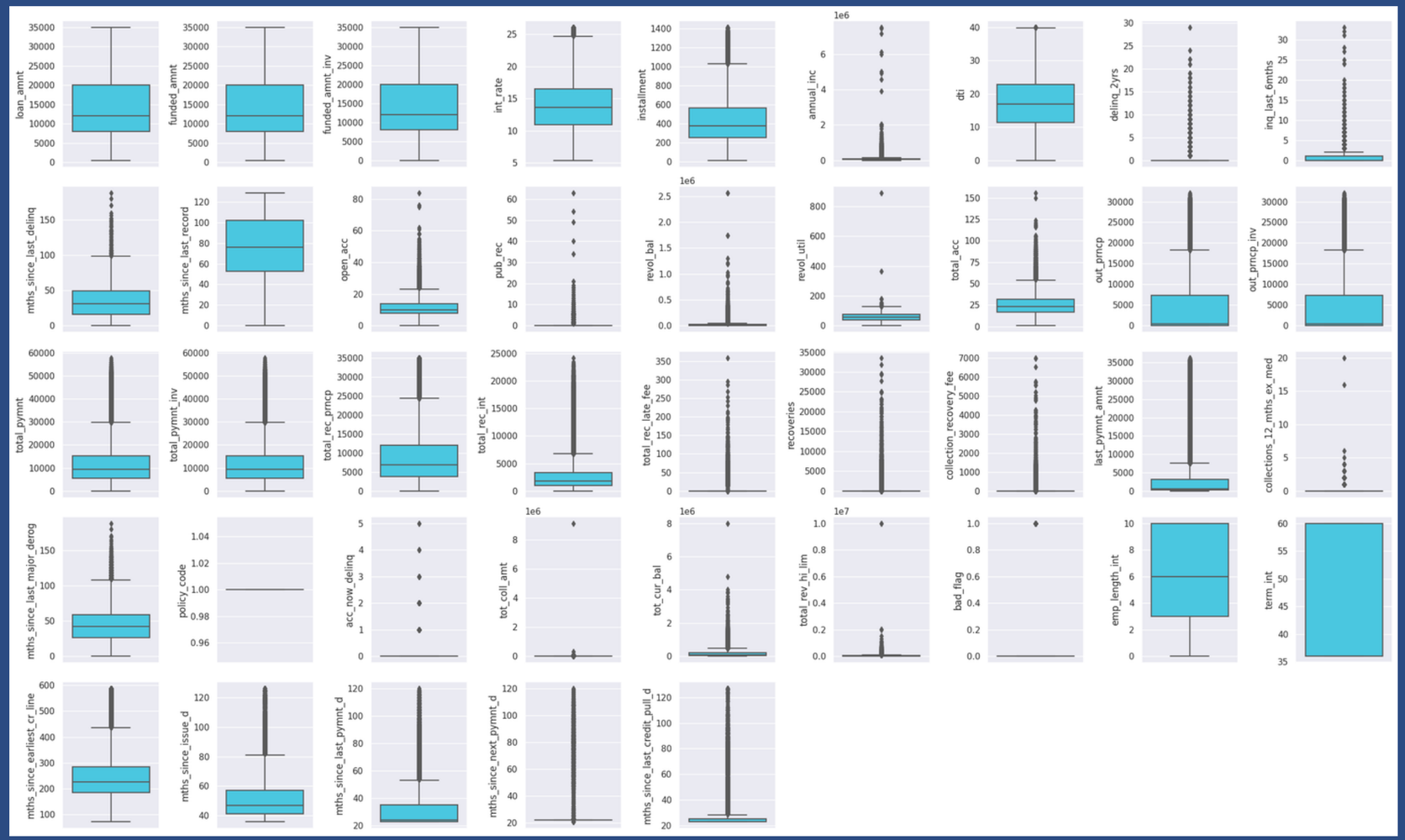
the data types:

- float64(46)
- int64(7)
- object(22)

More

2 Data Preprocessing

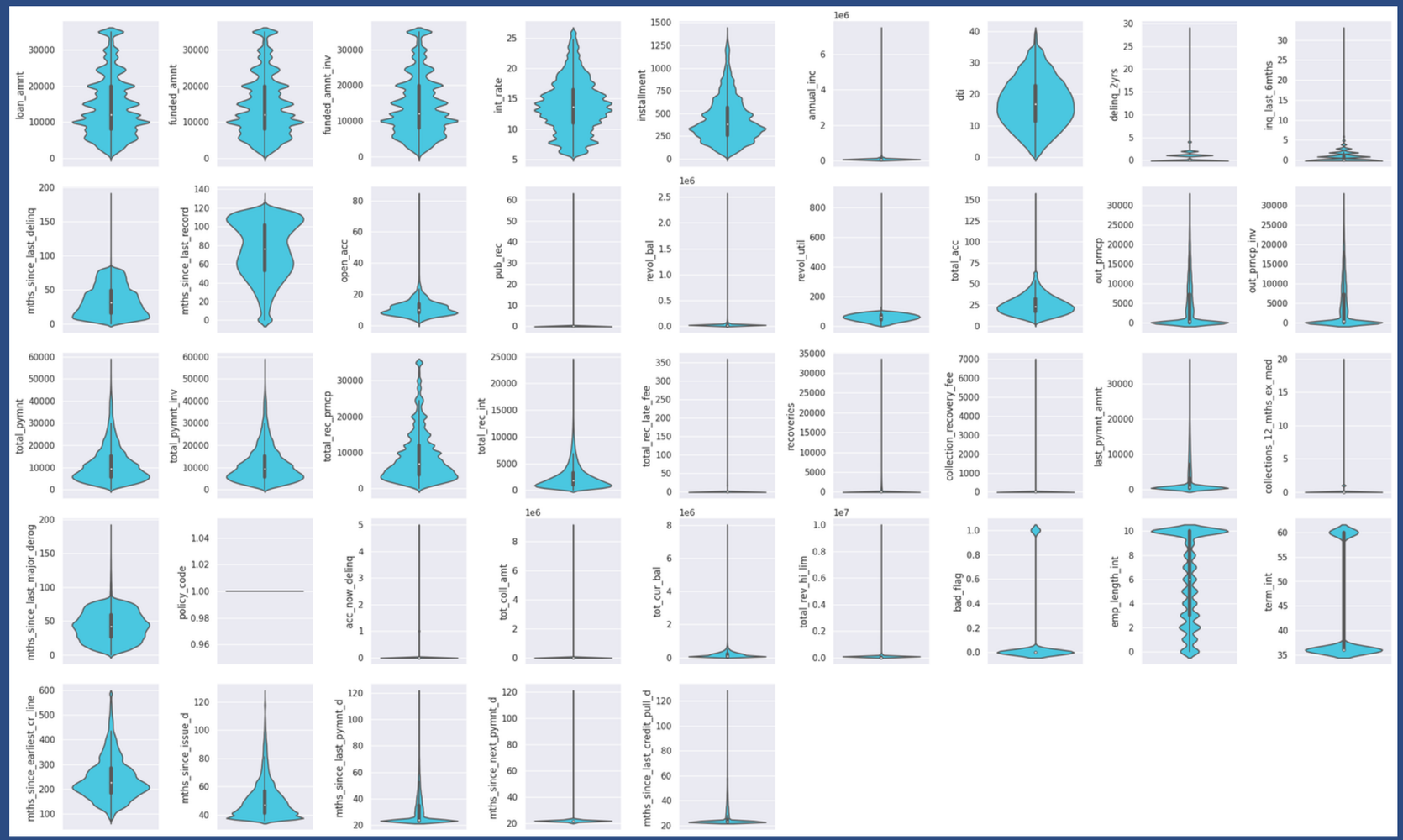
INDIVIDUAL BOXPLOT & VIOLINPLOT



More

2 Data Preprocessing

INDIVIDUAL BOXPLOT & VIOLINPLOT



More

2 Data Preprocessing

IN THIS DATASET, THE **LOAN_STATUS** VARIABLE IS A TARGET VARIABLE.

LOAN_STATUS VARIABLE HAS SEVERAL VALUES:

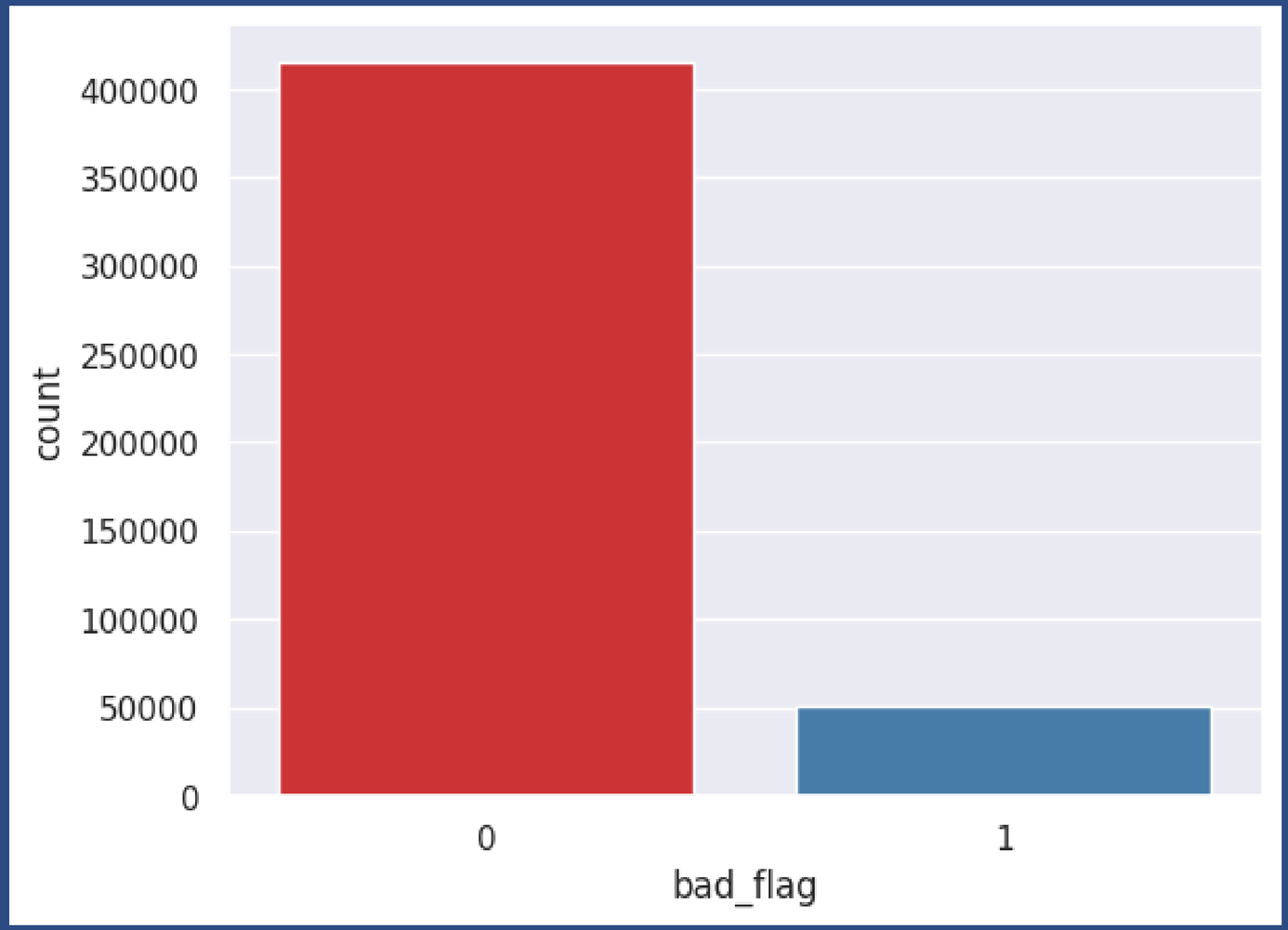
- CURRENT
- CHARGED OFF
- LATE
- IN GRACE PERIOD
- FULLY PAID
- DEFAULT

FROM THESE DEFINITIONS, EACH BORROWER CAN BE CATEGORIZED AS A GOOD BORROWER AND A BAD BORROWER.

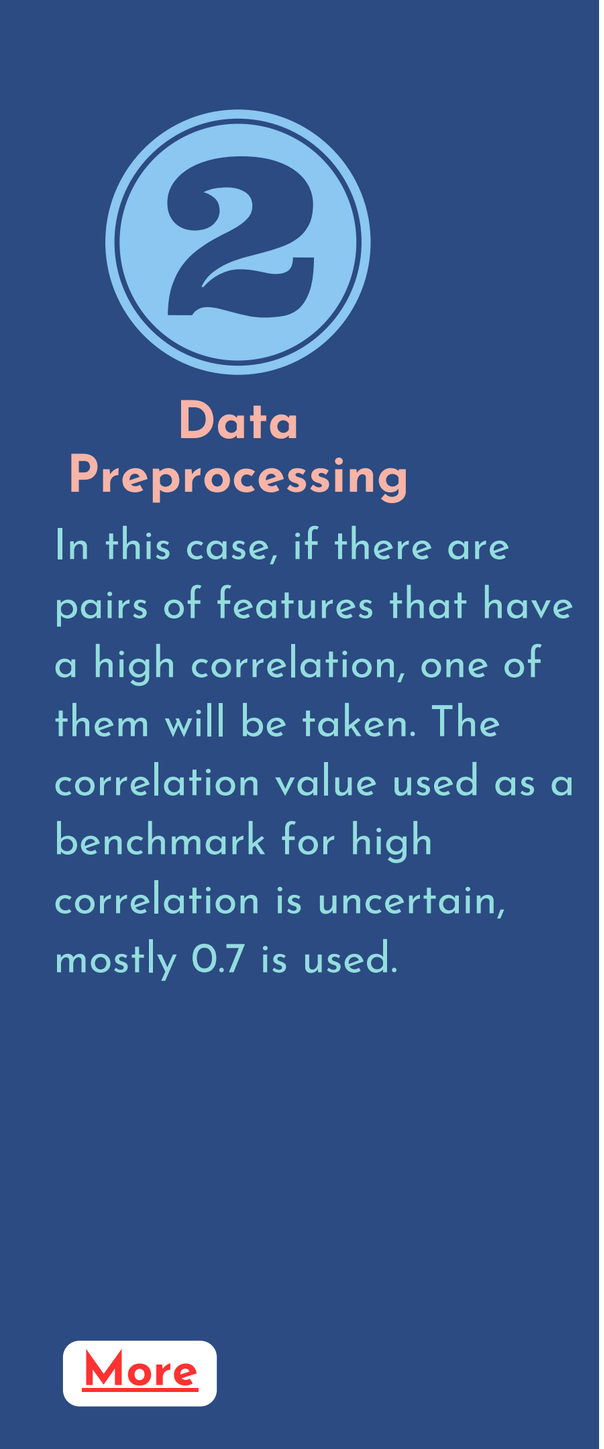
0 = 89.06%

1 = 10.93%

Separating between good borrowers and bad borrowers, it can be seen that there is imbalanced data because the good borrower data is 89% while the bad borrower is only 10%.



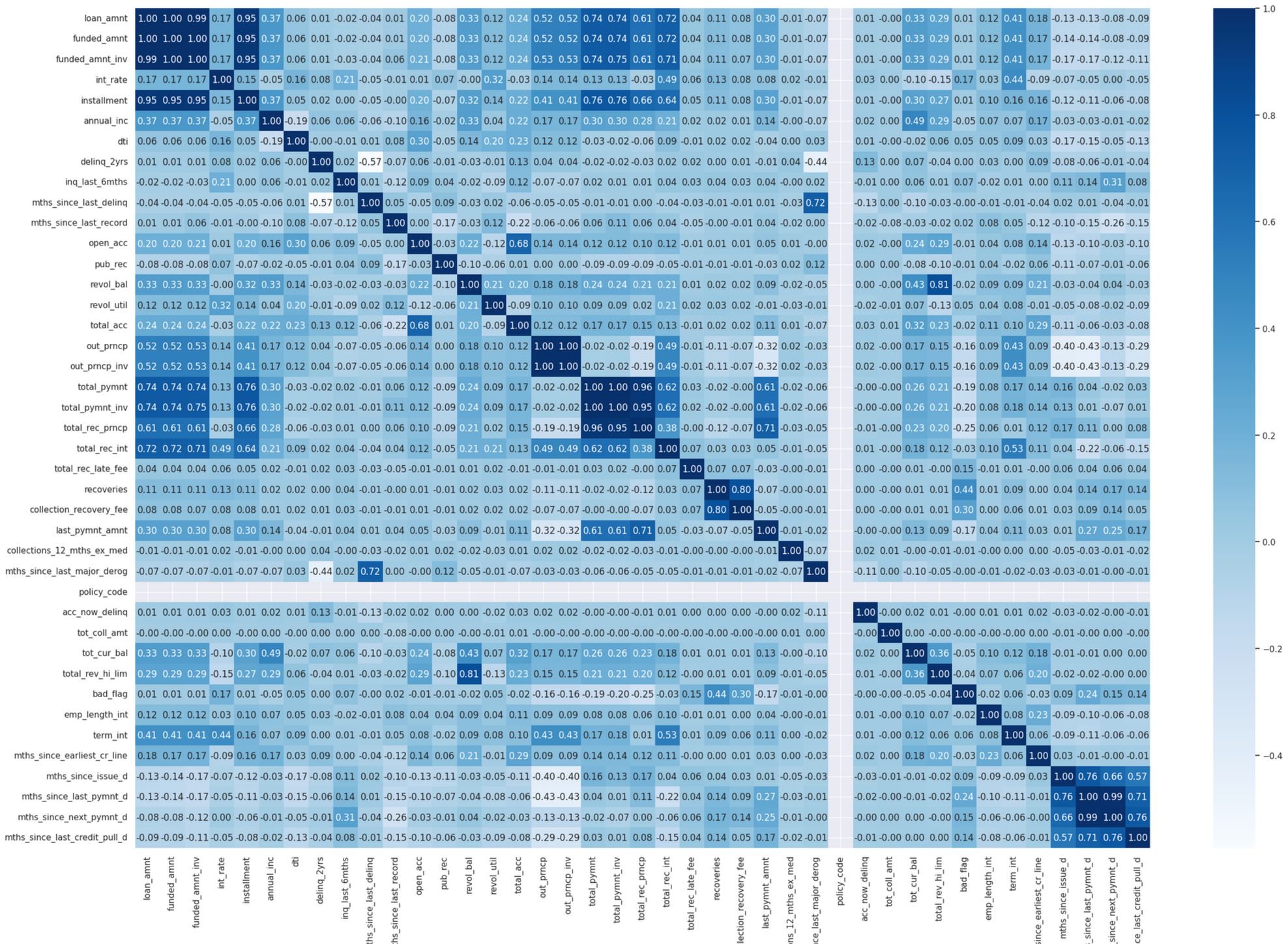
More



Data Preprocessing

In this case, if there are pairs of features that have a high correlation, one of them will be taken. The correlation value used as a benchmark for high correlation is uncertain, mostly 0.7 is used.

More



3 Handling Missing Values

- Feature that have missing value more than 50% will drop because too avoid bias result on modelin
- Numerical feature will replace missing value with "Median“

```
df_create['annual_inc'].fillna(df_create['annual_inc'].mean(), inplace=True)
df_create['mths_since_earliest_cr_line'].fillna(0, inplace=True)
df_create['acc_now_delinq'].fillna(0, inplace=True)
df_create['total_acc'].fillna(0, inplace=True)
df_create['pub_rec'].fillna(0, inplace=True)
df_create['open_acc'].fillna(0, inplace=True)
df_create['inq_last_6mths'].fillna(0, inplace=True)
df_create['delinq_2yrs'].fillna(0, inplace=True)
df_create['collections_12_mths_ex_med'].fillna(0, inplace=True)
df_create['revol_util'].fillna(0, inplace=True)
df_create['emp_length_int'].fillna(0, inplace=True)
df_create['tot_cur_bal'].fillna(0, inplace=True)
df_create['tot_coll_amt'].fillna(0, inplace=True)
df_create['mths_since_last_delinq'].fillna(-1, inplace=True)
```

```
df_create['mths_since_last_pymnt_d'].fillna(df_create['mths_since_last_pymnt_d'].median(), inplace=True)
df_create['mths_since_next_pymnt_d'].fillna(df_create['mths_since_next_pymnt_d'].median(), inplace=True)
df_create['mths_since_last_credit_pull_d'].fillna(df_create['mths_since_last_credit_pull_d'].median(), inplace=True)
df_create['total_rev_hi_lim'].fillna(df_create['total_rev_hi_lim'].mean(), inplace=True)
```

```
df_create.isna().sum()
```

```
loan_amnt                     0
funded_amnt                   0
funded_amnt_inv               0
int_rate                      0
installment                   0
grade                         0
home_ownership                0
annual_inc                    0
verification_status           0
purpose                       0
addr_state                    0
dti                           0
delinq_2yrs                   0
inq_last_6mths                0
mths_since_last_delinq        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
total_pymnt                   0
total_pymnt_inv               0
total_rec_prncp               0
total_rec_int                 0
total_rec_late_fee            0
recoveries                    0
collection_recovery_fee       0
last_pymnt_amnt               0
collections_12_mths_ex_med    0
acc_now_delinq                0
tot_coll_amt                  0
tot_cur_bal                   0
total_rev_hi_lim              0
bad_flag                      0
emp_length_int                0
term_int                      0
mths_since_earliest_cr_line   0
mths_since_issue_d            0
mths_since_last_pymnt_d       0
mths_since_next_pymnt_d       0
mths_since_last_credit_pull_d 0
dtype: int64
```

More

4 Feature Scaling, Transformation and Modeling

One-hot encoding in machine learning is the conversion of categorical information into a format that may be fed into machine
learning algorithms to improve prediction accuracy

One Hot Encoding

```
[ ] categorical_cols = [col for col in df_create.select_dtypes(include='object').columns.tolist()]
```

```
[ ] onehot = pd.get_dummies(df_create[categorical_cols], drop_first=True)
```

```
onehot.head()
```

	grade_B	grade_C	grade_D	grade_E	grade_F	grade_G	home_ownership_MORTGAGE	home_ownership_NONE	h
0	1	0	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	
2	0	1	0	0	0	0	0	0	
3	0	1	0	0	0	0	0	0	
4	1	0	0	0	0	0	0	0	

More

4 Feature Scaling, Transformation and Modeling

Standardization entails scaling data to fit a standard normal distribution. A standard normal distribution is defined as a distribution with a mean of 0 and a standard deviation of 1.

Standardization

```
[ ]  numerical_cols = [col for col in df_create.columns.tolist() if col not in categorical_cols + ['bad_flag']]
```

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
std = pd.DataFrame(ss.fit_transform(df_create[numerical_cols]), columns=numerical_cols)
std.head()
```

	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	dti	delinq_2yrs	inq_last_6mths
0	-1.124392	-1.122963	-1.114455	-0.729587	-1.105575	-0.896551	1.328632	-0.357012	0.178920
1	-1.426088	-1.425101	-1.412732	0.330634	-1.528763	-0.787387	-2.065791	-0.357012	3.843328
2	-1.438156	-1.437186	-1.424784	0.488979	-1.428140	-1.110294	-1.082491	-0.357012	1.095022
3	-0.521001	-0.518687	-0.508860	-0.077850	-0.380931	-0.438063	0.354248	-0.357012	0.178920
4	-1.365749	-1.364673	-1.352474	-0.261438	-1.496071	0.122311	0.091865	-0.357012	-0.737182

More

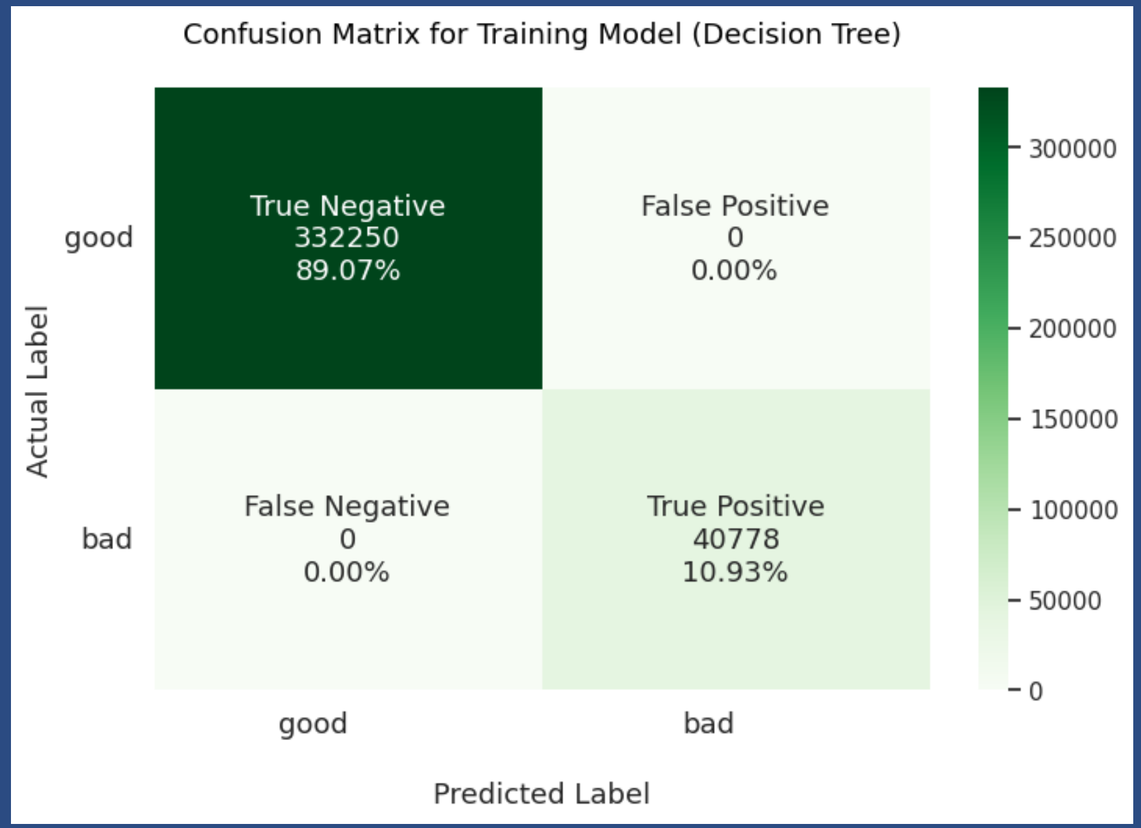
4 Feature Scaling, Transformation and Modeling

Modeling

Decision Tree Training

```
DecisionTreeClassifier(random_state=42)
```

	Evaluation Metrics	Train	Test	Diff Range
0	Accuracy	1.000000	0.991000	0.009000
1	Precision	1.000000	0.959000	0.041000
2	Recall	1.000000	0.963000	0.037000
3	F1 Score	1.000000	0.961000	0.039000
4	F1 Score (crossval)	1.000000	0.962000	0.038000
5	ROC AUC	1.000000	0.979000	0.021000
6	ROC AUC (crossval)	1.000000	0.979000	0.021000



More

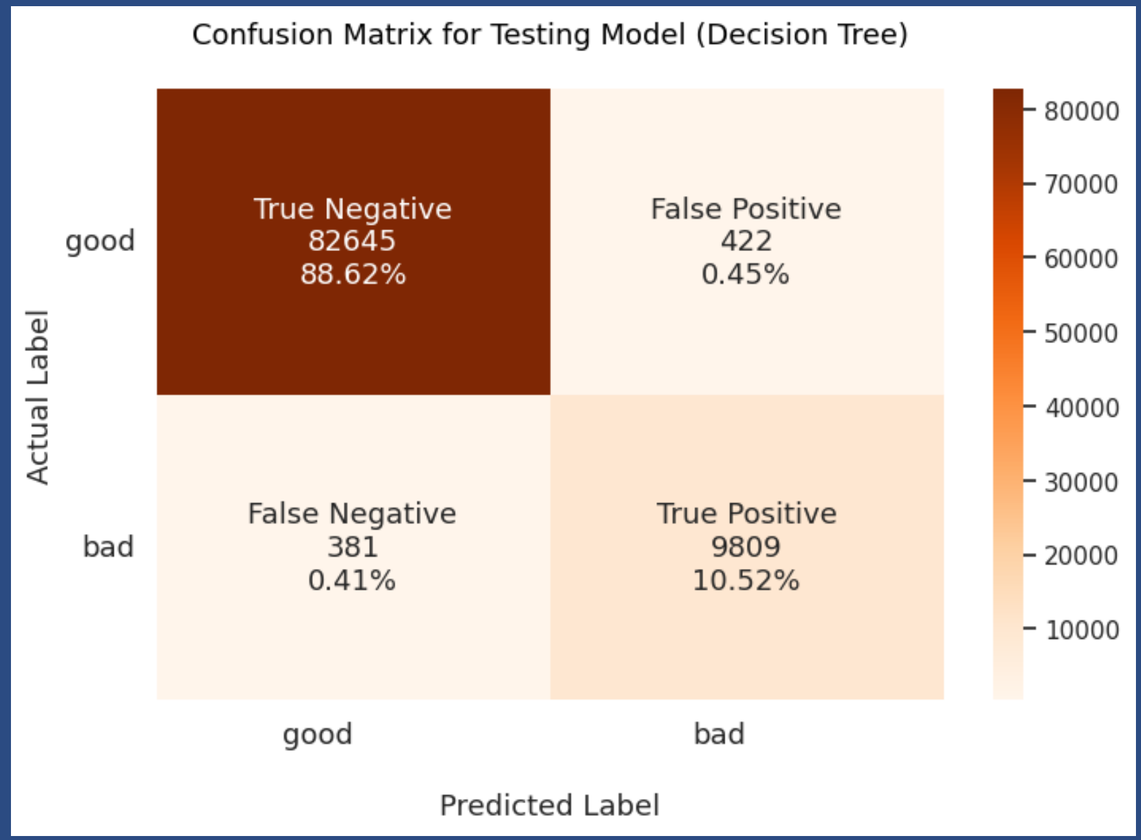
4 Feature Scaling, Transformation and Modeling

Modeling

Decision Tree Test

DecisionTreeClassifier(random_state=42)

	Evaluation Metrics	Train	Test	Diff Range
0	Accuracy	1.000000	0.991000	0.009000
1	Precision	1.000000	0.959000	0.041000
2	Recall	1.000000	0.963000	0.037000
3	F1 Score	1.000000	0.961000	0.039000
4	F1 Score (crossval)	1.000000	0.962000	0.038000
5	ROC AUC	1.000000	0.979000	0.021000
6	ROC AUC (crossval)	1.000000	0.979000	0.021000



More

4 Feature Scaling, Transformation and Modeling

Modeling

Decision Tree

	Train	Test
Accuracy	1.000000	0.991000
Precision	1.000000	0.959000
Recall	1.000000	0.963000

```
acc_dt_train=round(dt_model.score(X_train,y_train)*100,2)
acc_dt_test=round(dt_model.score(X_test,y_test)*100,2)
print("Training Accuracy: {} %".format(acc_dt_train))
print("Testing Accuracy: {} %".format(acc_dt_test))
```

```
Training Accuracy: 100.0 %
Testing Accuracy: 99.14 %
```

More

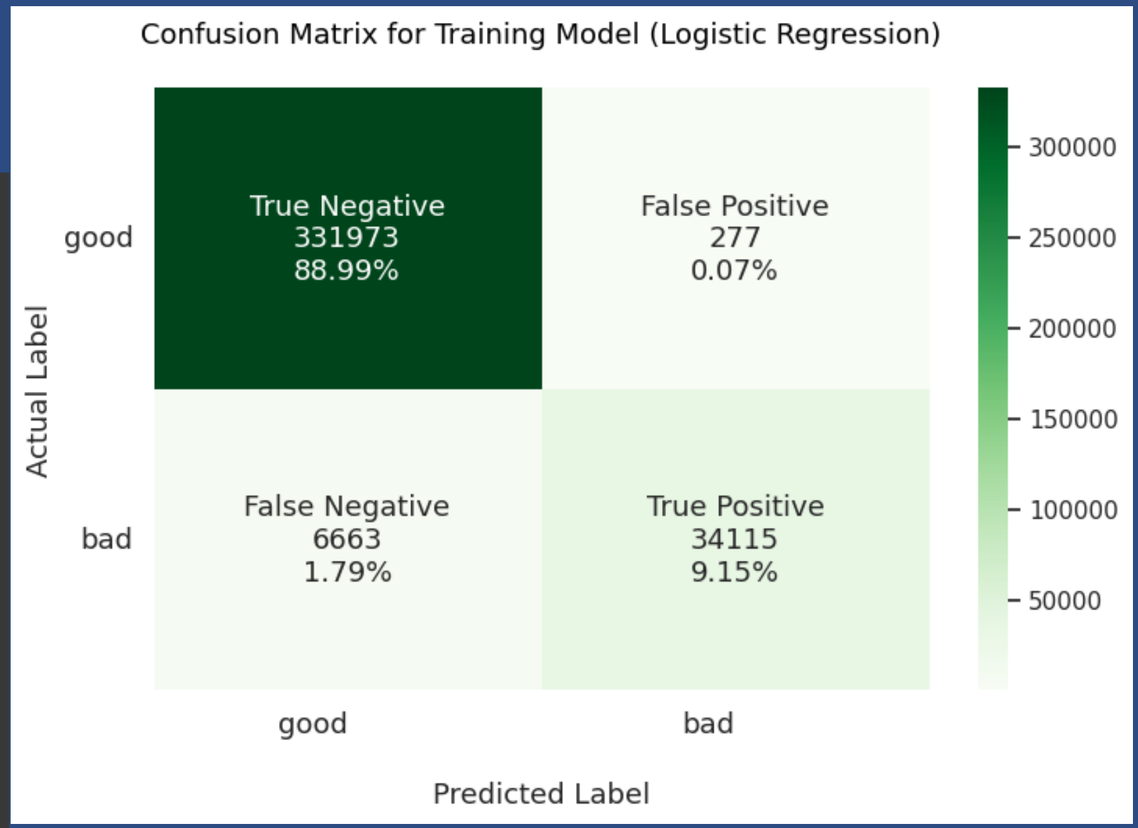
4 Feature Scaling, Transformation and Modeling

Modeling

Logistic Regression Train

```
LogisticRegression(max_iter=373028, random_state=42)
```

	Evaluation Metrics	Train	Test	Diff Range
0	Accuracy	0.981000	0.981000	0.000000
1	Precision	0.992000	0.990000	0.002000
2	Recall	0.837000	0.834000	0.003000
3	F1 Score	0.908000	0.905000	0.003000
4	F1 Score (crossval)	0.908000	0.908000	0.000000
5	ROC AUC	0.977000	0.977000	0.000000
6	ROC AUC (crossval)	0.977000	0.977000	0.000000



More

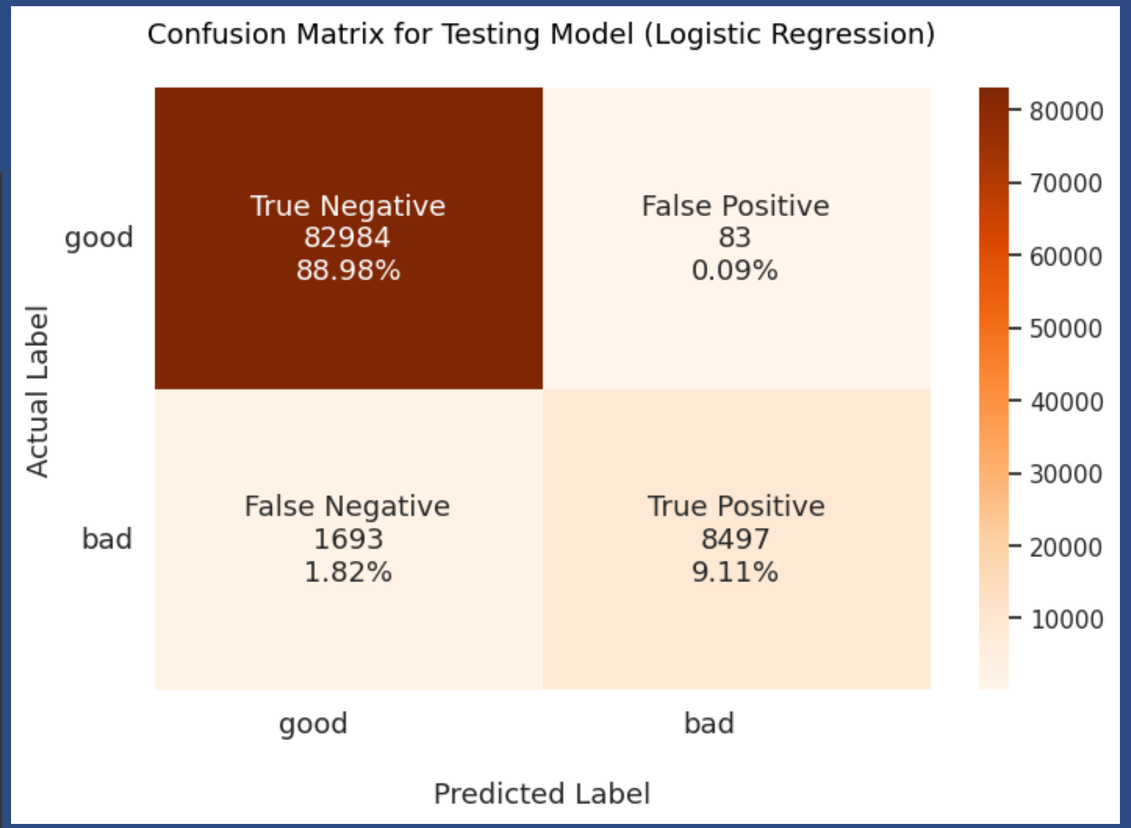
4 Feature Scaling, Transformation and Modeling

Modeling

Logistic Regression Test

LogisticRegression(max_iter=373028, random_state=42)

	Evaluation Metrics	Train	Test	Diff Range
0	Accuracy	0.981000	0.981000	0.000000
1	Precision	0.992000	0.990000	0.002000
2	Recall	0.837000	0.834000	0.003000
3	F1 Score	0.908000	0.905000	0.003000
4	F1 Score (crossval)	0.908000	0.908000	0.000000
5	ROC AUC	0.977000	0.977000	0.000000
6	ROC AUC (crossval)	0.977000	0.977000	0.000000



More

Feature Scaling, Transformation and Modeling

Modeling

Decision Tree

	Train	Test
Accuracy	0.981000	0.981000
Precision	0.992000	0.990000
Recall	0.837000	0.834000

```
acc_log_train=round(log_model.score(X_train,y_train)*100,2)
acc_log_test=round(log_model.score(X_test,y_test)*100,2)
print("Training Accuracy: {} %".format(acc_log_train))
print("Test Accuracy: {} %".format(acc_log_test))
```

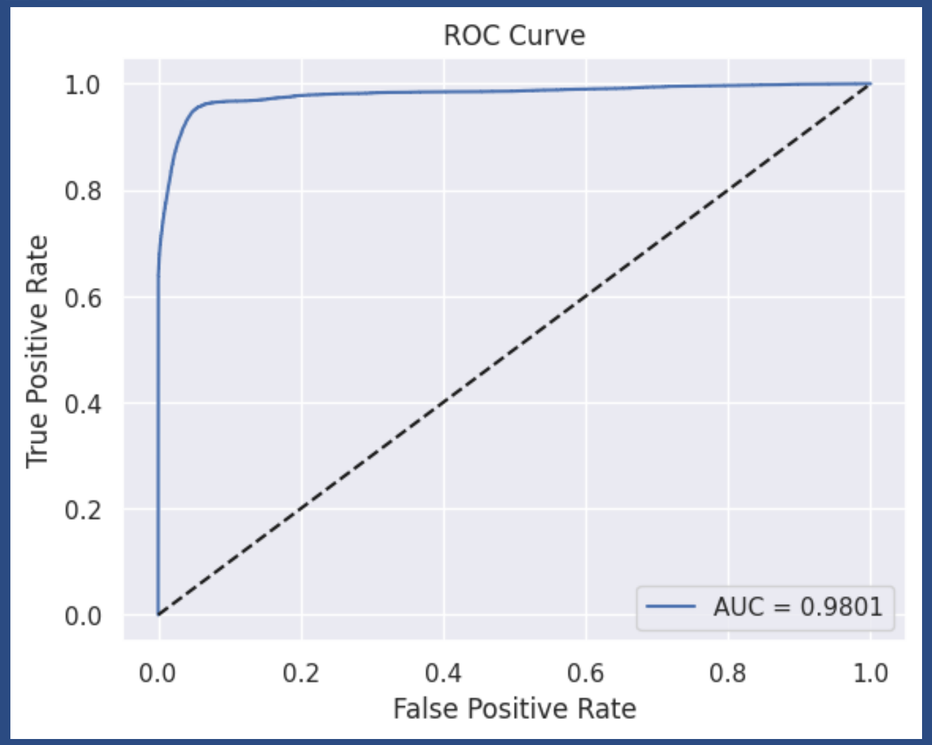
```
Training Accuracy: 98.14 %
Test Accuracy: 98.1 %
```

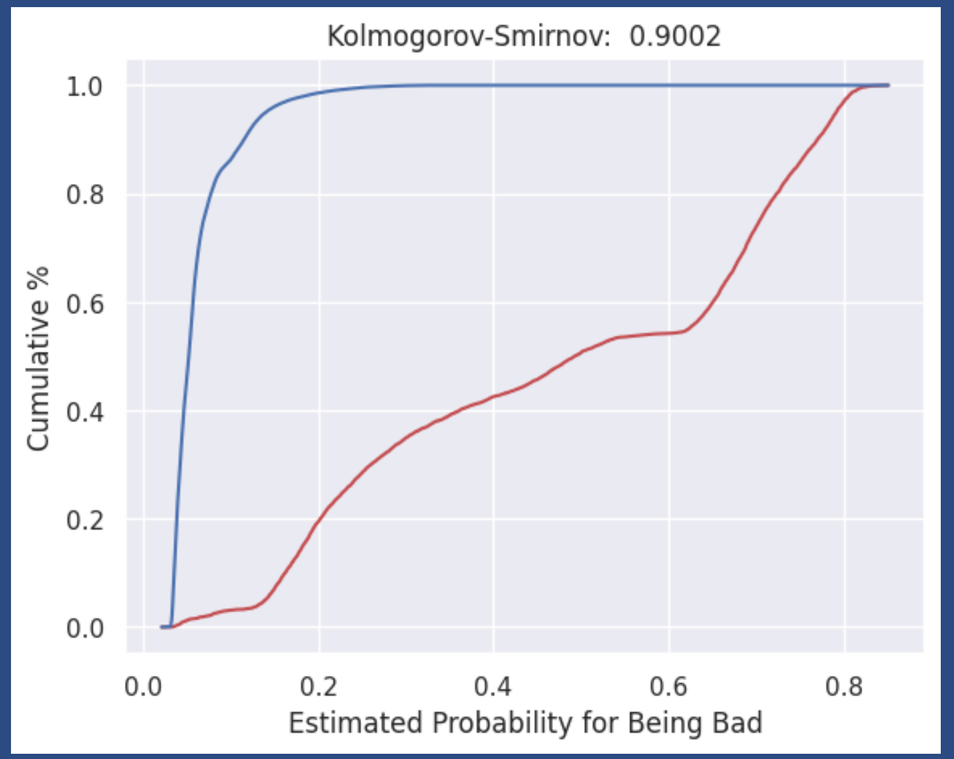
More

Feature Scaling, Transformation and Modeling

Modeling

MODEL EVALUATION





More

Feature Scaling, Transformation and Modeling
Metrics evaluation

	Models	Precision (Train)	Precision (Test)	Recall (Train)	Recall (Test)	F1 Score (Train)	F1 Score (Test)
0	Decision Tree	1.000000	0.959000	1.000000	0.963000	1.000000	0.961000
1	Logistic Regression	0.992000	0.990000	0.837000	0.834000	0.908000	0.905000

- Precision: Measures the proportion of positive predictions that are actually correct. Both models achieve high precision on both training and test sets, with Decision Tree having slightly higher scores.
- Recall: Measures the proportion of actual positives that are correctly identified. Decision Tree outperforms Logistic Regression in terms of recall, indicating its ability to capture more true positives.
- F1 Score: Combines precision and recall into a single metric, providing a balanced assessment of both aspects. Decision Tree again holds a slight advantage in F1 scores.

More

5 Conclusion

RECOMENDATION

- Implemented a stricter credit policy
- Closely monitor loan portfolios and conduct more in-depth analysis on borrowers with poor status to reduce the risk of non-performing loans (NPL).
- Analyzing the profile and economic conditions of borrowers' home countries
- Carefully assessing collateral value and calculating Loan-to-Value (LTV) to reduce the risk of loss in case of default.

THANKS !