

LOAN PREDICTION

by :
Robby Aulia Tubagus

Business Understanding

Upon receiving a loan application, the business must decide whether to approve the loan based on the applicant's profile. The bank's choice carries two different kinds of risks:

1. false positive (Type I) Situation: Although the applicant defaults, the model approves the loan and indicates that the applicant will return it.

Impact: Should the applicant default, this could result in monetary loss for the organization.

1. false negative (Type II) Situation: Although the application would have returned the loan, the model predicts that the applicant would default and be rejected.

Impact: The company loses out on a potentially lucrative customer as a result, which causes a decline in business.



Business Objective

Objective: Using past data, create a machine learning model that predicts whether a customer's loan application will be approved or denied.

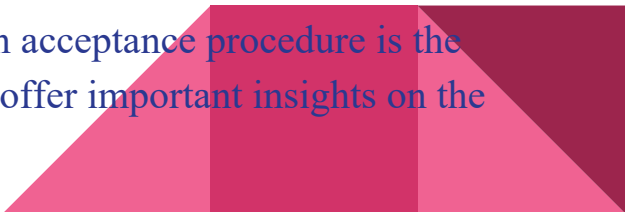
Key Steps:

1. Preparing Data:

This procedure makes sure the dataset is optimized by eliminating columns that don't include any useful information. it helps to create a more useful and targeted dataset for further analysis.

1. Feature Importance Analysis:

Examining the significance and effects of different aspects on the loan acceptance procedure is the focus of this investigation. An Exploratory Data Analysis (EDA) can offer important insights on the features of the dataset before this analysis.



Business Objective(Cont...)

3. Model Development:

To prepare the dataset for train and evaluate machine learning algorithms, identifying trends and connections related to loan approval.

3. Model Evaluation:

To guarantee accurate predictions, evaluate the model's performance using important metrics including accuracy, precision, recall, and F1-score.



Preparing Data

Preparing Data

1. Loading the Dataset:

- a. A pandas DataFrame (df variable) can be used to load a dataset from a CSV file using the `pd.read_csv` function. The location of the CSV file on the local computer is indicated by the file path.

```
df = pd.read_csv('path.csv')
```

1. Removing Columns with Only Missing Values:

- a. Using the expression `df.isnull().all()`, the function finds the columns in the DataFrame (df) when all values are absent (NaN). The `null_columns` variable contains these columns.
- b. The DataFrame's columns containing all missing values are eliminated using the `df.drop(null_columns, axis=1)` method. Dropping columns is specified by the `axis=1` argument.

Preparing Data(Cont...)

```
In [5]: null_columns = df.columns[df.isnull().all()]
print(null_columns)
print(null_columns.shape)

Index(['annual_inc_joint', 'dti_joint', 'verification_status_joint',
       'open_acc_6m', 'open_il_6m', 'open_il_12m', 'open_il_24m',
       'mths_since_rcnt_il', 'total_bal_il', 'il_util', 'open_rv_12m',
       'open_rv_24m', 'max_bal_bc', 'all_util', 'inq-fi', 'total_cu_tl',
       'inq_last_12m'],
      dtype='object')
(17,)
```

```
In [6]: df = df.drop(null_columns, axis=1)
```

Feature Importance Analysis

Goal : to enhance the data quality
and prepare the dataset for further
process

Feature Importance Analysis

1. Unique Identifiers Handling:

Checked the uniqueness of identifiers.

```
In [23]: df['url'].unique()
```

```
Out[23]: array(['https://www.lendingclub.com/browse/loanDetail.action'],  
              dtype=object)
```

1. Categorical Variable Transformation:

Explored and transformed unique values in categorical columns.

Replaced specific values in certain columns.

```
df['pymnt_plan'] = df['pymnt_plan'].str.replace('n', 'False')  
df['pymnt_plan'] = df['pymnt_plan'].str.replace('y', 'True')
```

Feature Importance Analysis(Cont...)

1. Handling Missing Values:

Explored and addressed missing values in specific columns.

```
df['emp_length'].fillna('0 years', inplace=True)
```

1. Drop Unnecessary Columns:

Dropped unnecessary columns from the dataset.

```
toDrop = ['application_type', 'policy_code', 'zip_code', 'addr_state']  
df = df.drop(columns=toDrop)
```

1. Exploring and Transforming Numeric Variables:

Checked for specific conditions in numeric columns.

Transformed and extracted numeric values from specific columns.



Feature Importance Analysis(Cont...)

```
import re

def extract_numeric_value(s):
    if '< 1' in s:
        return 0.5
    else:
        match = re.search(r'\d+', s)
        if match:
            return float(match.group())
        else:
            return None

df['emp_length'] = df['emp_length'].apply(extract_numeric_value)
```

4. Feature Engineering:

Created new boolean columns based on specific conditions.

Cleaned and created new columns based on existing ones.

```
df['pymnt_plan'] = df['pymnt_plan'].str.lower() == 'true'
```

Explanatory Data Analysis(EDA)

Goal : to gain a deeper understanding of the structure, patterns, relationships, and characteristics of a dataset

Explanatory Data Analysis(EDA)

1. Correlation matrix :

It helps us understand the linear relationships between pairs of numeric variables in a dataset

Usage :

- a. Identifying Relationships
- b. Multicollinearity Detection
- c. Feature Selection

1. Dropping Highly Correlated Features

The process of dropping highly correlated features involves removing one of the two correlated variables to address multicollinearity.



Explanatory Data Analysis(EDA)(Cont...)

Usage :

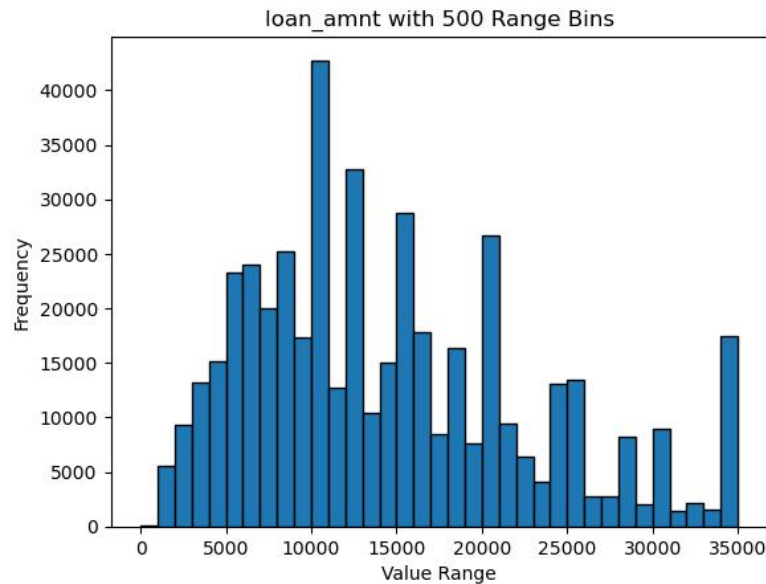
- a. Improved Model Stability
- b. Enhanced Model Interpretability
- c. Reduced Overfitting
- d. Computational Efficiency

```
# Identify and drop highly correlated features (threshold set to 0.9 in this example)
threshold = 0.9
upper_tri = correlation_matrix.abs().where(np.triu(np.ones(correlation_matrix.shape), k=1).astype(bool))
to_drop = [column for column in upper_tri.columns if any(upper_tri[column] > threshold)]
df = df.drop(to_drop, axis=1)
```


Explanatory Data Analysis(EDA)(Cont...)

3. Visualization :

the goal is to interact with the data, discover patterns, and make informed decisions about subsequent analysis and modeling steps



Model Development


Goal : to ensure that the data is in a suitable format for training

Model Development

goal : to ensure that the data is in a suitable format for training

1. Label Encoding:

Label Encoding is a technique used to convert categorical variables into numerical format, which is essential for feeding data into machine learning models. Label Encoding assumes an ordinal relationship between categories, as it assigns integers.



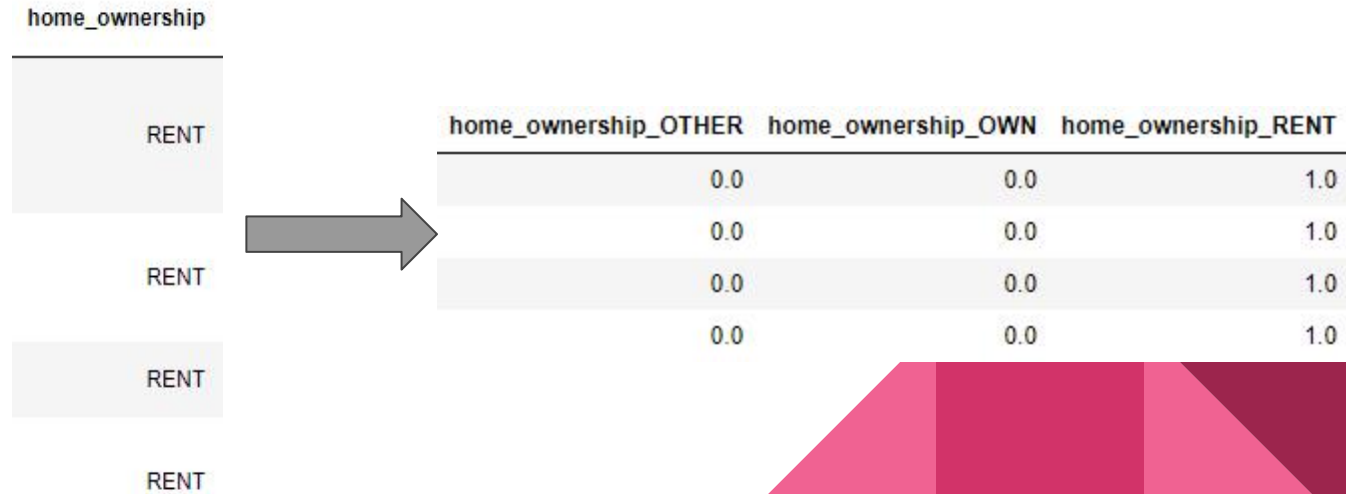
grade	sub_grade
B	B2
C	C4
C	C5
C	C1
B	B5

grade	sub_grade
1	6
2	13
2	14
2	10
1	9

Model Development(Cont...)

2. One-Hot Encoding:

One-Hot Encoding is used when dealing with categorical variables with more than two categories. It creates binary columns for each category, representing the presence or absence of that category for each observation.

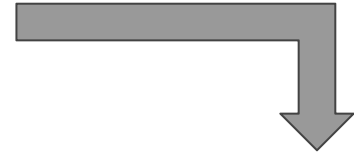


Model Development(Cont...)

3. Standard Scaling:

Standard Scaling (or Z-score normalization) is applied to numeric features to ensure that they have a mean of 0 and a standard deviation of 1. This scaling is particularly important for models that rely on distance metrics, It prevents features with larger scales from dominating the learning process.

loan_amnt	term	int_rate	grade	sub_grade	emp_length	annual_inc	pymnt_plan	dti	delinq_2yrs
5000	36	10.65	1	6	10.0	24000.0	False	27.65	0.0
2500	60	15.27	2	13	0.5	30000.0	False	1.00	0.0
2400	36	15.96	2	14	10.0	12252.0	False	8.72	0.0
10000	36	13.49	2	10	10.0	49200.0	False	20.00	0.0
3000	60	12.69	1	9	1.0	80000.0	False	17.94	0.0



```
[[ 1.16820533 -0.61599066 -1.13167941 ... -0.07325327 -0.07130443
  -0.73360139]
 [-0.52105375 -0.61599066  0.11540782 ... -0.07325327 -0.07130443
  -0.73360139]
 [ 0.73684096 -0.61599066 -1.13167941 ... -0.07325327 -0.07130443
  -0.73360139]
 ...]
```

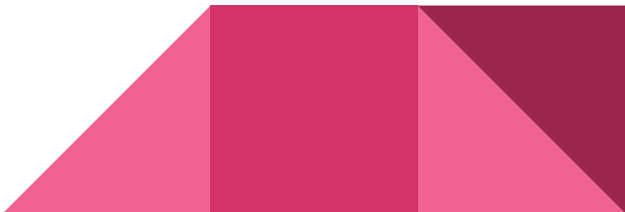
Model Development(Cont...)

4. DataFrame Splitting:

To evaluate the model's performance effectively, the dataset is split into training and testing sets. This step ensures that the model is trained on a portion of the data and evaluated on unseen data, providing insights into its generalization capabilities.

```
x = df.drop('flagged', axis=1)
y = df['flagged']

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



Model Evaluation

Goal : to assess the performance and effectiveness of a machine learning model in making predictions on new, unseen data.

Model Evaluation

1. Performance Measurement

Random Forest Accuracy: 0.98

Random Forest Classification Report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	82690
1	1.00	0.81	0.89	10567
accuracy			0.98	93257
macro avg	0.99	0.91	0.94	93257
weighted avg	0.98	0.98	0.98	93257

XGBoost Accuracy: 0.98

XGBoost Classification Report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	82690
1	0.99	0.85	0.91	10567
accuracy			0.98	93257
macro avg	0.98	0.93	0.95	93257
weighted avg	0.98	0.98	0.98	93257

Logistic Regression Accuracy: 0.97

Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.97	1.00	0.98	82690
1	0.97	0.78	0.87	10567
accuracy			0.97	93257
macro avg	0.97	0.89	0.93	93257
weighted avg	0.97	0.97	0.97	93257

Model Evaluation(Cont...)

2. cross-validation:

- a. Its primary purpose is to assess how well a model will generalize to an independent dataset

```
Cross-Validation Results (Accuracy): [0.96157929 0.97926161 0.98369023 0.96374535 0.9710263 ]  
Mean Accuracy: 0.9718605573844321
```

2. Output summarization

a. Model Selection:

- i. Considering the metrics, Random Forest and XGBoost appear promising for predicting loan default.

b. Confidence in Model:

- i. The models demonstrate strong accuracy and balanced precision-recall trade-offs, instilling confidence in their ability to make reliable predictions.



Thankyou

by :

Robby Aulia Tubagus