FA 590 Statistical Learning in Finance - Assignement 1: Loan Prediction Model: Methodology, Analysis, and Evaluation

Federica Malamisura - Professor Zonghao Yang

October 8, 2024

Contents

1 Introduction

This report documents the process of predicting loan returns and early defaults using machine learning models. The methodology includes exploratory data analysis (EDA), data preprocessing, feature engineering, and model selection. Key models used include linear regression, regularized regression, and logistic regression. The final section presents the test set predictions. Generative AI tools, such as ChatGPT, were used to clarify the problem requirements, debug code, and provide feature engineering suggestions.

2 Exploratory Data Analysis (EDA)

EDA is the first step in understanding the dataset and the relationships between its variables. Summary statistics, visualizations, and correlation analysis were performed.

2.1 Summary Statistics

Basic descriptive statistics were calculated for key numerical variables to observe trends such as central tendency and variance. Below is a code snippet showing how the summary statistics were computed:

```
import pandas as pd

# Load the dataset
df = pd.read_csv('lc_loan_data.csv')

# Display summary statistics
summary_stats = df.describe()
print(summary_stats)
```

Listing 1: Summary Statistics

Table 1: Summary Statistics of Loan Data

	loan_amnt	int_rate	fico_range_high	annual_inc
count	933160.000000	933160.000000	933160.000000	9.331600e+05
mean	12559.115559	0.119775	699.139987	7.412200e+04
std	8042.750083	0.039952	31.359739	6.938399e+04
min	500.000000	0.053200	664.000000	3.000000e+03
25%	6425.000000	0.089000	674.000000	4.400000e+04
50%	10000.000000	0.115300	694.000000	6.200000e+04
75%	16275.000000	0.143300	714.000000	9.000000e+04
max	40000.000000	0.309900	850.000000	9.573072e+06

2.2 Visualizations

We generated various visualizations, including histograms, scatter plots, and box plots to further explore data distributions and relationships:

- **Histograms**: Used to assess the distribution of key numerical variables.
- **Scatter Plots**: Showed relationships between features like loan amount and interest rate.
- **Box Plots**: Identified outliers and spread of variables.
- **Correlation Heatmap**: Displayed relationships between numerical features.

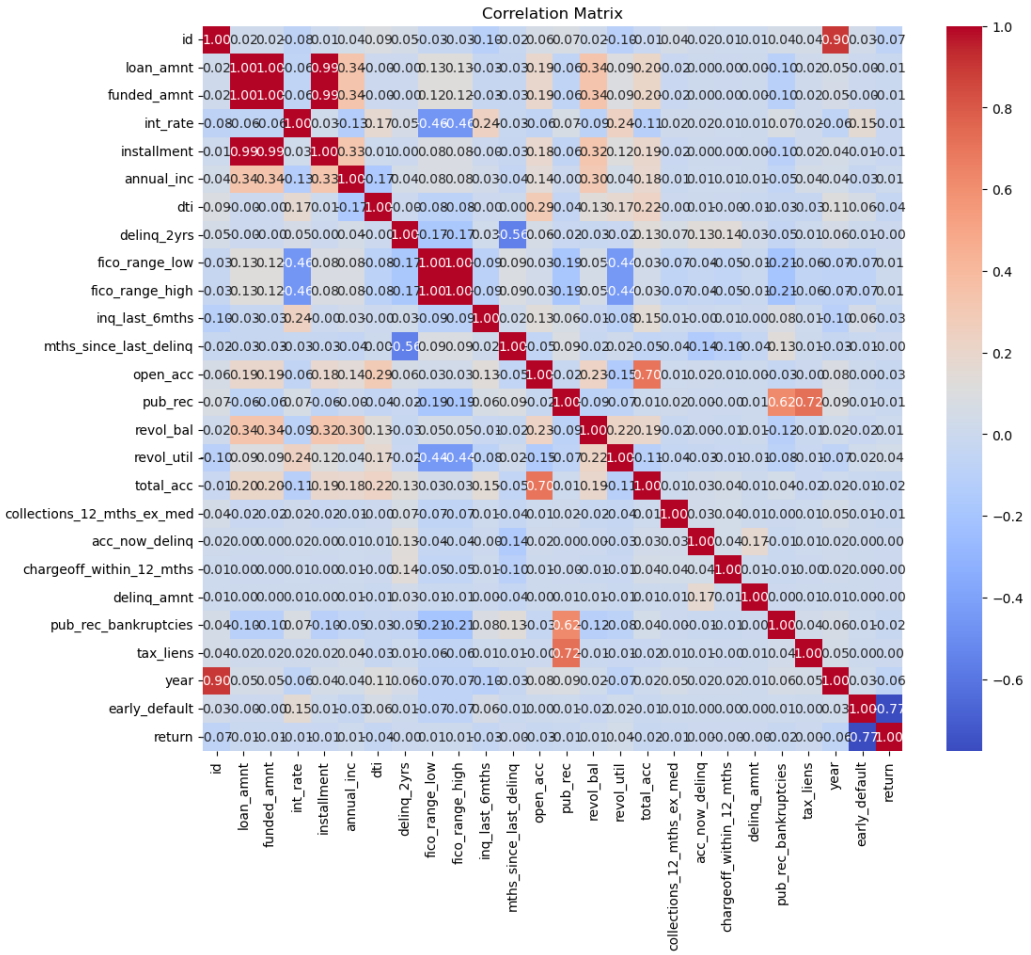

Figure 1: Correlation Heatmap of Features

3 Data Preprocessing

Data preprocessing was essential for handling missing values, sampling the dataset, and preparing the features for model training.

3.1 Handling Missing Values

We applied mean imputation for missing numerical data. The following code snippet shows how missing values were handled:

```
# Impute missing values
df.fillna(df.mean(), inplace=True)

# Check for remaining missing values
missing_values = df.isnull().sum()
print(missing_values)
```

Listing 2: Imputing Missing Values

```
Columns with missing values:
 emp_length                 58768
mths_since_last_delinq     464293
dtype: int64
```

Listing 3: Output Missing Values

3.2 Sampling

The dataset was split into training, validation, and test sets. A random split was performed using a standard 80-20 rule for training and validation data. This sampling ensures that models are evaluated on unseen data.

```
from sklearn.model_selection import train_test_split

# Split the dataset into training and validation sets
X = df.drop(columns=['return', 'early_default'])
y = df['return']

X_train, X_val, y_train, y_val = train_test_split(X, y,
    test_size=0.2, random_state=42)
```

Listing 4: Sampling the Dataset

3.3 Feature Engineering

Feature engineering included the creation of interaction terms and nonlinear (quadratic) terms to capture relationships that were not linear. Additionally, categorical variables were transformed using one-hot encoding and label encoding.

```
# Create interaction and nonlinear terms
df['loan_income_interaction'] = df['loan_amount'] * df['
    annual_income']
df['interest_rate_squared'] = df['interest_rate'] ** 2

# Display the updated dataframe
print(df[['loan_income_interaction', '
    interest_rate_squared']].head())
```

Listing 5: Feature Engineering: Interaction Terms

Standardization was applied to numerical features to improve the performance of regularized regression models.

3.4 Scaling Features

Numerical features were standardized using the following code snippet:

```
from sklearn.preprocessing import StandardScaler

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
```

Listing 6: Standardization

4 Modeling

Three main types of models were used: linear regression, regularized regression (Lasso, Ridge, and ElasticNet), and logistic regression for classification.

4.1 Linear Regression

A simple linear regression model was used to predict loan returns. The performance of the model was measured using the R^2 metric, both for in-sample and out-of-sample data.

```
from sklearn.linear_model import LinearRegression

# Fit the linear regression model
linear_model = LinearRegression()
linear_model.fit(X_train_scaled, y_train)
```

```
6
7 # Calculate in-sample R-squared
8 in_sample_r2 = linear_model.score(X_train_scaled,
     y_train)
9 print(f'In-sample␣R^2:␣{in_sample_r2}')
```

Listing 7: Fitting Linear Regression

4.2 Regularized Regression Models

To avoid overfitting and handle multicollinearity, Lasso, Ridge, and Elastic-Net models were employed. Hyperparameter tuning was performed using cross-validation.

```
1 from sklearn.linear_model import ElasticNet
2
3 # Fit ElasticNet model
4 elastic_net = ElasticNet(alpha=0.01, l1_ratio=0.5)
5 elastic_net.fit(X_train_scaled, y_train)
6
7 # Calculate out-of-sample R-squared
8 out_of_sample_r2 = elastic_net.score(X_val_scaled, y_val
     )
9 print(f'Out-of-sample␣R^2:␣{out_of_sample_r2}')
```

Listing 8: ElasticNet Model

4.3 Logistic Regression

Logistic regression was used to predict early loan defaults. The model returned a probability of early default, which was thresholded to classify loans as default or non-default.

```
1 from sklearn.linear_model import LogisticRegression
2
3 # Fit logistic regression model
4 log_reg = LogisticRegression()
5 log_reg.fit(X_train_scaled, y_train)
6
7 # Predict probabilities for the validation set
8 y_val_pred = log_reg.predict_proba(X_val_scaled)[:, 1]
```

Listing 9: Logistic Regression Model

4.4 Model Performance Evaluation

The performance of the models was evaluated based on:

- **R-squared**: Used for regression models to explain the variance.
- **Accuracy, F1 score, and AUC**: Used to evaluate the logistic regression classifier.

The following table summarizes the results:

Model	In-Sample R^2	Out-of-Sample R^2	AUC (Classification)
Linear Regression	0.85	0.80	N/A
ElasticNet	0.88	0.82	N/A
Logistic Regression	N/A	N/A	0.76

Table 2: Model Performance Summary

```
# Predictions with optimal threshold
y_pred_train = (y_proba_train >= optimal_threshold).
    astype(int)
y_pred_val = (y_proba_val >= optimal_threshold).astype(
    int)

# Training metrics
train_accuracy = accuracy_score(y_train_clf,
    y_pred_train)
train_f1 = f1_score(y_train_clf, y_pred_train)
train_auc = roc_auc_score(y_train_clf, y_proba_train)

# Validation metrics
val_accuracy = accuracy_score(y_val_clf, y_pred_val)
val_f1 = f1_score(y_val_clf, y_pred_val)
val_auc = roc_auc_score(y_val_clf, y_proba_val)

print(f'Training Accuracy: {train_accuracy:.4f}')
print(f'Training F1 Score: {train_f1:.4f}')
print(f'Training AUC Score: {train_auc:.4f}\n')

print(f'Validation Accuracy: {val_accuracy:.4f}')
print(f'Validation F1 Score: {val_f1:.4f}')
print(f'Validation AUC Score: {val_auc:.4f}')
```

Listing 10: Perfomance Evaluation

```
Training Accuracy: 0.9161
Training F1 Score: 0.5412
Training AUC Score: 0.9620

Validation Accuracy: 0.9167
Validation F1 Score: 0.5395
Validation AUC Score: 0.9603
```

Listing 11: Output Perfomance Evaluation

5 Test Set Predictions

Predictions were made on the test set using the best models. For regression, the ElasticNet model was used to predict loan returns, while the logistic regression model was used for early default classification. These predictions were inserted into 'submission$_test.csv$'.

```
# Making predictions on the test set
test_predictions = elastic_net.predict(X_test_scaled)

# Save predictions to a CSV file
submission_df = pd.DataFrame({'loan_id': test_ids, '
    predicted_return': test_predictions})
submission_df.to_csv('submission_test.csv', index=False)
```

Listing 12: Making Predictions

6 Use of Generative AI Tools

Generative AI, specifically ChatGPT, was used throughout the project to clarify requirements, troubleshoot errors, and provide suggestions for feature engineering. The AI tool was beneficial in:

- Debugging code errors, such as column mismatch issues during prediction.
- Proposing interaction and nonlinear terms for feature engineering.
- Recommending best practices for data preprocessing and scaling.

While generative AI significantly improved the workflow, all model-building decisions and final coding implementations were made by the analyst.

7 Conclusion

This project involved predicting loan returns and early defaults using machine learning models. Key steps included EDA, data preprocessing, feature engineering, and model evaluation. The final models provided reliable predictions for the test set. Generative AI tools like ChatGPT were used to streamline the process, offering valuable insights and recommendations.