Loan Approval Classification

December 24, 2024

0.1 A Data Analysis Project by Garrett Power

I used a dataset from Kaggle for this project that you can find at this link:
https://www.kaggle.com/datasets/taweilo/loan-approval-classification-data/data

In this project I wanted to predict whether or not a person's (or borrower's) loan would be accepted or rejected using this synthetic dataset containing 45,000 records and 14 variables. This dataset includes continuous, discrete, binary numeric and categorical independent variables alongside a binary class target variable. These are those variables:
1. **person_age**, the age of that person in years
2. **person_gender**, the gender of that person
3. **person_education**, the education level of that person
4. **person_income**, the annual income of that person
5. **person_emp_exp**, the amount of years that person has been employed
6. **person_home_ownership**, the category of home ownership for that person
7. **loan_amnt**, the loan amount that person has requested
8. **loan_intent**, the category of the loan purpose
9. **loan_int_rate**, the interest rate of that loan
10. **loan_percent_income**, percentage of the loan in regards to the annual income of that person
11. **cb_person_cred_hist_length**, the length of credit history for that person in years
12. **credit_score**, the credit score of that person
13. **previous_loan_defaults_on_file**, whether or not that person has defaulted on a loan before
14. **loan_status**, our target variable indicated by a '1' for an approved loan or a '0' for a rejected loan

```
   age  gender    education  income  emp_exp home_ownership  loan_amnt  \
0   22  female       Master   71948        0           RENT      35000
1   21  female  High School   12282        0            OWN       1000
2   25  female  High School   12438        3       MORTGAGE       5500
3   23  female     Bachelor   79753        0           RENT      35000
4   24    male       Master   66135        1           RENT      35000

  loan_intent  loan_int_rate  loan_percent_income  cred_hist_length  \
0    PERSONAL          16.02                 0.49                 3
1   EDUCATION          11.14                 0.08                 2
2     MEDICAL          12.87                 0.44                 3
3     MEDICAL          15.23                 0.44                 2
4     MEDICAL          14.27                 0.53                 4

   credit_score previous_loan_defaults  loan_status
0           561                     No            1
1           504                    Yes            0
2           635                     No            1
3           675                     No            1
4           586                     No            1
```

Remarks: I imported the required libraries and loaded the csv file from Kaggle into a dataframe named 'df'. Many variable names have also now been shortened or modified to improve comprehensibility.

0.2 Initial Data Preprocessing

In my initial dataset preprocessing I wanted to address any null values, which I had not found, and conduct feature encoding. I used binary mapping for Gender, and Previous_Loan_Defaults. Label mapping for Education and Loan_Intent. One-Hot encoding for Home_Ownership. I created a new dataframe that contained these encoded variables named 'encoded_df'.

Null values:

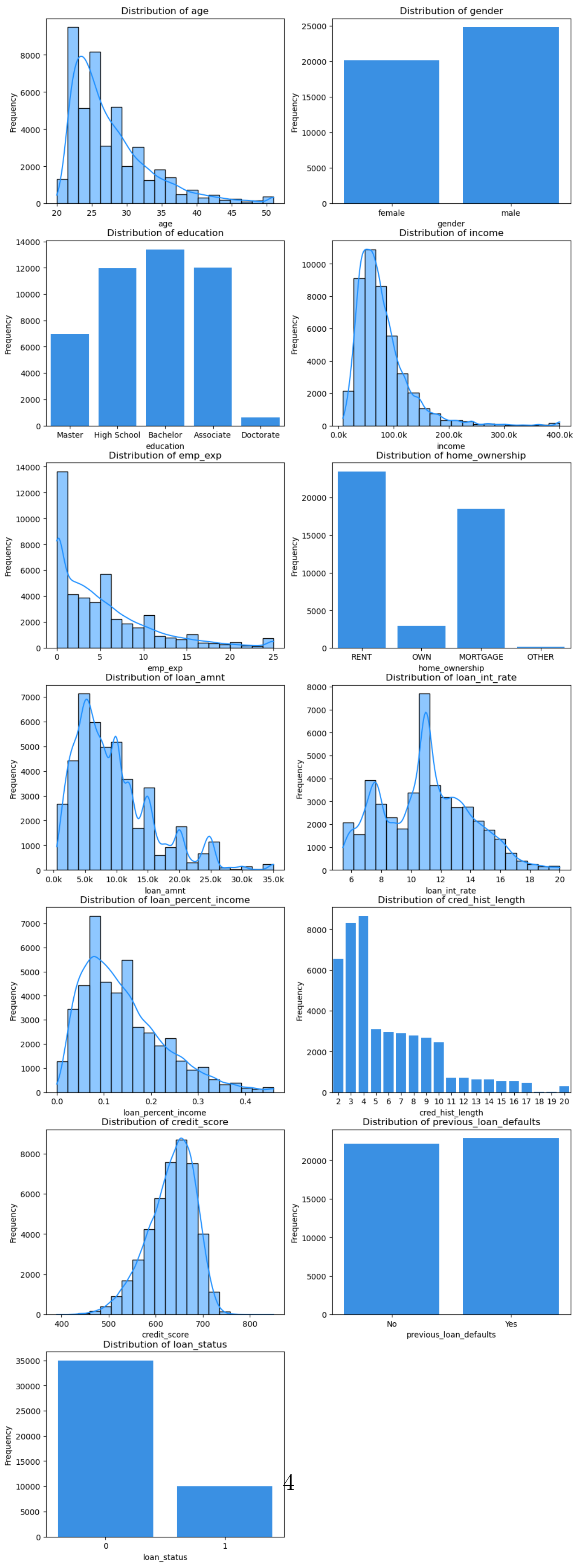
```
False
```

0.3 Exploratory Analysis

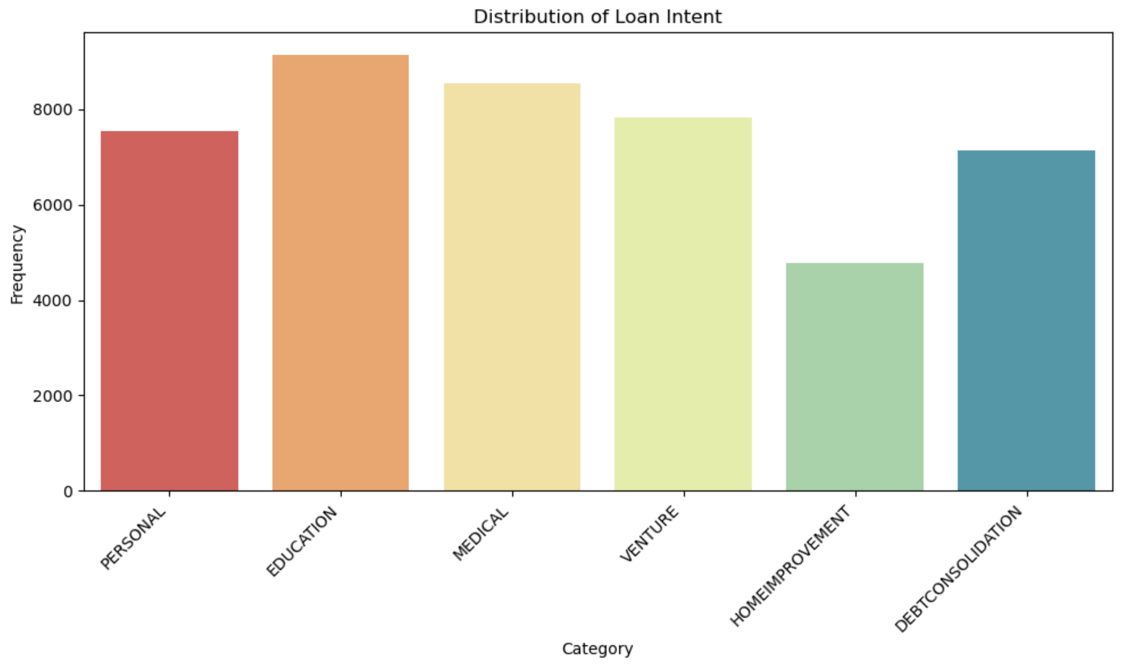
I started my exploratory analysis with summary statistics to check things such as value ranges (mix-max) per column and their respective count(). Many variables had odd outlier ranges, such as an age maximum of 144, that I will address by capping using the clip() function from NumPy. I viewed the dataset variable distributions with histograms and countplots, then investigated a multicollinearity obstacle using VIF analysis. Other than that, I knew I would need to be cautious in my approach to scaling and transforming my data later.

```
               age        income        emp_exp     loan_amnt  loan_int_rate  \
count  45000.000000  4.500000e+04  45000.000000  45000.000000   45000.000000
mean      27.764178  8.031905e+04      5.410333   9583.157556      11.006606
std        6.045108  8.042250e+04      6.063532   6314.886691       2.978808
min       20.000000  8.000000e+03      0.000000    500.000000       5.420000
25%       24.000000  4.720400e+04      1.000000   5000.000000       8.590000
50%       26.000000  6.704800e+04      4.000000   8000.000000      11.010000
75%       30.000000  9.578925e+04      8.000000  12237.250000      12.990000
max      144.000000  7.200766e+06    125.000000  35000.000000      20.000000

       loan_percent_income  cred_hist_length  credit_score   loan_status
count         45000.000000      45000.000000  45000.000000  45000.000000
mean              0.139725          5.867489    632.608756      0.222222
std               0.087212          3.879702     50.435865      0.415744
min               0.000000          2.000000    390.000000      0.000000
25%               0.070000          3.000000    601.000000      0.000000
50%               0.120000          4.000000    640.000000      0.000000
75%               0.190000          8.000000    670.000000      0.000000
max               0.660000         30.000000    850.000000      1.000000
```


Distribution of age
age
Frequency
Distribution of gender
female
male
gender
Distribution of education
Master
High School
Bachelor
Associate
Doctorate
education
Distribution of income
income
Distribution of emp_exp
emp_exp
Distribution of home_ownership
RENT
OWN
MORTGAGE
OTHER
home_ownership
Distribution of loan_amnt
loan_amnt
Distribution of loan_int_rate
loan_int_rate
Distribution of loan_percent_income
loan_percent_income
Distribution of cred_hist_length
cred_hist_length
Distribution of credit_score
credit_score
Distribution of previous_loan_defaults
No
Yes
previous_loan_defaults
Distribution of loan_status
loan_status

Remarks: As you can see, the variable range clipping allowed for more concise and interpretable plots. The **Age** column was quite left-skewed, which is an attribute I find often in datasets, that would imply that most of the sample population were of the age range 20-30 years old. **Gender** was fairly even in distribution for both male and female. I felt the **Education** distribution was representative of what you may find in the real world, with less borrowers having a Masters degree and far less having a Doctorate. **Income** was left-skewed and also what I would consider a real world representation, most of the population falling within the range of 25,000-125,000 (USD) salaries. **Employment_Experience** was left-skewed with majority of borrowers having no, or little work experience (in years). **Home_Ownership** depicted Renters as being the most commonly distributed, while those with Mortgages followed closely behind. Owners and the Other ownership category were not common in this dataset. **Loan_Amount** was left-skewed towards most borrowers asking for an amount in the range of 500-15,000 (USD). **Loan_Interest_Rate** had an interesting yet realistic distribution, with most borrowers falling around 8 or 11 percent on their loan. **Loan_Percent_Income** was left-skewed with majority of borrowers falling within the 0.1-0.2 loan-to-income range. **Credit_History_Length** was left-skewed with most borrowers having about 2-4 years of history, some having 5-10 years, and much less in the range of 11-20 years. **Credit_Score** was fairly normal in distribution with a mean value of about 630. **Previous_Loan_Defaults** was nearly equal in distribution, which I did not expect, indicating that about half of borrowers had previously defaulted on a loan while the other half had not. And finally, **Loan_Status** was imbalanced in distribution (about 80-20) with majority of borrowers falling into the Class ‘0’ indicating their loan had been rejected, compared to Class ‘1’ borrowers who had their loan approved.



Remarks: Loan_Intent was fairly even in distribution with most borrowers being of the intention categories 'Personal', 'Education', 'Medical', and 'Venture'. Less borrowers indicated their loan intention as 'Debt Consolidation' and much less for 'Home Improvement'.

0.3.1 VIF Analysis

```
              Feature         VIF
0               const  358.563818
1                 age   12.640777
2              income    2.872846
3             emp_exp    9.697015
4           loan_amnt    3.930621
5       loan_int_rate    1.026481
6  loan_percent_income    3.737263
7     cred_hist_length    4.304929
8        credit_score    1.034204
```

Remarks: Considering how many features were included in the dataset, I wanted to test for collinearity issues that could impact the learning models implemented later. What I found using this VIF analysis and other correlation methods was that the Age column was correlated heavily with Employment_Experience and Credit_History_Length (>0.85). I considered actions such as combining collinear features, but decided on removing the Age variable entirely from the learning models.

0.4 Additional Data Preprocessing

I needed to apply the same variable adjustments with clip() that I conducted on the original df onto the encoded_df I would be using for unsupervised/supervised learning models in this project. I also dropped the Age column to reduce multicollinearity impact from the encoded_df and scaled all non-binary numeric features using StandardScaler() by scikit-learn.

0.5 Principal Component Analysis (PCA)

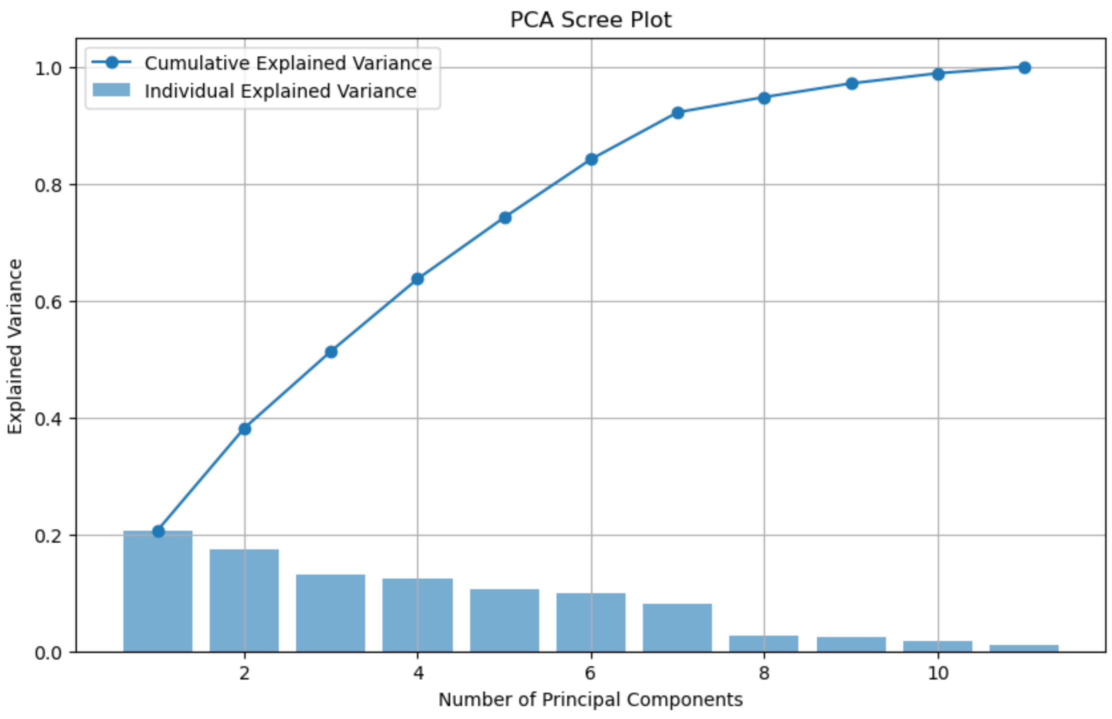
0.5.1 Unsupervised Learning Model

In my unsupervised learning mode, I chose to examine a PCA approach on this dataset. I dropped the boolean encoded Home_Ownership feature and our target variable Loan_Status to create a dataframe specifically for this, called 'unsupervised_df'. I used explained, cumulative, and noise variance as evaluation metrics for principal component analysis and visualized this further using a Scree plot.

```
Explained Variance per component:
[0.206769   0.174769   0.13158852 0.12375652 0.10524713 0.09945605
 0.08040163 0.02602915 0.02339177 0.01734869 0.01124255]

Cumulative Explained Variance:
[0.206769   0.381538   0.51312652 0.63688303 0.74213016 0.84158621
 0.92198784 0.94801698 0.97140876 0.98875745 1.        ]

Noise Variance: -2.220446049250313e-16
```



Remarks: With a Noise Variance value close to 0, I looked to the Scree plot and found the results to be rather intriguing. If I were to implement PCA into my learning models, I would probably utilize 8 principal components to capture about 95% of explained dataset variance and minimizing noise inclusion.

0.6 Train & Test Split for Supervised Learning Models

I used an 80:20 train & test split ratio on my encoded dataframe. I chose Logistic Regression as my first supervised learning model and an MLP neural network as my second. I did try a few different methods in transforming my data to improve learning model performance, but simply encoding and scaling seemed to be most effective. Anytime I introduced methods such as resampling, the models would perform slightly worse. I found this to be acceptable though, as imbalanced data in scenarios like this would be a realistic expectation to have.

0.7 Logistic Regression

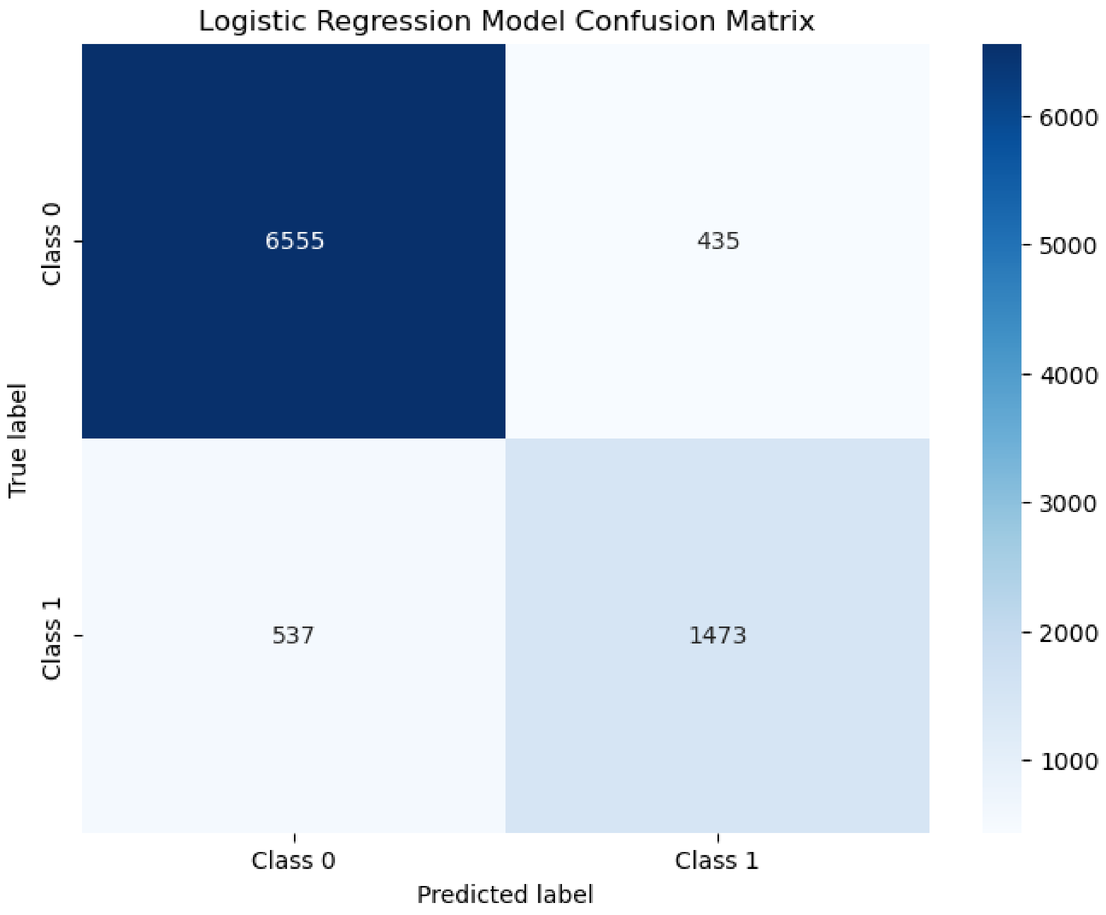
0.7.1 Supervised Learning Model

For this model I kept many default parameters such as the 'lbfgs' solver, utilizing the encoded and scaled dataframe with the respective train & test sets formed previously. The evaluation metrics I included are the ROC-AUC, a classification report, and a confusion matrix for the model's prediction ability on our target variable Loan_Status. (Class 1 = accepted, Class 0 = rejected)

```
Logistic Regression Model ROC-AUC: 0.95
Logistic Regression Model Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.94      0.93      6990
           1       0.77      0.73      0.75      2010

    accuracy                           0.89      9000
   macro avg       0.85      0.84      0.84      9000
weighted avg       0.89      0.89      0.89      9000
```



Remarks: My Logistic Regression model had decent scores overall, especially for majority borrowers within the Class '0' classification. The model still struggled on Class '1' of borrowers who had been approved based on the encoded and scaled data. I would definitely try to improve model fitting for Class '1' if I were to continue working on this Logistc Regression approach.

0.8 Multi-Layer Perceptron (MLP)

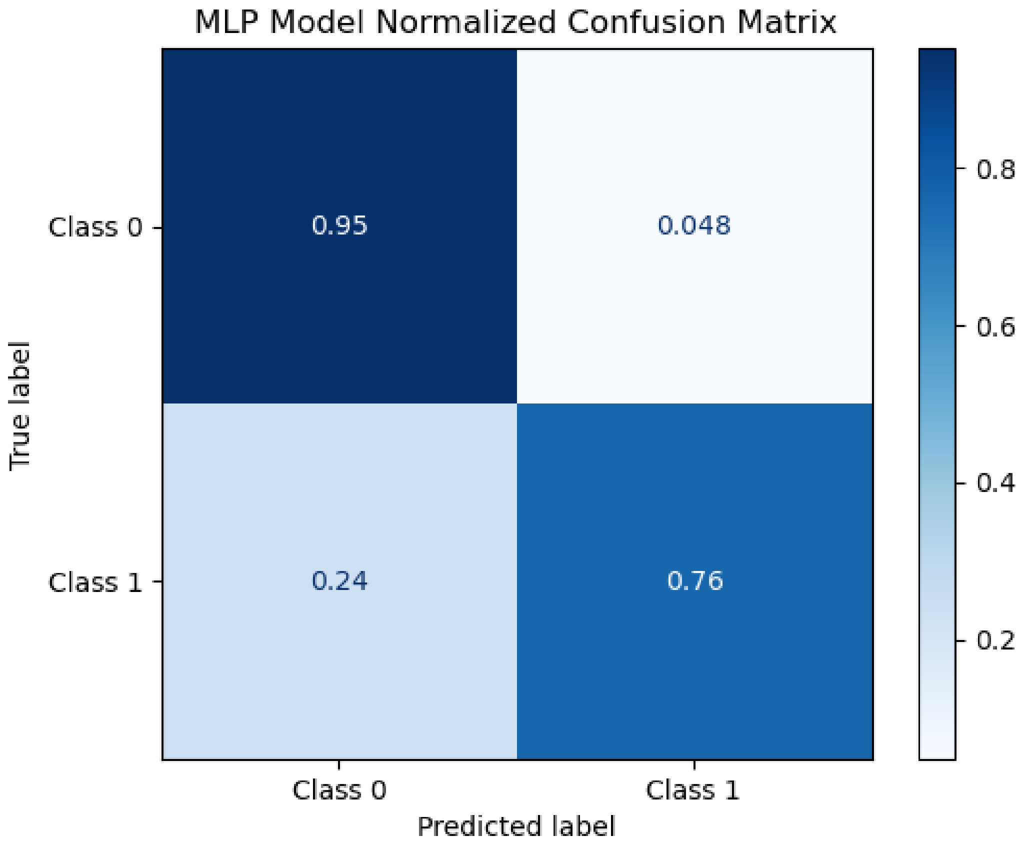
0.8.1 Supervised Neural Network Model

For this model I chose to use MLPClassifier in hopes of improving prediction capability on our target variable Loan_Status. I kept many default parameters such as the 'ReLU' activation function and 'adam' weight optimizer. The evaluation metrics I included for this model are similar to those included in the Logistic Regression model. The ROC-AUC, a classification report, and a now normalized confusion matrix to display percentages for classification performance. (Class 1 = accepted, Class 0 = rejected)

```
MLP Model ROC-AUC: 0.96
MLP Model Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      6990
           1       0.82      0.76      0.79      2010

    accuracy                           0.91      9000
   macro avg       0.88      0.86      0.87      9000
weighted avg       0.91      0.91      0.91      9000
```

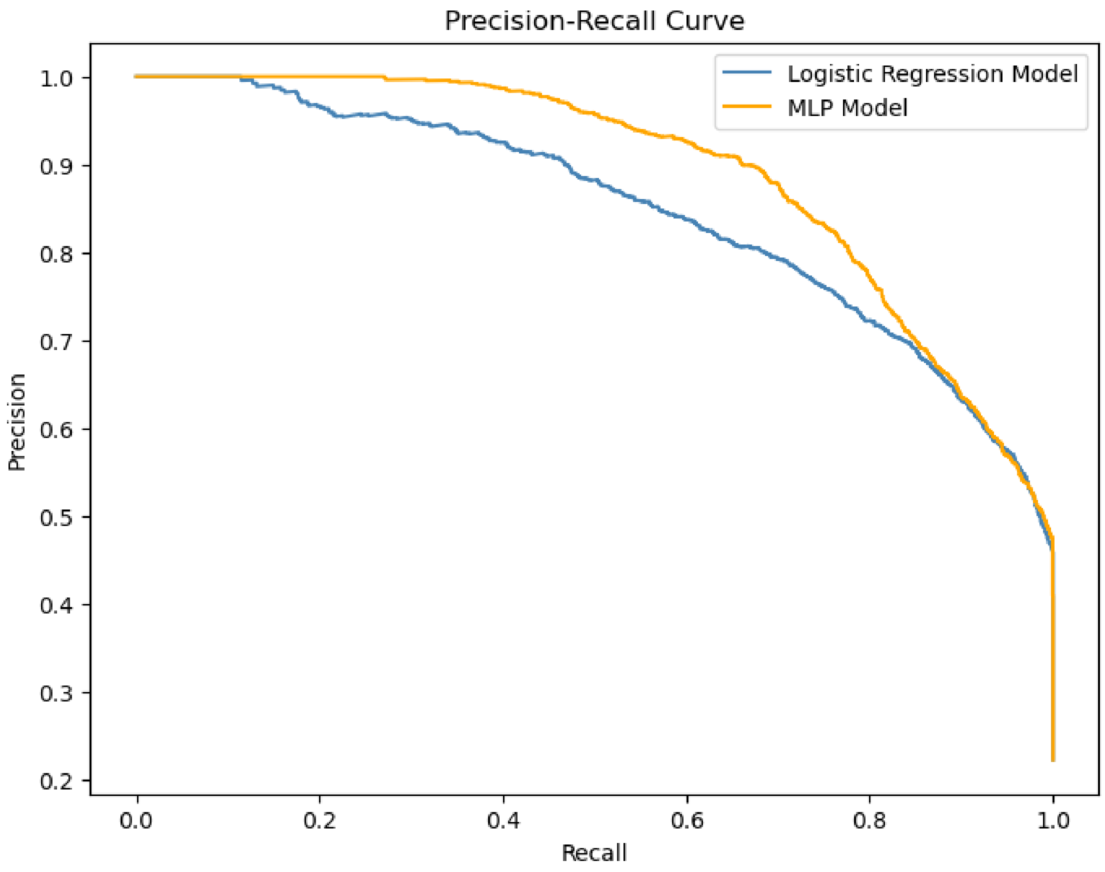


Remarks: My MLP Neural Network model performed better than the previous Logistic Regression approach. I included a normalized version of the confusion matrix to better represent classification performance for this model. Compared to the Logistic Regression predictions on Class '1' (around 35% mislabel), this model had reduced the misclassification of Class '1' borrowers to about 25%. The model evaluation scores are better overall. but the most important takeaway is the improvement of Class '1' performance. I can definitely see the ability to further improve model fitting for this approach, allowing the ability to effectively handle imbalanced target variable distribution.

0.9 Supervised Learning Models Comparison

0.9.1 Precision-Recall Curve

After training and testing both models I wanted another way to visually compare their performance. Especially after the evaluation metrics suggested that the models performed similarly (slightly better for the MLP model), I felt it would be interesting to view the difference in precision-recall ability.



Remarks: As you can see, the MLP Neural Network model did quite well on Precision-Recall performance. The Logistic Regression model performed slightly worse, with a lacking Precision capability. Ideally, I would try targetting Precision performance to hopefully increase both model's class prediction ability.

0.10 Conclusion

My MLP Neural Network model seemed to perform the best, while my Logist Regression model lagged behind slightly in evaluation metric results. I enjoyed working with this dataset and welcomed the challenges, such as the combination of variable types, to learn a more effective approach for analysis. I felt as though my approach was realistic with this kind of dataset and variable distributions, and am motivated by my results using MLPClassifier. If I had more time I would look for more interesting ways in evaluating the model's performance, improving model fit to try and bolster Class '1' prediction ability, and potentially utilizing methods such as the aforementioned PCA to gain more compelling attribute-related insights.