DIAML Assignment 3

Georgina Njoroge - gnjoroge

November 2025

Question 1: Exploratory Data Analysis

Methods

The data were first loaded from the csv file using the Panda library and stored in a dataframe. The data structure, information, and summary were then analyzed using the Panda library. Finally, histograms showing the distributions for the numerical columns were plotted to analyze their distributions.

Results

The following are the results of the exploratory data analysis.

```
Shape: (614, 13)

Data Information
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
None
```

Figure 1: This figure shows the data structure and properties of the data. There are 614 rows and 13 columns. There are 8 features with object type, 4 with float type and 1 with integer type. Some columns have missing data.

From this table, there are 8 clear categorical data types. The categorical data can be classified as ordinal for education level and number of dependents and nominal for the remaining columns, such as gender and property area, that do not have any meaningful order. Despite credit history being represented as float

value, it is a categorical data type with 1 representing a 'Yes' and 0 representing a 'No'. There are 4 numerical features; applicant income, co-applicant income, loan amount, and loan amount term.

	count	mean	std	min	25%	50%	75%	max
ApplicantIncome	614.0	5403.459283	6109.041673	150.0	2877.5	3812.5	5795.00	81000.0
CoapplicantIncome	614.0	1621.245798	2926.248369	0.0	0.0	1188.5	2297.25	41667.0
LoanAmount	592.0	146.412162	85.587325	9.0	100.0	128.0	168.00	700.0
Loan_Amount_Term	600.0	342.000000	65.120410	12.0	360.0	360.0	360.00	480.0
Credit_History	564.0	0.842199	0.364878	0.0	1.0	1.0	1.00	1.0

Figure 2: This figure shows a table of the summary statistics of the numerical columns including mean, max and standard deviation.

From this table, we can see that the data have extreme values. For example, applicant income has a minimum of 150 and a maximum of 81000 while 50% of the data range between 2877.5 and 5795.0. This trend is also evident for the co-applicant income, loan amount, and loan amount term.

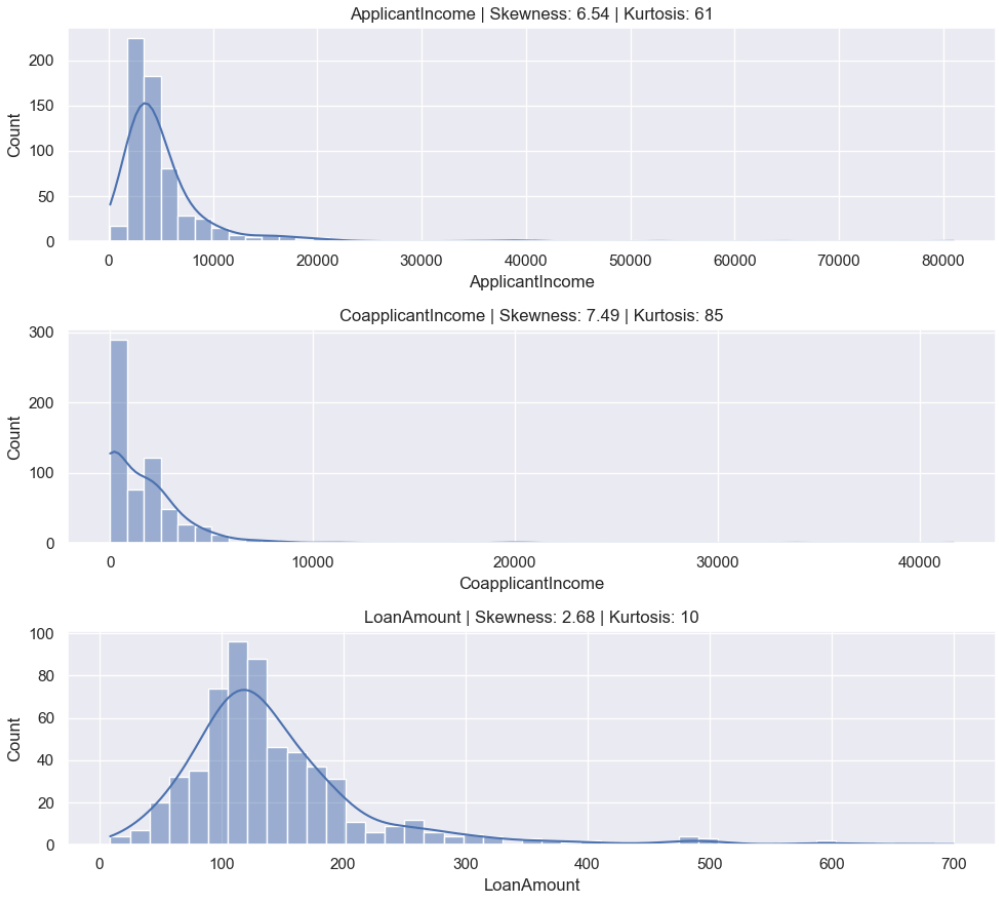

Figure 3: This figure shows the skewness, kurtosis and distributions of the numerical columns, represented by histograms. The numerical features represented include applicant income, co-applicant income and loan amount.

From this plot, we can see that applicant and co-applicant income cluster around the lower values with few high income earners. Regarding loan amounts, most of them range from 100 to 150.

Discussion

The applicant, co-applicant income, and loan amounts are all positively skewed, with the first two having highly skewed data. This is expected as there are fewer high earners, whereas most people earn low income. Loan amounts are also right-tailed because most people cannot afford expensive loans, and hence most amounts are concentrated in the lower amounts.

Most models assume normality in the data. Therefore, in order to prevent the outliers from skewing the results, it is important to transform the data to achieve model stability.

Combining categorical and numerical features may pose a problem due to different scales and encoding needs. Models interpret numerical data differently from categorical data whose encoding does not represent any numerical significance.

Question 2

Methods

For this section, the data was analyzed for missing values and outliers. To determine missing values, built-in Python functions were used. The missing values were dealt with using imputation strategies. The missing values in the categorical data were filled by by the mode, and the numerical values were filled in by the median, since the data were mostly skewed. Therefore, the median was a better estimator compared to the mean which is affected by skewness. To better visualize the distributions for applicant income, co-applicant income, and loan amount, box-plots were drawn before and after dealing with outliers.

Outliers were identified using the **interquartile range** , as the distribution of the three columns was skewed, with long tails. This is recommended over the z-score, as the latter assumes a normal distribution in the data and depends on mean and standard deviation, which is not the case for these features. The identified outliers were then dropped on the basis of the specified numerical column bounds.

Results

The following are the results of the data cleaning.

```
Loan_ID              0.000
Gender               2.117
Married              0.489
Dependents           2.443
Education            0.000
Self_Employed        5.212
ApplicantIncome      0.000
CoapplicantIncome    0.000
LoanAmount           3.583
Loan_Amount_Term     2.280
Credit_History       8.143
Property_Area        0.000
Loan_Status          0.000
dtype: float64
```

Figure 4: This image shows the missing values for each column as percentages of the total dataset.

From the figure, we can see that several columns, such as loan id, education, and applicant income, do not have missing values. On the other hand, the credit history column has the most missing values at 8% followed by credit history at 5% and loan amount at 3%.

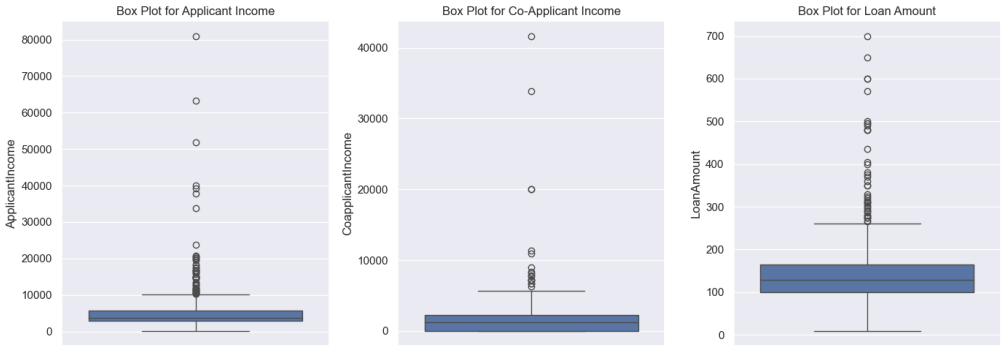

Figure 5: This image shows box plots for applicant income, co-applicant income and loan amount before the outliers were handled.

From this figure, we can see that the values are skewed to the right with long tails. Although most values are concentrated at the lower amounts, some outliers are extremely far from most of the data. For example, in applicant income, there is an extreme value greater than 80,000, which is significantly higher than the mean, which is lower than 10,000.

```
IQR value for column ApplicantIncome is: 2917.5
Lower Bound: -1498.75
Upper Bound: 10171.25
Outliers Percentage: 7.53

IQR value for column CoapplicantIncome is: 2297.25
Lower Bound: -3445.875
Upper Bound: 5743.125
Outliers Percentage: 2.85

IQR value for column LoanAmount is: 64.5
Lower Bound: 3.5
Upper Bound: 261.5
Outliers Percentage: 6.26
```

Figure 6: This image displays the bounds and number of outliers for the numerical features expressed as percentages of the total dataset length.

From this figure, we can see that the column with the most outliers is the applicant income, followed by the loan amount.

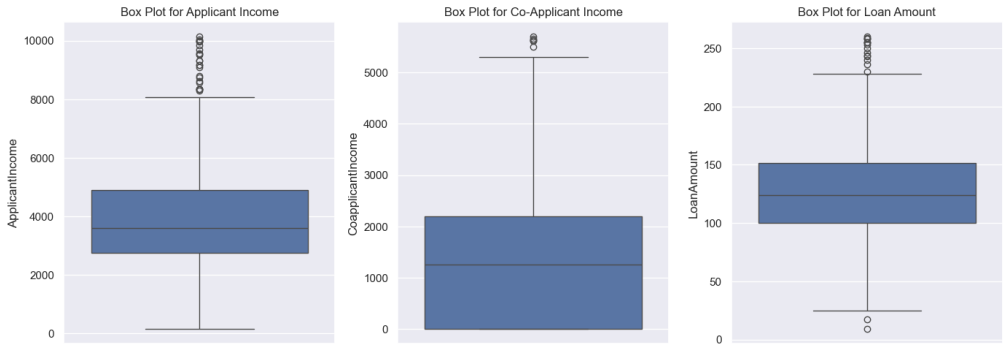

Figure 7: This image shows box plots for applicant income, co-applicant income and loan amount after the outliers were handled.

From this figure, we can see that the tails are now shorted with fewer extreme values. Although the features are still skewed, the effect of extreme values has been reduced.

Discussion

Removing outliers reduces the model variance, but increases the bias. In this case, mode was used for imputation, which often leads to reinforcement of the dominant class. This leads to bias in the model and can overshadow minority

patterns. Additionally, removing outliers also causes rare events that may be informative to be overlooked.

Question 3

Methods

In this section, new features were created by combining existing features. Total income as the sum of applicant and co-applicant income and loan amount per income as the loan amount divided by the total income.

The numerical columns, applicant income, co-applicant income, total income, and loan amount were transformed by applying logs to the values. This transforms the skewed data by compressing large values and expanding small values. In order to prevent logarithmic errors in logarithm (0), a constant value, 1, was added to the data points.

The categorical values were encoded using the label-encoding and the one-hot encoding. Label encoding was used for ordinal data, such as education and the number of dependents. This maintained the inherent order in the data, with values such as 3+ in dependents and graduate in education encoded with the highest numerical value. Additionally, other nominal data points with just two categories, such as gender, marriage status, loan status, and whether they were self employed, were also encoded using label-encoding. This is because, for binary columns, one-hot encoding has the same effect as label encoding, assigning 0s and 1s. Finally, the property area was encoded using one-hot encoding as it had three categories, resulting in 3 columns for each.

Since the data had been transformed and were approximately normal, the standard scaler was used to scale the numerical features, including the loan term amount. This ensures a uniform scale across features and prevents some from overshadowing others during model training.

Finally, a Pearson correlation heatmap was plotted to display the nature of the relationships among the features. This helped identify multicollinearity and guided feature selection by highlighting strong linear associations. Since the features were modified in-place in the dataframe, there were no redundant columns and therefore the only column that was dropped was the loan id, which contributed no information.

Results

The following are the results of feature transformation and engineering.

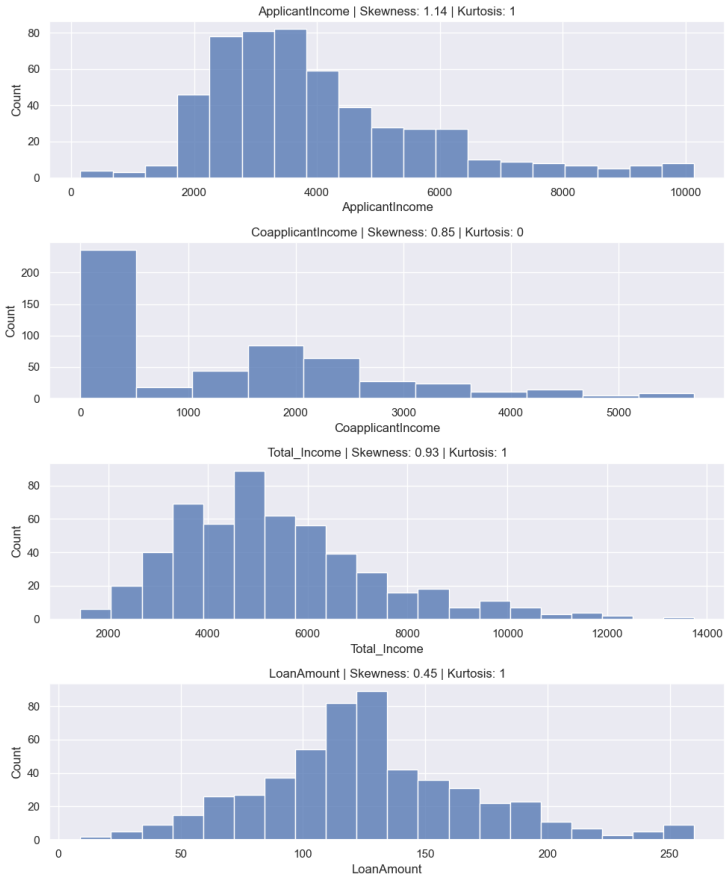

Figure 8: This figure shows the histograms for the numerical columns, applicant income, co-applicant income, total income and loan amount before they are transformed.

From this figure, we can see that the range of data is spread from zero to large values such as 14,000 in total income. There is still evidence of skewness despite dealing with outliers in the previous step.

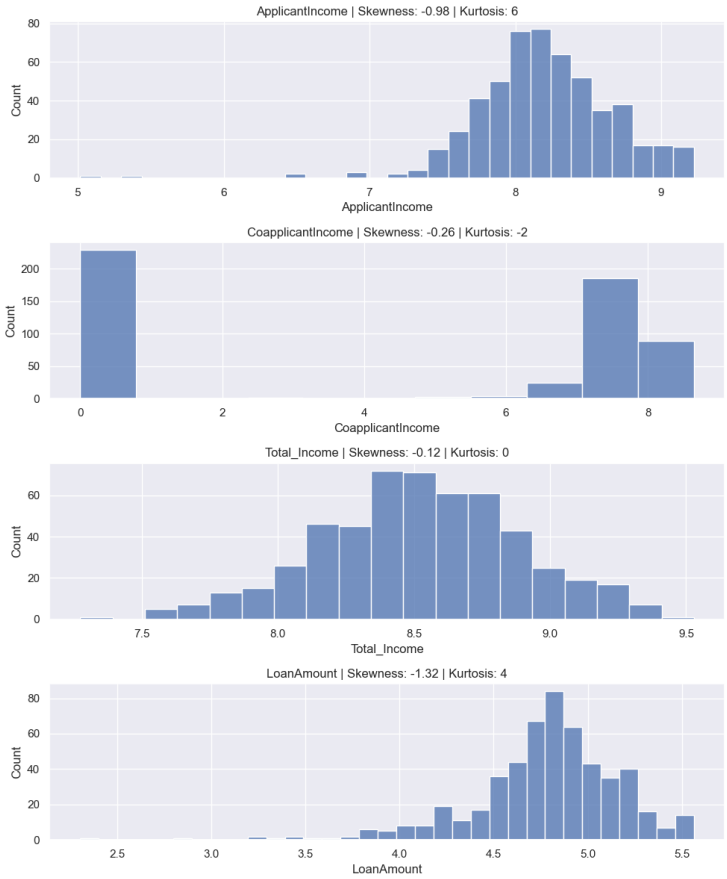

Figure 9: This figure shows the histograms for the numerical columns, applicant income, co-applicant income, total income and loan amount after they are transformed.

From this figure, we can see that the scale of the data changes from thousands to values between zero and ten. Based on the calculated skewness, we can see that the values have decreased with all of them being inverted to left-skewed data, as evidenced by the negative skewness values.

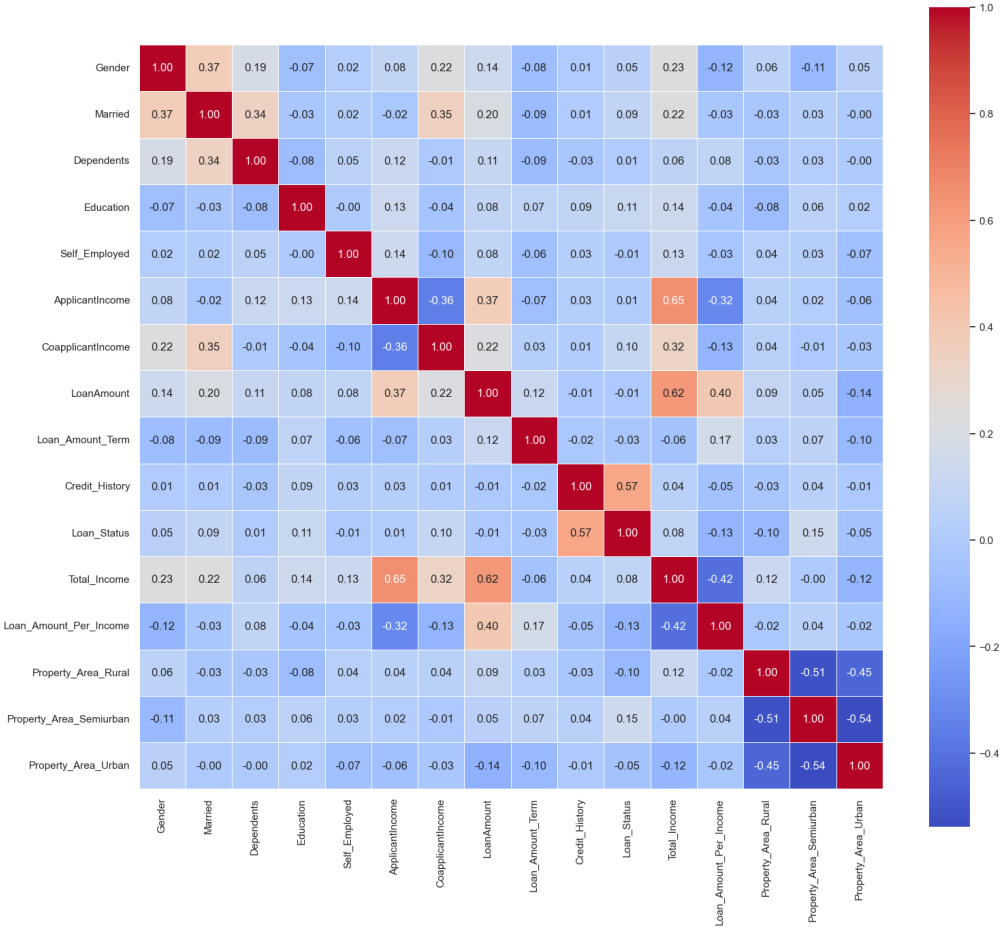

Figure 10: This heatmap shows the Pearson Correlation for all the features and the target column. It shows how the columns are correlated with each other.

From this figure, we can see that the highest correlation is between the columns and themselves. The other highly correlated columns were applicant income and total income at 0.65, followed by loan amount and total income at 0.62. Credit history has the highest correlation, 0.57, with the target column, loan status.

Discussion

Encoding Methods

Label Encoding — Label encoding converts categorical variables into integers by assigning each category a number. For example, Graduate = 0 and Not Graduate = 1. This method is simple and memory-efficient. However, it intro-

duces an ordinal relationship that may not exist, making the model think one category is "greater than" another. Label encoding is best for ordinal variables where order matters, such as education levels or credit ratings.
One-Hot Encoding — One-hot encoding creates binary columns for each category. A feature with three categories becomes three columns with 1s and 0s. This avoids false ordering but increases dimensionality. One-hot encoding is best for nominal variables with no natural order, like gender or property area.
Comparison — Use label encoding for ordinal data and one-hot encoding for nominal data. Label encoding works well with tree-based models, while one-hot encoding is preferred for linear models like logistic regression or SVM.

Scaling Methods

Min-Max Scaling — Min-Max scaling transforms features to a range [0, 1] using:

$$x_{scaled} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

It preserves the distribution shape but is sensitive to outliers. Use it when data has bounded ranges or for neural networks and image processing.
Standardization — Standardization transforms features to mean = 0 and standard deviation = 1 using:

$$x_{scaled} = \frac{x - \mu}{\sigma}$$

It is less sensitive to outliers than Min-Max scaling. Use it for logistic regression, SVM, or PCA, especially when features have different units.
Robust Scaling — Robust scaling uses the median and IQR:

$$x_{scaled} = \frac{x - median}{IQR}$$

It is resistant to outliers. Use it when data contains significant outliers or is heavily skewed.
Unit Vector Scaling — Unit vector scaling normalizes each sample to unit norm:

$$x_{scaled} = \frac{x}{\|x\|_2}$$

It scales rows, not columns. Use it for text classification, cosine similarity, or when direction matters more than magnitude.

Question 4

Methods

In this section, we split the dataset, 70% for training and 30% for testing. We used a stratified set because we wanted to ensure that the data were balanced.

In addition, a random state was set for reproducibility. We trained three models, logistic regression, SGD classifier and SVM with linear kernel. We ran these for 1000 iterations and also specified a seed for reproducibility. The trained models were then evaluated using the test set. Accuracy, precision, recall, and f1-score were used to evaluate model performance.

Results

	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.850932	0.838462	0.973214	0.900826
SGD Classifier	0.782609	0.829060	0.866071	0.847162
SVM (Linear Kernel)	0.863354	0.840909	0.991071	0.909836

Figure 11: This image shows the models' performances using accuracy, precision, recall and f1-score as the performance metrics.

From this figure, we can see that the SVM (Linear Kernel) is the best overall performer with 86.3% accuracy, 99.1% recall, and 0.91 f1-score. SVM maximizes the margin between approved and denied loans, leading to high recall and an overall balance between precision and recall.

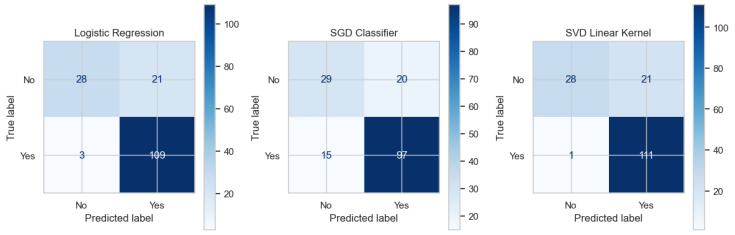

Figure 12: This figure shows the confusion matrices for each model.

From this figure, we can see that SVM has the highest number of true positives at 111 and SGD the lowest at 97. SGD has the highest number of false negatives at 15. Logistic regression performs well overall, with few false negatives, rejecting valid applicants. SGD has the worst performance with significantly more false negatives.

Discussion

Imbalanced dataset - This is a dataset that has target classes that are not represented equally. For example, in these data, an imbalanced data set would have few samples of unapproved loans, with the majority of the data for approved loans. This could affect the model performance especially when generalizing on unseen minority data.

Evaluation metrics - Evaluation metrics used to measure performance in classification problems with class imbalance issues include precision, recall, f1-score, and specificity. Precision determines the proportion of correctly predicted positive values among all predicted values. Recall computes the proportion of correctly predicted positive values among all actual positives. The F1-score is the harmonic mean of precision and recall. Finally, specificity measures how well the model detects the negative class.

Detecting bias or variance problems - Often, machine learning models have a bias-variance tradeoff. High bias occurs when the model is underfitting because it is too simple to model complex data. This is evident when the model accuracy is low for both the train and the test sets.To fix this issue, we could use a more complex model. High variance, on the other hand, occurs when the model is overfitting because it is too complex for simple data. This is evident when the model performs well during training and poorly on the test set. This can be solved by using more data points or simplifying the model.

Question 5

Methods

In this section, we analyzed bias in the SVM model and explored interpretability methods. We examined how to detect bias by checking if the model treats certain groups unfairly. We also explored bias mitigation strategies to reduce or correct identified biases. We compared LIME and SHAP methods, discussing their trade-offs in explaining predictions. SHAP was used to show how each feature contributes to individual predictions. We compared feature contributions across groups such as employment types and education levels to identify features driving prediction bias.

Results

From this figure, we can see that credit history is the most important feature in determining loan approval, followed by applicant income and loan amount. High feature values (red) push predictions toward approval, while low values (blue) push toward denial. The horizontal spread shows the magnitude of each feature's impact on predictions.

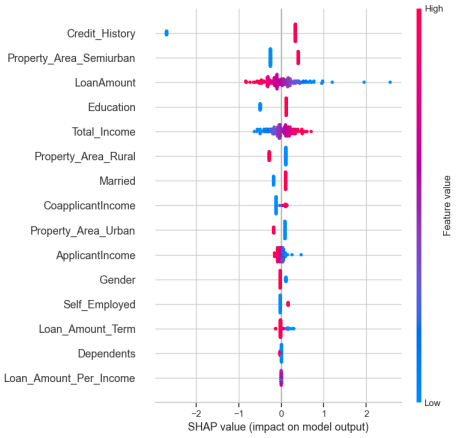

Figure 13: This figure shows the SHAP summary plot indicating feature importance and their impact on model predictions.

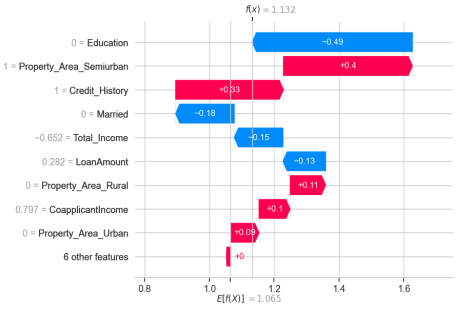

Figure 14: This figure shows the SHAP waterfall plot for a single prediction, showing how features contributed to the final output.

From this figure, we can see how each feature moved the prediction from the base value to the final predicted value. Positive SHAP values (red) push toward approval, while negative values (blue) push toward denial. The bar length represents each feature's contribution strength for this specific instance.

Discussion

Bias detection - Bias occurs when a model systematically discriminates against certain groups, leading to unfair outcomes. To detect bias, we analyze model performance across demographic groups, checking if accuracy, precision, or recall differ significantly. We also check if protected attributes correlate with predictions inappropriately. In loan approval, bias might show as lower approval rates for certain employment types or education levels, even when applicants are qualified.

Bias mitigation strategies - Several strategies can reduce bias once detected. Pre-processing methods modify training data to remove bias, such as resampling underrepresented groups. In-processing methods add fairness constraints during training. Post-processing methods adjust outputs after training to meet fairness metrics. These strategies involve trade-offs, as improving fairness often reduces overall accuracy.

LIME vs SHAP - LIME approximates the model locally using a simpler interpretable model. It is fast but can be unstable, giving different explanations for similar instances. SHAP uses game theory to calculate each feature's contribution, providing consistent and theoretically grounded explanations. SHAP is more reliable but computationally expensive. We chose SHAP because it provides stable explanations, which is important for justifying loan decisions.

Feature contributions across groups - Comparing SHAP values across groups helps identify features driving bias. If credit history impacts self-employed applicants more than salaried employees, this indicates potential bias. Our analysis showed that applicant income and loan amount had similar impacts across groups, but credit history showed some variation.

Making the model fairer - To improve fairness, we should reduce the influence of sensitive attributes that could cause discrimination. We can implement fairness constraints during training to ensure equal treatment across groups. Regular audits using SHAP can monitor the model's behavior over time. Providing clear explanations for every prediction improves transparency. These steps ensure the model operates ethically and builds user trust.