MÄLARDALEN UNIVERSITY
SCHOOL OF INNOVATION, DESIGN AND ENGINEERING
VÄSTERÅS, SWEDEN

---

Project report

# PREDICTIVE DATA ANALYTICS DVA478

Rickard Sörlin – Rsn21005@student.mdu.se

Examiner: Shahima Begum, Mobyen Uddin Ahmed
Mälardalen University, Västerås, Sweden

Date: 2025-October-9

# Abstract

*ABC Finance aims to automate its loan process by building two classification models trained on its data. After constructing an Analytic Base Table (ABT), it became apparent that the data was imbalanced and required thorough preprocessing. The results indicate that the decision tree classifier performed better than the Logistic Regression model on the test set. However, questions remain whether the model's performance in a real production environment would perform as well and whether potential gender bias could affect the fairness for customers.*

Keywords: Machine Learning, Prediction, Pre-processing

# Table of Contents

**1. Introduction and problem formulation ..... 4**

**2. Method ..... 5**

*2.1. Implementation ..... 5*
*2.2. Libraries ..... 5*
*2.3. Data Exploration ..... 5*
*2.4. Data Pre-processing ..... 7*
*2.5. Feature Engineering ..... 8*
*2.6. Development of Predictive Models ..... 8*

**3. Result ..... 9**

**4. Model Evaluation ..... 10**

**5. Discussion and future work ..... 11**

**6. Conclusions ..... 12**

**References ..... 13**

# 1. Introduction and problem formulation

The finance company, ABC Finance, wants to automate the process of home loans. The task is to help them build a model based on their data. The model should predict whether a new customer can be approved for a loan or not based on several customer attributes in the data set. Since the target LaonStatus is binary (Y/N), the task is a binary classification problem. The model to be used is Logistic Regression[1, p. 409] as a baseline model and a decision tree [1, p. 544], where the intention is to follow best practice regarding data exploration, data preprocessing, and feature engineering. The evaluation of the model is performed using accuracy, precision, recall, F1-score, and Confusion Matrix metrics [1, pp. 463–477].

# 2. Method

## 2.1. Implementation

As this project is a classification problem, the choice of model is Logistic Regression [1, p. 409] as the baseline and a decision tree model [1, p. 544].

## 2.2. Libraries

Pandas is chosen as a data framework as it provides many built-in methods for handling and processing data. Matplotlib is used for visualization. Scikit-learn is used for its easy use regarding setting up a data pipeline and loading different models in Python.

## 2.3. Data Exploration

During data exploration, an Analytic Base Table (ABT) was constructed as seen below in Table 1[1, p. 71]. Some features appear to have the wrong data type, such as ´Dependents´, where the reason appears to be incorrect value formatation using a plus sign. Additionally, some features have missing values that will be addressed later using imputation [1, p. 121]. However, no feature has too many missing values, so all features will be kept except Unnamed:0. The dataset has class imbalance in the target variable LoanStatus with 422 Yes labels, which is about 69% of the class [1, p. 539], and will be addressed using the up-sampling technique SMOTE [2] later on.

I believe there is some bias in the features as well regarding the Gender feauture, where 489 is male, which is about 80% of the data set [1, p. 147].

Using the describe method of the Pandas library, I obtained an overview of how the data was distributed between low and high percentiles. This makes it easier to spot outliers, which could interfere with the training. There appear to be some outliers when looking at the $50^{th}$ and $75^{th}$ percentiles and comparing them to min and max in Table 1, for features ApplicationIncome, CoapplicantIncome, and LoanAmount [1, p. 540].

**Table 1 Analytic Base Table (ABT)**

| Feature | count | unique | top | freq | mean | std | min | 25% | 50% | 75% | max | Missing | dtype |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Unnamed: 0** | 614 | | | | 306,5 | 177,39 | 0 | 153,25 | 306,5 | 459,75 | 613 | 0 | int64 |
| **LoanID** | 614 | 614 | LP001002 | 1 | | | | | | | | 0 | object |
| **Gender** | 601 | 2 | Male | 489 | | | | | | | | 13 | object |
| **Married** | 611 | 2 | Yes | 398 | | | | | | | | 3 | object |
| **Dependents** | 599 | 4 | 0 | 345 | | | | | | | | 15 | object |
| **Education** | 614 | 2 | Graduate | 480 | | | | | | | | 0 | object |
| **SelfEmployed** | 582 | 2 | No | 500 | | | | | | | | 32 | object |
| **ApplicantIncome** | 614 | | | | 5403,5 | 6109 | 150 | 2877,5 | 3812,5 | 5795 | 81000 | 0 | int64 |
| **CoapplicantIncome** | 614 | | | | 1621,2 | 2926,25 | 0 | 0 | 1188,5 | 2297,25 | 41667 | 0 | float64 |
| **LoanAmount** | 592 | | | | 146,4 | 85,59 | 9 | 100 | 128 | 168 | 700 | 22 | float64 |
| **LoanAmountTerm** | 600 | | | | 342 | 65,12 | 12 | 360 | 360 | 360 | 480 | 14 | float64 |
| **CreditHistory** | 564 | | | | 0,84 | 0,36 | 0 | 1 | 1 | 1 | 1 | 50 | float64 |
| **PropertyArea** | 614 | 3 | Semiurban | 233 | | | | | | | | 0 | object |
| **LoanStatus** | 614 | 2 | Y | 422 | | | | | | | | 0 | object |

Further investigation and visualization of the outliers were performed using boxplots as seen in Figures 1,2 and.

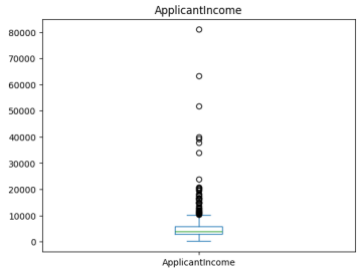


**Figure 1 shows an indication of outliers in ApplicantIncome.**

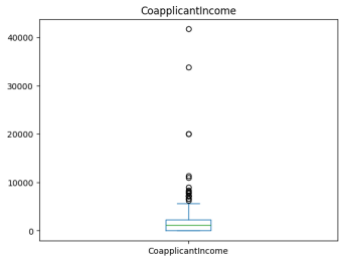


**Figure 2 shows indications of outliers in the feature CoapplicantIncome.**

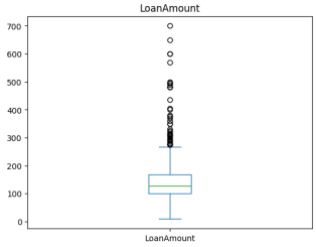


**Figure 3 shows an indication of outliers in LoanAmount**

Before investigating feature correlation, two features were dropped, LoanID and Unnamed, as they are not useful for feature prediction and correlation. The CreditHistory feature showed a very high correlation with the target feature LoanStatus_Y, as seen in Table 2 below. Further investigation will be conducted, but without domain knowledge, I believe it can be valid.

**Table 2 Top 5 highly correlated variables with the target variable LoanStatus.**

| Feature | |
|---|---|
| LoanStatus_Y | 1 |
| CreditHistory | 0,561678 |
| PropertyArea_Semiurban | 0,13654 |
| Married_Yes | 0,084281 |
| Dependents_2 | 0,062384 |
| Gender_Male | 0,025407 |

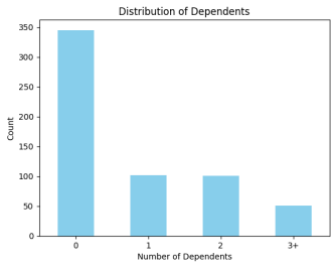


**Figure 4 shows the distribution of the Dependents feature.**

## 2.4. Data Pre-processing

During the data exploration, where the data was analysed using the ABT in the Table 1, it clearly showed which features had missing values. A data preprocessing pipeline was set up using the Scikit-learn library, as seen in Figure 5 for Logistic Regression and Figure 6 for the decision tree.

If the features that had missing values were numerical, they were imputed using the mean, and if categorical, using Mode with Scikit-learn´s SimpleImputer function.

The numerical feature was normalized using StandardScaler in the pipeline, as many of the features seen in Table 1 of the ABT had different ranges [1, p. 142]. Since some of the numerical features have outliers and the Logistic Regression performed worse than the decision tree in accuracy, I attempted to transform only the numerical features using a log transform during model training. For simplicity, I used the same pipeline for both models; that's why logtransform and standard scaler can also be seen in Figure 6 for the decision tree classifier even that isn't needed for the model.

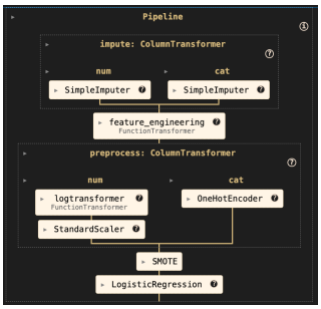


Figure 5 Data Pre-processing pipeline for Decision Tree.

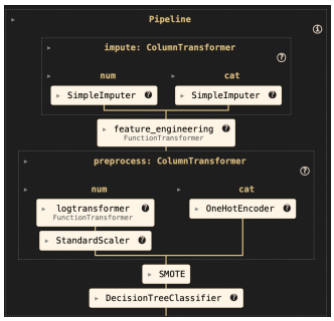


Figure 6 shows the Data Pre-processing pipeline for Logistic Regression

Categorical features were transformed using OneHot encoding [3] in the pipeline because machine learning models have difficulty interpreting strings. The result can be seen in Figure 7, where each categorical value is transformed to its own feature with a valid value between one and zero.

```
X_train_proccessed.head()
✓ 0.0s
```

| Gender_Female | Gender_Male | Married_No | Married_Yes | Education_Graduate | Education_Not Graduate | SelfEmployed_No | SelfEmployed_Yes | Pr |
|---|---|---|---|---|---|---|---|---|
| 0.0 | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | |
| 0.0 | 1.0 | 0.0 | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 | |
| 0.0 | 1.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | |
| 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 1.0 | 0.0 | |
| 0.0 | 1.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | |

Generate + Code + Markdown

Figure 7 The result after one-hot encoding of the features in the data preprocessing pipeline.

As the training data set is unbalanced, as seen in the ABT in Table 1 regarding the target variable LoanStatus with 422 Yes labels, (about 69% of the class), the up-sampling technique SMOTE [2] is implemented at the end of the pipeline as seen in Figures 5 and 6 to upsample the minority class to 422 / 422.

## 2.5. Feature Engineering

Three new features were created using current features in the dataset to help the models recognize patterns more clearly. The feature engineering is performed in the pipeline after the imputations, as seen in figures 5 and 6.

The new feature, TotalIncome, is created by adding ApplicantIncome with CoapplicantIncome:

$$TotalIncome = ApplicantIncome + CoapplicantIncome$$

To have the correct scale of the new feature, InstalmentAmount, the LoanAmount is multiplied by 1000 before calculating the new feature by dividing the LoanAmount by LoanAmounTerm. A multiplication of 1000 is performed because LoanAmount is in thousands:

$$InstalmentAmount = (LoanAmount * 1000) / LoanAmountTerm$$

The last feature, BalancedIncome, is created by subtracting the InstalmentAmount from TotalIncome:

$$BalancedIncome = TotalIncome - InstalmentAmount$$

## 2.6. Development of Predictive Models

The project requirement from the customer is that the machine learning model should be able to decide whether a new customer can be approved for a loan or not. The task fits well with a binary classification task: false for not allowed and true for allowed a loan. The dataset reveals this since since the target LoansStatus is binary (Y/N).

If the customer requirement had been to predict, for example, the customer's income as a continuous value, then it would fit a regression problem where a non-binary value could represent the prediction.

The Logistic Regression model [1, p. 409] was chosen, as well as the decision tree classifier [1, p. 544], to model the prediction of the categorical value yes or no, and both were loaded from the Scikit-learn library. The models were trained on the training dataset after all steps in the preprocessing, as explained and shown in Figures 5 and 6 of the pipeline.

The number of splits is computed by taking node_count and subtracting it from n_leaves from the trained tree, and was calculated to be 6.

# 3. Result

The final models were tested on the unseen test data, where the target class had been balanced only on the training dataset using SMOTE. Initially, the decision tree model parameter max_depth was set to 30, and min_samples_leaf was set to two. The accuracy of the decision tree achieved at that stage was only 75,7 %, which was lower than the Logistic Regression model that achieved 86,9%.

Changes were made to the hyperparameters of the decision tree, and the max_depth was set to three. With other hyperparameters kept the same, the decision tree classifier achieved an accuracy of 98,9%.

As a test, the models were trained again, but this time on the imbalanced dataset without using SMOTE in the pipeline [2]. The Logistic Regression model achieved 98,4% in accuracy, and the decision tree classifier archived an accuracy of 97,3%.

# 4. Model Evaluation

The Logistic Regression and decision tree models are evaluated using accuracy, precision, recall, and F1-score. These metrics are calculated using functions from Scikit-learn, as seen below in Tables 3 and 4.

**Table 3 Logistic Regression Classification Report.**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| | | | | |
| **0** | 0.58 | 0.94 | 0.72 | 63 |
| **1** | 0.98 | 0.86 | 0.92 | 304 |
| | | | | |
| **accuracy** | | | 0.87 | 367 |
| **macro avg** | 0.78 | 0.90 | 0.82 | 367 |
| **weighted avg** | 0.92 | 0.87 | 0.89 | 367 |

**Table 4 Decision Tree Classification Report.**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| | | | | |
| **0** | 1.00 | 0.94 | 0.97 | 63 |
| **1** | 0.99 | 1.00 | 0.99 | 304 |
| | | | | |
| **accuracy** | | | 0.99 | 367 |
| **macro avg** | 0.99 | 0.97 | 0.98 | 367 |
| **weighted avg** | 0.99 | 0.99 | 0.99 | 367 |

The confusion matrix is also used to calculate the false/true positive, false/true negative for each class [1, pp. 463–465], as seen below in Figures 8 and 9. These metrics are also calculated using functions from Scikit-learn.

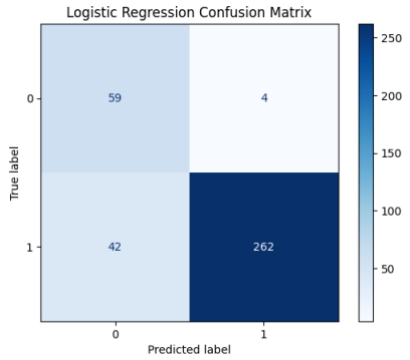


**Figure 8 Evaluation of the Logistic Regression model using the Confusion Matrix**

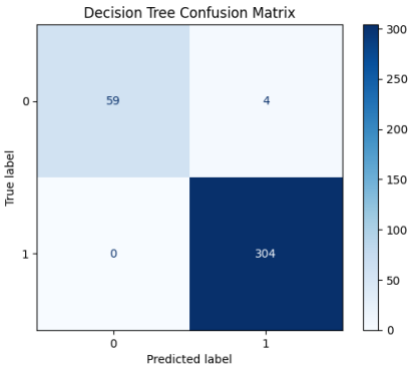


**Figure 9 Evaluation of the Decision tree classifier model usthe ing Confusion Matrix**

# 5. Discussion and future work

As seen in the results, it´s important to handle class imbalance; otherwise, the model is trained on an over-represented class in the training data, which leads to overestimated performance and poor generalisation to real-world production data.

When looking at the result of the Confusion Matrix in Figure 8, the Logistic Regression model predicted incorrectly 42 times wrong by denying the loan when the true label was to allow it. For the decision tree classifier model in Figure 9, it predicted incorrectly only four times on the test dataset, but these were cases when it shouldn't have allowed a loan.

During the training of the decision tree model, the hyperparameter were set randomly at start, and it was shown that a deeper tree could hurt the performance of the decision tree. Some small changes made a big difference in taccuracy. Further work regarding hyperparameter tuning could improve the models even more.

The models were trained on the training set and evaluated on the test set. Future work could incorporate a 5-fold stratified cross-validation to help tune the hyperparameters for fine-tuning the model performance.

Future work could investigate incorporating an ensemble learning technique [1, p. 214] [4].

As the outliers were noticed during the data exploration and can be seen in the ABT in Table 1, an attempt was made to address them using log transformation. Future work should investigate further whether this helped or whether the data points should be removed to improve performance of the Logistic regression model.

It appears that the dataset contains some gender bias, as noticed during the data exploration, and hasn't been handled, which could affect the model's fairness in production as well I belive, as males are overrepresented in the data. Future work could focus on addressing bias in the dataset to improve the model further before bringing it to production.

# 6. Conclusions

The decision tree model achieved almost the same accuracy both with and without imbalanced data; it appears to be less sensitive to an imbalanced dataset and outliers [1, p. 544]. And the Logistic Regression model appears to be more sensitive to it when the dataset is imbalanced as well , according to the result.

After investigation the feature CreditHistory, it was found to have the strongest correlation with the target class LoanStatus, where the feature represents whether the customer meets the guidelines or not.

In the test set, it appears that the majority of the samples where the customer meets the guideline are allowed a loan. Very few of the samples are cases where the customer didn't meet the guidelines, and almost all of those customers were not allowed a loan. I believe this is why the decision tree classifier model achieves the best performance among the models when it had such strong feature to split on and make its decision and less sensitive to outliers.

For the preprocessing step, I believe it is important to have some domain knowledge of the data to manage well in the preprocessing stage and feature engineering. It might be difficult to know, for example, if an outlier is a real outlier, and making mistakes during the preprocessing step might also hurt the performance of the model in real production.

# References

[1] J. D. Kelleher, B. MacNamee, and A. D'Arcy, *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. Cambridge, Massachusetts London, England: The MIT Press, 2015.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, 'SMOTE: Synthetic Minority Over-sampling Technique', *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, June 2002, doi: 10.1613/jair.953.

[3] V. Efimov, 'Decoding One-Hot Encoding: A Beginner's Guide to Categorical Data', Towards Data Science. Accessed: Oct. 09, 2025. [Online]. Available: https://towardsdatascience.com/decoding-one-hot-encoding-a-beginners-guide-to-categorical-data-058582240e86/

[4] J. Dieckmann, 'Ensemble learning: Bagging and Boosting', Towards Data Science. Accessed: Oct. 08, 2025. [Online]. Available: https://towardsdatascience.com/ensemble-learning-bagging-and-boosting-23f9336d3cb0/