

My First LaTeX Document

Tanja Matura (01307001), Fausto, Stephan

November 3, 2025

Contents

1	Introduction	2
2	Data Set 1: Credit Score	2
2.1	Overview	2
2.1.1	Feature Distribution (Numeric)	3
2.1.2	Feature Distribution (Binary)	4
2.2	Data cleaning and pre-processing	4
2.3	Logistic Regression (m1a)	6
2.4	Decision Tree (m1b)	7
2.5	Support Vector Machine (m1c)	8
2.6	Summary	8
3	Data Set 2: Vehicle	9
3.1	Overview	9
3.2	Pre-processing	9
3.3	Algorithm 1	9
3.4	Algorithm 2	9
3.5	Algorithm 3	9
3.6	Summary	9
4	Data Set 3	9
4.1	Overview	9
4.2	Pre-processing	9
4.3	Algorithm 1	9
4.4	Algorithm 2	9
4.5	Algorithm 3	9
4.6	Summary	9

5	Data Set 4	9
5.1	Overview	9
5.2	Pre-processing	9
5.3	Algorithm 1	9
5.4	Algorithm 2	9
5.5	Algorithm 3	9
5.6	Summary	9
6	Summary	9

1 Introduction

Hello, world! This is a simple L^AT_EX document compiled in VS Code.

2 Data Set 1: Credit Score

The first data set we looked at is the `analcatdata_creditscore` dataset from OpenML[1]. The dataset comes from the book ‘Analyzing Categorical Data’, by Jeffrey S. Simonoff, Springer-Verlag, New York, 2003 [2] and provides data about the credit score of 100 people. A more detailed look at the columns is given in the next section.

Our goal will be to predict whether someone’s credit application will be approved based on information about them. For this we will use logistic regression, a decision tree and a support vector machine. A comparison of results can be found in 2.6 Summary.

2.1 Overview

The dataset is small with only 7 columns and 100 entries. All features are numeric, although `Derogatory.reports` represents an ordinal variable. The target is `Application.accepted`, a binary variable that indicates whether someone’s credit application was approved or denied. This table gives an overview over the features and the target:

Feature	Types	Observation
Age	Numeric (Integer)	Range 20-55, mean 32
<code>Income.per.dependent</code>	Numeric (Integer)	
<code>Monthly.credit.card.exp</code>	Numeric (Integer)	Range 0-1898, mean 189
<code>Own.home</code>	Categorial (Boolean)	64 Yes, 36 No
<code>Self.employed</code>	Categorial (Boolean)	5 Yes, 95 No
<code>Derogatory.reports</code>	Numeric (Integer)	Range 0-7
<code>Application.accepted</code>	Categorial (Boolean)	73 Yes, 27 No

It is notable that **the target variable is skewed**, with 73 being accepted and 27 rejected. This is even more true for self-employed (5 yes to 95 no). `Monthly.credit.card.exp` also seems to contain extreme values, with the range being 0-1898 with a mean of only 189. We will take a look at this outlier later.

A look at the first few rows of the dataset:

	Age	Income.per.dependent	Monthly.credit.card.exp	Own.home	Self.employed	\
0	38.0		4.52	124.98	b'1'	b'0'
1	33.0		2.42	9.85	b'0'	b'0'
2	34.0		4.50	15.00	b'1'	b'0'
3	31.0		2.54	137.87	b'0'	b'0'
4	32.0		9.79	546.50	b'1'	b'0'

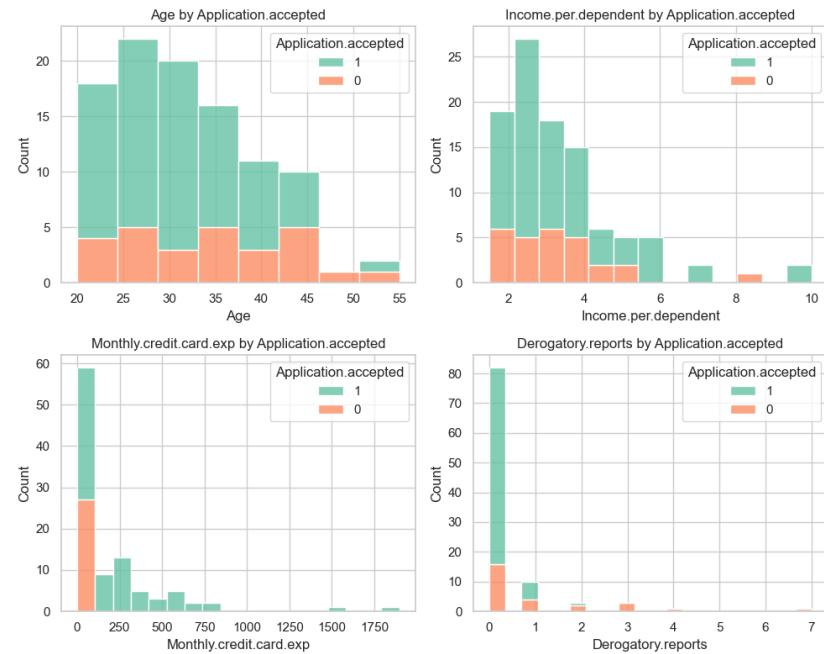
	Derogatory.reports	Application.accepted
0	b'0'	b'1'
1	b'0'	b'1'
2	b'0'	b'1'
3	b'0'	b'1'
4	b'0'	b'1'

`Monthly.credit.card.exp` already shows high variance in this snippet and is a candidate for transformation later.

The categorical columns - that are binary and therefore encoded as "0" and "1" - and `Derogatory.reports` are still byte enconded and had to be decoded first before they could be worked with.

2.1.1 Feature Distribution (Numeric)

Target based distribution of numeric features:



The bar plots show some extreme values for `Monthly.credit.card.exp` and

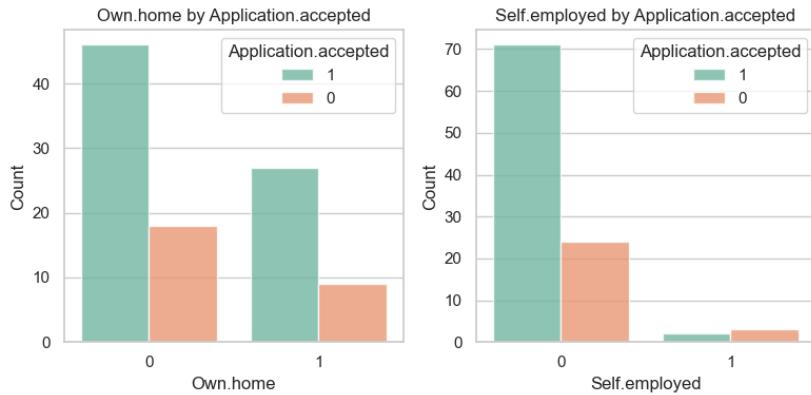
Derogatory.reports. However, neither seem to be unrealistic and therefore not errors in the data set but real extreme values.

Something else that stands out is that both of these features seem to be highly correlated with the target variable - the application of people with high monthly expenses were accepted, while the applications of people with more than two derogatory reports were rejected. To reduce the influence on the prediction we will transform these features before fitting a model (see Pre-Processing).

Age and **Income.per.dependent** also show right skewing and might be transformed later.

2.1.2 Feature Distribution (Binary)

Target based distribution of binary features:



The plots for the binary features show that the **Self.employed** feature is very unevenly distributed - there is almost no data for people who are self-employed. Their application rate also goes against the trend, with more applications rejected than accepted.

2.2 Data cleaning and pre-processing

After gaining a better understanding of the data we are working with, we prepared the data to be suitable for model fitting.

Byte-Decoding: As seen in the first five rows of the dataset, some columns were still byte-encoded after importing the .arff file and had to be decoded first.

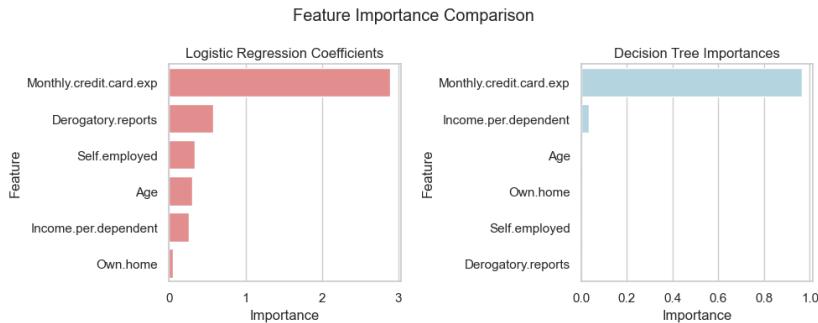
Missing Values: The Credit Score Dataset is complete and therefore didn't need any handling of missing values.

Normalization of Binary Features: To make sure no entry is missed during conversion (in case some binary features aren't encoded as 0 or 1) we searched the columns for subsets of "Yes", "No", "True" and "False" and converted them to 0 and 1, should any be found.

Conversion of Numeric Values: All numeric values in the dataset were converted from String to ensure later calculations work. For this the pandas function `pd.to_numeric` was used, which detects and converts numeric Strings automatically.

Outlier Handling: Some rows contain extreme values. At first we considered to keep them, however after beginning to work with the dataset an issue arose: Even after log transforming `Monthly.credit.card.exp` correlated so strongly with the target that it essentially encoded the same information - leading to unrealistic 0% prediction error in the first attempts using Logistic Regression.

Visual diagnostics confirmed that this feature is dominating the others.

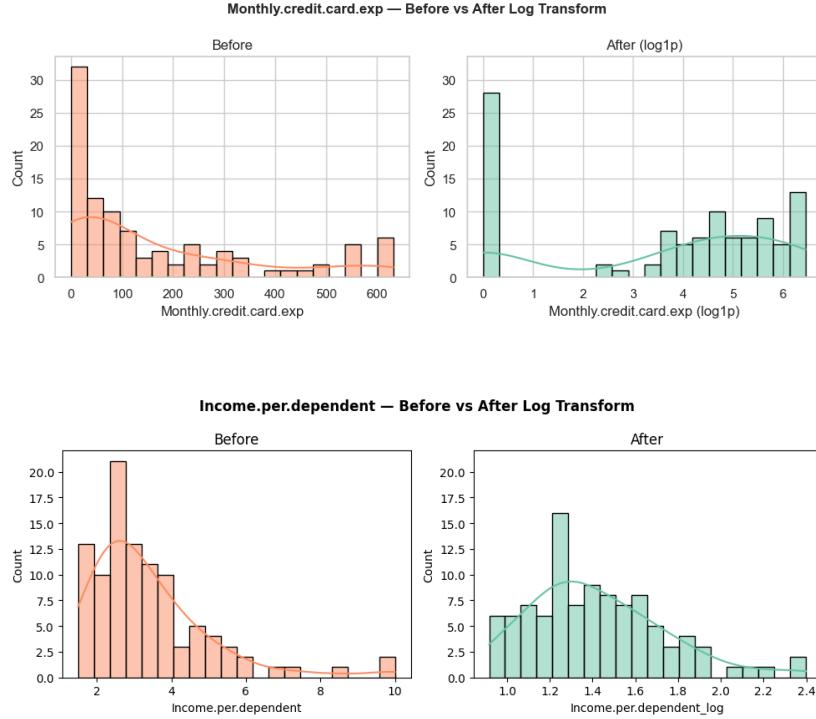


To reduce the influence of `Monthly.credit.card.exp` we decided to cap the column. For this we used the IQR: The upper threshold turned out to be 632.01 and 6 rows needed to be capped at this value. After capping the extreme values in `Monthly.credit.card.exp`, the performance of the Logistic Regression model decreased from 100% accuracy to a more credible level of 85%.

Transformation of Numeric Values: Instead of relying on visuals only we used the function `skew()` to output the skewness of each numeric feature. Features with $|skew| > 1$ were considered strongly skewed. This applied to `Monthly.credit.card.exp` and to `Income.per.dependent`.

After log transformation the features changed as followed:

Feature	Skewness before	Skewness after
Income.per.dependent	1.882	0.800
Monthly.credit.card.exp	1.220	-0.586



Scaling of Numeric Values: Numeric features were scaled using RobustScaler to mitigate the influence of remaining outliers.

Class Balance of Binary Features: Our binary columns (including the target) show class imbalance. Features were kept as they are, the imbalance of the target will be addressed by using `class_weight='balanced'` during model fitting.

Training- and Test-Split: 80:20

2.3 Logistic Regression (m1a)

As baseline model, we first trained a Logistic Regression classifier to predict `Application.accepted`. Logistic Regression is well-suited for this dataset because the target variable, `Application.accepted`, is binary (approved vs. rejected), which fits the model's purpose of predicting probabilities for categorical outcomes.

Since the target variable was moderately imbalanced (approximately 1:3 ratio), two balancing strategies were compared: First, assigning class weights inversely proportional to class frequency (`class_weight='balanced'`), and second, random oversampling of the minority class to equalize class counts.

As discussed earlier the model achieved 100% accuracy on the training set before `Monthly.credit.card.exp` was capped. After capping and re-fitting the model, performance metrics became more realistic.

Both balancing methods produced identical results, with the model achieving an overall accuracy of 85%. The recall for rejected applications (Class 0) remained perfect, indicating no false negatives, while the recall for accepted applications (Class 1) slightly decreased to 0.8, suggesting more realistic model behavior.

These results confirm that outlier capping effectively reduced the excessive influence of `Monthly.credit.card.exp` without removing its predictive contribution. The model now generalizes better and provides a more balanced interpretation of multiple predictors rather than relying on a single dominating feature.

Cross Validation

Since we established that the target variable is imbalanced we chose Stratified K-Fold cross-validation. Working with a small dataset also meant that we should keep k small, so the validation sets won't become too small.

Mean Accuracy: 0.970 ± 0.024

Mean F1 Score: 0.964 ± 0.030

Metric	Results
CV	0.970
SD	0.024
Macro F1	0.83
Recall (Class 0)	1.00
Recall (Class 1)	0.80
Recall (Class 1)	0.80

2.4 Decision Tree (m1b)

The second model we trained was a decision tree. In contrast to Logistic Regression, which assumes a linear relationship between predictors and the log-odds of the target, a Decision Tree partitions the feature space recursively into regions that are more homogeneous with respect to the target variable. However, since the dataset is relatively small (only 100 samples), trees can easily overfit if left unpruned. Therefore, we limited the depth and minimum number of samples per leaf to help ensure better generalization.

Even with this the tree achieved a perfect prediction score on it's first run - but later cross validation showed that the model does not always predict with perfect accuracy, suggesting that the training and test data split happen to make the dataset perfectly separable. This is again due to the variable `Monthly.credit.card.exp`. All applications that were rejected did not have any expenses, making it a very strong predictor for a tree (if expenses = 0 = reject). However, some people without expenses still received a positive application result, which means the feature cannot be deterministic factor and the

model should not achieve 100% accuracy.

Cross Validation

Mean Accuracy: 0.98 ± 0.024 Mean F1 Score: 0.976 ± 0.029

2.5 Support Vector Machine (m1c)

The third model that was trained for the Credit Score Dataset was a support vecotr machine. A Support Vector Machine (SVM) is a supervised learning algorithm that tries to find the optimal separating boundary (hyperplane) between classes. Instead of just any line that separates accepted and rejected applications, it chooses the one that maximizes the margin — the distance between the boundary and the nearest data points from each class (called support vectors). This makes it robust to noise and reduces overfitting, especially in small datasets. Its margin-based approach makes it more generalizable than a Decision Tree and more flexible than Logistic Regression when the separation between classes is not perfectly linear.

The SVM performed well overall but didn't reach the near-perfect performance of the Decision Tree.

With class weights, accuracy was 80

With oversampling, performance improved to 90

Cross-validation confirmed consistent generalization with Mean Accuracy: 0.88 ± 0.05 Mean F1: 0.86 ± 0.06

2.6 Summary

While the Decision Tree achieved perfect training accuracy, this performance likely reflects overfitting to the small dataset. In contrast, Logistic Regression produced slightly lower but more realistic results, maintaining good accuracy while capturing the main relationship between Monthly.credit.card.exp and the target without memorizing individual samples. The SVM shows strong but not perfect generalization — less prone to overfitting than the Decision Tree and slightly more flexible than Logistic Regression.

3 Data Set 2: Vehicle

- 3.1 Overview**
- 3.2 Pre-processing**
- 3.3 Algorithm 1**
- 3.4 Algorithm 2**
- 3.5 Algorithm 3**
- 3.6 Summary**

4 Data Set 3

- 4.1 Overview**
- 4.2 Pre-processing**
- 4.3 Algorithm 1**
- 4.4 Algorithm 2**
- 4.5 Algorithm 3**
- 4.6 Summary**

5 Data Set 4

- 5.1 Overview**
- 5.2 Pre-processing**
- 5.3 Algorithm 1**
- 5.4 Algorithm 2**
- 5.5 Algorithm 3**
- 5.6 Summary**

6 Summary

References

- [1] OpenML. *analcatdata_creditscore*. 2025. URL: <https://www.openml.org/d/461>.

- [2] Jeffrey S. Simonoff. *Analyzing Categorical Data*. New York: Springer-Verlag, 2003.