Table of Content

Introduction

Purpose and Goals

Goal: building a classifier to predict the binary target for a dataset with numerical and categorical features

Scope:

- Data preprocessing: handling missing values, datetime and categorical features
- Model building: using logistic regression, decision trees, random forests, or ANN and fine-tuning the selected model(s) to predict the target feature
- Performance evaluation: assessing the effectiveness of the model(s) though standard metrics and visualisation techniques

Dataset Overview and Preprocessing

Preprocessing involved an initial exploration of the dataset to understand its structure and data types (ref. section 1.2 of the ipynb file). The dataset contains 925 rows and 8 columns. It includes a mix of numerical, categorical, and datetime features, with the target variable being binary (0 or 1).

For var4 we have a significant portion of missing data (600 out of 925 values). For the imputation of the missing values I applied a commonly recommended technique of using linear regression to predict and fill in the missing data, as discussed in the review by Lin and Tsai (2019).

Categorical data was transformed in the following way:

- since we have a very big number of values for var3 (236!), I decided to convert them to numerical values using Weight of Evidence (WOE). WOE is a measure to encode categorical variables into numerical, and the calculation is based on the proportion of positive and negative outcomes for each category, the formula is as follows (Pershukova, 2023):

$$WOE = \ln\left(\frac{\frac{Positives\ in\ category}{Total\ Positives}}{\frac{Negatives\ in\ category}{Total\ Negatives}}\right)$$

Equation 1. Weight of Evidence calculation (from Pershukova, 2023)

1. the binary categorical feature var6 was converted using LabelEncoder.

For the datetime feature var7, it was transformed to a proper datetime format, then invalid data were corrected manually and then converted to Unix timestamps.

Lastly, Decision Tree and Random forest do not necessarily require feature scaling, but because I was aiming at initially comparing the model training results among different algorithms, all features have been normalized by using MinMax scaler.

The data was then split into test (20%) and train (80%) set with the use of stratification to have the same proportion of classes 0 and 1 in both sets.

Below is the screenshot of the head of the features that are ready for training:

	var1	var2	var3	var5	var4	var6	var7_float
0	0.282675	0.798813	0.512494	0.405575	0.541285	1.0	0.474746
1	0.879001	0.628218	0.469898	0.635220	0.300652	1.0	0.692138
2	0.274551	0.731204	0.547711	0.612103	0.565518	1.0	0.489684
3	0.663312	0.972995	0.560204	0.574537	0.274199	1.0	0.761066
4	0.093174	0.434971	0.530102	0.323474	0.780488	0.0	0.154856

Figure 1. Normalized features for Dataset 1 training set

Task 1: Numerical and Categorical Classification

a. **Methodology and Techniques**:

For this task, I initially fit three algorithms: Logistic Regression, Decision Tree and Random Forest to later choose the one that provided the best output and fine-tune it. ANN is powerful, but I excluded it because it requires larger datasets for optimal performance , which is not our case.

Given the thorough preprocessing of data applied, the process of model fitting was a straightforward and simple process. Below are the performance results of the three algorithms fit with default parameters:

	Logistic Regression	**Decision Tree**	**Random Forest**
Accuracy	0.9622	0.9459	0.9784
F1 Score	0.9550	0.9456	0.9778
Precision	0.9609	0.9456	1.0

Recall	0.9347	0.9465	0.9565

Table 1. Performance scores of Logistic Regression, Decision Tree and Random Forest algorithms on Dataset 1

All the models perform well. But the Random Forest has the highest accuracy, the absolute precision and shows strong overall performance. Given that Random Forest is an ensemble of Decision Trees, it made sense to explore both models further to understand the benefits of each. This approach also allowed me to maximize the learning experience.

To optimise the Decision Tree, I used **Grid Search** to identify the best combination of hyperparameters. The best parameters identified were:

```
Best parameters found by extended GridSearchCV for Decision Tree:
{'max_depth': None, 'max_leaf_nodes': 22, 'min_samples_leaf': 1, 'min_samples_split': 2}
```

Figure 2. Decision Tree optimal parameters identified with Grid Search

The model was then retrained using these parameters, which resulted in improved score (with accuracy of 0.97).

For the Random Forest, initial attempts with Grid Search did not produce improvements. So, I used **thresholding** technique to improve performance. This involved analysing the **ROC curve**, which plots the trade-off between true positives and false positives at various threshold levels. Then by using **Youden's J statistics**, I identified the optimal threshold (0.38). And finally, applying thus threshold improved significantly the model's predictions (accuracy of approx. 0.99).

Subsequently, I evaluated the performance of both models using key metrics, such as accuracy, F1 Score, precision and recall. By comparing these metrics before and after fine-tuning, I assessed the improvement of the predictive capabilities of the models.

b. **Results and Discussion**:

The table below shows the performance metrics for the fine-tuned Decision Tree and Random Forest models:

Metric	Decision Tree	Random Forest
Accuracy	0.9676	0.9892
F1 Score	0.9674	0.9891
Precision	0.9674	0.9891
Recall	0.9674	0.9891

Table 2. Performance results of fine-tuned Decision Tree and Random Forest

The fine-tuned Random Forest model outperforms the Decision Tree across all metrics, with an accuracy almost 99%. It shows higher precision, recall, and F1 score, thus proving better ability to correctly classify labels. The confusion matrix further highlights this: the Random Forest model produces only 1 false positive and 1 false negative, compared to the Decision Tree's 3 false positives and 3 false negatives. This suggests that the Random Forest model is more reliable for this classification task.

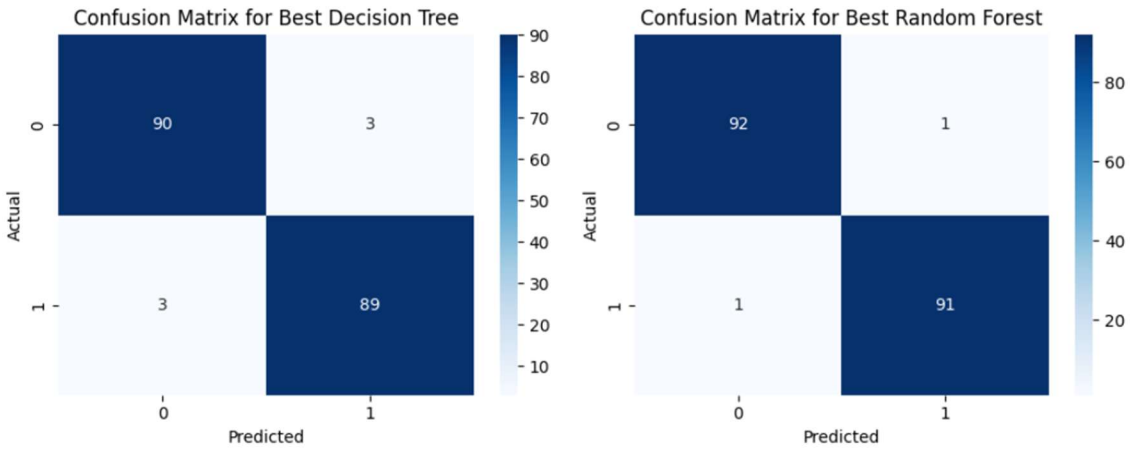

Figure 3. Fine-tuned Decision Tree and Random Forest Confusion Matrices

Another important aspect is the feature importance derived from these models. The plots below indicate that the Decision Tree relies predominantly on var5, and other features have minimal contribution. In contrast, the Random Forest model uses multiple important features, among which var5 and var1 are the most influential. This demonstrates that the Random Forest is able to aggregate information from various features, rather than depending heavily on a single variable.

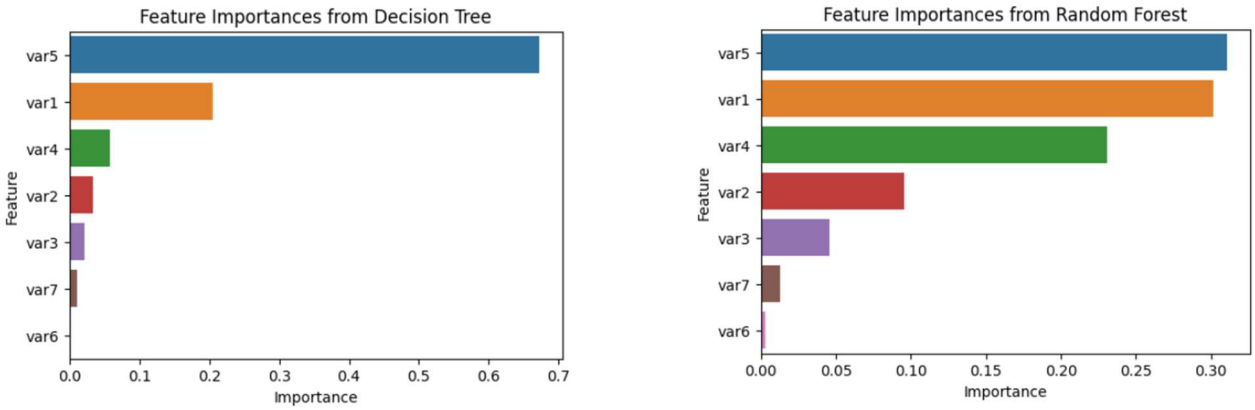

Figure 4 Feature importances from Decision Tree and Random Forest

Another aspect I would like to take into consideration is that the Decision Tree, being a classic example of the white-box algorithms is transparent, and in its architecture below we can see how decisions are made by the algorithm to arrive at solution. Each node represents a decision point, and the branches represent the outcomes of those decisions:

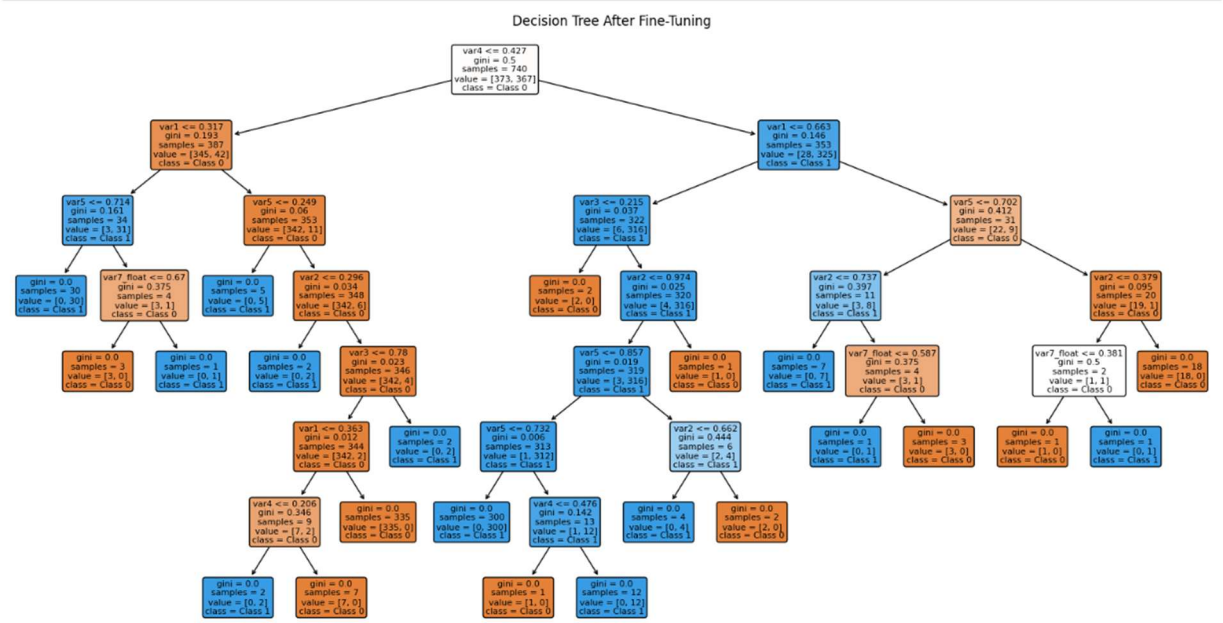

Figure 5. Fine- tuned Decision Tree Diagram

So, overall, the Random Forest model provides a more robust and balanced approach, making it a better choice for this classification problem. On the other hand, the Decision Tree is transparent, which makes it suitable for applications where it is important to understand the decision-making process.

Conclusion

Binary Classification Task:

- The Random Forest model outperformed other algorithms, achieving **98.92%** accuracy. It is a robust and reliable model, using multiple features to improve its performance.

- The decision tree while less accurate and is dependent primarily on a single feature, provides greater transparency, which is beneficial in applications where understanding the decision-making process is important.

- Careful data preprocessing is important for achieving better performance. Applying Weight of Evidence for high cardinality categorical features and Linear Regression imputation for large portion of missing values proved to be efficient.

- Fine-tuning is essential for achieving better performance results. For the Random Forest thresholding provided greater accuracy. The Decision Tree benefited from hyperparameter tuning with Grid Search.

Recommendations for future work:

For Binary Classification:

- Future work could explore alternative imputation methods for variables with a significant amount of missing data

- Investigating the performance of ensemble models that combine very different architectures could also be an option for a future work.

References

Lin, W.-C., & Tsai, C.-F. (2019). Missing value imputation: A review and analysis of the literature (2006–2017). Journal of the Knowledge Economy. Published online April 5, 2019. Springer Nature B.V. https://doi.org/10.1007/s13198-019-00863-5 (Accessed: 8 August, 2024).

Pershukova, A. (2023). Too many categories: how to deal with categorical features of high cardinality. Medium. Available at: https://annahava.medium.com/too-many-categories-how-to-deal-with-categorical-features-of-high-cardinality-d4563cfe62d6 (Accessed: 19 July 2024).