# Master in Data Science

Module: Data Science Languages

Assignment Report: Customer Churn Prediction (Enhanced Model Evaluation)

## 📊 Objective

This report summarizes the performance of three classification models—Decision Tree, Random Forest, and Logistic Regression—applied to a highly imbalanced banking churn dataset. The goal was to evaluate how well these models could identify customers likely to churn.

## 📂 Dataset Overview

- Full dataset: ~377,000 records  
- Target variable: `flag\_request\_closure` (renamed to `Target`)  
- Classes: 0 = No Churn (~99.5%), 1 = Churn (~0.5%)  
- Features: Numerical and categorical fields including spending behavior, age, and account activity

## ⚖️ Methodology Summary

1. Preprocessing: Dropped high-missing columns and ID-like fields; encoded categorical variables.  
2. Class Balancing: Random undersampling of the majority class to achieve a 5:1 ratio for training.  
3. Models Trained:  
 - Decision Tree (min\_samples\_leaf=50)  
 - Random Forest (class\_weight='balanced')  
 - Logistic Regression (class\_weight='balanced')  
4. Metrics: Recall, Precision, F1-Score, and AUC were used for evaluation.

## 📊 Model Performance Summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Recall (Churn) | Precision (Churn) | F1-Score | AUC Score |
| Decision Tree | 0.32 | 0.04 | 0.08 | 0.836 |
| Random Forest | 0.40 | 0.07 | 0.12 | 0.881 |
| Logistic Regression | 0.73 | 0.01 | 0.03 | 0.788 |

## 🚀 Conclusion

The results show that:  
- Random Forest provided the best balance between recall and overall discrimination (AUC = 0.88).  
- Logistic Regression captured the highest number of churners (recall = 0.73), but with extremely low precision (0.01), making it better suited for cases where high recall is critical.  
- Decision Tree served as a strong baseline but underperformed on minority class detection compared to the others.  
  
These results highlight the importance of model selection and performance trade-offs in imbalanced classification tasks. Future work could include applying SMOTE, cost-sensitive learning, or using ensemble stacking.