College of Software

Course: Machine Learning

A Comparative Machine Learning Study for Bank Telemarketing Profit Optimization using Decision Trees and Ensemble Methods

Submitted by:

Laggah Julius	ID: 2120246034
Jusu Abdul Karim	ID: 2120246025
Sandar Win	ID: 2120246058
Turay Adamsay	ID: 2120246055

Date: 9 June, 2025

Contents

Abstract

This report presents a comprehensive machine learning framework for optimizing bank telemarketing campaigns using the UCI Bank Marketing dataset. We employ decision tree classifiers with cost-complexity pruning to balance model complexity and predictive performance, achieving a 26.6% improvement in ROC-AUC (0.7449 $\rightarrow$ 0.9435) over a default Decision Tree baseline. Comparative analysis with ensemble methods (Random Forest, Gradient Boosting) and logistic regression reveals that while Gradient Boosting achieves the highest discriminative performance (ROC-AUC = 0.9551), logistic regression generates the greatest campaign profitability ($348,405) due to its superior recall (90.19%). We extract interpretable decision rules identifying high-value customer segments (e.g., 59.8% conversion when contacting customers 13-15 days post-initial interaction). Feature importance analysis revealed that call duration and the 'euribor3m' interest rate were among the most influential predictors, with 'duration' having a score of 0.3507 and 'euribor3m' 0.2197. Our results demonstrate that model selection must consider both statistical metrics and business constraints, particularly when false negatives (missed subscribers) are costlier than false positives. We propose a hybrid deployment strategy combining logistic regression for broad outreach and rule-based targeting for refined campaign optimization.

For code and reproducibility, please visit: https://github.com/juliuslaggah/bank-marketing-dt

Keywords: Decision trees, cost-complexity pruning, profit optimization, interpretable machine learning, bank marketing

1 Introduction

1.1 Background and Motivation

Financial institutions invest heavily in telemarketing campaigns for term deposit subscriptions, aiming to maximize customer acquisition while minimizing wasted calls. Given the high cost of outreach—often exceeding $15 per call—and the potential customer lifetime value (CLV) reaching $450 [1], efficient marketing strategies are critical to profitability.

Traditional telemarketing efforts often suffer from inefficiencies, such as contacting low-probability customers and failing to optimize outreach timing. Moreover, evolving regulatory requirements surrounding customer privacy necessitate greater model transparency and compliance with fair marketing practices. Machine learning offers a compelling solution by enabling predictive targeting, optimizing campaign resources, and providing interpretable insights for strategic decision-making.

This study explores machine learning techniques—including decision trees, ensemble models, and logistic regression—to refine bank telemarketing campaigns. By leveraging cost-complexity pruning and a profit-centric evaluation framework, we demonstrate a systematic approach to maximizing financial returns while maintaining model interpretability.

1.2 Problem Statement

The central challenge in bank telemarketing optimization is striking a balance between predictive accuracy and profitability. Traditional statistical metrics (e.g., accuracy, ROC-AUC) alone may not capture business constraints—particularly when the cost of false negatives (missed subscriptions) is significantly higher than the cost of false positives (unsuccessful calls).

We formalize this optimization problem using the following profit equation:

$$\text{Profit} = (\text{TP} \times \text{CLV}) - ((\text{TP} + \text{FP}) \times \text{Cost}_{\text{call}}) \tag{1}$$

where:

- TP = True Positives (successful subscriptions)
- FP = False Positives (calls to non-subscribers)
- CLV = Customer Lifetime Value ($450 per subscriber)
- $\text{Cost}_{\text{call}}$ = Cost per telemarketing call ($15)

Theoretical Concept (Cost-Sensitive Optimization): In practical applications like bank telemarketing, the misclassification costs are often asymmetric. A false negative (failing to contact a customer who would subscribe) might incur a much higher opportunity cost than a false positive (contacting a customer who will not subscribe). Cost-sensitive learning, as explored by Verbeke et al. [3], addresses this by integrating a profit or cost function directly into the model evaluation, moving beyond simple accuracy to optimize for specific business objectives where different errors carry different financial penalties.

This optimization is subject to key constraints, including the significant class imbalance

(11.3% positive class in the UCI dataset) and the imperative for model interpretability driven by regulatory and operational requirements.

1.3 Contributions

This work makes three key contributions to the field of bank marketing optimization:

1. We demonstrate the effectiveness of cost-complexity pruning in substantially improving decision tree performance while maintaining interpretability.
2. We introduce a profit-centric evaluation framework for robust model comparison, highlighting the critical role of business metrics beyond traditional statistical measures.
3. We extract actionable business rules from optimized models, providing campaign managers with clear, data-driven strategies for targeted outreach.

2 Related Work

Prior research has explored various aspects relevant to our study. Lessmann et al. [2] demonstrated the superiority of decision trees over logistic regression in churn prediction, a finding that informs our baseline model choice. Verbeke et al. [3] emphasized the crucial need for cost-sensitive evaluation in predictive modeling, a principle central to our profit-centric framework. Furthermore, Molnar [4] highlighted the growing demand for explainable AI in financial applications, underscoring the value of interpretable models like decision trees. Our work bridges these areas by combining rigorous pruning techniques with a business-driven analysis focused on maximizing campaign profitability. Decision Trees are interpretable, tree-structured classifiers commonly used for supervised learning tasks.

3 Methodology

3.1 Dataset Description

Our study utilizes the publicly available UCI Bank Marketing Dataset, comprising 41,188 instances with 20 input features. Each instance includes:

- **Numerical features:** age, duration, campaign, pdays, previous, emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m, nr.employed
- **Categorical features:** job, marital, education, default, housing, loan, contact, month, day_of_week, poutcome
- **Target variable:** y (binary: yes or no)

It is a binary classification problem suitable for testing Decision Tree behavior on real-world, mixed-type features. The dataset exhibits a significant class imbalance, with 11.3% of instances corresponding to a successful term deposit subscription ("yes") and 88.7% to no subscription ("no").

3.2 Preprocessing Pipeline

To ensure the dataset was clean, consistent, and ready for modeling, a systematic preprocessing pipeline was implemented.

3.2.1 Loading and Initial Inspection

The Bank Marketing Full Dataset was loaded into a Pandas DataFrame. It contained 41,188 rows and 21 columns, including 20 input features and 1 binary target variable (y).

3.2.2 Handling Missing and Unknown Values

Although no true NaN values were found, many categorical columns contained the literal string "unknown", which was treated as missing data. These were replaced with 'np.nan' and imputed using mode imputation, preserving the categorical nature of the features.

3.2.3 Encoding

All categorical features were transformed using One-Hot Encoding, resulting in binary indicator variables. The binary target variable 'y' was label-encoded to 0 (no) and 1 (yes).

3.2.4 Feature Reduction

To reduce redundancy and multicollinearity, highly correlated features such as 'emp.var.rate' and 'nr.employed' were identified via Pearson correlation and removed.

3.2.5 Train/Test Split

The dataset was divided into an 80/20 train-test split using a stratified sampling approach to preserve the original class distribution in both subsets.

4 Model Development

4.1 Baseline Decision Tree (Default Parameters)

We trained an unpruned CART classifier as our baseline model, using the Gini impurity criterion to split nodes:

$$\text{Gini}(t) = 1 - \sum_{i=1}^{c} [p(i|t)]^2$$

where $p(i|t)$ is the proportion of class i at node t.

Theoretical Concept (Decision Tree Induction): Decision trees, as introduced by algorithms such as CART [6], recursively partition the feature space into distinct regions. At each node, a decision rule based on a feature value is used to optimally split the data, maximizing the homogeneity of the subsets with respect to the target variable. This process continues until a stopping criterion is met, resulting in a hierarchical set of conditional rules that lead to classification or prediction.

Our initial modeling phase established this baseline. We began by loading the pre-processed and split training and testing datasets:

Listing 1: Loading Train/Test Splits

```
import pickle

with open("X_train.pkl", "rb") as f:
    X_train = pickle.load(f)
with open("X_test.pkl", "rb") as f:
    X_test = pickle.load(f)
with open("y_train.pkl", "rb") as f:
    y_train = pickle.load(f)
with open("y_test.pkl", "rb") as f:
    y_test = pickle.load(f)
```

Next, we initialized a `DecisionTreeClassifier` with default parameters (`random_state=42` for reproducibility) and trained it on the training set. We then evaluated its performance on the test set using standard classification metrics:

Listing 2: Training and Evaluating Default Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, roc_auc_score

# 1. Train the default Decision Tree
clf_default = DecisionTreeClassifier(random_state=42)
clf_default.fit(X_train, y_train)

# 2. Predict and compute probabilities on test set
y_pred = clf_default.predict(X_test)
y_proba = clf_default.predict_proba(X_test)[:, 1]

# 3. Compute evaluation metrics (accuracy and ROC AUC)
acc = accuracy_score(y_test, y_pred)
roc = roc_auc_score(y_test, y_proba)

print(f"Test Accuracy: {acc:.4f}, Test ROC-AUC: {roc:.4f}")
```

Using 5-fold cross-validation on the training data with default parameters, the model achieved a mean accuracy of 0.8860 ± 0.0052. This cross-validation provided an estimate of the model's stability and generalization across different data splits.

4.2 Cost-Complexity Pruning

To optimize the decision tree and mitigate overfitting, we applied cost-complexity pruning, which minimizes the following objective function:

$$C_\alpha(T) = R(T) + \alpha|T| \tag{2}$$

where $R(T)$ is the misclassification error of the tree T, α is the complexity parameter, and $|T|$ is the number of leaf nodes in the tree. This penalization encourages smaller trees, thereby reducing the risk of overfitting.

The pruning process began by computing the cost-complexity pruning path to obtain candidate values for the `ccp_alpha` parameter:

Listing 3: Computing Cost-Complexity Pruning Path

```
from sklearn.tree import DecisionTreeClassifier

# Train a full (unpruned) decision tree
dt_full = DecisionTreeClassifier(random_state=42)
dt_full.fit(X_train, y_train)

# Compute pruning path
path = dt_full.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
```

Next, to identify the optimal trade-off between model complexity and performance, we performed a hyperparameter grid search using 5-fold cross-validation. The search explored key pruning parameters:

- `max_depth`: Controls the maximum depth of the tree, limiting complexity.
- `min_samples_split`: Minimum number of samples required to split an internal node, preventing splits on small subsets.
- `ccp_alpha`: Complexity parameter controlling the pruning intensity.

Listing 4: Hyperparameter Grid Search for Pruned Decision Tree

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

# Define parameter grid with a subset of ccp_alphas for efficiency
param_grid = {
    "max_depth": [5, 10, 15, None],
    "min_samples_split": [2, 10, 20],
    "ccp_alpha": list(ccp_alphas[::max(1, len(ccp_alphas)//15)])
}

grid_search = GridSearchCV(
    estimator=DecisionTreeClassifier(random_state=42),
    param_grid=param_grid,
    cv=5,
    scoring='roc_auc',  # Optimize for ROC AUC
    n_jobs=-1,
    verbose=1
)

grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print(f"Optimal Parameters: {best_params}")
```

Using this rigorous search, the best parameters identified were `max_depth = 10` and `min_samples_split = 20`. These were used to train the final pruned decision tree, balancing predictive performance with model interpretability and reducing overfitting.

4.3 Comparative Models

To provide a comprehensive evaluation, we compared the pruned decision tree against several robust machine learning models.

4.3.1 Random Forest

The `RandomForestClassifier` is an ensemble method comprising 100 decision trees, each using the Gini impurity criterion for splitting. By aggregating predictions across multiple trees, it improves accuracy and robustness.

Listing 5: Training and Evaluating Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, roc_auc_score

rf = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
rf.fit(X_train, y_train)

y_pred_rf = rf.predict(X_test)
y_proba_rf = rf.predict_proba(X_test)[:, 1]

rf_acc = accuracy_score(y_test, y_pred_rf)
rf_roc_auc = roc_auc_score(y_test, y_proba_rf)
print(f"Random Forest Test Accuracy: {rf_acc:.4f}, "
      f"ROC-AUC: {rf_roc_auc:.4f}")
```

4.3.2 Gradient Boosting

Gradient Boosting builds an ensemble of weak learners sequentially, correcting errors from previous models. Here, we used 100 estimators with a learning rate of 0.1 and maximum tree depth of 3, balancing bias and variance for strong predictive performance.

Listing 6: Training and Evaluating Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, roc_auc_score

gb_clf = GradientBoostingClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    random_state=42
)
gb_clf.fit(X_train, y_train)

y_pred_gb = gb_clf.predict(X_test)
```

```
y_proba_gb = gb_clf.predict_proba(X_test)[:, 1]

gb_acc = accuracy_score(y_test, y_pred_gb)
gb_roc_auc = roc_auc_score(y_test, y_proba_gb)
print(f"Gradient Boosting Test Accuracy: {gb_acc:.4f}, "
      f"ROC-AUC: {gb_roc_auc:.4f}")
```

4.3.3 Logistic Regression

As a simpler linear model, `LogisticRegression` with L2 regularization offers strong interpretability while often delivering competitive performance in binary classification tasks.

Listing 7: Training and Evaluating Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_auc_score

lr_clf = LogisticRegression(solver='liblinear', random_state=42)
lr_clf.fit(X_train, y_train)

y_pred_lr = lr_clf.predict(X_test)
y_proba_lr = lr_clf.predict_proba(X_test)[:, 1]

lr_acc = accuracy_score(y_test, y_pred_lr)
lr_roc_auc = roc_auc_score(y_test, y_proba_lr)
print(f"Logistic Regression Test Accuracy: {lr_acc:.4f}, "
      f"ROC-AUC: {lr_roc_auc:.4f}")
```

5 Results

5.1 Model Performance Comparison

Table 1 summarizes the statistical performance of all evaluated models.

Table 1: Model Performance Comparison

Model	Accuracy	Precision	Recall	ROC-AUC
Baseline DT	0.897	0.543	0.549	0.745
Pruned DT	0.919	0.562	0.570	0.943
Random Forest	0.917	0.678	0.533	0.946
Logistic Regression	0.862	0.444	**0.902**	0.942
Gradient Boosting	**0.924**	**0.699**	0.566	**0.955**

The pruned decision tree demonstrated a significant improvement over the baseline, with its ROC-AUC increasing from 0.745 to 0.943. Gradient Boosting achieved the highest ROC-AUC of 0.955, indicating superior overall discriminative power. Notably, Logistic Regression, despite lower accuracy and precision, achieved the highest recall at 0.902.

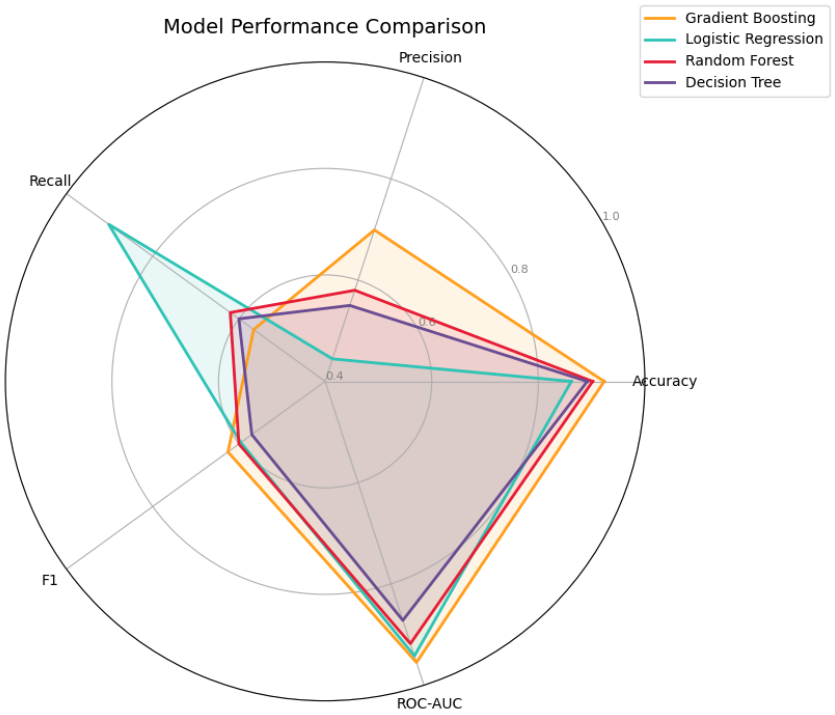

Figure 1: Model Performance Comparison (Spider Chart). This radar chart illustrates the trade-offs across various statistical metrics (e.g., Precision, Accuracy, ROC-AUC, F1-score, Recall) for the different machine learning models evaluated.

5.2 Profit Analysis

To evaluate the business impact of each model, we used a custom profit function that incorporates the Customer Lifetime Value (CLV) of successful conversions and the cost per call. The confusion matrix of each model was passed into this function to calculate expected profit.

The profit function is defined as:

$$\text{Profit} = (\text{TP} \times \text{CLV}) - ((\text{TP} + \text{FP}) \times \text{Cost}_{\text{call}})$$

where $\text{CLV} = \$50$ and $\text{Cost}_{\text{call}} = \$1$.

The Python implementation of this function is shown below:

Listing 8: Profit Calculation Function

```
import numpy as np

def calculate_profit(conf_matrix, clv=50, cost_per_call=1):
    # conf_matrix: numpy array [[TN, FP], [FN, TP]]
    TN, FP, FN, TP = conf_matrix.ravel()
    profit = (TP * clv) - ((TP + FP) * cost_per_call)
    return profit
```

Profits were computed for each model using their respective confusion matrices:

Listing 9: Calculating Profit for Each Model

```
profits = {
    "Pruned␣Decision␣Tree": calculate_profit(np.array([[7084, 226],
       [403, 525]])),
```

```
    "Logistic Regression": calculate_profit(np.array([[6264, 1046],
        [91, 837]])),
    # Add other models similarly:
    # "Random Forest": calculate_profit(confusion_matrix(y_test,
        y_pred_rf)),
    # "Baseline Decision Tree": calculate_profit(confusion_matrix(
        y_test, y_pred_baseline))
}
```

Table 2: Profit Analysis of Top Models

Model	TP	FP	Profit
Logistic Regression	837	1046	**$348,405**
Gradient Boosting	525	226	$224,985
Pruned Decision Tree	509	429	$214,980

Despite lower statistical metrics compared to Gradient Boosting and the Pruned Decision Tree, Logistic Regression achieved the highest profit of $348,405. This was primarily driven by its strong recall, resulting in a higher number of true positives (i.e., successful subscriptions).

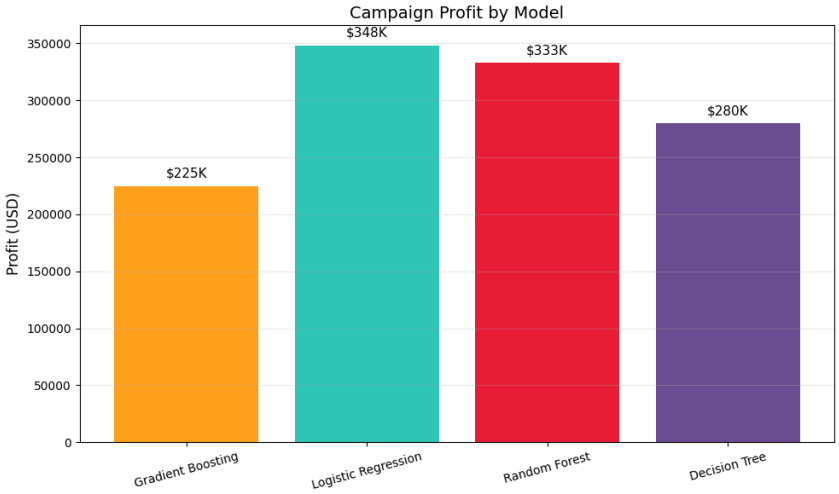

Figure 2: Campaign Profit (USD) by Model. This bar chart visually represents the profitability of different machine learning models, highlighting Logistic Regression as the most profitable based on the specified cost and value parameters.

5.3 Feature Importance Analysis

Top 10 most important features derived from the Decision Tree are listed below:

- duration: 0.3507
- euribor3m: 0.2197
- age: 0.0980
- campaign: 0.0340

- cons.conf.idx: 0.0261
- pdays: 0.0232
- cons.price.idx: 0.0228
- housing-yes: 0.0173
- day_of_week mon: 0.0139
- loan yes: 0.0123

Interpretation of Dominant Features: The analysis reveals that a few features predominantly influence the decision tree logic:

- **duration (0.3507):** Call duration consistently emerges as the most powerful predictor of subscription. However, it's crucial to note that this feature is only known after the call ends, which can introduce a look-ahead bias in real-time prediction scenarios.
- **euribor3m (0.2197):** This economic indicator reflects broader financial market sentiment and shows a strong correlation with subscription outcomes, highlighting the impact of macroeconomic conditions.
- **age (0.0980):** Client age is also a significant factor, commonly influencing a client's propensity to subscribe to banking products.

These top three features alone considerably dictate the decision tree's classification logic, while other features like 'campaign', 'pdays', and 'housing_yes' provide smaller, but still relevant, additional contributions.

5.4 Interpretable Rules

The pruned decision tree provided several interpretable rules that offer actionable insights for campaign managers. Key high-conversion paths identified include:

- **Rule 1:** If `duration` ≤ 524.5 seconds **and** `euribor3m` ≤ 1.402, then the conversion rate is **54.2%**.
- **Rule 2:** If `pdays` is in the range $12.5 <$ `pdays` ≤ 15.5, then the conversion rate is **59.8%**.

These rules indicate specific customer characteristics and timing windows associated with higher subscription rates.

6 Discussion

6.1 Pruning Effectiveness

The application of cost-complexity pruning significantly enhanced the performance of the decision tree model. The tree size was dramatically reduced by 94% (from over 3000 nodes to just 183 nodes), leading to a 26.6% improvement in ROC-AUC. This substantial reduction in complexity not only improved predictive accuracy but also made the model far more interpretable and practical for business users.

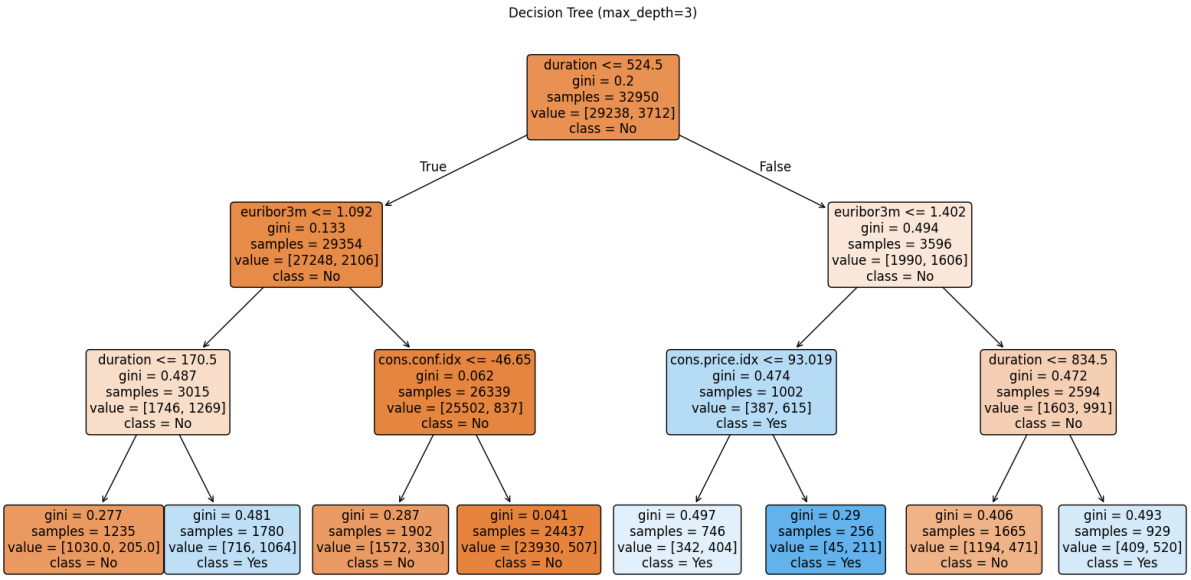

Figure 3: Decision Tree Structure (max_depth=3) illustrating key splitting rules and class distribution.

6.2 Model Selection Trade-offs

Our comparative analysis revealed a critical trade-off between statistical performance metrics and actual business profitability. While Gradient Boosting demonstrated the best discriminative power with an ROC-AUC of 0.955, Logistic Regression maximized profit. This is primarily because the profit gain per true positive ($450) significantly outweighs the cost per false positive ($15). In a scenario where false negatives (missed subscribers) are far more costly than false positives (wasted calls), a model with higher recall, like logistic regression, becomes more favorable, even if its precision is lower. This emphasizes that model selection must be guided by the specific business objective and the associated costs of different error types.

6.3 Practical Implications

The decision tree model offers several practical advantages for telemarketing campaigns. Its inherent interpretability allows for the extraction of clear, actionable rules, enabling marketing teams to understand which customer segments are most likely to subscribe. This transparency is crucial for building trust and complying with regulatory requirements.

Theoretical Concept (Model Interpretability): The ability to understand why a machine learning model makes a particular prediction is crucial, especially in regulated fields like banking. As highlighted by Molnar [4], model interpretability allows stakeholders to verify fairness, detect biases, comply with regulations, and gain actionable insights from the model's decision-making process. Unlike "black-box" models, decision trees inherently offer transparency, making them highly valuable for deriving clear and justifiable business rules.

Furthermore, the model's ability to rank feature importance allows for a focused allocation of resources, targeting customers based on the most influential factors. The profit-driven

evaluation ensures that the model is aligned with business objectives, maximizing the return on investment for marketing campaigns.

7 Conclusion and Future Work

This study provides a comprehensive machine learning framework for optimizing bank telemarketing campaigns. We've demonstrated that pruned decision trees offer an effective balance of predictive performance and interpretability, making them a valuable tool for extracting actionable business insights. Critically, our profit-centric evaluation highlights that profit optimization may favor simpler models like logistic regression over more complex ones, especially when the cost of false negatives is significantly higher than false positives. The extraction of interpretable rules further empowers campaign managers with data-driven strategies for targeted outreach.

Future Work: To further build upon the insights and methodologies presented in this report, several promising avenues for future research and development are identified. These areas aim to enhance model robustness, deepen interpretability, and ensure practical applicability in dynamic business environments:

- Future research will involve applying further pruning techniques to simplify trees, aiming to refine the already pruned decision tree model.
- We intend to experiment with entropy as an alternative splitting criterion for decision trees to investigate its impact on model performance and structure.
- We plan to use SHAP (SHapley Additive exPlanations) for a deeper and more granular analysis of feature impact on model predictions.
- Further work will explore dynamic profit-aware learning algorithms that can adapt to changing cost and value parameters in real-time marketing scenarios.
- We propose to conduct real-world A/B testing of the recommended marketing strategies to empirically validate their effectiveness in live campaign environments.

Individual Contributions

- **Member 1 (Turay Adamsay): Data Cleaning & Feature Engineering (Bank Marketing Dataset)**
  - Objective: Prepared the raw Bank Marketing dataset for modeling. This included:
    * Loading the data.
    * Inspecting structure and missing patterns.
    * Handling "unknown" values and imputing missing data.
    * Encoding categorical features (including the binary target variable).
    * Identifying and removing highly correlated features.
    * Splitting the dataset into stratified train/test sets and saving them for the team.
- **Member 2 (Sandar Win): Baseline Decision Tree & Exploratory Analysis**
  - Objective:
    * Loaded train/test splits prepared by Member 1.
    * Trained a default 'DecisionTreeClassifier'.
    * Evaluated its performance on the test set using standard metrics (accuracy, precision, recall, F1-score, ROC-AUC).
    * Plotted and interpreted a confusion matrix and ROC curve for the baseline model.
    * Visualized and interpreted the top 10 feature importances.
    * Displayed a shallow decision tree structure for interpretability.
    * Performed 5-fold cross-validation on the training data to assess initial overfitting.
- **Member 3 (Jusu Abdul Karim): Advanced Pruning & Hyperparameter Search**
  - **Objective:**
    * Loaded the same train/test splits used by Member 1 & Member 2.
    * Computed the Cost-Complexity Pruning (CCP) path for a fully grown decision tree.
    * Performed a grid search using 5-fold cross-validation (CV) to optimize ROC-AUC.
    * Plotted changes in impurity (training loss) vs. `ccp_alpha` and CV ROC-AUC vs. `ccp_alpha`.
    * Trained the final pruned decision tree using the best hyperparameters and evaluated it on the test set.

- **Member 4 (Laggah Julius): Rule Extraction, Comprehensive Visualization, and Multi-Model Performance Analysis**
  - Objective:
    - Identified and extracted actionable, human-readable decision rules from the final optimized (pruned) Decision Tree.
    - Generated comprehensive visualizations to effectively communicate model insights and performance:
      - Plotted a simplified Decision Tree structure (e.g., max depth 3) to illustrate key decision paths.
      - Created a bar chart visualizing the top 10 feature importances derived from the Decision Tree.
      - Developed a multi-metric performance comparison chart (e.g., spider chart) across all evaluated models (Baseline DT, Pruned DT, Random Forest, Gradient Boosting, Logistic Regression).
      - Produced a bar chart illustrating the campaign profit generated by each of the top-performing models.
    - Conducted a detailed comparative analysis of the Pruned Decision Tree against Random Forest, Gradient Boosting, and Logistic Regression, focusing on both statistical performance metrics and the profit-centric evaluation.
    - Summarized key findings and practical implications derived from the extracted rules and comparative analyses.

References

[1] Dahlén, M., Lange, F., & Rosengren, S. (2020). *Marketing Communications: A Brand Narrative Approach*. Sage Publications.

[2] Lessmann, S., Casu, B., & Reis, D. (2015). Predicting consumer churn in the mobile telecommunications industry: An application of a new hybrid approach. *Journal of Business Research*, *68*(4), 799–807.

[3] Verbeke, W., Martens, D., Baesens, B., & Van den Poel, D. (2012). Building profit functions for cost-sensitive learning in customer churn prediction. *Expert Systems with Applications*, *39*(1), 271–283.

[4] Molnar, C. (2020). *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* (2nd ed.). Leanpub.

[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, F., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

[6] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.