NUS Datathon 2025

Category A

(Group 8)

README File

Team Members

Muhammad Abdul Rauf Bin Abdul Malik

Tan Teck Hwe Damaen

Tay Wei Ting

1. Introduction

The objective of this project was to build machine learning models to predict successful financial advisor-customer matches, improving engagement and policy conversion rates. The approach involved data preprocessing, feature engineering, model training, and evaluation, with both **Decision Tree** and **Random Forest** models deployed for predictions. This reflection outlines key steps, challenges encountered, insights gained, and how they were addressed.

2. Setting Up the Environment

To run the model successfully, the following dependencies were installed:

```
pip install pandas numpy scikit-learn matplotlib seaborn
```

If using Google Colab, ensure GPU acceleration is enabled by navigating to:

- Runtime → Change runtime type → Select GPU

Additionally, dataset files must be placed in the correct working directory before executing the notebook.

3. Running the Notebook

The notebook execution involves the following steps:

1. Load the Dataset: Ensure all **.parquet** files are available and paths are correctly set.
2. Perform Data Cleaning & Transformation: Handle missing values, encode categorical features, normalize numerical data.
3. Train Decision Tree and Random Forest Models: Fit classifiers using historical data.
4. Evaluate Model Performance: Assess using accuracy, precision, recall, and F1-score.
5. Interpret Results & Key Findings: Analyze feature importance and business implications.

4. Data Preprocessing and Cleaning

Data preprocessing was a crucial step in ensuring model accuracy. Several techniques were applied to clean and transform raw data:

- Handling Missing Values: Numerical values (e.g., agent age, household size) were imputed using the median, while categorical missing values (e.g., marital status) were replaced with an "Unknown" category.
- Data Type Conversions: Categorical variables were encoded using label encoding.
- Feature Selection: Irrelevant columns such as identifiers with no predictive power were dropped to reduce model complexity.
- Date Handling: Policy start and end dates were broken into year, month, and day components for better feature extraction.

5. Feature Engineering

To enhance model performance, feature engineering was performed:

- Derived Features: Customer tenure at the time of purchase and agent tenure in the company were calculated to provide insights into experience levels.
- Scaling and Normalization: Numerical features such as annual premium and economic status were normalized using logarithmic scaling to prevent bias.
- Class Balancing Considerations: Policy conversion rates were analyzed to determine if class imbalance existed, ensuring a fair model.
- Encoding Categorical Variables: The **agent_gender**, **agent_marital**, and **cltsex** columns were converted using label encoding, ensuring better representation for the model.

6. Model Implementation: Decision Tree and Random Forest

Two models were chosen due to their interpretability and effectiveness in handling structured financial data:

Decision Tree Classifier

- Train-Test Split: Data was split into 80% training and 20% testing.

- Model Training:

```
from sklearn.tree import DecisionTreeClassifier

dt_model = DecisionTreeClassifier(max_depth=10, criterion="gini", random_state=42)
dt_model.fit(X_train, y_train)
```

Random Forest Classifier

- Train-Test Split: Data was split into 80% training and 20% testing.

- Model Training:

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)
rf_model.fit(X_train, y_train)
```

7. Model Evaluation

Both models were evaluated using classification metrics:

```
from sklearn.metrics import classification_report

# Decision Tree Evaluation
predictions_dt = dt_model.predict(X_test)
print("Decision Tree Performance:")
print(classification_report(y_test, predictions_dt))

# Random Forest Evaluation
predictions_rf = rf_model.predict(X_test)
print("Random Forest Performance:")
print(classification_report(y_test, predictions_rf))
```

- **Hyperparameter Tuning:** max_depth, min_samples_split, and n_estimators were optimized to prevent overfitting.
- **Feature Importance Analysis:** The models provided insights into which features impact policy conversions the most:

```
import matplotlib.pyplot as plt
import seaborn as sns

feature_importances = rf_model.feature_importances_
feature_names = X_train.columns

plt.figure(figsize=(12, 6))
sns.barplot(x=feature_importances, y=feature_names)
plt.xlabel("Feature Importance")
plt.ylabel("Feature Name")
plt.title("Random Forest Feature Importance")
plt.show()
```

8. Challenges Encountered and How They Were Overcome

Several obstacles were faced during model development, along with their respective solutions:

- **Sparse Data in Recommendations:** Many customers interacted with only a few advisors, reducing the number of training samples. To address this, we ensured dataset completeness and extracted additional derived features to improve input quality.
- **Overfitting Risk:** Both models initially showed signs of overfitting. This was mitigated by tuning **max_depth** and **min_samples_split** while using ensemble learning for generalization.
- **Class Imbalance:** Policy conversions were relatively rare, requiring class weight adjustments. The imbalance was handled using **class_weight='balanced'** in the model training process, ensuring minority class representation.
- **Computational Constraints:** Processing large datasets required optimized memory management strategies. Memory-efficient data types were used, unnecessary columns were removed, and batch processing was implemented where needed.

9. Key Insights and Findings

From this project, several key learnings emerged:

- **Decision Trees and Random Forests Are Effective for Advisor Matching:** The models successfully identified key factors contributing to policy conversions.
- **Feature Engineering Plays a Critical Role:** Encoding categorical variables and scaling numerical features significantly improved accuracy.
- **Importance of Data Cleaning:** Handling missing values and ensuring consistent data types positively impacted predictions.
- **Hyperparameter Tuning Matters:** Optimizing decision tree and random forest parameters improved generalization, making the models more robust against overfitting.
- **Feature Importance Reveals Business Insights:** The models identified significant features, allowing stakeholders to make data-driven decisions regarding advisor-client interactions.

10. Conclusion

This project successfully implemented **Decision Tree and Random Forest models** to improve financial advisor-customer matching. The structured approach to **data preprocessing, feature engineering, and model evaluation** ensured robust results. While challenges such as sparse data and class imbalance were encountered, both models provided valuable insights into financial advisory performance. The feature importance analysis further contributed to understanding key business factors, enabling data-driven strategies for customer engagement and policy conversions. The use of both models highlighted the advantages of decision trees for interpretability and random forests for improved accuracy through ensemble learning, demonstrating the power of machine learning in financial decision-making.