Kristina Grkovic

Waverly Allen

BAN 6025

Dr. Balan

2/1/2025

**Sky’s the Limit: Final Deliverable Iterative Model Report**

#### **Project Overview**

This project focuses on building a predictive model for Target\_Y using a dataset with 1,000,000 rows and 32 features. The primary goal is to maximize recall while keeping false positives under control, making the F1 score the key evaluation metric. We utilize a 70/30 train-test split and iteratively refine our approach by testing various models, adjusting features, and fine-tuning parameters.

#### **Data Overview**

* **Dataset Size:** 1,000,000 rows, 32 columns
* **Target Variable:** Target\_Y (binary classification problem)
* **Features:** 31 masked variables (real-world meaning unknown)
* **Class Imbalance:** Target\_Y is highly imbalanced (0: 99%, 1: 1%)
* **Data Split:** 70% training / 30% test

**Evaluation Metric:** Focus on F1 score, balancing recall and precision, with a controlled false positive rate to maintain model reliability.

### **Iterative Modeling Process**

We started with simple models, including logistic regression and a decision tree with default parameters. However, due to the severe class imbalance (99% class 0, 1% class 1), these models performed poorly on recall and F1 score.

#### **Handling Class Imbalance**

1. **Random Undersampling:** Initially, we tried undersampling the majority class, but this led to a loss of important information.
2. **Random Oversampling:** We then attempted oversampling, but the results were inconsistent.
3. **SMOTE (Synthetic Minority Over-sampling Technique):** Finally, we applied SMOTE to the entire dataset before splitting into train/test sets. This approach yielded the best performance, as it generated synthetic examples to balance the class distribution while preserving variability. Tuning the SMOTE parameters had a significant impact on the model results, however this does raise concerns of too much temperament with data.

#### **Feature Engineering**

* Many features had only a few unique values, suggesting they were categorical rather than numerical.
* We converted the following variables into dummy variables to better represent the information: 'X8', 'X15', 'X18', 'X27', 'X1', 'X5', 'X14', 'X21', 'X23', 'X29', 'X31'
* This transformation improved model interpretability and performance.

**Model Experiments**

We tested multiple models, applying a **Grid Search** and **Bayesian Optimization** to a sampled data (10% of full data) on Decision Trees, Random Forests, and Gradient Boosting Machines (GBM) to find the optimum parameters prioritizing the F1 score. While Decision Trees showed high variance and overfitting, Random Forests improved performance but still struggled with precision-recall balance. GBM provided the best results, achieving a strong trade-off between precision and recall. Additionally, we experimented with two Neural Network architectures, but training times were significantly longer, and improvements in F1 score were marginal compared to GBM.

#### **Tracking Progress**

We maintained an Excel sheet to log all experiments, capturing model parameters for each iteration, and performance metrics (Accuracy, Precision, Recall, F1 score, and AUC) for train and test sets. It helped us stay organized by being able to have a comparison of models and hence select the best-performing configurations and run them on the full dataset (we had a separate dashboard of the model results for the sample data and another one for the full data).

**Final Model Selection**

Based on empirical results, the final model selected was **Gradient Boosting**, which outperformed other models in terms of F1 score and recall while keeping false positives in check.

#### **Hyperparameter Tuning Approach**

* Used Bayesian Optimization to refine hyperparameters.
* Applied tuning on sample data due to long training times on the full dataset.
* Final optimized parameters included:
  + n\_estimators: 150
  + learning\_rate: 0.15000000000000002
  + max\_depth: 9
  + subsample: 1.0
  + min\_samples\_split: 20
  + min\_samples\_leaf: 15
  + max\_features: ‘sqrt’

#### **Challenges and Limitations**

Due to computational limitations with our computers, long training times required us to rely on sample data for tuning. In the future, it would be beneficial to have a stronger computer that can run the optimizations on the full data without taking hours to run. Further, we learned that SMOTE only works on numerical data, requiring categorical variables to be preprocessed. It was also difficult to understand the data since features were masked; we had to rely on model-driven insights rather than domain expertise.

#### **Key Learnings:**

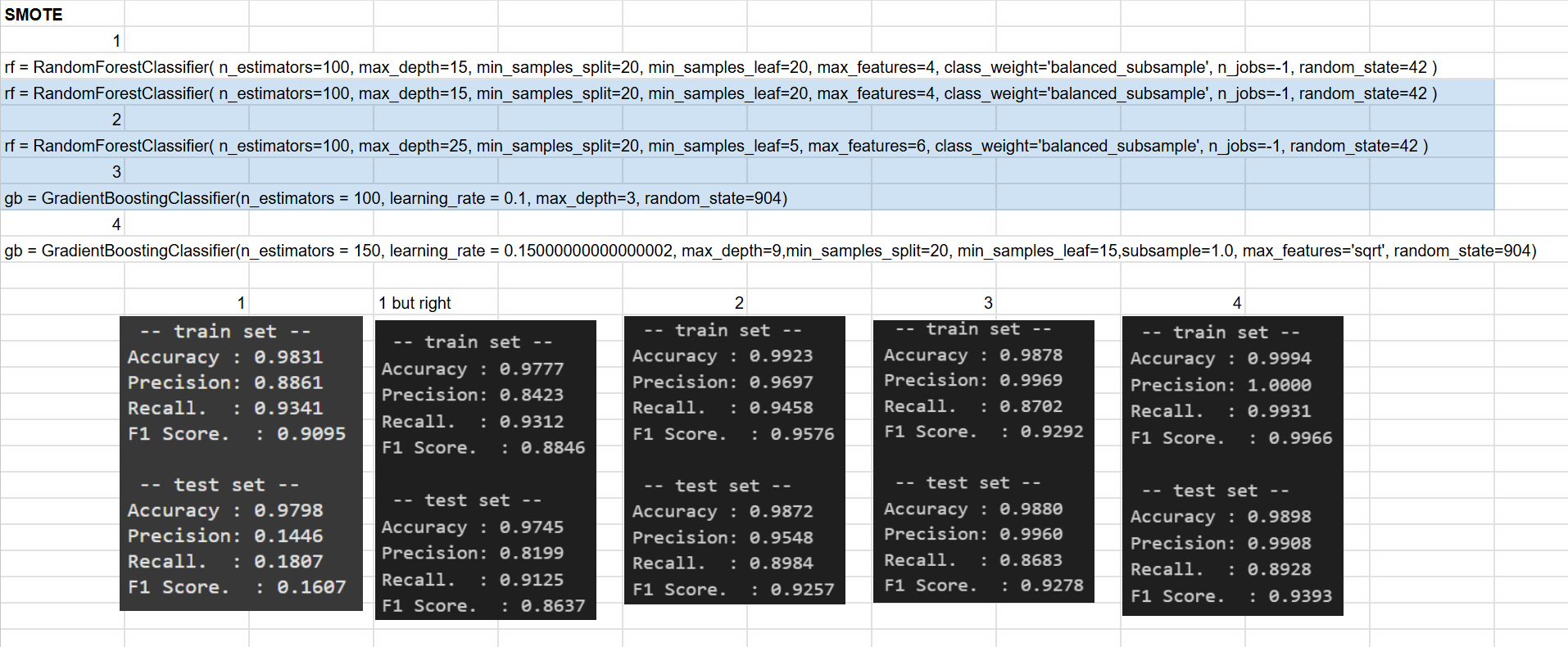
* **Handling Class Imbalance:** Random oversampling was insufficient; SMOTE provided the best results.
* **Feature Engineering:** Converting low-unique-value features into dummy variables significantly improved model performance.
* **Model Selection:** Gradient Boosting with Bayesian Optimization achieved the best recall and F1 score.
* **Hyperparameter Tuning:** Bayesian Optimization was efficient for selecting optimal parameters.
* **Data Limitations:** Computational constraints forced us to tune models on sample data before applying them to the full dataset.

#### **Potential Next Steps**

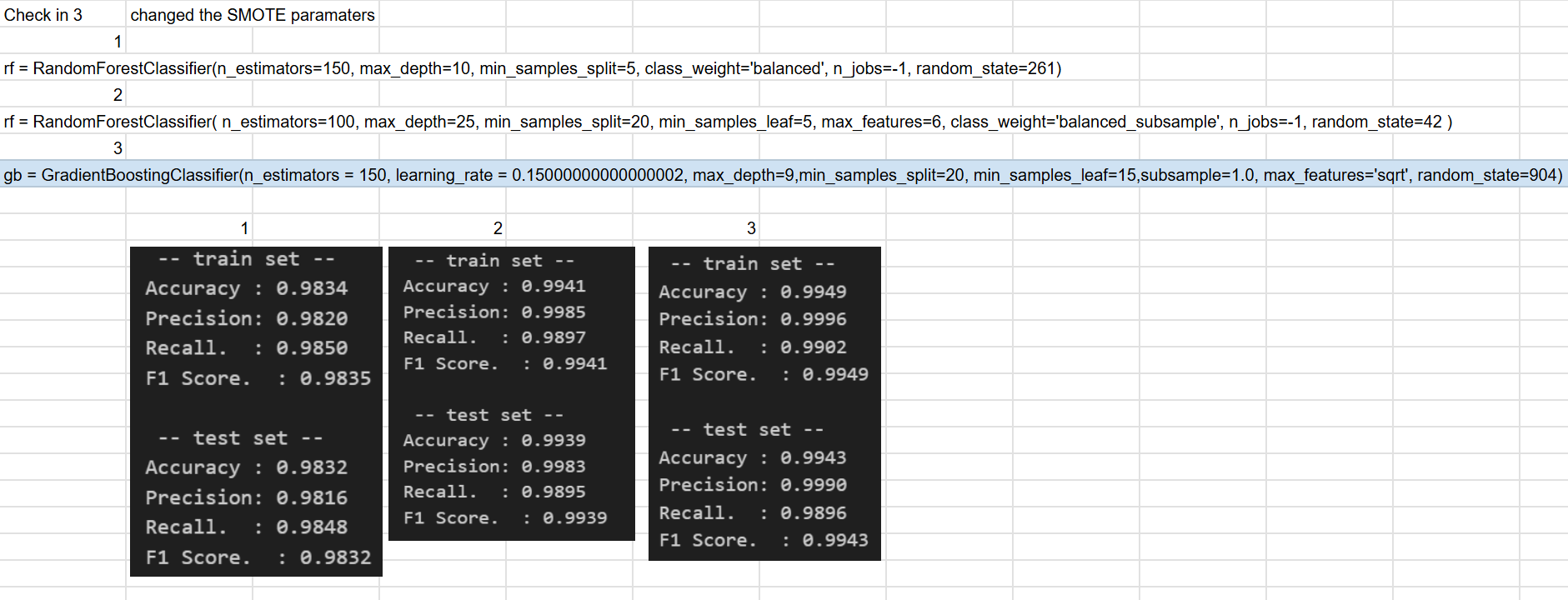
* **Feature Interpretability:** Since feature meanings are unknown, further analysis with domain experts could refine feature importance.
* **Alternative Models:** Testing **LightGBM or CatBoost** may reduce runtime while maintaining performance.
* **Neural Network Fine-tuning:** Further hyperparameter tuning could improve neural network results.
* **Deploying the Model:** If this were a real-world project, the final model would need to be tested in production with real-time data.
* **Balancing data without SMOTE:** Using a different approach to balance the data might “tamper” less with it.

**Technical Appendix**

**A snapshot of our model organization on the Excel where we had around 20+ models total:**



**Another snapshot of our final 3 models and their parameters:**



**A close up of our final models results:**

gb = GradientBoostingClassifier(n\_estimators = 150, learning\_rate = 0.15000000000000002, max\_depth=9,min\_samples\_split=20, min\_samples\_leaf=15,subsample=1.0, max\_features='sqrt', random\_state=904)

