Marketing Targets - Customer Subscription Prediction

Xufei Li

Abstract

The goal of this project is to help a bank institution build a classification model to predict potential customers who's going to subscribe to "Term Deposit". Then, the institution will apply direct marketing campaigns(phone calls) to the Marketing Targets. I used data from Kaggle-Banking dataset, then started with a baseline Logistic Regression model **F-beta(beta = 2): 0.7582, ROC_AUC: 0.6502**. Then adding complexity to improve the prediction ability. I trained 6 candidate models and selected the best performed one-XGBoost. Then, I retrained the model to get my final model with **F-beta(beta = 2): 0.9946, ROC_AUC: 0.9998.**

Design/Metric Choosing

For the metric, I chose a F-beta score as a hard prediction metric. I want to minimize False Negatives which are the actual potential customers but were predicted as not going to subscribe in this case. At the same time, I also don't want too many False Positives either because I don't want our sales team to waste too much time calling people who are not going to subscribe. So, I set beta = 2 which has a good balance of precision and recall with a slight favor to "recall". At the same time I don't want my soft predictions to be too off, so I will use ROC_AUC as my secondary metric as a reference.

Data

The training dataset contains 45,000+ rows, 16 features while the test dataset contains 4,500+rows, 16 features. Target is 'y'-- has the client subscribed a term deposit? (binary: "yes","no").

Algorithms

Feature Engineering:

- Start with numerical data after the EDA process, fit models in single features separately to explore the performance.
- Combining features to improve prediction ability.
- Keep improve complexity - Scaling data, and adding dummified categorial features

Models(6 classification models):

Logistic Regression, k-nearest neighbors(20-NN), Random Forest, XGBoost, Naive Bayes, SVM.
I splitted the train.csv into a training & validation set. Then train on training and get the score on validation. Below is the comparison among different models.

Model Evaluation & Selection:

- Logistic Regression: F-beta: 0.7933 ROC_AUC: 0.7687
- 20-NN: F-beta: 0.7632 ROC_AUC: 0.7392
- **Random Forest: F-beta: 0.8333 ROC_AUC: 0.7928**

- **XGBoost:** **F-beta: 0.8329** **ROC_AUC: 0.7992**
- Naive Bayes: F-beta: 0.8016 ROC_AUC: 0.7487
- **SVM:** **F-beta: 0.8388** ROC_AUC: N/A

XGBoost performs the best. So, I tuned the hyperparameter to get a better predictive power. Then retrain my model based on (train+validation), then score on the test set.
Final score(on test):

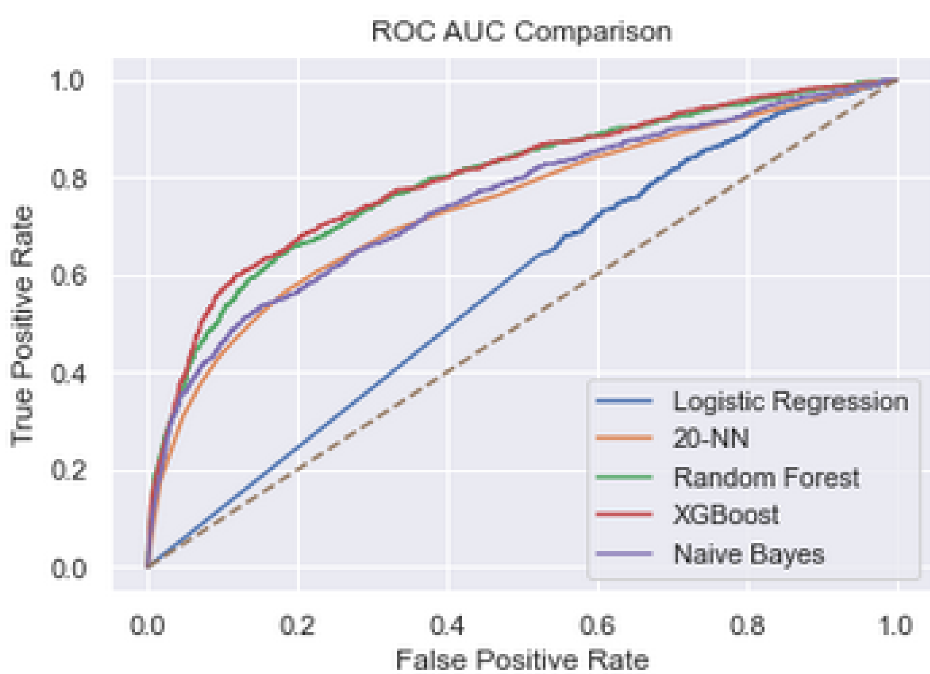
- **Final XGBoost:** **F-beta: 0.9946** **ROC_AUC: 0.9998**

Tools

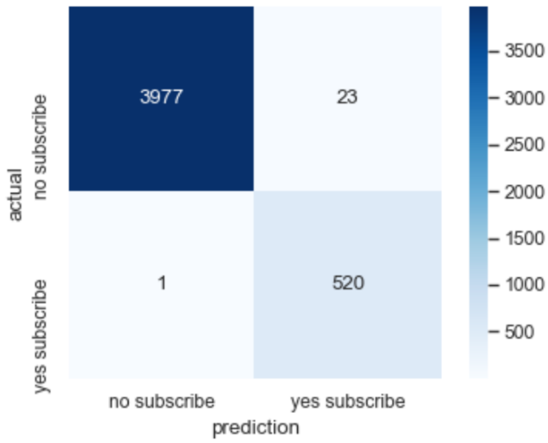
- Numpy and Pandas for data manipulation
- Scikit-learn for modeling
- Matplotlib and Seaborn for plotting

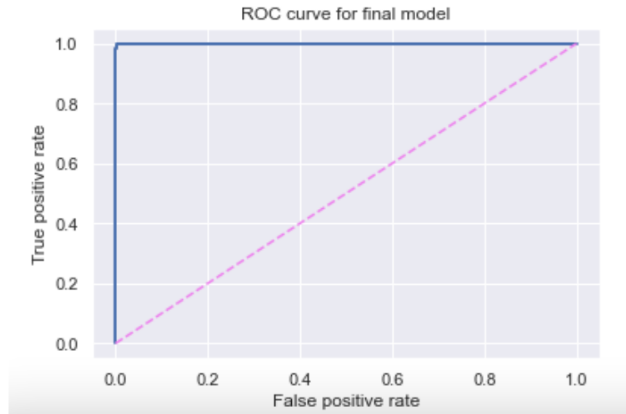
Communication

Model Selection Comparison



Final Model Evaluation





Conclusion & Future work

In order to build a robust model, I want to go back to check my code settings to make sure everything has been set up correctly and then try the second candidate model(Random Forest) as well to compare results. Also, I will try different “random_state” to see if it’s the validation set that drives the higher score. Further on, SVM performs pretty good as well on F-beta score, so I can also tune hyperparameters for SVM to see if it can be a good predictor for our client’s as well.