Questions:

(1) Describe your model and why you chose this model over other types of models? Describe any other models you have tried and why do you think this model performs better?

* I tried three different models during this assignment: XGBoost, Random Forest, and LightGBM. I decided to use decision tree algorithms because there are so many variables of different formats (decision trees don’t require data to be normalized or scaled). Decision trees were also used because of the small size of data.
* I ended up using XGBoost for my model. The results from training and testing (explained further in third question) were about the same for all the models, but I believe XGBoost handled the class imbalance better than the other models. I assumed the data from marketing\_test has a similar imbalance as in marketing\_train, and my results using XGBoost had a much closer ratio of yes to no responses as seen in marketing\_train. Marketing\_train has yes response approximately 11% of the time; XGBoost predicted yes approximately 14% of the time, whereas the other models predicted yes around 3% of the time.

(2) How did you handle: missing data, string data, unbalanced data?

* There was data missing from custAge, schooling, and day\_of\_week. For custAge, I took the average custAge for three profession groups: ‘student’, ‘retired’, and the remainder. I replaced the missing custAge with the average based on its profession. Schooling has an option of ‘unknown’, so all missing values for schooling were replaced with ‘unknown’. Missing values for day\_of\_week were ignored because all the days had about the same ratio of yes to no responses.
* String data exists in the dataframe as an object data type. All columns were converted to categorical then one-hot encoded.
* The XGBoost model expects unbalanced data and trains accordingly by using the scale\_pos\_weight parameter when creating the model. I initially set scale\_pos\_weight = 9 because the dataset has about a 9:1 ratio of no to yes responses, but it was predicting a much higher percentage of yes responses (around 30%) than I would expect if the data from the two marketing datasets were randomly selected from the same environment. I adjusted scale\_pos\_weight to 5 which gave a much closer yes response about 14% of the time.

(3) How did you test your model?

* I start with using the data from marketing\_train for training and testing the different models. I used the scikit-learn train\_test\_split function to randomly select 70% of the values for training and the remainder for testing. I looked at the confusion matrices and ROC plots/AUC scores for comparing the different models.