Project 3 Summary: Connecting with Shoppers in the Digital Age

Project Design:
I used classification models to predict whether a customer will make a purchase. This model could be used by ecommerce businesses to identify customers who are more likely to make a purchase, and as a starting point for helping the business identify ways to meet the shopper's needs. Some examples include: 1. optimizing the use of the business's resources, website update planning, marketing (e.g., advertising, recommender systems, sales and promotions); and 2. performing cost models to calculate the revenue based on gains from customer purchases vs. costs of implementing changes.

I tested the following models for this project:

- K Nearest Neighbors
- Gaussian Naïve Bayes
- Logistic Regression
- Support Vector Classifier
- Decision Tree Classifier (just to see how it performed—not to be used by itself)
- Random Forest Classifier
- XGBoost Classifier

I ran my models using a single split of train, validation, and test because I wanted to ensure I followed the correct order of steps of standard-scaling (when necessary), performing oversampling methods, and keep oversampling steps separate from my cross-validation and test data—and I felt CV was making that more confusing. Because my dataset was relatively large enough to do a single train-test split, then split train into "subtrain" and validation sets, I proceeded with this path. However, since this approach is not using k-folds and cross-validation and averaging out the performance metric scores, my measures of model performance may not be as robust.

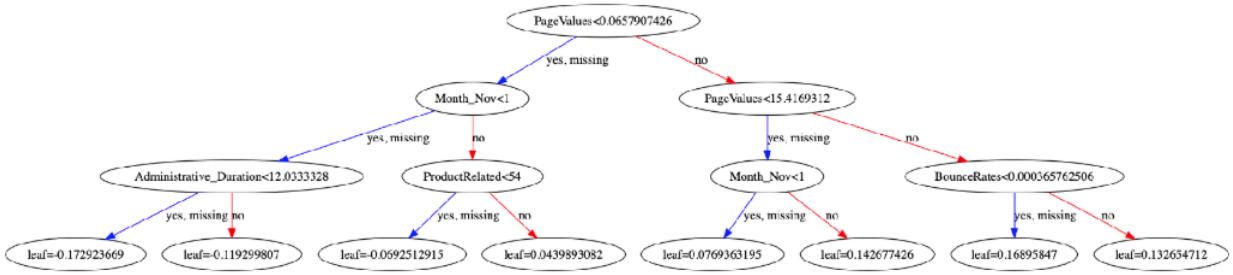
I tuned my parameters manually to produce 108 models (after manual tuning and oversampling methods), then I tried RandomizedSearchCV on my best classifier (XGBoost). Because of the manual process, I may not have tuned all the parameters correctly and specified all of the correct settings in all of my models. Additionally, some of the parameters produced in my best model of XGBoost using RandomizedSearchCV led to overfitting, so I did not end up using these results anyway.

My best model was XGBoost with the following parameters:

```
XGBClassifier(max_depth=3,n_estimators=95,random_state=41, scale_pos_weigh
t*=(y_subtrain.values==0).sum()/(y_subtrain.values==1).sum())
```

*I used scale_pos_weight instead of the oversampling methods we learned in class (i.e., random oversampling, SMOTE, and ADASYN)—see my notes under the Data section below.

This is the top of the FIRST tree in my XGBoost model:



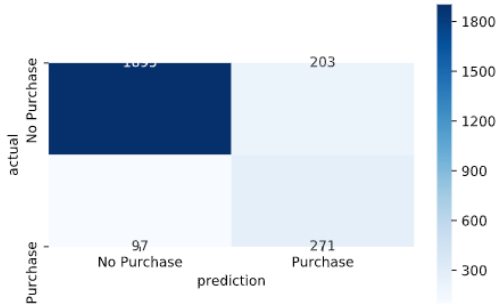
I used ROC AUC as my metric because my classes were imbalanced and ROC is less sensitive to imbalance. The ROC AUC of my final model was 0.85.

At first, I believed that minimizing false negatives would result in a bigger loss for the business because by ignoring a customer who may have been persuaded to make a purchase, we would lose a sale. However, devoting resources to improving features for people who have no interest in making a purchase can also be a big cost to the business as wasted effort. We would want to act (e.g., offer assistance or recommendations) to customers who are actually likely to make a purchase but also likely to leave the website (the latter of which was not in the scope of this project). To

Kristen Tokunaga

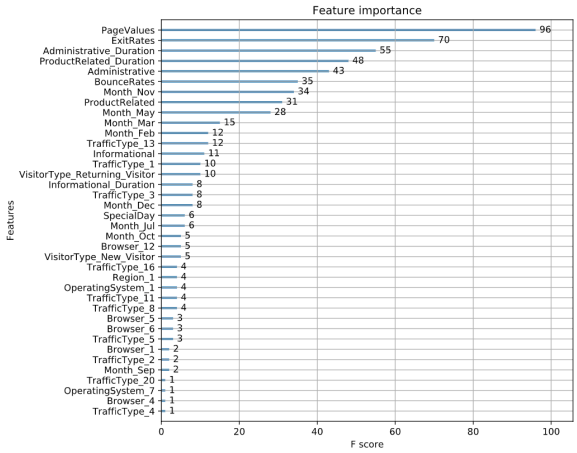
Project 3 Summary: Connecting with Shoppers in the Digital Age

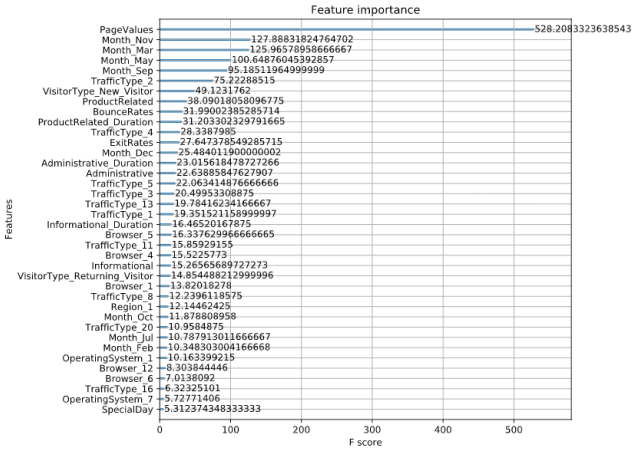
find the optimal balance of false positives and false negatives, after selecting my model with the best ROC AUC, I selected a threshold that optimized the F1 score. This resulted in the following confusion matrix at my selected threshold (0.656602):



Out of the 2466 y values: 1895 true negatives, 203 false negatives, 97 false positives, and 271 true positives.

Below are the feature importances of my best model:





Based on the page values and month of visits appearing high on my feature importances list, I tailored my recommendations to focus on these features. Additionally, I did not believe there were actionable insights that could be derived from some of the other features (e.g., page types and exit rates or bounce rates) without knowing more specifics about the business's ecommerce site.

Tools

- Python
  - Data Cleaning & Analysis: Pandas
  - Data Visualization: Matplotlib, Seaborn
- Tableau
  - I created a dashboard of customer demographic data with filters
  - I also created sheets with individual charts to view monthly trends of purchases vs. website traffic volume, which types of pages were visited (admin/account management pages, informational pages, product-related pages) and for how long, and Google Analytics metrics of the pages (bounce rates, exit rates, and page values).
  - For all of these graphics, I added filters that I believed the business user would find relevant to narrow down results and I could potentially use as a starting point to identify interesting interactions of the features (e.g., if certain months had more volume on the weekends—because then maybe if the website normally has downtime for updates/maintenance on weekends they should consider rescheduling to accommodate for higher traffic, or they should staff accordingly to have more customer and technical support during the weekends of those months)

Data

I used the "Online Shoppers Purchasing Intention" Dataset from the UCI Machine Learning Repository. There were 12,330 observations (sessions) collected over one year—although there were only 10 months accounted for in the

Project 3 Summary: Connecting with Shoppers in the Digital Age

dataset (January and April were not represented). The dataset contained 10 numerical and 8 categorical features. The categorical features required one-hot encoding. I explored my dataset to see if it needed Standardscaler, or if multi-collinearity was present—which may matter for some models and less for others.

Because the requirement of this project was to use supervised learning techniques, I chose this dataset because it had a labeled target variable.

My dataset also had class imbalance in my target variable (15.5% were purchases, 84.5% were non-purchases), and this made sense in terms of representing real-world data because a majority of ecommerce shopping sessions do not result in a purchase. I had used several options to overcome this:

1. Use a metric that is less sensitive to class imbalance (e.g., AUC of the ROC)
2. Oversampling methods (Random oversampling, SMOTE, ADASYN)
3. Scale_pos_weight for XGBoost: Per the recommendation from readthedocs, I adjusted this from the default of 1 to sum(negative instances) / sum(positive instances).

I compared models using the oversampling methods and scale_pos_weight across my models and found that XGBoost using scale_pos_weight was my best model.

There were several limitations in my dataset that made it difficult to derive direct actionable takeaways for the business user from the model itself. Specifically,

1. **The actual business was unknown:** There was no context of the product(s) offered or the industry of the ecommerce site where the data came from. There is a lot of variety in how much implementing a website update or recommender system would cost and there was no pricing information included in the dataset to determine how much the business would earn for a purchase. I looked into the researchers who supplied this data to see if I could find more information about the ecommerce site's business, but they did not reveal specifics in their research paper.
2. **De-identified "demographic" features:** Features such as region, browser, operating system, and "traffic type" (i.e., whether the customer came to the website from an email, advertising partner website, etc.) were all labeled with a number as the categorical feature—rather than the name of the actual category. For example, operating system was labeled 1,2,3,4,5,6,7, or 8. The specifics of this information could tell us more about the customer base and where to focus our efforts—such as making the website compatible with certain types of devices, browsers, etc. And knowing the geographic region of the customer would help us see whether we were reaching our target demographic. After exploring as much as I could on the Google Analytics website about these features, I also attended a Google Analytics 101 session from Seattle Tech4Good to learn more about whether these "numbers" are actually mapped to a specific category (e.g., if Browser 1 = Google Chrome, vs. Browser 2 = Internet Explorer. Unfortunately, there is no such key in Google Analytics to find this information.
3. **Aggregated page detail information:** Each session captured the bounce rates, exit rates, and pages values as an aggregate value (averages of all the pages visited per session). Typically these values are used at the page-level to identify pages that are more or less successful at attracting and/or maintaining customers' interest.

What I Would Do Differently

If I had more time, I would have liked to:

1. Explore interactions. Some examples include:
    a. Weekends and months visited by the shoppers
    b. Types of pages visited (administrative, informational, product-related) and their bounce rates, exit rates, and page values
        i. For example, were there administrative or informational pages that have high page values, bounce rates, or exit rates
2. Tune more parameters for other models
    a. Specifically, SVM, random forest, and logistic regression
        i. Both for my learning and to see if they could produce similar or better results
        ii. I figured out how to perform RandomSearchCV a little
3. Explore my feature importances further

Kristen Tokunaga

Project 3 Summary: Connecting with Shoppers in the Digital Age

   a. Because many of my categorical variables were one-hot encoded (dummies) in my model, their importance may be have been diluted because they were split into individual features—which made their importance as an overall category harder to interpret
   b. Look more into feature importance types—as there are multiple ways to calculate feature importance from tree-based models, some producing different results
   c. Better understand permutation of feature importances—I think I was able to get the function to work, but couldn't really make sense of the results
4. Learn more about how to use RandomizedSearchCV and GridSearchCV, and use them earlier for parameter tuning so I could instead spend more time finding better guidance on tuning the models' parameters and optimize my results

Future Work

1. Real-time modeling (i.e., for recommender systems, offering instant sales/promotions, offering assistance with an alert or message)
   a. Other models to try: multilayer perceptron classifier
      i. These appear to be used in real-time modeling because they have online learning implementations
   b. Feature selection (narrow features) to see if I can get faster model performance—although my model's predictions already ran pretty quickly, it's nice to have a simplified model if there are unnecessary features
   c. Scoping out work for building a recommender system (if the business doesn't already have one in place)
2. Explore more features
   a. More customer demographic information (e.g., gender, age range) may help improve the model
   b. If the following features are available, we could look into: patterns of when website changes occurred (e.g., website downtime, major design changes, ad campaigns), sales, or if the site already has recommender systems in place—these features may also help improve the model
3. Deeper dive into the business's trends
   a. Getting page-level and product-specific details can help us move forward in identifying the more vs. less successful pages and products—and we can take action on improving their page values or finding ways to actually direct more customers to the high-value pages
   b. Year-over-year trends. Since this dataset only captured a one-year period, we may
   c. We could also look at whether products were
4. Cost model for a cost-benefit analysis
   a. To assess the overall revenue based on gains from customers making more purchases vs. costs of offering a sale or devoting resources to implementing a change

Kristen Tokunaga