Final Report

MSBX 5415: Advanced Data Analytics

Teddy Li, Alex McLaughlin, Alex Qaddourah, Junji Wiener

In the past year, retail e-Commerce has generated more than 3.5 trillion dollars, representing a 14.9% growth in comparison to 2018. In fact, e-Commerce today accounts for more than 50% of the overall retail market meaning that online retail sales outweigh that of brick-and-mortar sales. The decline of in-store retail experiences and feigning need for physical sales representatives creates a large demand for understanding the interests and behaviors of online shoppers. Developing an understanding of consumer actions is a necessity in order to tailor, or personalize, the online shopping experience to consumer's needs; in turn, conversion rates could be optimized. While traditional data analytics techniques have focused on mining user demographics, sales and attribution, the opportunity to establish an understanding of sequential session actions can highlight more intricate user patterns. The behavior of users is captured by mapping user records according to the structure of the e-commerce website as "user events." These user events capture exactly how online shoppers move through the e-Commerce purchase funnel.

Actionable insights can be found by investigating variations and behavioral patterns among users as they progress through the purchase funnel. Identifying these trends and patterns can support multiple departments within a retail organization, primarily website designers, UX/UI developers, digital marketers, and C-suite executives making budgeting decisions. Notable “user events” within the purchase funnel relate to the path in which a typical online shopper would convert on the e-Commerce website; therefore, important actions include Product Landing Pages, Cart Landing Pages, and Purchase Confirmation Landing Pages. The primary goal of the online shopping experience is to mitigate the number of users who “bounce,” leaving the website and potentially abandoning their cart. The average cart abandonment rate for online retailers is a staggering 67.91%. By identifying actions and timelines of user sessions, marketers and analysts can identify opportunities to disrupt users from “bouncing” due to bottlenecks in specific landing pages and motivate them to purchase by fine-tuning marketing strategies.

A data analytics solution can illuminate characteristics of consumer groups, opportunities to change behavior, and ultimately optimize conversion. In order to motivate would-be non-purchasers (given identified patterns among user sessions), marketers must tune their tactics based on analytics insights. Remarketing can be leveraged to increase conversion among consumers who have “bounced” from the website after a product view or cart addition. With remarketing, consumers are reminded via email or personalized advertisements that they recently viewed a product or have an item in the cart waiting for them to make a decision. Because user session history also includes temporal information such as the date and time in which they

completed an action on the website, analysts can identify timelines when consumers are most likely to convert and work with marketers to offer sales and promotions during that period. Finally, a data analytics solution can help to form a discussion surrounding pricing strategies and how they affect the entire funnel, from attribution to a product view through the eventual purchase decision.

To begin, our data set was downloaded from Kaggle.com and focused on E-Commerce events from an unnamed E-Commerce website. The data was listed as behavioral, but our group wanted to test Machine Learning on this data set. The original data was 42 million rows of user events. The data included features such as “user_id” (consumer’s unique account ID), “user_session” (consumer’s active session; it is possible for there to be more than one unique user sessions per user_id), “brand” (brand of product), “price” (price of product), “product_id” (unique ID for each different product), “category_code” (category of product such as smartphones, etc), and “event_time” (the date and time of when the user event happened). To describe the data, we first ran some descriptive code to learn more about each column. For user_id, there were about 2.3 million unique user_ids inside the data set. There were around 6.4 million unique user_sessions inside the data set. It is important to note that one user_id can have one or more user_sessions, as they may log in to the website on a different day or time as his or her first session. Brand was in abundance in the data set, represented by about 3,400 unique brands. Category Code had 127 unique levels such as electronics, furniture, and apparel. As one can see, because of the wide variety of category codes in the data, it makes sense to have so many different brands. Price had a large range, with its minimum value of just under a dollar, to its maximum of around

$2500 dollars. Finally, there were around 166,000 unique product_ids in the data set. Product ID refers to the specific product's unique identifier. For example, if there are 2 Samsung smartphones in the data set listed at different prices, these two phones would have a different product_id, indicating they are most likely different models of the same phone. It also should be noted that this data set spanned just one month, October of 2019.

One of the key attributes of this data set was the "event_type" that each row was classified as, such as a "view" event (consumer clicked on this product on the website), "cart" (consumer added this product to his/her cart), "purchase" (consumer bought the product). The problem with this data set was that the "View" events accounted for 96% of the data set. This makes sense, as most people click around on a website and may never purchase anything, but still creates a serious imbalance of the dataset. "Cart" event types accounted for 2.2% of the data, while "Purchase" accounted for just 1.8% of the data. Because of our business question, we targeted the "Purchase" event type as our target variable for this project.

During our initial consideration of the dataset, we found that the data quality score on kaggle was high but our first impressions in Exploratory Data Analysis proved that it was of lower integrity than anticipated. Manually reviewing data was not directly feasible; therefore, more time and effort was spent examining samples and subsets of data than would typically be expected. Realizing that the data contained a plethora of issues such as poor formatting and skewed distributions, we felt that the project would be an excellent opportunity to take a stab at a dataset requiring a great amount of manual review and preparation.

As mentioned previously, one of the biggest challenges at first was the massive size of the data set. At 42 million rows, this dataset was very difficult to load and maneuver. The initial struggle was loading the file itself, it took several minutes. We knew that it would not be easy to just run code without thinking about what we were doing first because of the time it would take to run, so we needed to be very strategic about our decisions. Another challenge with this data set is that for Machine Learning preparation, the data did not currently have enough features to draw meaningful conclusions. From the initial data, we hypothesized that brand and price would be the most likely drivers of a purchase decision - but we wanted to feature engineer more features that we thought might hold weight.

One of the first things our group did was to deal with any blank cells. Upon inspection, all of the blank cells were located in “brand” and “category_code”. We converted these to NA, and removed them from the data set. After removing the NAs our dataset was now around 26 million rows. Our reasoning to remove these NAs was simply that we had enough data to still have reliable reporting and statistics for our business question. We thought about imputing these missing values, but because they were missing “brand” and the “category code” they were in, we felt it was not needed and decided to just remove them.

After this, we wanted to make “category code” easier to query and read. Originally, the data had the code in one long string, inside just one column. An example was “electronics.smartphones”, where the period seemed to separate the general category from a more specific one. We used a package in R to help us with this manipulation, and we were able to create new columns with the delimiter as the period.

This allowed us to subset off of a general category, or a more specific one, later in our data analysis.

At this point, our data was still much too big to work with. As a group, we decided to look at the electronics category, specifically smartphones, and subset off of this group. Once we did this, everything in the dataset became more manageable to work with. There were only 40 brands of smartphones in the dataset, some large brands, and some small. This brought our dataset to under 12 million rows, which was a lot easier to deal with and run code off of.

Our next step was to set up our 3 event_types using OneHotEncoding methods. This will help us feed it directly into our machine learning models later on in our analysis. To do this, we assigned a 0 or a 1 for each event type respectively, in its own column. This gave us 3 new features to test against, which is something that we were trying to do with our dataset. NOTE: when we created the view and cart columns with 0 or 1, these were direct predictors of the purchase column which led to all models giving 100% accuracy. We ended up removing the view and cart columns, and received better scores.

The next feature we wanted to look at was the time of day. To preface, the date and time was inside one column. To make things easier, we split up each entity (day, hour, etc) into its own column. Using descriptive methods, we concluded that Saturdays were one of the highest traffic days, and decided to only look at days of the week that fell on a Saturday. Once this was done, we binned the hour of each day into 4 groups: “Early Morning” (12 am - 6 am), “Morning” (6 am - 12 pm), “Afternoon” (12 pm - 6 pm), and “Evening” (6 pm - 12 pm). We OneHotEncoded these new columns as well, giving

us 4 new features to test with. In addition, we also kept the Hour column to potentially use in modeling.

Next, we thought it would be interesting to see if the brand's country would give any insight into a purchase decision or not. To do this, we had to manually look up and create lists for each country that a brand belonged to, and then map this back to the dataset. We OneHotEncoded for each country that was represented in the dataset.

Finally, our last set of features were frequency based and dealt with the user_ids and the product_ids. Our first idea was to count how many total product_ids a specific user looked at. This might be helpful in predicting a purchase because if a user has just looked at one product, he or she may not buy. However, if they have looked at 25 products, they are serious about purchasing. Then, we generated two more features: the amount of views and the amount of carts per user_id per product_id.

To conclude, after subsetting the data into just smartphones, and after creating more features based on the limited number of columns in our original dataset, we felt as if our data was ready to be placed in machine learning models.

Because our target variable is binary in nature representing two outcomes, purchase or no purchase, we sought out classification algorithms to enable prediction in identifying the category to which new data will fall under. Exploratory research and data analysis led us to focus primarily on a Gradient-boosted decision tree, namely XGBoost, due to its versatility with categorical variables, unrivaled capability to improve overall prediction accuracy, and proven track record as a popular machine learning algorithm. XGBoost employs bootstrap aggregation and efficiency "boosting"; therefore, interpretability can be complicated, varied, or even lost. For comparison and support to

a Gradient-boosted decision tree, we elected to perform a straightforward predictive analysis of binary outcomes using a logistic regression model. With logistic regression having no hidden layers between input and output, we decided to investigate the effect of including hidden layers through a neural network analysis.

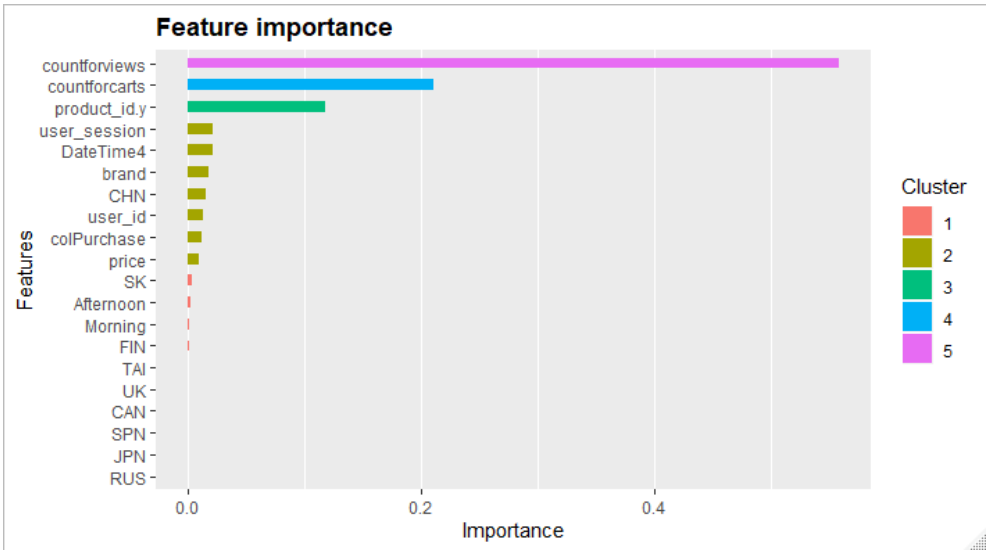
While we initially attempted a K-Means clustering approach to understand user groups, we felt that the resulting characteristics of the cluster analysis did not provide significant contribution to support our solution for the business problem being addressed. As is clear with our model selections, we wanted to focus on predicting the value of our target variable, *colPurchase*, to understand the probability that a customer will convert on the e-commerce website. Additionally, the complicated geometric structure of our data and large distance between data points led kmeans to incorrectly determine clusters. Data points in small clusters received less attention due to a focus on larger clusters.

We wanted to include logistic regression because it is the easiest model to understand and implement. It allows us to run a model with all predictors, to see which ones are statistically significant, and remove variables as needed. It also allows us to view the coefficients for each predictor, to see which has the highest or lowest impact. Logistic regression also arguably has the best output visually, compared to other models. The cons for logistic regression are that it assumes the relationship is linear, which is likely never the case using real world data. Another con with Logistic is that it is less robust under the hood, which could lead to less findings in the data.

Neural networks, the basis of deep learning, offer a flexible predictive modeling solution that works well with classification problems. By splitting the classification task

into a multi-layered hidden network of simpler elements, a neural network ensures reliability over a model with many features. One drawback of neural networks is that we cannot gain insight into feature importance of independent variables influencing the target variable; however, this is mitigated by implementing XGBoost and logistic regression.

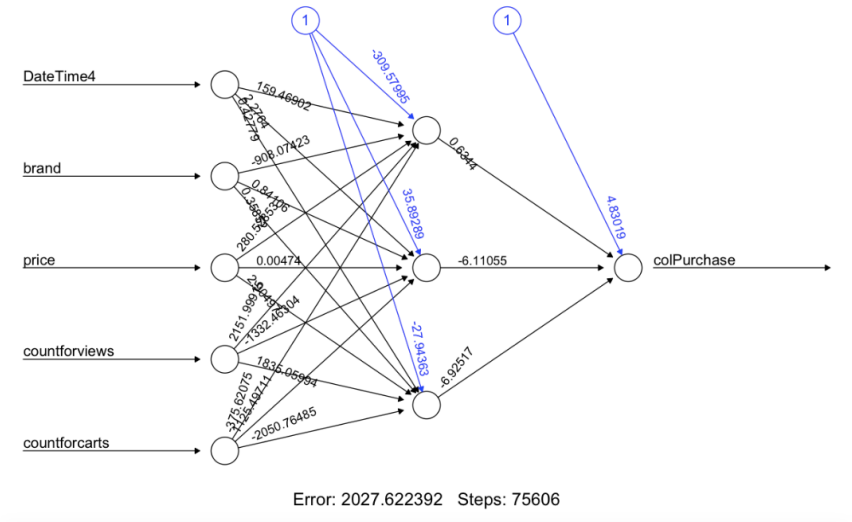
With the XGBoost model, our primary intention with the output was to achieve high prediction accuracy and precision, as well as feature importance. After running an XGBoost model with 1000 trees and a Bernoulli distribution, we ran a prediction on our testing data with a cutoff of 0.25. We achieved an accuracy of 88.3%, a sensitivity of 12.6%, and a specificity of 95.6%. In plotting the feature importance, we found that the number of times a user viewed Product Landing Pages, Cart Landing Pages, and total unique products were the top 3 predictors.



The results of this model provide useful insights into our business problem and offer a potential way of predicting purchasers vs. non-purchasers based on user actions across a multitude of sessions. As one would expect, a user who spends a significant amount of time on the website is more likely to convert, or make a purchase, on the e-

commerce website. Interestingly, Product Landing Page views are considered more important than Cart Landing Page actions. This can be explained by the typical customer journey in which an online shopper considers a variety of products across different brands. The more a user considers a set of products within the same category, the more likely he is to commit to a purchase decision. When an online shopper makes the decision to pursue an item, they will selectively add the one or two items, smartphones in our case, to their cart in preparation for the final phase of the purchase decision. In turn, the *product_id.y* feature refers to the total number of unique products viewed by a customer whereas *countforviews* can include multiple product views of the same product in separate sessions by the same user. When a customer increases the number of products in their consideration set, the user has a higher prediction probability of purchasing.

With the Logistic Regression model, our primary intention was to create a "descriptive model" that shows us which features were more significant than others. After running the model, we had an accuracy score of 91%, a sensitivity of 12.4% and a specificity of 99.5%. For business insight, we had several predictors that were deemed relevant from our model. Several big name brands like Asus, Huawei, Samsung, Sony, Xiaomi and Apple were all significant predictors, which tells us consumers are likely to purchase based off of brand recognition and brand loyalty. Price was also a large driver. The assumption here is that if a consumer has choices, he or she will likely choose the lowest priced item. Another important feature is the "countforcarts" column, suggesting the higher number of items in a consumer's cart will lead to a purchase.



We ran the neural network through 1% of the entire dataset, and we found the accuracy was around 40%; in order to improve accuracy of this model, by rescaling the dataset and release less correlated columns, we finally got improved result which is 91%, and we thought this could be better if it ran through larger dataset or create more related predictor for purchase.

As a team we were not completely satisfied with the confusion matrices and output statistics associated with our models produced in R; therefore, we ran the data set through Python. Using the ML pipeline that we created, each model produced the metrics we were looking for in order to rank the classifiers used. The Random Forest Classifier produced the best results out of XGBoost, Logistic Regression and Neural Network.

Random Forest Classifier Results:

- Score - 64%
- TPR - 53%
- F1 Score - 60%
- Precision - 66%

- ROC AUC - 70%

The Random Forest was built using a max number of trees set to 100, and a max depth set to 20. We chose to set max tree and depth because with the Random Forest algorithm to achieve more reliable estimates.

Our result seemed off due to the fact that our dataset was not geared towards predictive analytics. Most of the research being done on this dataset from kaggle was exploratory data analysis. If we took a second pass at the dataset taking everything that we have learned from the first pass, we would know how to feature engineer more appropriately and transform the data better. The dataset could include more data points and be structured better. For example, the view, cart, purchase structure. Additional features like where each purchase was being made from (country or provincial data). In terms of model efficiency and dataset upload time, hosting this project on the cloud would have been a more appropriate approach. As far as modeling metrics goes, redesigning the dataset to be more machine learning friendly.

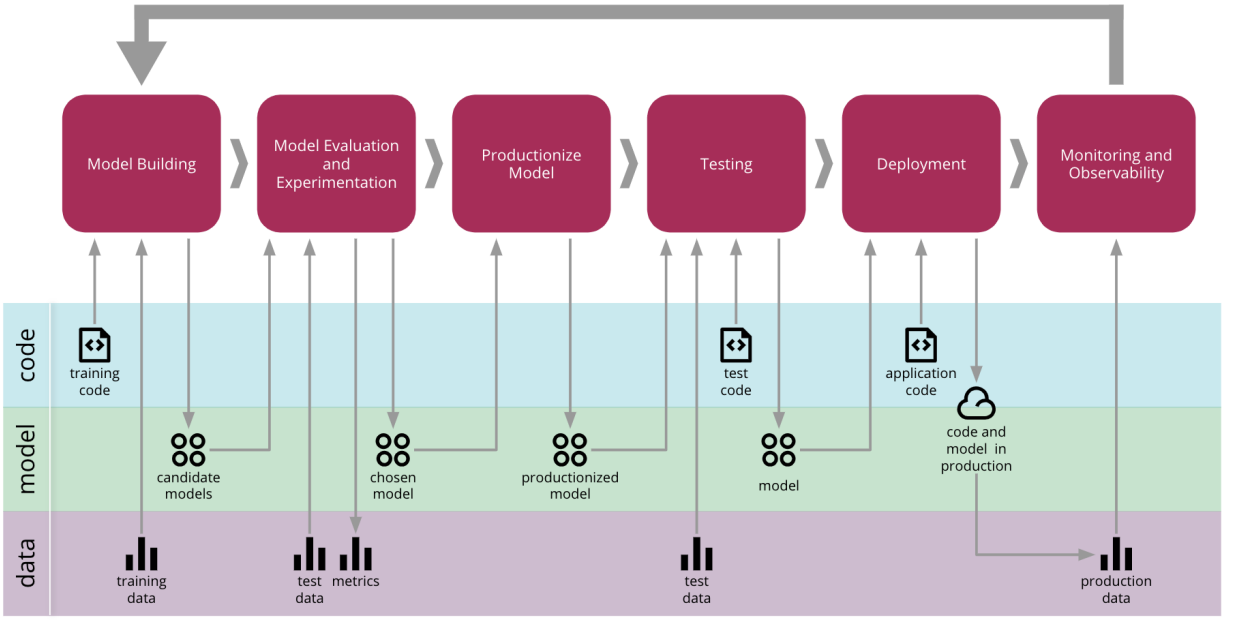
- ROI?

The benefit to identifying predicted purchasers would allow companies to dive into target marketing, knowing what brand and price range a consumer favors. If a consumer is set on purchasing a specific brand, a higher priced phone could be pushed on them knowing they are brand loyal. There could be marketing campaigns created to persuade a consumer to purchase two phones at once or give them a percent discount on their next visit to that ecommerce site. Customer churn is a very important metric in any retail space so this data could be used to help build out a loyalty program.

    - Benefit of identifying “predicted” non-purchasers

- Could pushing additional product views change behavior?
- Could marketing strategies change behavior? i.e. remarketing, promotion timelines?

There are many ways to deploy a ML model depending on what type of scalability you want to have. AWS, GCP, Azure, Flask and Heroku are just some of the options. Below is an End to End deployment model:



Once the model is deployed there will be maintenance involved. This can include monitoring the model, updating code, dashboards for client end use, or even a streaming data solution where end users are able to see realtime updates as data is fed through the pipeline.

- Are there important ethical considerations?
    - Do not compromise user privacy through data-driven marketing strategies
    - Keeping user information secure when handling deployment applications
- Identify the risks associated with your proposed plan and how you would mitigate them

- Potential for low predictive value across models
- Useful dataset for Exploratory Data Analysis; however, significant preprocessing, subsetting, upsampling, and re-balancing is necessary to perform predictive analysis.