Classifying Fingerhut Customer Purchasing Intent

Nicholas Davidson[1*†], Ryan Dunker[1†], Andra Velea[1†], Darren Sohn[1†]

[1*]UCLA Department of Mathematics.

[†]These authors contributed equally to this work.

Abstract

Executive Summary: Our value proposition to Bluestem Brands focuses on real-time prediction of customer purchasing behavior based on their initial website interactions, including page visits, events triggered, and other on-site activities. Our models are designed to provide insights into the likelihood of a user making a purchase on the site. This information is crucial for Bluestem Brands, as it enables targeted advertising and promotional efforts toward potential customers who may need an extra nudge to convert. Overall, we hope to provide Bluestem with a framework that will allow them better understand their customer base and provide them with the tools to make key business decisions.
Results: We began by evaluating several classification models to identify buyers and non-buyers, including logistic regression, XGBoost, and Support Vector Machines (SVM). Logistic regression served as our baseline model, achieving an accuracy of 81%. However, it struggled to correctly classify purchasing customers. While the XGBoost model achieved an accuracy of 91%, it faced challenges in identifying non-buying customers. Ultimately, the SVM model proved to be the most effective, with a 95% accuracy and an ability to classify both buyers and non-buyers, thanks to its capacity to handle non-linear relationships. Additionally, we employed a Long Short-Term Memory (LSTM) model to predict users' next clicks after their first four interactions. With an accuracy of 92%, this model was chosen for its ability to analyze and predict time-indexed sequences. LSTMs have been shown to outperform traditional Recurrent Neural Networks (RNNs) as they can effectively model both long-term and short-term dependencies.
Conclusion: By training these models offline leveraging cloud computing, we can deploy them in real-time using AWS SageMaker or Google Vertex AI, enabling live predictions of users' future clicks and potential purchases based on their browsing activity during a session. This approach offers valuable insights and opportunities for Bluestem Brands to enhance customer engagement and conversion rates.

1 Introduction

Bluestem Brands is interested in exploring user behavior based on their browsing data. One of the central research questions for the company revolves around the possibility of predicting customer conversion based on their initial clicks. Gaining insights into this aspect could prove invaluable for Fingerhut, a Bluestem Brands subsidiary, as it would enable them to deploy targeted advertisements within the browser, capturing the attention of users who may be hesitant about completing a purchase.

By accurately predicting customer conversion with a high degree of precision, Fingerhut would be better positioned to create tailored marketing strategies that resonate with its audience. Such personalized campaigns could lead to higher conversion rates, improved customer satisfaction, and increased return on investment (ROI) for their advertising efforts. Continuous evaluation and optimization of these data-driven strategies would ensure that Fingerhut remains adaptive to changing consumer behaviors and market dynamics. [1]

2 Data

Bluestem provided us with a week's worth of data from December of 2022. Just one day from this data set consists of nearly 42 million rows containing 283 features and approximately around 850 thousand unique users that entered the site. Each row corresponds to a new server call each user takes on the Bluestem page. For example, clicking on the preview button for an item is an action performed by the user and would therefore have a corresponding row in the data set.

Each action on the Bluestem site is recorded via Javascript and numerous features and attributes are collected. Each day varies in the number of actions performed by users on Bluestem's site; however, the number of columns or features is maintained and consistent each day.

2.1 Exploratory Data Analysis

One significant issue that exists in the e-commerce world is that many users tend to bounce (view one page and leave). As we can see in the graph below, about 60% of users in one day of December had bounced which creates a difficult task of predicting the user's behavior when they give Bluestem Brands such little data to work with.

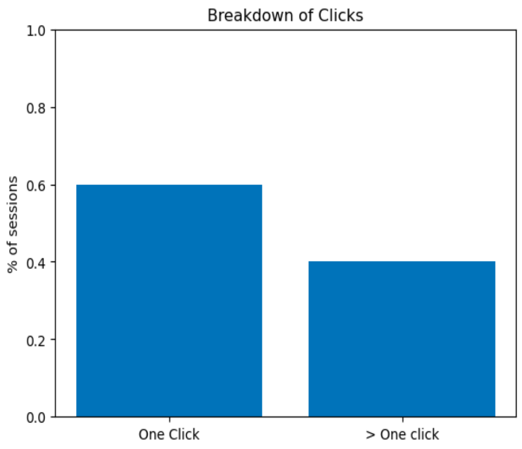

Fig. 1: Breakdown of Number of Clicks

Given the tokenized format of event list and the difficulty needed to sequence it, we decided to look into the variable: PageName. Below is a histogram showing the number of clicks for the top ten PageName's visited by users on Day 0. From the figure, we see a broad range of page names visited including searches, checkouts, and products as well. Women's Shoes and Toys interestingly had high visibility which could be subject to further investigation.

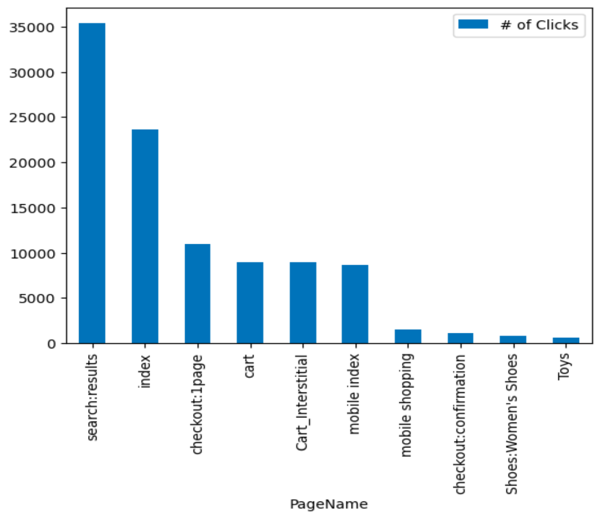

Fig. 2: Top 10 PageNames visited

We decided to take it one step further from the graph above and explore the first page visited by customers who ended up converting. In other words, we selected the top ten Pagenames visited first for customers who ended up purchasing items. Overall, it proves to be quite similar to Fig. 2, but we do see cart and credit application. Cart implying that they are a returning user who left the site and returned later to complete purchase and credit application indicating they are seeking approval.

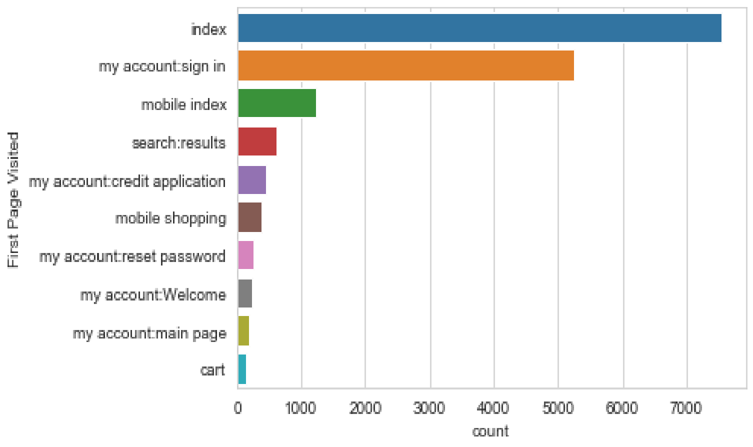

Fig. 3: First PageName visited for Converted Customers (Top 10)

One statistic provided in the data that was worth exploring was 'prop29' which seems to be Bluestem Brands native customer segmentation based on user history.

Due to the fact that segmentation is mainly based on a history of previous data, users who bounce or don't access the site more than once are more likely to be labeled as 'None'. Despite this, we think that the Bluestem Brands native customer segmentation can bring about a lot of insight. It should be noted that since a large portion of the data was removed (those with 'None' attribute), the conversion rate is of a subsetted user base and not indicative of how the company is able to convert users overall.

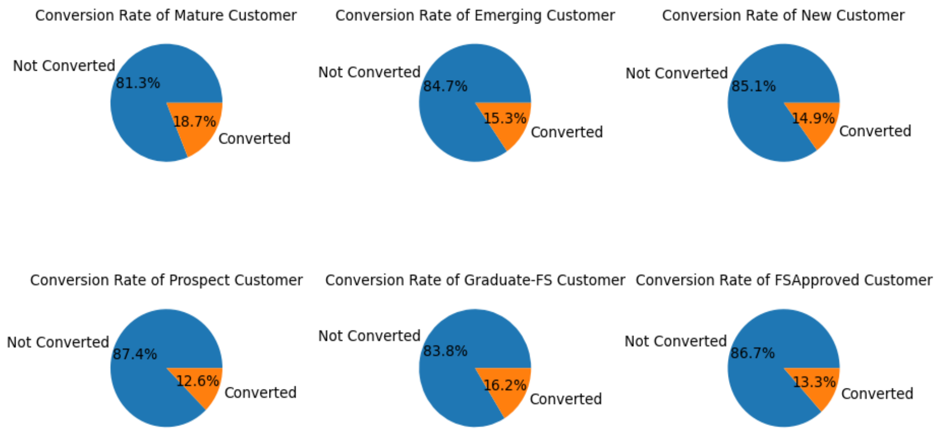

Fig. 4: Conversion Rate by Customer Segmentation

We know that "Mature" customers are those who have been improving their credit score and eventually stop shopping with the company. This tells us that users are successfully purchasing from this site and improving their credit score to eventually not be a credit-based user which is demonstrated in 4. However these segmentations are not telling of how we can approach marketing to the individual. For the most part, each of these groups seem to be converting at a similar rate and our goal is to possibly create new ones that work in conjunction with the existing ones to understand the user and their tendencies.

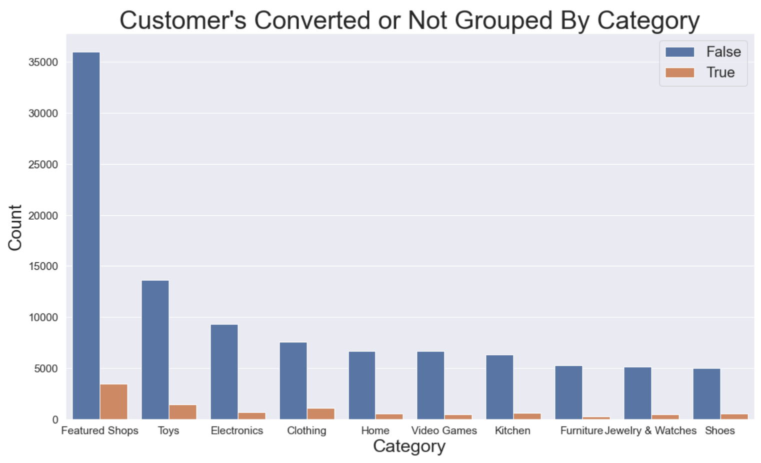

Fig. 5: Conversion Statistics by Category

Another potential statistic to look into as a predictor or hyperparameter in our predictive models would be the category the users were shopping for. We can infer from **??** that many users are initially drawn to the Featured Shops as opposed to any other category. Also since the time frame of this data set was mainly around the holiday season, we can infer that many users may be parents and families shopping for their children as we can see that toys and electronics were the next most visited category aside from featured shops. Whether or not where the user enters first is to be determined in our predictive models.

Ultimately, it seems that the category the user visits does not have much effect on whether or not a user ultimately converts on a purchase. It is also evident that the data set is skewed heavily toward customers who ended up not making a purchase.

2.2 Data Cleaning

The dataset that was given to us was not ready for production-ready models. As mentioned during our **EDA** a majority of users bounce when visiting Fingerhut's site and offer no real value when trying to build out classifiers and predictive models. Therefore, we decided to remove from the data all users who bounced.

We decided to index the data such that each user session was it's own record or row. To do this we indexed the data given to us by the unique identifier given to each site user (visitid), their session start time (visitstarttimegmt), and their actions taken during the session (pagename).

We then selected features that we all believed could have a potential impact on customer purchase intent and filtered them out to best fit the models. The user's product category, their initial page that they started the session with, their account engagement score, what products they have viewed, lifetime order total, search terms, metatab navigation, their current time in the sequence of events, if they used an autosuggest interaction, the product review count, their last order recently, and then the indexed columns. In addition, we removed all rows of data that included NAs.

Table 1: Features Used

No.	Feature
1	Category
2	First Hit Pagename
3	Pagename (excluding checkout)
4	Account Engagement Score
5	View Product List
6	Remove Email Subscriptions
7	Life Time Order Totals
8	Search Terms
9	Megatab Navigation
10	Hit Time
11	Auto Suggest Interactions
12	Product Review Count
13	Order Recency
14	VisitID
15	Visit Start Time
16	Event List

By removing inconsistencies, missing values, and irrelevant data, the refined dataset provides a reliable foundation for extracting meaningful insights and patterns, ultimately leading to better decision-making and optimized marketing strategies.

3 Models

3.1 Classification Models

To begin our analysis, we explored multiple classifications models to determine whether a user will convert or not convert. We were also able to utilize feature importance methods to understand which features are most unique and impactful when classifying conversion.

3.2 Logistic Regression

Predicting purchasing intent based on user web browsing behavior is an important task in the field of online marketing and e-commerce. Logistic Regression is a supervised machine learning algorithm suitable for this binary classification problem due to its ability to model the relationship between a categorical dependent variable and one or more independent variables, handle high-dimensional data, and provide interpretable results.

Logistic Regression models the probability of an event (e.g., purchasing intent) occurring by using a logistic function, which is an S-shaped curve that maps any real-valued input to a value between 0 and 1. The logistic function is defined as follows:

$$p(x) = \frac{1}{1 + e^{-z}}, \tag{1}$$

where $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$ is the linear combination of features $x_1, x_2, \ldots, x_n$ and their corresponding coefficients $\beta_1, \beta_2, \ldots, \beta_n$, and β_0 is the intercept.

The model is trained by estimating the coefficients $\beta_0, \beta_1, \ldots, \beta_n$ using the labeled training data. The goal is to find the values of these coefficients that maximize the likelihood of the observed data. This is typically achieved through an optimization technique called Maximum Likelihood Estimation (MLE).

The likelihood function is given by:

$$L(\beta) = \prod_{i=1}^{m} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}, \tag{2}$$

where m is the number of training samples, and y_i is the class label of the i-th sample. To simplify the optimization problem, the natural logarithm of the likelihood function, called the log-likelihood, is used:

$$\ell(\beta) = \sum_{i=1}^{m} \left[y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i)) \right]. \tag{3}$$

The optimization is usually performed using iterative methods such as Gradient Descent.

3.2.1 Why Logistic Regression for Predicting Purchasing Intent

Logistic Regression is well-suited for binary classification problems, as it models the probability of an event (e.g., purchasing intent) occurring.

User behavior data can be high-dimensional, consisting of numerous features. Logistic Regression is capable of handling high-dimensional data by using regularization technique, we used Ridge (L2), to prevent overfitting and improve model performance.

The coefficients of the logistic regression model can be easily interpreted as the odds ratios, which quantify the relationship between the independent variables and the dependent variable. This makes it easier for marketers and e-commerce professionals to understand the impact of different features.

3.2.2 Logistic Regression Implementation

To begin with our Logistic Regression model, we first look at our input data. In our final logistic regression model, we utilized the following features: clickaction, pagename, post_evar7, firsthitpagename, post_evar22, post_evar46 which would predict the checkoutthankyouflag feature. As a reminder, post_evar7 represents how the user found the product, post_evar22 represents the number of product reviews they have, and post_evar46 represents the number of lifetime orders.

For each of the categorical predictors: clickaction, pagename, post_evar7, and firsthitpagename, we utilized the TargetEncoded function from the Categories Encoded package [2] which replaces the missing values with the mean of the target values of that category. By target encoding, we ensure that we were using a full dataset and allows for a faster training of the model. As for the numerical predictors, we replaced all NA values with 0 again to have a complete dataset and for simplicity.

Using the LogisticRegression function from sklearn.linear_models [3] as well as the train test split function [4] where we exhibited a train test split of 80/20, we achieved a model accuracy of score of 81%. While this accuracy seems high considering the complexity of the data, it would be short sighted to assume this high accuracy means we've developed a great model.

The main issue stems from the recall score which was only 34%. The recall score for logistic regression represents the false negatives meaning that it had a hard time classifying buyers. This is most likely due to the class imbalance, meaning the model can not handle the large differential between buyers and non-buyers and simply classifies most people as non-buyers.

Looking into the features of the model, we start to gain insight as to what predictors prove to be significant. In 6, we can see that clickactionencoded and pagenameencoded have high values of 0.0030 and 0.0015. The way we interpret these values we can look to [5] which states that it represents the decrease in model score when a feature value is randomly shuffled. We take note of these predictors and look towards how they will affect both our future classification and predictive models.

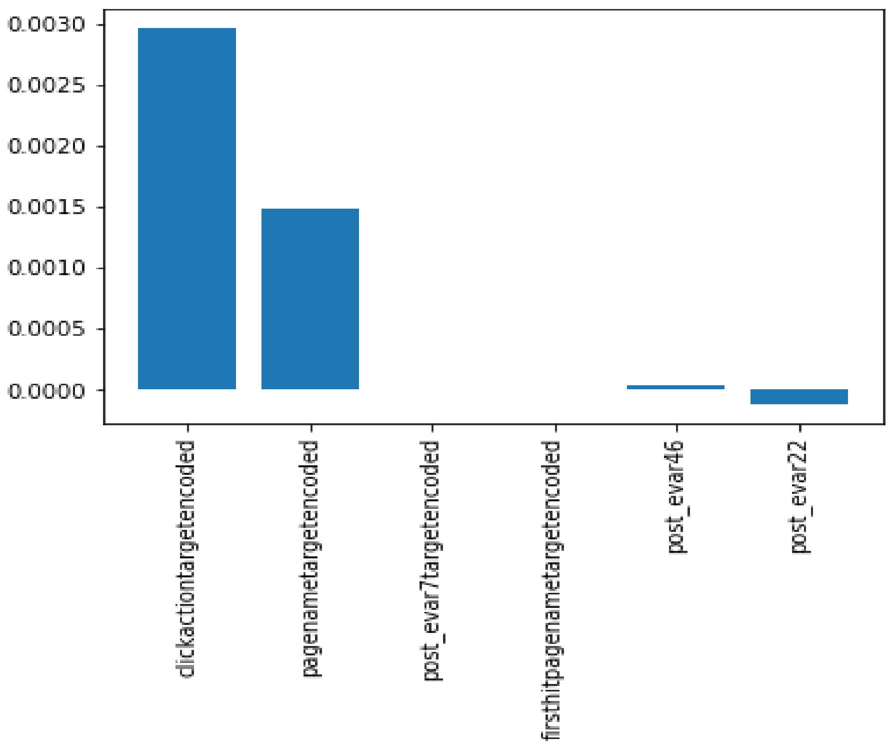

Fig. 6: Permutation Importance of Logistic Regression Features

3.3 XGBoost

Extreme Gradient Boosting (XGBoost) is a supervised machine learning algorithm able to handle high-dimensional data and missing values, optimize regularization techniques, and prevent over fitting. XGBoost is an enhanced version of the gradient boosting technique which uses decision trees as the base model. The algorithm is iterative by adding new decision trees that attempt to correct the errors previous decision trees made to a model.

XGBoost takes labeled training data as its input, consisting of feature vectors and their corresponding class labels, and begins with a single decision tree that makes predictions based on the input features. The results of this tree are compared to the class labels (actual target value) by calculating the errors. The subsequent decision tree aims at ameliorating the errors made by the first by adjusting the weights on the same instances the first tree predicted incorrectly. As such, this gives more importance to the incorrectly classified features. This second tree is then added to the first tree, and the predictions of both are combined to return a more accurate overall prediction. The process continues in an iterative fashion of predicting and calculating errors until either a specified number of trees have been built or the prediction error stops improving compared to a certain threshold. XGBoost also applies L1 and L2 regularization techniques during the training process.

3.3.1 XGBoost Math

Let y_i be the target value for the i-th instance, and $\hat{y}_i^{(t)}$ be the prediction for the i-th instance at the t-th iteration. The objective function to minimize is:

$$L(y, \hat{y}^{(t)}) = \sum_{i=1}^{N} l(y_i, \hat{y}_i^{(t)}) + \Omega(f_t), \tag{4}$$

where $l(y_i, \hat{y}_i^{(t)})$ is a differentiable loss function, $\Omega(f_t)$ is a regularization term, and f_t is the weak learner added at the t-th iteration.

3.3.2 Regularized Learning Objective

In XGBoost, the objective function includes an additional regularization term that penalizes the complexity of the model, which helps prevent overfitting:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2, \tag{5}$$

where T is the number of leaves in the tree, w_j is the weight assigned to the j-th leaf, γ is a complexity control parameter, and λ is the L2 regularization term.

3.3.3 Additive Training

At each iteration t, a new weak learner f_t is added to the existing model to minimize the objective function:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i), \tag{6}$$

where $\mathbf{x}_i$ is the feature vector for the i-th instance.

The optimal weak learner f_t is found by solving the following optimization problem:

$$f_t = \arg\min_f \sum_{i=1}^{N} l(y_i, \hat{y}_i^{(t-1)} + f(\mathbf{x}_i)) + \Omega(f). \tag{7}$$

3.3.4 XGBoost Implmentation

Bluestem Brands's data is high-dimensional and riddled with missing values, making it a prime candidate for XGBoost. The accuracy of the algorithm alongside its overall robustness and intrinsic regularization techniques to combat overfitting make it a powerful and useful tool for scenarios like this binary classification problem. Additionally, XGBoost includes feature importance scores and other diagnostic tools to help evaluate the model's performance and interpretability.

For these reasons, we utilized XGBoost as one of our classification models in the R programming language via tidymodels [6][7][8]. Since there were many intermittent models, we choose to only focus on the discussion of the very last XGBoost model run. This model, which will further be referred to as “xgb”, included 11 predictor variables: myaccountengagement, post_evar46, post_evar22, num_events_triggered, post_evar30, post_evar7, firsthitpagename, evar83, formanalysis, visitpagenum, and lastpurchasenum. Respectively, each variable corresponds to: user cookie count, number of user lifetime orders, number of product reviews for a specific product, number of events triggered per shopping session, order recency, how a user found a product, first page a user clicks on, user behavior score, user SafeLine eligibility, and although the last two did not have an associated description, they were significant variables. The only predictor resulting from manual manipulation is “num_events_triggered”, calculated from the original “eventlist” variable by unlinking the nested lists via regular expressions and counting the length of each subsequent vector. The response variable is “checkoutthankyouflag.” These variables were chosen through random forest variable importance plots via the varImp() function in the caret package and manual selection based on intuition and data exploration [9]. The goal for the model was to use the chosen features as a way to see if they could correctly classify a customer's buying habits.

The data used here come from only day 6 (December 16th, 2022) and were split into testing and training sets with stratified sampling across "checkoutthankyouflag" at a proportion of 0.75, the default. Stratified sampling was used because of the imbalance in the response variable; there were many more non-buyers than buyers. Due to processing and computational power, xgb could only be run on individual days' worth of data, as combining the entire week took too long to compile all at once.

To optimize the output, xgb utilized 3-fold cross-validation with tuned hyperparameters *mtry*, *trees*, and *scale_pos_weight*. The range for the first hyperparameter was between one and ten, meaning the algorithm will consider up to ten randomly selected variables at each decision tree split. The latter two will use default tuning ranges. Overall, tuning these hyperparameters allowed for the algorithm to yield the best results without too much compilation time, hence the low number for cross-validation and small tuning range for *mtry*.

For xgb's recipe, we conducted some transformative steps for all variables. Table 2 shows the steps and their definitions. For the sake of simplicity, we did not look into any interaction effects and focused instead on variable significance. Some of the nominal (character or factor) variables, in particular "firsthitpagename", had high cardinality, so many of the steps included reducing the dimensionality of the dataset for easier handling.

Xgb resulted in around a 91.5% accuracy score with tuned hyperparameter values $mtry = 9$, $trees = 278$, and $scale\_weight\_pos \approx 1.06$. From the confusion matrix in Figure 7, we can calculate the precision and recall scores. Respectively, the values are approximately 3.56% and 91.8%. The low precision in tandem with high recall indicates xgb had an issue with misclassifying non-buyers, i.e., there were lots of false positives. To combat this issue, our team tried different encoding techniques for the high cardinality nominal variables such as using likelihood scores derived from a generalized linear model (step_lencode_glm() in embed package in R [10]) and a generalized linear mixed model (step_lencode_mixed() [11]); however, these did not produce any significantly better results in terms of precision. Additionally, although xgb was only run on day 6 of the data, the other days were comparable in terms of accuracy, precision, and recall scores.

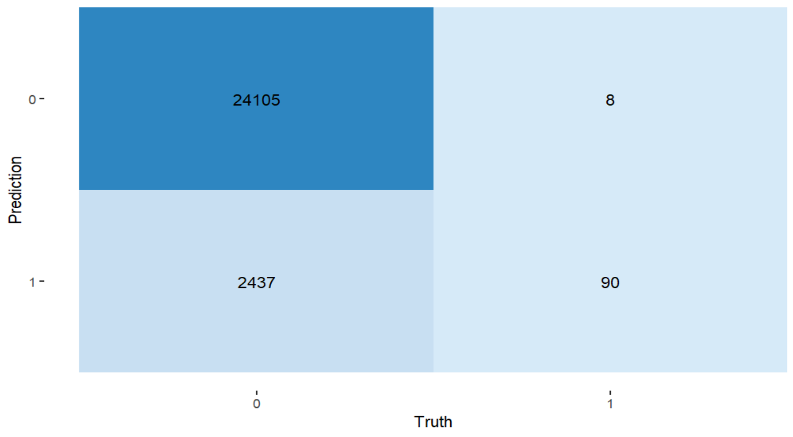

Fig. 7: Confusion matrix for xgb.

Figure 8 shows the variable importance plot in terms of the SHAP (SHapley Additive exPlanations) values. To preface the interpretability of this plot, the SHAP value, in layman's terms, shows the contribution/importance of each feature on the prediction of the model, but it does *not* evaluate the quality of the prediction itself [12]. In technical terms, the SHAP value measures the difference between the predicted value for the instance with and without the contribution of each feature. It is then calculated as the average of the contributions across all possible permutations of features. The SHAP plot in Figure 8 shows a range of values for the features, where yellow is low and purple is high. For binary variables, 0 or false will be low and 1 or true will be high. The spread of the values also indicates how important that variable is; the SHAP plot puts the most important feature in the model at the top, and the further the spread the more important that type of value is relative to the other values of the variable. [13].

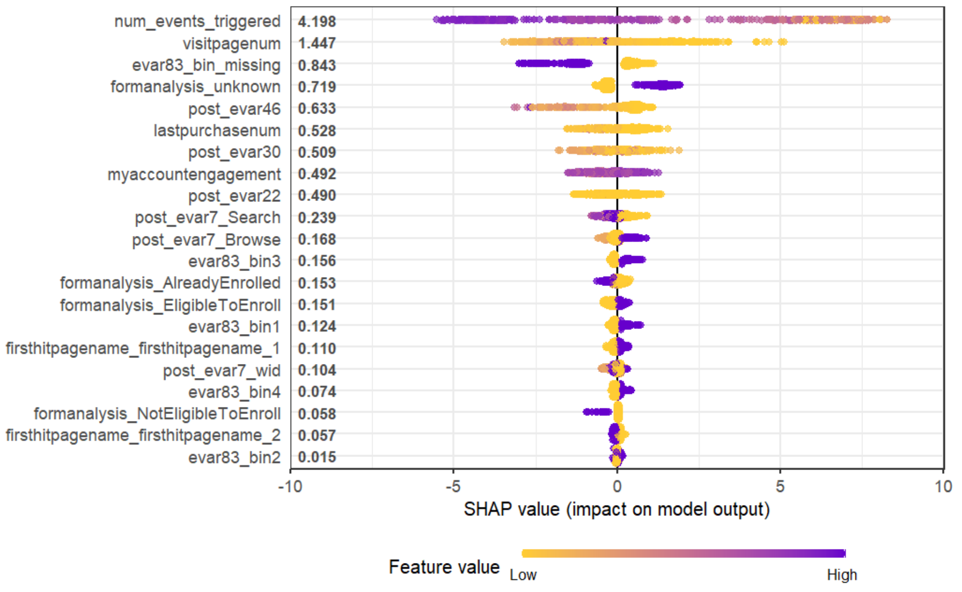

Fig. 8: Feature importance graph for xgb.

To interpret some of the findings from xgb, Figure 8 shows that higher "num_events_triggered" corresponds to a negative SHAP value, which is a negative impact on model output, meaning the user does not buy. In fact, it seems that an average number of events triggered (more orange-like shade) corresponds to a customer completing a purchase. Moreover, for "formanalysis_NotEligibleToEnroll", which is the "Not Eligible To Enroll" level of the "formanalysis" variable, the true value (customer is not eligible to enroll) means they are more likely not to make a purchase, where a false value (customer is not not eligible to enroll) means they are more likely to buy. Applying this logical interpretation to the rest of the plot could yield some helpful information for Bluestem Brands should they choose to investigate these potential relationships. [14] [15] [16].

Table 2: Recipe Steps and Definitions [17]

Step	Definition
step_downsample(checkoutthankyouflag)	Downsamples the majority class (non-buyer) to account for class imbalance
step_novel(firsthitpagename, post_evar7, formanalysis)	Assigns a previously unseen factor level to a new value for each variable
step_unknown(firsthitpagename, formanalysis)	Assigns a missing value in a factor level to "unknown"
step_other(starts_with("firsthitpagename"), starts_with("formanalysis"))	Pools infrequently occurring values into an "other" category
step_discretize(evar83)	Converts evar83 into a factor with bins having approximately the same number of data points
step_collapse_cart(starts_with("firsthitpagename"), outcome = vars(checkoutthankyouflag), skip = T)	Collapses factor levels into a smaller set using a supervised tree
step_dummy(all_nominal_predictors(), one_hot = T)	Creates traditional dummy variables from non-numeric variables and one-hot encodes them
step_zv(all_predictors())	Zero variance filter: removes variables that contain only a single value
step_normalize(all_predictors())	Normalizes numeric data to have a standard deviation of one and a mean of zero
step_impute_knn(all_predictors())	Imputes missing data using k-nearest neighbors
step_center(all_predictors())	Centers numeric data

3.4 SVM

Support Vector Machines (SVM) is a supervised machine learning algorithm suitable for classification problems due to its ability to handle high-dimensional data, manage noise and outliers, and capture non-linear relationships.

SVM takes labeled training data as input, consisting of feature vectors and their corresponding class labels. In the context of user web browser behavior, features can include time spent on different webpages, click-through rates, browsing history, etc., while class labels represent purchasing intent. The goal is to find a hyperplane that separates data points of different classes with the largest possible margin.

Margin is defined as the distance between the hyperplane and the closest data points from different classes, known as support vectors. SVM aims to maximize the margin to improve the generalization performance of the classifier.

For non-linearly separable data, SVM employs the kernel trick, which involves mapping input data points into a higher-dimensional space where they become linearly separable. Popular kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels.

SVM incorporates regularization to prevent overfitting. The regularization parameter (C) determines the trade-off between maximizing the margin and minimizing the classification error. A smaller C value produces a wider margin with more misclassifications, while a larger C value results in a narrower margin with fewer misclassifications.

3.4.1 Why SVM for Predicting Purchasing Intent

User behavior data can be high-dimensional, consisting of numerous features. SVM is suitable for this classification task due to its ability to handle high-dimensional data and find the optimal separating hyperplane.

Predicting purchasing intent is a binary classification problem, as users are either classified as having purchasing intent or not. SVM is well-suited for binary classification problems.

User behavior data can be noisy, with outliers and irrelevant features. SVM's regularization parameter (C) helps manage noise and outliers by controlling the trade-off between maximizing the margin and minimizing misclassifications.

User behavior data may have non-linear relationships among features. SVM can capture these non-linear relationships using the kernel trick, enabling better classification performance.

SVM is known for its robustness and generalization capabilities, which are essential when dealing with complex, real-world data, such as using web browser behavior.

3.4.2 Non-linearly Separable Case and Soft Margin

For non-linearly separable data, we introduce a soft margin to allow some misclassifications. We add slack variables $\xi_i \geq 0$ for each data point:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i. \tag{8}$$

The optimization problem becomes:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i, \tag{9}$$

subject to the constraints:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i,$$

$$\xi_i \geq 0 \quad \forall i.$$

The parameter $C > 0$ controls the trade-off between maximizing the margin and minimizing the classification error. Larger C values put more emphasis on minimizing misclassifications, while smaller C values allow for more misclassifications in favor of a larger margin.

3.4.3 Kernel Trick

To handle non-linear decision boundaries, SVM employs the kernel trick, which implicitly maps input data to a higher-dimensional feature space using a kernel function $K(\mathbf{x}, \mathbf{x}')$. The most common kernels are:

- Linear kernel: $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
- Polynomial kernel: $K(\mathbf{x}, \mathbf{x}') = (\gamma \mathbf{x}^\top \mathbf{x}' + r)^d$
- Gaussian (Radial Basis Function, RBF) kernel: $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma |\mathbf{x} - \mathbf{x}'|^2)$
- Sigmoid kernel: $K(\mathbf{x}, \mathbf{x}') = \tanh(\gamma \mathbf{x}^\top \mathbf{x}' + r)$

where γ, r, and d are kernel-specific parameters.

3.4.4 Dual Problem

The optimization problem can be reformulated as a dual problem, which is more convenient for solving SVM with kernels. To do this, we introduce Lagrange multipliers $\alpha_i \geq 0$ for each constraint, and construct the Lagrangian:

$$L(\mathbf{w}, b, \xi, \alpha) = \frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i [y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i]. \tag{10}$$

The dual problem is obtained by minimizing the Lagrangian with respect to $\mathbf{w}$, b, and ξ, and then maximizing with respect to α:

$$\max_{\alpha} \min_{\mathbf{w}, b, \xi} L(\mathbf{w}, b, \xi, \alpha). \tag{11}$$

The solution to the dual problem has the form:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}j), \tag{12}$$

subject to the constraints:

$$\sum i = 1^N \alpha_i y_i = 0, \ 0 \leq \alpha_i \qquad \leq C \quad \forall i.$$

This is also a quadratic programming problem, which can be solved using specialized algorithms, such as the Sequential Minimal Optimization (SMO) algorithm.

Once the dual problem is solved, the weight vector $\mathbf{w}$ and bias term b can be recovered:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \phi(\mathbf{x}i), \ b \qquad = \frac{1}{N\text{SV}} \sum_{i \in \text{SV}} \left(y_i - \sum_{j \in \text{SV}} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \right),$$

where $\phi(\mathbf{x}_i)$ is the implicit mapping function associated with the kernel, and SV denotes the set of support vectors. Note that only support vectors ($\alpha_i > 0$) contribute to the computation of $\mathbf{w}$ and b.

For a new data point $\mathbf{x}_{\text{new}}$, the prediction is made based on the sign of the decision function:

$$f(\mathbf{x}_{\text{new}}) = \text{sign}\left(\sum i \in \text{SV}\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_{\text{new}}) + b\right). \tag{13}$$

3.4.5 SVM Implementation

We computed SVM using the same steps as our logistic regression pipeline. This means we used the following features we utilized the following features: clickaction (with checkout removed), pagename, post_evar7, firsthitpagename, post_evar22, post_evar46 which would predict the checkoutthankyouflag feature. Post_evar7 represents how the user found the product, post_evar22 represents the number of product reviews they have, and post_evar46 represents the number of lifetime orders. The categorical variables were target encoded, which is described below.

Target encoding, also known as mean encoding, is a technique used to convert categorical variables into numerical values based on the mean of the target variable for each category. It is particularly useful when dealing with high cardinality categorical features.

1. For each category in the categorical feature, calculate the mean of the target variable.

   Let C_i be the i^{th} category, n_i be the number of samples with the i^{th} category, $y_{i,j}$ be the target value of the j^{th} sample in the i^{th} category, and m_i be the mean of the target variable for the i^{th} category. Then,

$$m_i = \frac{\sum_{j=1}^{n_i} y_{i,j}}{n_i} \tag{14}$$

2. Replace the original categorical values with the calculated mean values.

   For each sample s with the category C_i, replace the categorical value with the mean m_i.

Utilizing SK-learn's implementation of SVM, we utilized a Radial Basis Function kernel to do the kernel trick and balanced the class weightings such that the number of users who checked out and didn't was even. This would help prevent the model from classifying only nonpurchasing customers and having poor precision and recall as the prior methods did. We achieved an accuracy of 95% with our SVM model with above 50% precision and recall when predicting whether a customer would make a purchase utilizing the features described above.

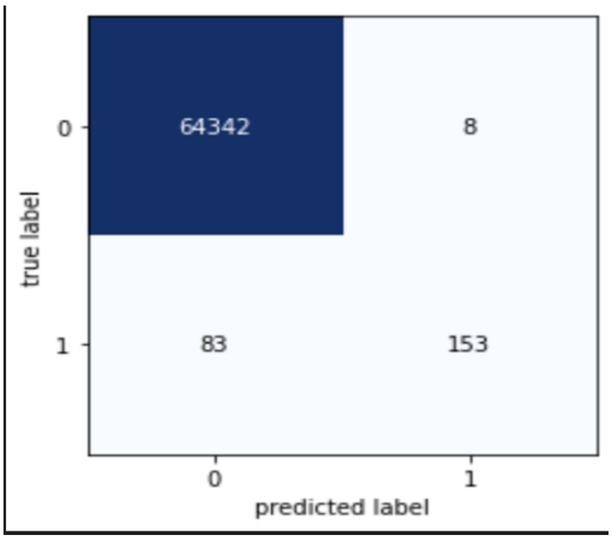

Fig. 9: Confusion Matrix for SVM. 0 Represents Non-Purchasing Customers while 1 represents Purchasing Customers. As you can see the model correctly classified both purchasing and nonpurchasing customers better, but does substantially better at predicting non purchasing customers.

3.5 Predictive Models

To complement our classification models we wanted to develop a predictive model to predict where a user might move to next on Fingerhut. In order to do so, we selected a Long Short-Term Memory network based on prior knowledge of its capabilities, but further research into other models would be beneficial for future study.

3.6 LSTM

Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN), are suitable for this sequence prediction problem due to their ability to learn long-term dependencies, handle variable-length input sequences, and model complex relationships between input features and the target variable.

LSTM networks consist of memory cells, which store and manipulate information over time, and gates that control the flow of information. Each memory cell has three gates: input, forget, and output gates. These gates allow LSTMs to learn long-term dependencies by selectively remembering or forgetting information.

LSTM networks are trained using backpropagation through time (BPTT), an adaptation of the backpropagation algorithm for RNNs. The objective is to minimize the loss function, which measures the difference between the predicted and actual next-page navigation. Common loss functions include mean squared error (MSE) and categorical cross-entropy.

3.6.1 Vanishing and Exploding Gradient Problem

Traditional RNNs suffer from the vanishing and exploding gradient problem during training, which makes it difficult for them to learn long-term dependencies. LSTMs were introduced to address this issue by incorporating a memory cell and a set of gating mechanisms that enable the network to selectively retain or forget information.

3.6.2 LSTM Architecture

An LSTM unit consists of the following components:

- Memory cell: c_t
- Input gate: i_t
- Forget gate: f_t
- Output gate: o_t
- Hidden state: h_t
- Input vector: x_t

The LSTM unit updates its internal state using the following equations:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), && (15)\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), && (16)\\
c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), && (17)\\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), && (18)\\
h_t &= o_t \odot \tanh(c_t), && (19)
\end{aligned}
$$

where $\sigma(\cdot)$ denotes the sigmoid function, $\tanh(\cdot)$ denotes the hyperbolic tangent function, $\odot$ denotes the element-wise product, and W and b denote weight matrices and bias vectors, respectively.

3.6.3 Gating Mechanisms

The input, forget, and output gates control the flow of information in the LSTM unit:

- Input gate (i_t): Controls the amount of new information added to the memory cell.
- Forget gate (f_t): Controls the extent to which the previous memory cell state is retained.
- Output gate (o_t): Controls the amount of information from the memory cell that is passed to the hidden state.

These gating mechanisms enable the LSTM to selectively remember and forget information, which helps address the vanishing and exploding gradient problem.

3.6.4 LSTM Applicability to Predicting Next Page Navigation

LSTM networks are specifically designed for sequence prediction problems. In this case, the input sequences are the first four pages visited by users, and the goal is to predict the next page the user will navigate to.

The choice to use the first four clicks as our input and the fifth click as our output mainly stems from Ben's suggestion to utilize the first five clicks. We also extracted and tested on the first four and six clicks, but four clicks was not a long enough sequence for the algorithm to learn effectively and the number of six-click sequences dropped significantly from five-click sequences.

User browsing behavior may exhibit long-term dependencies, where the next page visited is influenced by pages visited earlier in the sequence. LSTM networks can capture these long-term dependencies through their memory cells and gated architecture.

4 Results

4.0.1 Logistic Regression

Logistic Regression overall served its purpose as a base model to learn and build off of. With a considerable accuracy score of 81%, we knew that we needed to build models that would perform better than that which we were able to achieve. We also learned that "pagename" and "clickaction" were fairly important to our classification of buyers and we continued to use these factors throughout or model training.

4.0.2 XGBoost

Overall, due to its low precision, xgb was an invalidated model. There is no real-world use for its classification because of the number of false positives; it ultimately provides no useful insight for Bluestem Brands in terms of its classification measures. Instead, xgb may provide some information about which variables might be worth looking into for future research purposes, such as "num_events_triggered" and "visitpagenum."

4.0.3 SVM

SVM was ultimately our best classification model trained. We reached a 95% accuracy on classifiers accompanied by 95% precision and 65% recall. Although this preformed much better than other models, we still want to be conscious of the fact that it only encapsulates a days worth of data and a heavily imbalanced dataset. Further testing would be needed to ensure its validity.

4.0.4 LSTM

Our LSTM Network did not perform as well as expected and obtained a training accuracy of only 59% with a test accuracy of 33% when predicting the next action of the user based on the first four clicks. Despite not being very accurate, it is important that further study should be done on the neural network. Tuning the number of events in a sequence could allow for more predictive power beyond the first five clicks. Additionally, it would be beneficial to experiment with adding additional layers to the model to prevent over fitting.

5 Discussion

Through this report, we attempted to create and tune two models, one predictive and one classification, that would work in conjunction with each other to create a live prediction system that Bluestem brands could use to target non-buyers based on their next click. As for our predictive model, we successfully built and tuned a LSTM model that accurately predicted 92% of user's 5th click based on their first four clicks. As for our classification models, we built out three different ones each with their own strengths and faults. The logistic regression performed decently for a baseline model however, it had troubling classifying buyers and simply labelled most of the users as non-buyers, leading to lots of false negatives. Moving onto our XGBoost model, it had different predictors and a higher accuracy, achieving 91% at its best state, which was significantly better than our baseline model. The faults with xgb was that it had low precision and was not run on the entirety of the dataset. Our final classification model that we want to present to Bluestem Brands would be the SVM which had the best accuracy and dealt with precision and recall. It achieved an accuracy of 95% and we believe it derives it success to the fact that the relationship is most likely not linear and SVM can deal with non-linear relationship by using a non-linear kernel. However we are also cautious that SVM draws meaningful results in that we only tested on one day's worth of data. As for the LSTM model, it works with great accuracy and proudly present it to Bluestem Brands. With more time we would have loved to explore experimenting with different click numbers before predicting but leave that to Bluestem Brands to test if necessary.

6 Conclusion

Value Added and Technical Recommendations. With our research, we believe that we are adding value to Bluestem Brand by providing them a live inference model that can predict whether or not a user after a few clicks on the website will purchase or not. With our given models, they can be trained offline every quarter due to the computational intensity which would then be deployed to a cloud software such as AWS SageMaker. After this, Bluestem Brands can generate real-time predictions on whether a customer will purchase or not based on their first few clicks. There are multiple routes that can be taken from here. Firstly, they could run targeted ads for those that are not likely to purchase in order to get them to convert since our prediction model would already tell them which customers are purchases and spending marketing dollars on them would be lost cost. Another avenue they could take with our results would be recommending products similar to the items they would purchase in their visit. Implementing a k-Nearest Neighbors algorithm or similar clustering algorithms would have to be run in order to determine product groupings. One final suggestion could be altering the SVM to understand it probabilistically as opposed to just buyer or not buyer. There would need to be the use of Platt scaling and we suggest looking to [18] for more information and suggestions.

Future Work. While we are proud with the results and suggestions we have achieved, there are some future work we would look to given more time and resources. To begin we believe that our Support Vector Machines model performed the best for classifying buyers versus non-buyers; However we failed to run our model on the entirety of the given data. If given the time, resources, and data, we would love to test our model on different seasons of the year. As we stated earlier, only about one day's worth of data was accounted for in our model and therefore may not give similar accuracy results when applied to an entire week or a season. In conjunction with more data and time being given, we would also look to run some dimensionality reduction techniques such as Principal Component Analysis (PCA) in order to cut down on computation time while still retaining a majority of the information from the data.

Another improvement we are looking towards would be experimenting with different predictors for both our classification and predictive models. For our classification model, we mainly handpicked our predictors based on intuition. We could look into more experimenting with predictors and possibly using some data analysis methods that would mathematically demonstrate which predictors might add the most value to our models. As for the predictive models, EventList could produce better sequential data as inputs for the LSTM. EventList is more expansive, therefore bringing about more data; However it is ultimately harder to parse and was giving us trouble when running the LSTM model. Ideally we would like to work on this predictor and test its strength in comparison to PageName.

One last improvement for future work would be expanding upon our LSTM model to producing likelihoods of purchasing as opposed to determining whether or not they will purchase in the end. This way we could possibly create new customer groupings that would be beneficial to Bluestem Brands because they can perform A/B testing to see if their current customer groupings is any different than the ones we would propose with our predictive model.

Acknowledgments. We would like to thank our professor, Dr. Mason Porter, for his guidance, as well as our Teaching Assistant, Samuel Christensen.

References

[1] Bruun, S.B.: User-click Modelling for Predicting Purchase Intent. https://arxiv.org/pdf/2112.02006.pdf

[2] Scikitlearn: Target Encoder. https://contrib.scikit-learn.org/category_encoders/targetencoder.html

[3] Scikitlearn: Logistic Regression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[4] Scikitlearn: Train Test Split. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[5] Scikitlearn: Permutation Feature Importance. https://scikit-learn.org/stable/modules/permutation_importance.html

[6] Tidymodels: tidymodels. https://tidymodels.tidymodels.org/

[7] Parsnip: Boosted Trees via XGBoost. https://parsnip.tidymodels.org/reference/details_boost_tree_xgboost.html

[8] R-Bloggers: Using XGBoost with Tidymodels. https://www.r-bloggers.com/2020/05/using-xgboost-with-tidymodels/ (2020)

[9] Caret: Package 'varImp'. https://cran.r-project.org/web/packages/varImp/varImp.pdf (2022)

[10] Embed: Supervised Factor Conversions into Linear Functions using Likelihood Encodings. https://embed.tidymodels.org/reference/step_lencode_glm.html

[11] Embed: Supervised Factor Conversions into Linear Functions using Bayesian Likelihood Encodings. https://embed.tidymodels.org/reference/step_lencode_mixed.html

[12] Trevisan, V.: Using SHAP Values to Explain How Your Machine Learning Model Works. https://towardsdatascience.com/using-shap-values-to-explain-how-your-machine-learning-model-works-732b3f40e137#:~:text=SHAP%20values%20(SHapley%20Additive%20exPlanations,interpretability%20of%20machine%20learning%20models. [Online; accessed 20-March-2023] (2022)

[13] Fisher, H.: Opening the Black Box: Exploring XGBoost Models with fastshap in R. https://www.hfshr.xyz/posts/2020-06-07-variable-importance-with-fastshap/ (2020)

[14] Lundberg, S.: Interpretable Machine Learning with XGBoost. https://towardsdatascience.com/interpretable-machine-learning-with-xgboost-9ec80d148d27 (2018)

[15] Exchange, D.S.S.: How to interpret Shapley value plot for a model? https://datascience.stackexchange.com/questions/65307/how-to-interpret-shapley-value-plot-for-a-model (2019)

[16] Casas, P.: A Gentle Introduction to SHAP Values in R. https://blog.datascienceheroes.com/how-to-interpret-shap-values-in-r/ (2019)

[17] Recipes: Preprocessing and Feature Engineering Steps for Modeling. https://recipes.tidymodels.org/index.html

[18] Arat, M.M.: Probabilistic Output of SVM. https://mmuratarat.github.io/2019-10-12/probabilistic-output-of-svm. [Online; accessed 22-March-2023] (2019)