Predicting Online Shoppers Purchasing Intention using Machine Learning Techniques

Mostafa Kamal
Department of Computer Science
University of Massachusetts Lowell
mostafa_kamal@student.uml.edu

***Abstract*—Online shopping has significantly grown over the years, and understanding user behavior is critical in improving sales and traffic to a website. The recent development of ecommerce has introduced a level of convenience that wasn't readily available before. For instance, amazon provides 1-2 day free shipping for most of their products sold on their website. To deal with the high demand, companies tend to collect a lot of data when a user visits an ecommerce site. This paper employs machine learning techniques to analyze user behavior and predict purchase likelihood. We are going to address this by using machine learning algorithms such as Naive Bayes, Logistic Regression and Neural Network. Our output would be predicting whether a shopper has an intention to purchase or not. We are also going to evaluate our model using F1, accuracy, recall and precision scores.**

***Index Terms*—Naive Bayes, Logistical Regression, Nueral Network, ecommerce, purchasing intention**

I. Introduction

With the recent developments in the digital space, many companies have moved their business online. Ecommerce websites have made it easier for people to browse through a wide range of products, make purchases and have them delivered within days. The convenience, easy access, and the popularity has therefore revolutionized the way we shop. Additionally, just as a real brick-and-mortar store would analyze customers foot traffic to determine product placement, pricing strategies and promotional campaigns, online stores also collect real time information of their visitors to their sites. This helps in optimizing a user's shopping experience by running tailored advertisements, targeted discounts, marketing and ads. By analyzing what causes a user to purchase a product online can lead to more sales and revenue for a company. This can be done by calculating weights by a machine learning model.

In this final project, we are going to utilize the data collected by an ecommerce site and analyze whether a user has the intention of purchase. We are going to use the dataset online shopper purchasing intention from UCI Machine Learning Repository to perform our prediction. The dataset includes both behavioral and technical features such as the types and durations of pages visited, bounce and exit rates, page value, and proximity to special days. These are collected from user interactions and Google Analytics metrics. This in turn help capture user intent and engagement. Additional features include operating system, browser, region, traffic source, visitor type, day of the week, and month, providing a view of user behavior during e-commerce sessions.

II. Related Work

Examples of related work in this field:

Paper 1: Methods to analyze customer usage data in a product decision process: A systematic literature review: This paper talks about how companies can use real customer usage data to make better product decision. It talks about a five stage process and also shows how AI and machine learning can help at each step. [1]

Paper 2: This paper uses a flexible time series regression approach to analyze the impact of various visitor types and other factors. This is done by collecting google analytics data.[2]

III. Description

Dataset: The dataset was formed so that each user does not appear twice. The dataset is collected over a 1 year period so it avoids biases like special day, campaign etc. Because each user in the dataset appears only once, and is unique we can say it's not a time series dataset. The content of the database is collected by Google Analytics. It has 10 numerical features and 8 categorical features.

A. *Exploratory Data Analysis:*

Data Exploration: In our exploratory data analysis, we investigated the dataset for duplicates and null values, visualized the data by performing box plots, histogram plots, pair plots and heatmaps. Additionally we did not remove any duplicates as that data could belong to a single session. By doing so, we observed the following:

- The dataset was highly skewed dataset
- Uneven dataset with 84.5% being False Revenue and 15.5% being True Revenue
- Zero Null values across features and target variables
- 25 duplicate samples

Data Analysis: We observed that a high page value typically relates to a low bounce rate. This could indicate that a user is actively engaged and navigating through the site. On the other hand, low to mid-range page values are associated with higher bounce rates. This suggests that users are leaving the site after viewing the page which could also imply the user isn't engaged enough. Similarly, low to mid range page values are associated with higher exit rates while mid to high corresponds to lower exit rates. In summary, higher page values tend to result in lower bounce and exit rates, while lower page values are linked to higher bounce and exit rates. We can see that these engagement metrics are also related to revenue in the heatmap.

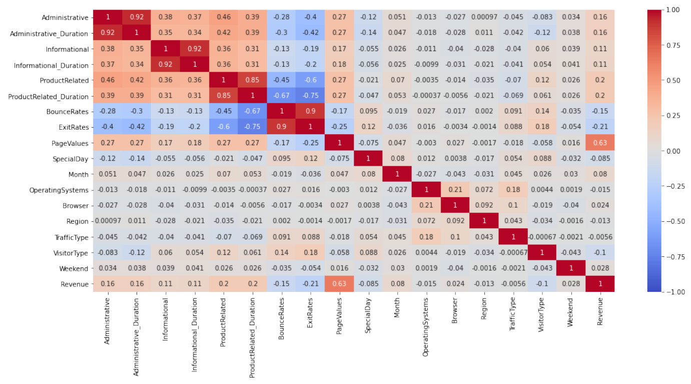

Fig. 1. Heatmap for the entire dataset

Observations:

- Revenue is heavily correlated with page values and slightly with all the other numerical columns.
- Revenue has a slight relationship with visitor type
- All the numerical features are highly correlated between themselves
- Categorical features such as visitor type and traffic type show a slight relationships with our numerical features

Therefore by performing EDA, we can observe different relationships of the data and make concrete statements about them. Additionally, it greatly helps in detecting abnormalities such as outliers and helps with machine learning techniques such as feature selections.

B. Data Transformations and Preprocessing:

We first reduce the skewness of our dataset by performing log1p transformation. This takes the natural logarithm of the dataset plus adding one. This reduces skewness and handles the zero values. We then also converted our dataset into integer categories from their string counterparts.

Previously, we also observed that our dataset was highly imbalanced, which can negatively affect the performance of classification models by biasing predictions toward the majority class. To address this, we performed SMOTE (Synthetic Minority Over-sampling Technique) analysis to resample the dataset. SMOTE works by generating synthetic examples of the minority class by generating new data points between minority class samples and their nearest neighbors. This is a powerful tool to balance the dataset without duplicating the existing data which can lead to overfitting.

C. Train Test Split:

We did a 70-30 Train Test split. We performed SMOTE on our training data while keeping testing untouched. We then used training set to train our models and used test set for evaluation purposes

D. Algorithm Details:

1) Naive Bayes: For our Naive Bayes algorithm, we began by further preprocessing our dataset. Since Naive Bayes works best with categorical data, we converted our numerical columns into categories bins. We then separating our features (X) and the target variable (Y)

The algorithm followed these main steps:

- Calculated the Prior Probability:
  Naive Bayes first computes the prior probability of each class in the dataset. The prior probability is the likelihood of each class occurring in the data before looking at any of the feature variables.

$$P(Y = y)$$

- Calculated the Conditional Probabilities:
  Then, for each of the feature variables, the algorithm computes the probability of that feature being in a certain class label. This is also called conditional probability or Likelihood.

$$P(X = x|Y = y)$$

- Predicted using Bayes Rule:
  In our final step, the algorithm takes each of our test samples and calculates the probability of it belonging to each class. This is done by multiplying the prior probability with the product of the probabilities of each observed feature given that class. The class with the highest probability becomes the prediction or label for that test sample.

$$P(X = x_1|Y = y) \dots P(X = x_n|Y = y)P(Y = y)$$

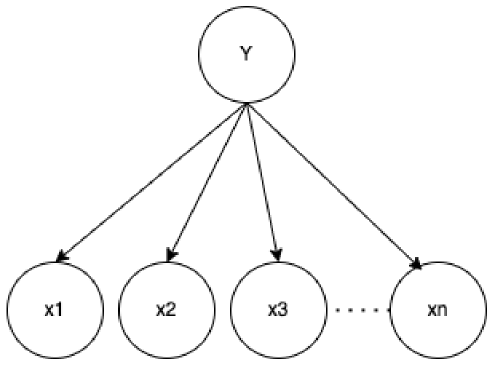

Fig. 2. Bayes Net Diagram

This algorithm assumes that all the features are independent and only depends on the class label. [Fig-2]

2) Logistical Regression: We implemented logistic regression from scratch using Numpy. This algorithm is used for classifying our target variable ('Revenue'). The model is trained using the maximum likelihood estimation to find parameters that best fit the data.

- Sigmoid Function: The sigmoid function is our primary choice for logistical regression because it maps the output of the model to a probability which is between 0 and 1.

$$Sigmoid(z) = \frac{1}{1+e^{-z}}$$

- Cost Function: The cost function is important as it gives our model the ability to assess how well our weights are performing in predicting the probabilities of a certain label. The algorithm does this by first utilizing the weights to predict the label which is a probability between 0 and 1. Then it uses the predicted label and the true label to calculate our loss. We use Binary cross entropy for our loss function: Here $\hat{y}$ is our predicted label.

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

- Initialization: We start by randomly initializing our weights. These weights are the parameters assigned to the feature variables.
- Gradient Descent: Now, the weights will be updated based on the gradient of cost function with respect to each weight. By doing so, we minimize the cost function.
  - Calculate the prediction using the sigmoid of the weighted sum which is the dot product of the weights and a sample
  - Calculate the gradient of the loss with respect to each weights
  - Update our weights by subtracting the gradient times our alpha (learning rate)

$$w \leftarrow w - \alpha * \nabla_w g(w)$$

$$\nabla_w g(w) = \begin{bmatrix} \frac{\partial g}{\partial w_1} \\ \frac{\partial g}{\partial w_2} \\ \vdots \\ \frac{\partial g}{\partial w_n} \end{bmatrix}$$

Lastly, after fully training our model we can make predictions. If our calculated probability by the weights are greater than 0.5, we will classify it as 1 or 0 otherwise

3) Nueral Network: : This is our final algorithm that we are going to employ in making a prediction whether a users will have an intention to purchase.

- Model Architecture:
  - Input layer: 17 nodes for each of our features.
  - Hidden layer 1: 5 nodes
  - Output layer: 1 node

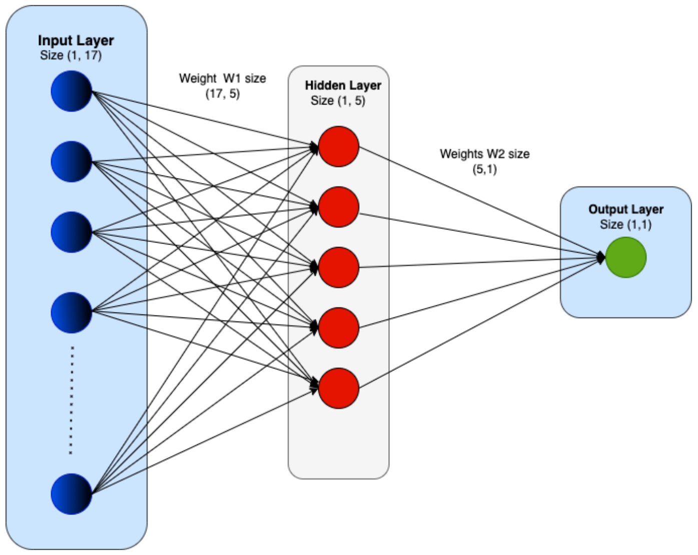

Fig. 3. Nueral Network Model Architecture

- Initialize our weights:
  - Weight 1: Randomize the weights for the connection between the input and the hidden layer. The shape of these weights are (17, 5)
  - Weight 2: Randomize the weights for the connection between the hidden and the output layer. The shapes of the weights are (5, 1)
  - Bias terms are also initialized to zeros
- Activiation Function: We are going to be using sigmoid as our activation function for both our hidden and output layer.

$$Sigmoid(z) = \frac{1}{1+e^{-z}}$$

- Loss Function: For our loss function we are going to use Binary Cross Entropy. This will tell us how well our model is doing with our weights. Nueral network also aims to get weights that best fit our data so that the loss function is minimized.

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

- Forward Propagation: In forward propagation, the model calculates the predicted output by utilizing weights and biases. In essence, a sample passes through the input layer to the hidden layes and lastly comes out through the output layer where we classify it.
  Hidden Layer:

$$z_1 = X \cdot W1 + b1$$

$$a_1 = sigmoid(z_1)$$

Output Layer:

$$z_2 = a_1 \cdot W2 + b2$$

$$a_2 = sigmoid(z_2)$$

- Backward Propagation: This is similar to the forward propagation but goes backwards. It calculates how much of each weight is related to the loss (gradients). We use such gradients to update our weights. The gradients are calculated by taking the derivatives of the loss function.

$$\delta_2 = \frac{\partial \text{Loss}}{\partial z_2} = a_2 - y$$

We use chain rule to backpropagate to the hidden layer

$$\delta_1 = \frac{\partial \text{Loss}}{\partial z_1} = \frac{\partial \text{Loss}}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_2}$$

Lastly we calculate our gradients with respect to the weights.

$$\frac{\partial \text{Loss}}{\partial W_1} = \delta_1 \cdot x$$

$$\frac{\partial \text{Loss}}{\partial W_2} = \delta_2 \cdot x$$

As we are doing a batch, we are going to average the derivative of the loss with respect to the weights by the size of the batch.
- Lastly Optimization: We are going to optimize our weights by subtracting our gradients using the formula or gradient descent.

$$W_1 \leftarrow W_1 - \alpha \frac{\partial \text{Loss}}{\partial W_1}$$

$$W_2 \leftarrow W_2 - \alpha \frac{\partial \text{Loss}}{\partial W_2}$$

To conclude, we train our Nueral Net with train dataset that is resampled using SMOTE data anaylsis. We do a batch gradient descent. The nueral network works best when $\alpha = 0.0005$, $epoch = 500$, $InputNuerons = 17$, $HiddenNueron = 5$ and $OutputNueron = 1$. Additionally, by doing a batch of 64 samples at a time, the nueral networks learns its weights more efficiently and correctly.

IV. Experimental Evaluation

A. Library Details

We used pandas and numpy for data manipulation. Matplotlib and seaborn for visualization purposes. Scikit Learn for data preprocessing and model evaluation. Lastly, we used SMOTE data resampling to synthetically resample our data. This was done to resolve the imbalance in the dataset.

B. Starter Code

Neural Network implementation provided in class: I used this implementation heavily to implement my neural network.[3]

Github Neural Network Implementation: Used this implementation to use the class structure and do the batch gradient descent.[4]

C. Algorithm Tuning and Personilization

Note: SMOTE was applied to every single dataset

1) Naive Bayes:: The numerical columns in the dataset was converted into categorical bins of size 5. All the 17 features were used in training the model.

2) Logistical Regression:: The dataset was kept as it just like Naive Bayes. One Hot encoding negatively impacted the results. The learning rate for this algorithm was 0.1

3) Nueral Network:: The dataset for the Nueral Network was kept the same. The paramters used for the nueral network algorithm were input layer with 17 neurons, hidden layer with 5 neurons and lastly output layer with 1 nueron.

- epoch = 500
- learning rate = 0.0005

D. Result Details

1) Naive Bayes:: The Naive Bayes algorithm results

- Accuracy Score: 0.743
- Precision Score: 0.359
- Recall Score: 0.828
- F1 Score: 0.501
- Confusion Matrix:

$$\begin{bmatrix} 2271 & 852 \\ 99 & 477 \end{bmatrix}$$

We see that after SMOTE data resampling the F1 score more than double. This was a big increase from the previous models. We also see that the precision score is 0.359 which means that there could be a lot of false positives. This is problematic as it indicates a lot of the non purchase data indicates as purchased.

2) Logistical Regression:

- Accuracy Score: 0.858
- Precision Score: 0.531
- Recall Score: 0.747
- F1 Score: 0.621
- Confusion Matrix:

$$\begin{bmatrix} 2744 & 379 \\ 146 & 430 \end{bmatrix}$$

SMOTE also improved our F1 score in our logistical regression. We also see training loss to decrease gradually in our graph. However, the recall score did not improve all that much from naive bayes.

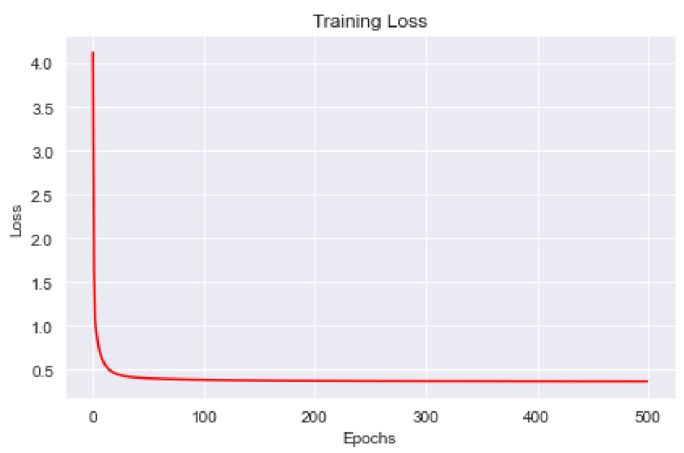

Fig. 4. Logistic Regression Training Loss vs Epoch chart

3) Nueral Network: Implemented using batch gradient descent, alowing it to learn faster. The Result of this algorithm are as follows:

- Accuracy Score: 0.8577
- Precision Score: 0.8812
- Recall Score: 0.8270
- F1 Score: 0.8532
- Confusion Matrix:

$$\begin{bmatrix} 6485 & 814 \\ 1263 & 6036 \end{bmatrix}$$

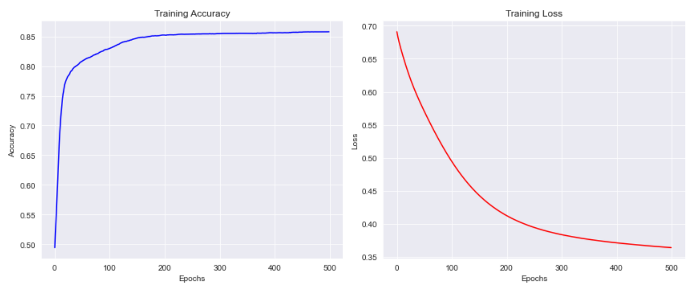

Fig. 5. Neural Network Training Loss and Accuracy Graph

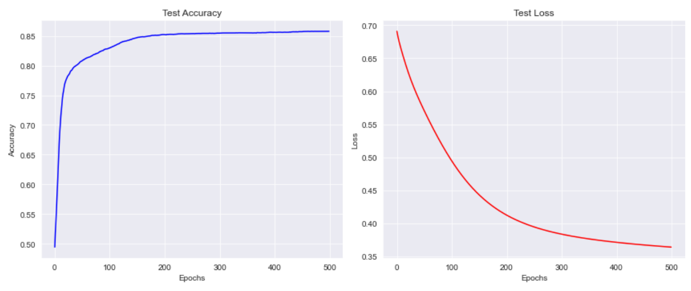

Fig. 6. Neural Network Testing Loss and Accuracy Graph

From these 3 models, we have seen that nueral network perform the best among all the model tested. It provides the highest F1 score (0.8522) while keeping its prediction of user online shooping intenntions to a 85.77% accuracy. We have also seen SMOTE data analysis improved our overall model quality by increasing the F1 score.

E. Dataset Details

The dataset used in this project is called Online Shoppers Purchasing Intention Dataset from UCI Machine Learning Repository. [5]

V. Limitation and Future Works

Limitation: This dataset is relatively small with only 12,000 records. It also has a high imbalance which makess it harder to make an accurate model. Addiitioinally because Nueral Network works well with unstructured data such as images, text, audio but not so much with structured data such as this. Furture work in this would be to implement Random Forrest algorithm which performs well under imbalances and structured data.

VI. Conclusion

The aim of this project was to predict whether an online shopper would make a purchase based on session behavior and user attribute. The neural network achieved the highest score across the board with an accuracy of 85.77%, Recall score of 82.70%, Precision score of 88.12% and F1 score of 85.32%. But due to the small dataset, class imbalance and structured data may cause the model perform worse than other kinds of machine learning algorithm and techniques. Future work could utilize an ensemble methods such as Random Forrest and other machine learning techniques.

References

[1] C. Micus, S. Schramm, M. Boehm, and H. Krcmar, "Methods to analyze customer usage data in a product decision process: A systematic literature review," *Operations Research Perspective*, vol. 10, 2023, Art. no. 100277. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221471602300012X?ref=pdf_download&fr=RR-2&rr=93d63d423a2b8f8d

[2] Omidvar, Mohammad & Mirabi, Vahid & NARJES, SHOKRY. (2011). Analyzing the Impact of Visitors on Page Views with Google Analytics. International Journal of Web & Semantic Technology. 2. 10.5121/ijwest.2011.2102.

[3] Google Colab. (n.d.). https://colab.research.google.com/drive/1Xt6O325NEL1FQeAfFqzXX7x0hm9YA7ZW?usp=sharing

[4] Lionelmessi. (n.d.). GitHub - lionelmessi6410/Neural-Networks-from-Scratch: In this tutorial, you will learn the fundamentals of how you can build neural networks without the help of the deep learning frameworks, and instead by using NumPy. GitHub. https://github.com/lionelmessi6410/Neural-Networks-from-Scratch

[5] Sakar, C. & Kastro, Y. (2018). Online Shoppers Purchasing Intention Dataset [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5F88Q.