# Walmart Shopping Trip Classification

*Dennis Gavrilov, Andrew Koe, and Miraj Patel, Cooper Union, New York, USA*

***Abstract* – This paper discusses the authors' approach to competing in Walmart's recruitment challenge on Kaggle. The competition provides a large dataset of customer visits to Walmart stores; each visit is identified uniquely by a "VisitNumber" feature and is associated with multiple UPC and Fineline numbers, indicating multiple items bought at once. visit required classifying customer visits into shopping trip types. Using Random Forests, Extreme Gradient Boosted Trees, and feature engineering, we were able to beat their preset benchmarks and for a temporary period be in the middle of the leaderboard.**

***Index Terms* – classification, extreme gradient boosted trees, machine learning, random forest**

## I. INTRODUCTION

A popular application of machine learning in industry is in recommendation systems. Companies can track search and purchase histories for all of their users and use machine learning to make targeted suggestions. The current state of the art recommendation systems can be eerily accurate, as in the case of Target's pregnancy ad campaign. Target was able to identify pregnant customers before some of the customers were ready to announce their own pregnancy to their families. Target used this information to begin sending targeted advertisements for baby products to those customers, demonstrating just how lucrative prediction algorithms can be in the retail industry.

## II. CONTEST DESCRIPTION

The Kaggle competition that we participated in was hosted by Walmart. They provided data regarding what customers were purchasing and the goal was to try and identify what type of shopping trip the customer was making (e.g. grocery shopping or clothes shopping). The submission required probabilities to be assigned to each of the 45 possible trip types. The data provided for each visit contained: weekday, UPC numbers, transaction type (purchase or return), department of item (e.g. dairy, shoes,

appliances), and fineline number (an internal Walmart identifier that is more specific than the department. The contest bans the use of any information outside of what was provided. Thus, the UPC numbers cannot be used with an external API to determine items specifically, which may have had a positive impact on the categorization of trips.

The submissions were evaluated using the multi-class logarithmic loss shown below, where "N" is the number of visits, "M" is the number of trip types, "y" is a binary value indication whether or not the observation is in class j, and "p" is the assigned probability.

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

The leaderboard provided two different benchmarks. The lower benchmark was the "All Zeros" benchmark, which was simply the multi-class logarithmic loss obtained if one were to assign the same probability to all classes. The second benchmark was more sophisticated and was the multiclass logarithmic loss obtained when using a Random Forest with just the department names as features.

## III. The Approach

The contest required assigning probabilities to each class which made it inappropriate to use linear classifiers such as support vector machines, but was ideal for trees.

### *A. Random Forests*

The first approach we took was to use the random forest algorithm provided by Python's Sklearn library. The random forest algorithm is a popular method for solving many Machine Learning problems, and relies on an ensemble of decision trees. In the first attempt, the UPC numbers, transaction types, and fineline numbers were used as features. The results showed moderate success, and placed us the middle of the leaderboard.

At times, the shoppers captured by the data did not strictly buy items corresponding to a single agenda, leading to slight variations from what would generally be expected for a single shopping trip of a certain type. These slight abnormalities, or outliers, may have greatly influenced the decision trees, leading to lackluster results.

### *B. Extreme Gradient Boosted Trees (xgboost)*

In order to obtain results that were slightly more competitive with the other

participants, we switched to new library and algorithm. Python's Sklearn library is simple and ideal for testing out theories quickly, but it doesn't offer as many tunable parameters. The XGBoost library used gradient boosted trees and provided more hyperparameters to tune. With this library, we were able to control how long to train the model as well as the maximum depth of the trees.

In addition to changing algorithms, we also engineered new features using a bag-of-words like approach, as suggested by Professor Keene of The Cooper Union. We created a vector of fineline numbers for each visit; each dimension in the vector corresponded to a unique fineline number and the values were the frequency of the item appearing in the visit. There were about 5000 unique fineline numbers so each visit was associated with a list of 5000 elements. In order to obtain the values of each vector, the code would aggregate the data entries corresponding to each visit, and increment the correct fineline number for each purchase made by the customer. This approach did indeed lead to better results and pushed us higher on the leaderboard.

## IV. Results

Using the Random Forest algorithm, we were able to barely beat the "all-zeros" benchmark. We received a score of 28.03 and beat the lower benchmark score of 34.54. Using the xgboost library we received our best score of 2.85 which beat the second benchmark score of 5.72.

## V. Conclusion

We were able to beat both benchmarks and (for a short period of time) be in the middle of the participants. If we had more time, we may want to try cross validation techniques and other features. Currently, we were limited by the senior lab machine we were using. Each model took almost half an hour to train and without highly optimized code, we ran out of memory. We were unable to glean much new insight from our results since much of the data was anonymized. However, it was interesting to see what we were able to achieve without taking into account any domain knowledge.

## VI. Acknowledgements

We would like to thank Chris Curro for his advice on technologies to tackle the problem. His suggestions regarding the xgboost library and feature engineering were invaluable to this project. We would also like to thank Sam Keene, Professor at The Cooper

Union, who advised with possible methods of introducing more, beneficial features.

## VII. References

**Xgboost library:**

https://xgboost.readthedocs.org/en/latest/

**Kaggle Contest:**

https://www.kaggle.com/c/walmart-recruiting-trip-type-classification