**W207 Final Project**

**Baseline Submission**

**Daniel Balck, Richard Hitchens**

**Walmart: Trip Type Classification**

**Business Understanding**

**What problem are we trying solve?**

The problem we are trying to solve as explained on Kaggle is:

*“Walmart uses both art and science to continually make progress on their core mission of better understanding and serving their customers. One way Walmart is able to improve customers' shopping experiences is by segmenting their store visits into different* ***trip types****.*

*Whether they're on a last-minute run for new puppy supplies or leisurely making their way through a weekly grocery list,* ***classifying trip types*** *enables Walmart to create the best shopping experience for every customer.*

*Currently, Walmart's* ***trip types*** *are created from a combination of existing customer insights ("art") and purchase history data ("science"). In their third recruiting competition, Walmart is challenging Kagglers to focus on the (data) science and* ***classify customer trips*** *using only a* ***transactional dataset*** *of the items they've purchased. Improving the science behind* ***trip type classification*** *will help Walmart refine their segmentation process.”*

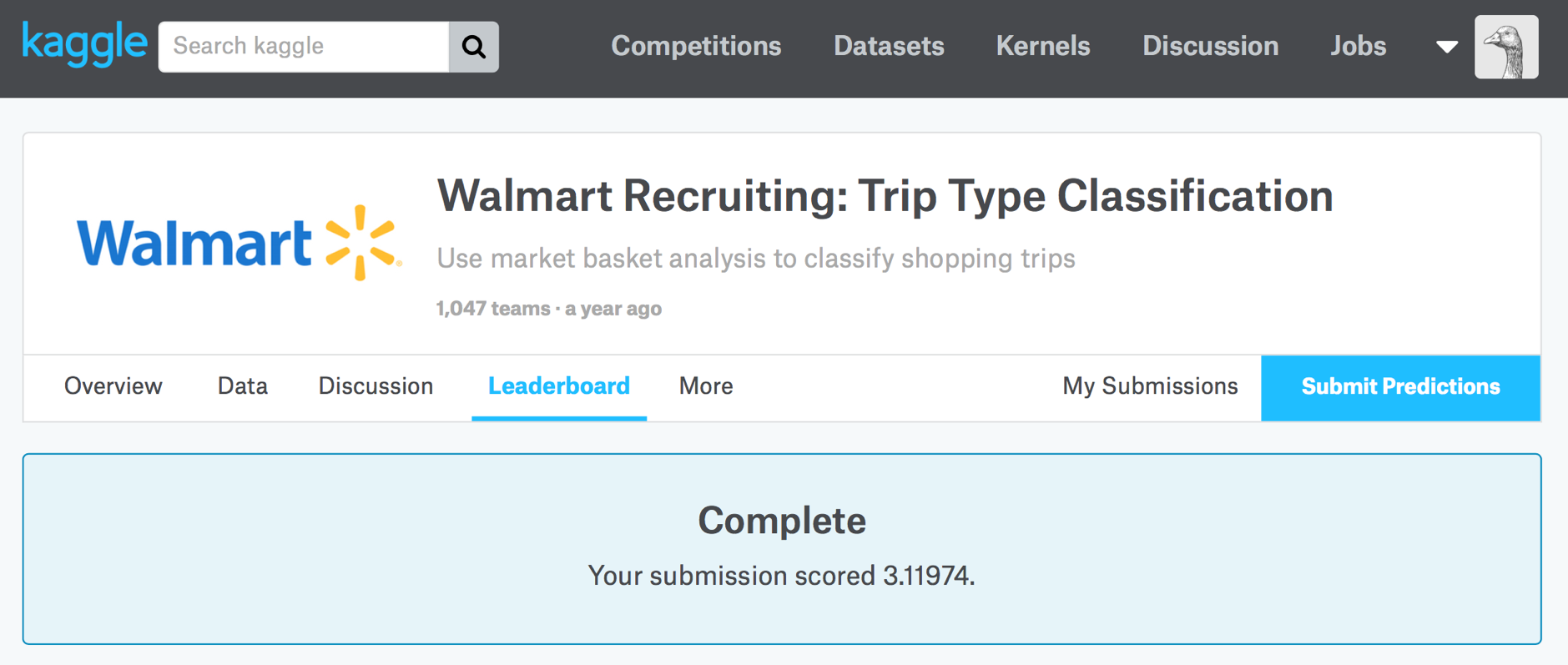
Accordingly, we are trying to solve a **multi-class classification problem**.

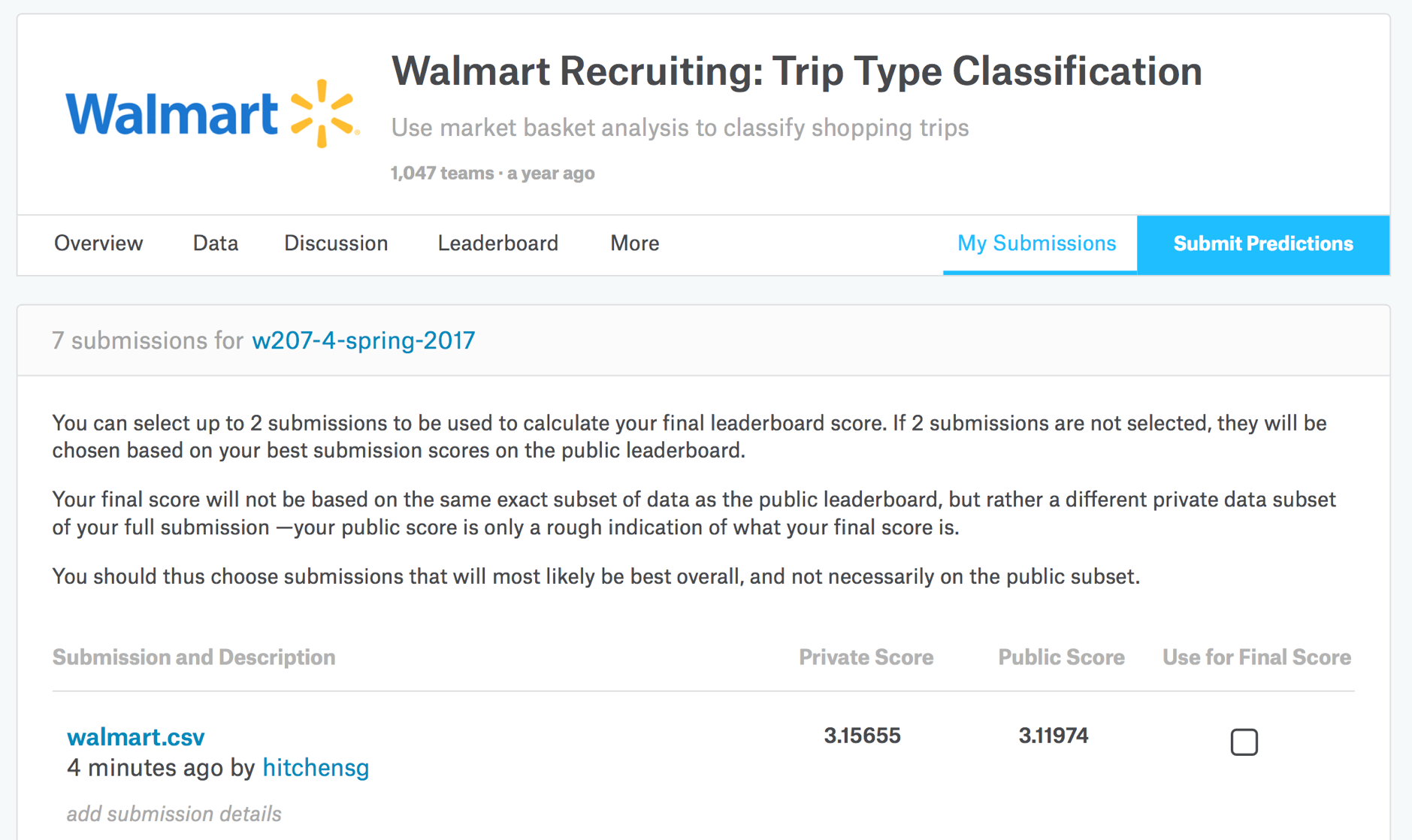
In particular, Walmart has categorized the trips contained in the data set into **38 distinct types** using a proprietary method applied to a more extended set of data.

The problem is made more challenging because the **trip types** are simply identified with numbered labels, products are also identified by number (although some higher- level department descriptions are also provided) and the data set provided is restricted than the data set used to classify the trip types.

**What are the relevant metrics? How much do we plan to improve them?**

As the problem is a **multi-class classification problem** the model will be evaluated using the multi-class logarithmic loss, which has to be minimized. Our baseline model produces a private score of 3.16, which would have positioned our model at a rank of 727 out of 1047. To improve our score to at least get in the top quartile we will need to lower the log-loss score to 0.73 or lower.





**What will we deliver?**

We will deliver a machine learning model that (given the available inputs below) best predicts Walmart’s specified Trip Types.

**Data Understanding**

**What are the raw data sources?**

The raw data is sourced from Walmart’s transactional data set of items their customers have purchased on a number of individual visits.

**What does each 'unit' (e.g. row) of data represent?**

Each unit in the data set represents a unique product type that was purchased by a single customer as part of their broader single trip to Walmart, i.e. it represents one product class in their overall visit basket (there is a field (ScanCount) in the row that identifies the number of items of the product that were purchased or returned).

**What are the fields (columns)?**

Walmart explains the data fields (columns) as follows:

* **TripType:** A categorical ID representing the type of shopping trip the customer made. This is the ground truth that you are predicting. TripType\_999 is an "other" category.
* **VisitNumber:** An ID corresponding to a single trip by a single customer
* **Weekday:** The weekday of the trip
* **Upc:** The UPC number of the product purchased
* **ScanCount:** The number of the given item that was purchased. A negative value indicates a product return.
* **DepartmentDescription:** A high-level description of the item's department.
* **FinelineNumber:** A more refined category for each of the products, created by Walmart.

Some comments:

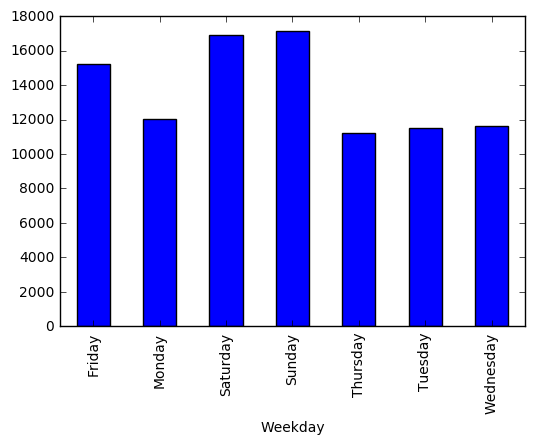
* **TripType:** As mentioned above this is a categorical variable that includes 38 distinct numbered trip types, which is the target of the model.
* **VisitNumber:** Within the data set there are 95,674 unique visit number IDs.
* **Weekday:** Is a simple categorical variable identifying the day of the week the visit occurred.
* **ScanCount:** Is the number of items of the given product type that were purchased or returned on the individual shopping trip. We need to consider how we treat the negative values, i.e. do they actually have much value in predicting trip type?
* The other fields are all categorical variables identifying at varying levels the type of product being purchased or returned. One level is a numerical product level classification (**Upc**), another is a more summarized Walmart defined numerical product level classification (**FinelineNumber**) and the highest-level product classification are text based categories (**DepartmentDescriptions**), which can easily be converted to dummy variables, but using ScanCount instead of 1 or 0, i.e. the number of items purchased or returned.

**EDA**

**Distribution of each feature**

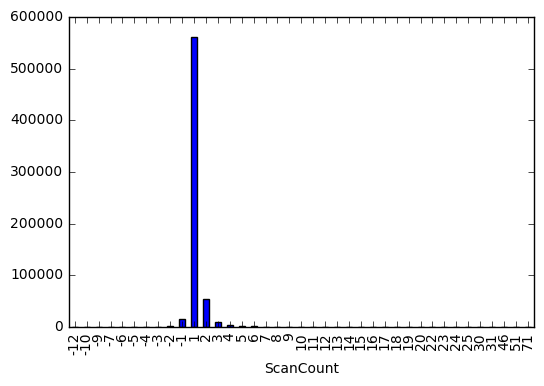
Weekday:

We plot the number of visits by week day. Frequency is higher on Friday, Saturday and Sunday, which is what you would intuitively expect, so the data set would appear to be a slice across time with equal numbers of days of the week. Accordingly, there doesn’t appear to be anything concerning with the distribution of the data with respect of week days of visits.

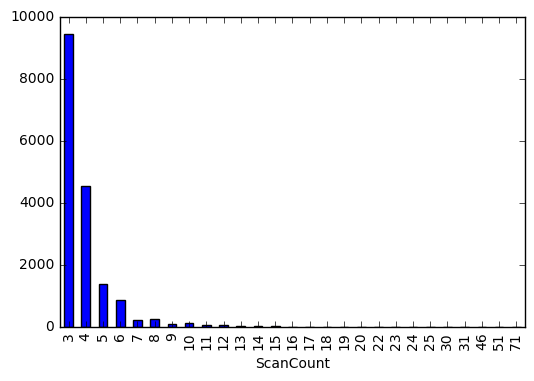


ScanCount:

We plot the frequency of the number of each product type in each shopping basket across all visits. As can be seen one item is by far the most common number of items of a product in a shopping basket.



It is hard to see in the chart above, but as shown below the frequency declines exponentially as the number of each product item in the shopping basket rises.

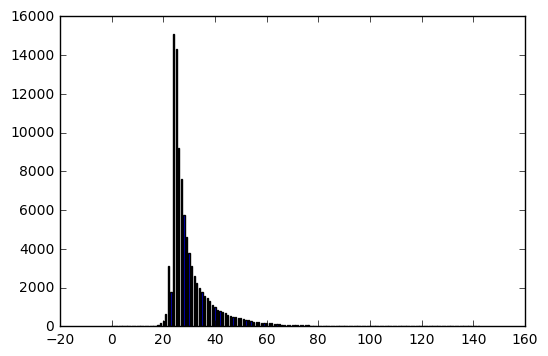


The same general relation is found for items being returned, i.e. negative ScanCount, but on a much lower frequency scale.

Accordingly, the data is as you would intuitively expect and this distribution will need to be considered when looking at variable normalization because many features will be derived from the ScanCount field.

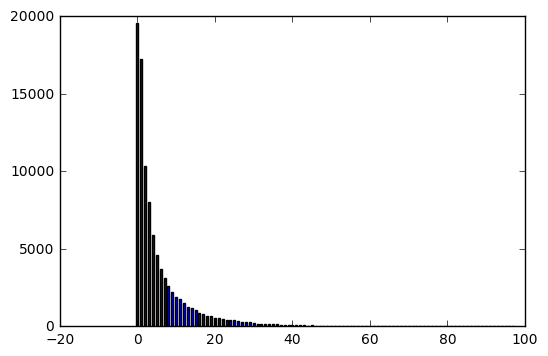
NumItems

The following chart, which uses an aggregation explained below, plots the histogram of the number of individual items in each shopping basket across the ~650K unique visit numbers.



NumProducts:

The following chart, which uses an aggregation explained below, plots the histogram of the number of individual products in each shopping basket across the ~650K unique visit numbers.



**Missing values**

As shown below a simple count of null values indicates that the only missing values in the data set relate to product IDs, i.e. **Upc** product numbers and hence consequently their **FinelineNumber** and **DepartmentDecription**.

**Field Count**

TripType 0

VisitNumber 0

Weekday 0

Upc 4129

ScanCount 0

DepartmentDescription 1361

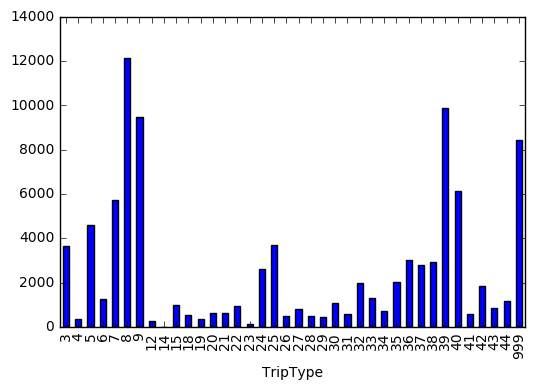
FinelineNumber 4129

There are 4,129 rows with null values out of X. Our exploratory analysis indicates nothing systematic about the null values.

We replace null values in the data set with ‘Unknown’ to effectively create an extra product classification in each of the three levels of product identification.

**Distribution of target**

A histogram of the target classifications is shown in the chart below. As can be seen the target frequencies are not evenly distributed across the classifications. In particular, a number of the classifications appear in the training data set with quite low frequency, so this will potentially make prediction of these classifications quite difficult.



**Relationships between features**

The primary features for this problem will be derived from the **ScanCount** field such as number of items or products in the basket or number of items by product classification.

**Other idiosyncracies?**

The negative values for returned items are a peculiarity of this particular problem. It is interesting to consider the influence of the returned item/s on the TripType. Is the returned item the cause of the trip? If so does it lead to different sized baskets than otherwise. Maybe consider a basket-includes-a-returned-item dummy? Maybe the returned items are misleading to overall trip type prediction in which case they should be converted to zero values. We need to consider their influence in more detail.

Many machine learning models are founded on the assumption of independence of observations assumptions, e.g. Naïve Bayes. Such an assumption is quite flawed in this instance because the combination of the items in the baskets are what enable trip type prediction, i.e. items in the basket are dependent on each other because of the trip type.

The unique products in the train and test data sets are not the same, i.e. there are around 25K different unique products between the two data sets so making predictions about individual unknown products could be difficult. However, there is no difference at the department level, so modelling at the department level will likely produce better results.

**Data Preparation**

**What steps are taken to prepare the data for modeling?**

Data analysis shows that there are just under 100,000 product items at the Upc level, so if you treat each Upc product as a feature you end up with a matrix of around 100,000 visits by 100,000 products most of which are zero entries, i.e. each shopping basket only has a very small number of all the possible products. Accordingly, a primary preparation of the data set is to load the data into Sparse Compressed Row format.

Initial model fits of a default Logistic Regression indicate that optimization of model even just across the 69 departments is time consuming, while for Naïve Bayes it is very fast and can in fact be easily optimized over the lowest level Upc product classification, however leads to overfitting and does not generalize well due to the large number of new products in the test data set.

**Feature transformations? engineering?**

**Weekday:** We convert the categorical Weekday variable into a numerical variable where Monday converts to 1 and Sunday to 7.

**Return:** We construct a dummy variable (Return) that denotes whether there was an item returned as part of the shopping trip, i.e. one of the items had a negative ScanCount. We have done this because trips with a returned item may have different characteristics to trips without a returned item, e.g. they may be of smaller size that non-return trips.

**ScanCount:** To run a baseline run in Naïve Bayes it was unhappy with negative values in the feature set, so to get it going we converted all negative ScanCount values to zeroes, the eventual final classifier will unlikely by Naïve Bayes, so how we treat the negative items is still up for further review

**Table joins? Aggregation?**

As discussed above the unit in the data set is a description of one product in a given shopping basket or visit, so it is necessary to use aggregations of the ScanCount field, i.e. the number of items of the product in the shopping basket, across the other fields to end up with a table describing each shopping visit.

**NumProducts/NumItems:** We first aggregate the ScanCount field by the VisitNumber field, using sum() and count() to work out the number of products (NumProducts) and items (NumItems) in each shopping basket, this will help to differentiate between big weekly shop trip types versus say a trip type to grab an item you might be missing for a recipe.

We then aggregate ScanCount using sum() across each of the more aggregated product categories, FinelineNumber and DepartmentDescription to determine the number of items purchased in these classifications.

**Precise description of modeling base tables.**

**What are the rows/columns of X (the predictors)?**

Rows:

**VisitNumber:** The rows in the base features table are the unique shopping visits that we are trying to predict their trip type.

Columns:

The columns in the base features table are:

**NumItems:** Is the number of individual items in the shopping basket.

**NumProducts:** Is the number of individual products in the shopping basket.

**Return:** Is a dummy indicating whether the shopping basket includes a returned item or not.

**Weekday: A set of dummies to indicate the day of the shopping visit.**

**DepartmentDescription:** A set of 69 features indicating which department the product comes from and how many units of the product were purchased or returned.

**FinelineNumber:** A set of ~5,300 features indicating which FinelineNumber category (a more granular version of DepartmentDescription) the product comes from and how many units of the product were purchased or returned. Due to impact on processing time/ability to find these features are not used in all model specifications.

**What is y (the target)?**

The target is the field **TripType.** It is a categorical ID representing the type of shopping trip the customer made. Walmart has categorized the trips contained in the data set into **38 distinct types** using a proprietary method applied to a more extended set of data. It is the ground truth that we are predicting. TripType\_999 is an "other" category.

**Modeling**

**What model are we using? Why?**

Our baseline model uses a default Logistic Regression because it generally performs reasonably well in most scenarios and it offers inbuilt multi-class functionality with log loss scoring.

**Assumptions?**

For our baseline model we have used ‘multi\_class’ set to ‘multinomial’, ‘solver’ set to ‘newton-cg’, ‘tolerance’ set to 1 because it was having trouble converging and a maximum number of iterations of 400.

**Regularization?**

We used a low level of regularization with c=1000 to help with optimization speed, but caused overfitting.

**Evaluation**

**How well does the model perform?**

The model is evaluated on the basis of a multi-class log loss score, some classifiers have this as a default scoring option while others don’t so we derive our own one for use when there isn’t one as follows:

from sklearn import metrics

my\_log\_loss = metrics.make\_scorer(metrics.log\_loss, greater\_is\_better=False, needs\_proba=True)

Intending to use a feature set with only department level product counts, we accidentally used the full feature set with around 100K features (12 hours later on a MacBook Pro it finished!), so our baseline model produced a log-loss score of 0.07 when optimized on the full training set and using no cross validation, so it didn’t generalize that well to the test data set hence the test score of 3.15 on Kaggle.

**Accuracy**

Similarly, the F1 score on our baseline model was 99.8%, which also suffered from overfitting. No test score equivalent was given by Kaggle as it is not a metric of interest to the submission.

**Cross-validation**

Using a grid search default cross validation process it ran into difficultly when we tried to use more than 4 folds, because the minimum number of target trip types were 4 for trip type 14. Grid search would not allow more folds than 4 we assume so that it could make each fold representative of the full sample. We will possibly look to do our own folds to relax this constraint.

**Other metrics not yet used:**

ROC curves

other metrics? performance?

AB test results (if any)

**Deployment**

**How is the model deployed?**

The model is deployed to Walmart via Kaggle in in their specified submission format.

In particular, you must submit a csv file with the **VisitNumber** (from the test data), all the 38 candidate **TripType** classes, and a probability for each class, i.e. a predicted probability matrix of shape (95,584, 38). The order of the rows does not matter. The file must have a header and look like the following:

"VisitNumber","TripType\_3","TripType\_4",...  
1,0,0.1,...  
2,1,0,...  
etc.

It is noted that the submitted probabilities for a given visit are not required to sum to one because they are rescaled prior to being scored (each row is divided by the row sum). In order to avoid the extremes of the log function, predicted probabilities are replaced with *max(min(p,1−10e−15),10e−15)*. Notably this probability condition is the default setting in the scikit learn log-loss metric.

The submission file format is easily created using the predict\_proba() function for fitted classifiers in scikit learn, which returns a predicted probability for each class for each example in the data set.