Definition

Project Overview

The overall objective of the project was to develop a recommender system utilizing an open source dataset containing data from a national grocery retailer. Most major grocery chains[1] utilize some form of personalized recommender system in order to offer their customers personalized prices, discounts and promotions as well as other features such as predictive shopping lists and meal planners to help make shopping easier.

The data utilized[2] was provided by dunnhumby, a global leader in retail analytics. The dataset contained a sample of 117 weeks of 'real' customer data from a large grocery store constructed to replicate typical pattern found in in-store data to allow the development of algorithms in a (near) real-world environment. Critical to enabling the development of a recommender system this data contained a customer identifier (obtained through a store loyalty card) then enabled the tracking of customers over time.

Problem Statement

Utilizing the data provided, algorithms will be developed that will use the customer shopping behavior from the previous 52 weeks in order to predict whether that customer will purchase an item in the following week. Such an algorithm could be utilized for creating 1:1 personalized content such as:

1. Creating a personalized digital flyer highlighting to each customer the items on sale that week that are most relevant for them
2. Generating automated weekly shopping lists
3. Reminding customers shopping on-line of items they may have forgotten to purchase i.e. items with high predicted relevancy for that week they did not purchase

The recommender system will consist of classic matrix factorization techniques commonly used in recommender systems but will also be combined with ML classification models that will add contextual features e.g. average customer purchase cycle for an item in order to predict the probability that a customer will purchase an item in a given week. An in-depth overview and rationale for this approach will be discussed in subsequent sections.

Metrics

The final models will be binary classification algorithms that will predict for every customer and item combination the likelihood that they will purchase an item in a given week. As such, metrics commonly used to assess the performance of classification problems will be utilized. The final comparison metric will be F1 score but other metrics such as precision, recall and AUC will also be calculated for each model developed.

Analysis

Data Exploration

The data provided contains 117 weeks of point of sale data from a grocery store for a random sample of 5,000 customers. A short data dictionary is provided below:

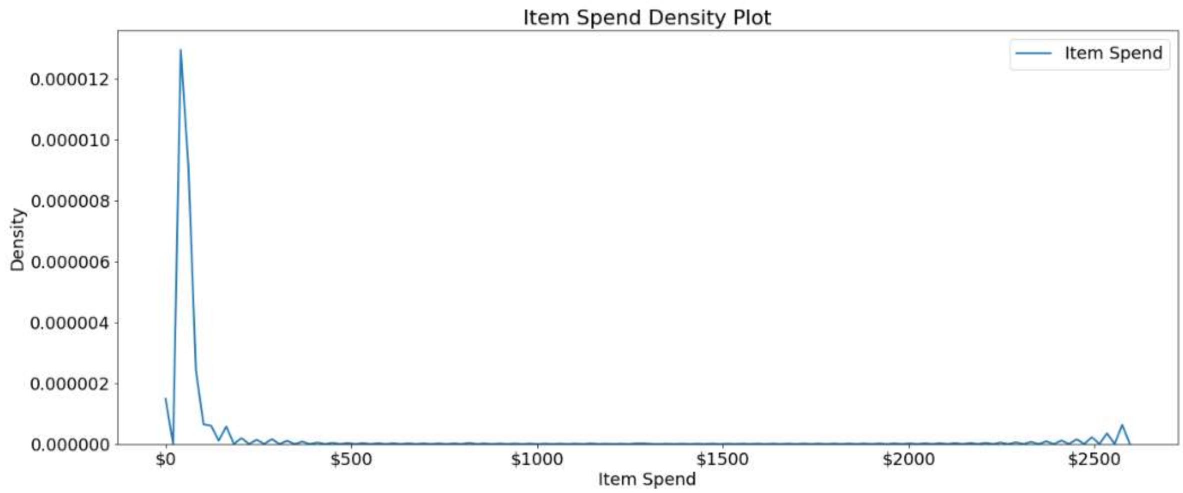
Variable	Description	Example Values	Missing Values?	Missing Value Identifier
SHOP_WEEK	The week of the transaction	200607		
SHOP_DATE	The date of the transaction	20060413		
SHOP_WEEKDAY	The weekday of the transaction	1, 2, 3		
SHOP_HOUR	The hour of the transaction	14, 15, 16		
QUANTITY	The quantity of the transaction	2		
SPEND	The spend of the transaction	1.03		
PROD_CODE	The code of the transaction	PRD0900097		
PROD_CODE_10	The 10 of the transaction	CL00001, CL00002		
PROD_CODE_20	The 20 of the transaction	DEP00001, DEP00002		
PROD_CODE_30	The 30 of the transaction	G00001, G00002		
PROD_CODE_40	The 40 of the transaction	D00001, D00002		
CUST_CODE	The code of the transaction	CUST0000410727	Y	NaN
CUST_PRICE_SENSITIVITY	The price sensitivity classification of the customer	LA, MM, UM	Y	NaN, XX
CUST_LIFESTAGE	The lifestage of the transaction	YA, OA, YF, OF, PE	Y	NaN
BASKET_ID	The id of the transaction	994100100398294		
BASKET_PRICE_SENSITIVITY	The price sensitivity classification of the transaction	LA, MM, UM	Y	XX
BASKET_TYPE	The type of the transaction	Small Shop, Top Up, Full Shop	Y	XX
BASKET_DOMAIN_MISSION	The mission of the transaction	Fresh, Grocery, Mixed, Nonfood	Y	XX
BASKET_SIZE	The size classification of the transaction	S, M, L		
STORE_CODE	The code of the transaction	STORE00001		
STORE_FORMAT	The format of the transaction	SS, MS, LS, XLS		
STORE_REGION	The region of the transaction	E01, E02, N02		

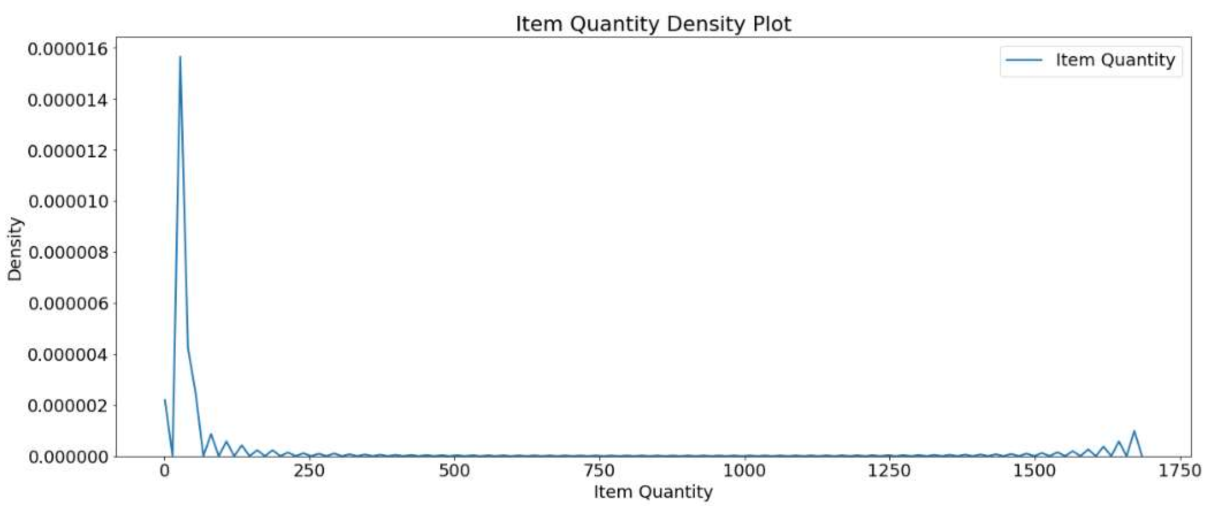
A number of the features have missing values, for example the CUST_CODE, CUST_PRICE_SENSITIVITY and CUST_LIFESTAGE varaibles are missing for “non-customer” records where a loyalty card was not used. In addition, the CUST_PRICE_SENSITIVITY variable has “XX” for customer records where the customer has not been classified.

All basket segmentations have been populated with “XX” where that basket was not classified with a segment.

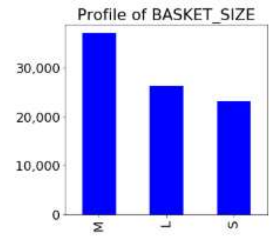
Exploratory Visualization

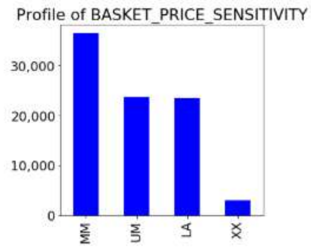
The following charts illustrate that there are a number of outliers in both the spend and quantity variables that will need to be removed:

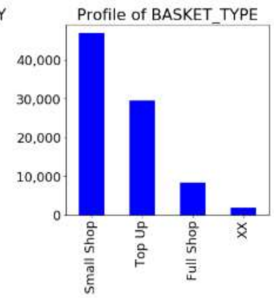


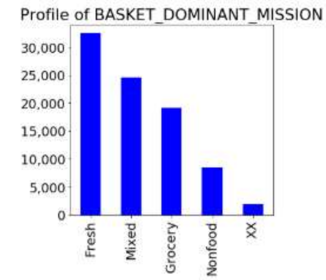


The following bar charts show the unique basket counts for each of the basket level segmentations:

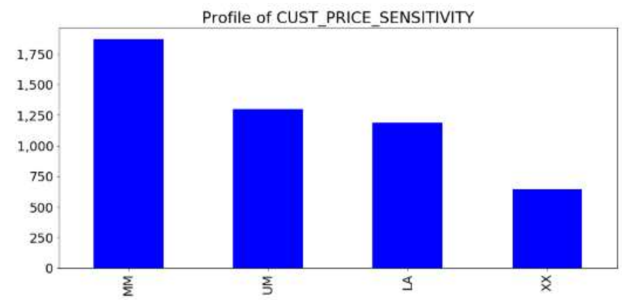


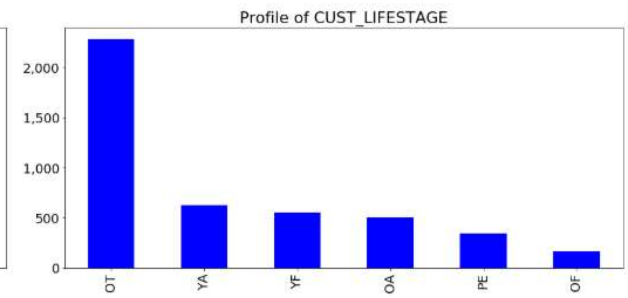






The following bar charts show the unique customer counts for each of the customer level segmentations:





Algorithms and Techniques

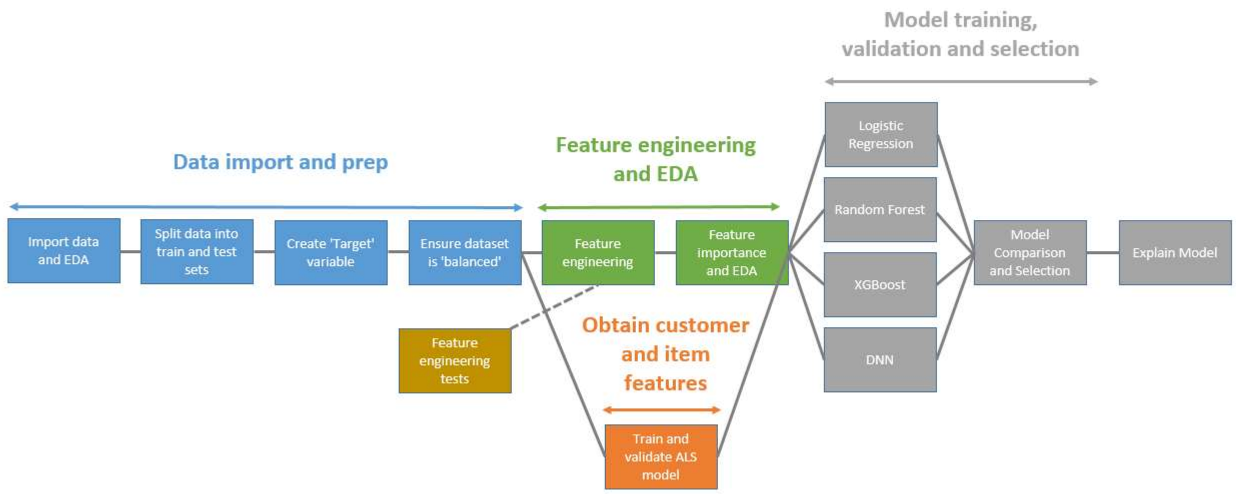
The recommender system will be comprised of two techniques:

1.) A matrix factorization technique will be utilized in order to generate user (customer) and item vectors that represent customers and items in a latent space. Specifically, an ALS model will be created by running a matrix factorization on customers and items using quantity purchased as implicit feedback

2.) The user and item features will be combined with contextual features such as how much each customer shopped in the category of the item being predicted, how much they have bought that item before, what their average purchase cycle is for that item and when they last bought that item to create a hybrid recommender. Standard techniques for binary classification can then be applied such as Logistic Regression, Random Forest and XGBoost. In addition a simple DNN classifier will be developed with TensorFlow to compare performance against the ML approaches

This approach combines the use of matrix factorization which is commonly used for such recommender tasks with classification algorithms to allow for the inclusion of powerful contextual features. Since the target variable is a binary ‘customer did not buy/bought the item’, standard classification approaches such as Logistic Regression, Random Forest etc. are well suited.

The diagram below outlines the end-to-end process in creating the recommender:



Each of these steps is discussed in more detail in subsequent sections.

Benchmark

The closest benchmark for this problem within the grocery domain is the Instacart market basket Kaggle competition[3]. The objective of the competition was to predict if a customer would re-purchase an item. This is a very similar problem in that it is in the grocery domain and is predicting whether a customer will purchase a given item in a defined timeframe. The leading F1 score for systems developed to solve this problem was typically ~0.4.

Research into top solutions[4] for this problem showed that the following types of features were found to be highly predictive:

- Counts of the number of items, aisles, departments shopped in different intervals e.g. last 15, 30 days etc.
- Timing features such as when the item was last purchased
- User-based features such as the tendency of a customer to buy items they'd never purchased before

Boosting algorithms and simple DNN's with a few layers were typically found to perform the best.

These insights were utilized in the development of the system in this project.

Methodology

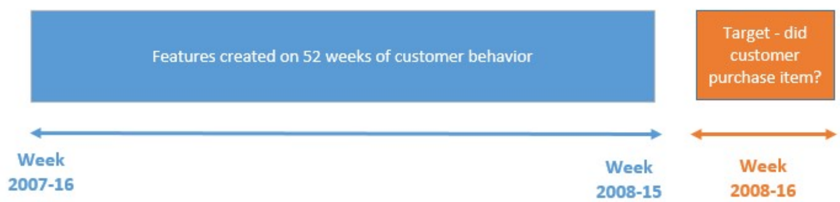
Data Pre-Processing

Addressing outliers

In order to address the outliers in spend and quantity variables the data was Winsorized, capping at the 5th and 95th percentile removed all 0 records and extreme positive records.

Setting up the 'target' variable

The problem has been constructed such that customer behavior over 52 weeks is used to predict whether a customer will purchase a given item in the following week. This is illustrated below:



In addition it was decided that all 'inactive' customers would be removed from the modeling, these were defined as all customers without a transaction between weeks 2008-07 and 2008-15. Removing inactive customers left 3,519 out of the original 5,000. These active customers were then split into training and test sets at a ratio of 70/30.

The first step of setting up the target variable was to identify all active customer and item purchase combinations in week 2008-16 (shown below):

	CUST_CODE	PROD_CODE	TARGET
2	CUST0000307323	PRD0900939	1
4	CUST0000307323	PRD0901465	1
8	CUST0000634693	PRD0903074	1
9	CUST0000634693	PRD0903399	1
11	CUST0000307323	PRD0903542	1

All non-target customer and item combinations were then added to the dataset, this is all active customers that did not purchase an item in week 2008-16. This created a very imbalanced dataset:

```
0.0    10502459
1.0       14551
Name: TARGET, dtype: int64
```

In order to address the imbalance the non-target records were down-sampled proportionately to the number of target records for each product creating a balanced sample:

```
1.0    14551
0.0    14551
Name: TARGET, dtype: int64
```

The final target dataset is now a list of customers and items with the target variable:

		CUST_CODE	PROD_CODE	TARGET
PROD_CODE				
PRD0900001	**1235174**	CUST0000203043	PRD0900001	0.0
	7187554	CUST0000240308	PRD0900001	0.0
	10458374	CUST0000285663	PRD0900001	0.0
	9045004	CUST0000620533	PRD0900001	1.0
	10133854	CUST0000728571	PRD0900001	1.0

Feature Engineering

To build the classification models features were created that described customer purchasing behavior over the 52 week observation period. Examples and descriptions of the types of features created are shown in the table below:

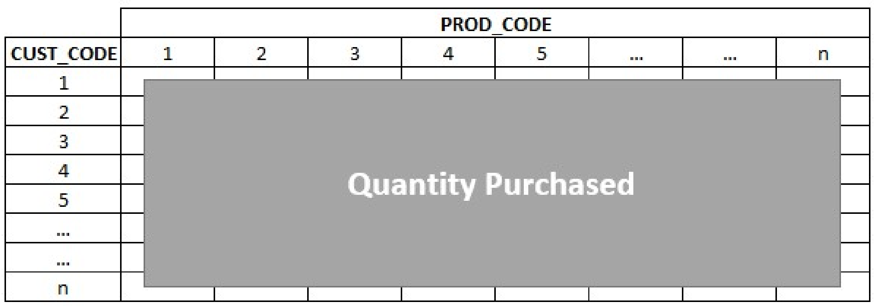
<table>
<tr><th>Features</th><th>Feature Description</th><th>Example of a feature</th></tr>
<tr><td>Spend/visits/quantity</td><td>Total spend, visits and quantity purchased by each customer in the last 1, 8, 26 and 52 weeks for each item and level of the product hierarchy that the item belonged to</td><td><table><tr><th></th><th>CUST_CODE</th><th>PROD_CODE</th><th>SPEND_PROD_CODE_52</th></tr><tr><td>0</td><td>CUST0000001052</td><td>PRD0902277</td><td>1.59</td></tr></table>Customer 1052 spent $1.59 on item 902277 in the last 52 weeks</td></tr>
<tr><td>Change in spend/visits/quantity</td><td>Measures the change in customer spend, visits and quantity e.g. what is the ratio of spend in the most recent week over spend in the last 8 weeks?</td><td><table><tr><th></th><th>CUST_CODE</th><th>PROD_CODE</th><th>CHNG_VISITS_PROD_CODE_8_52</th></tr><tr><td>0</td><td>CUST0000001052</td><td>PRD0902277</td><td>0.333333</td></tr></table>Customer 1052 spent 1/3 of their total spend on item 902277 in the last 8 weeks</td></tr>
<tr><td>Time since last purchased</td><td>Measures the time since the customer last purchased the item. Also measures the median time between purchases of that item for that customer and overall for all customers</td><td><table><tr><th>CUST_CODE</th><th>PROD_CODE</th><th>TIME_BTWN_MEDIAN_CUST_PROD_CODE</th><th>TIME_BTWN_MEDIAN_OVERALL_PROD_CODE</th><th>TIME_BTWN_LAST_PROD_CODE</th></tr><tr><td>CUST0000001052</td><td>PRD0902277</td><td>169.0</td><td>8.0</td><td>98</td></tr></table>Customer 1052 last purchased item 902277 98 days ago, historically they have purchased the item every 169 days, customers on average purchase this item every 8 days</td></tr>
<tr><td>Time since last purchased ratio</td><td>Obtains the ratio of the time since the customer last purchased the item to their median time between purchases and to the average customer</td><td><table><tr><th>CUST_CODE</th><th>PROD_CODE</th><th>TIME_BTWN_RATIO_CUST_PROD_CODE</th><th>TIME_BTWN_RATIO_OVERALL_PROD_CODE</th></tr><tr><td>CUST0000001052</td><td>PRD0902277</td><td>0.579882</td><td>12.250</td></tr></table>Customer 1052 bought product 902277 98 days ago compared to their average of 169 days giving a ratio of 0.58. Compared to the average customer 98 days is 12.25 times longer than the average customer purchase cycle for that item</td></tr>
<tr><td>Spend/visits/quantity by basket segment</td><td>Creates the total spend/visits/ quantity and % of total for each segment classified at the basket level e.g. weekday/weekend, time of day, basket price sensitivity, basket size, basket type, basket dominant mission, store type</td><td><table><tr><th>CUST_CODE</th><th>BASKET_PRICE_SENSITIVITY_SPEND_CUST_CODE_LA</th></tr><tr><td>CUST0000001052</td><td>1.06</td></tr></table>Customer 1052 spent $1.06 on baskets classified as LA (Lower Affluence)</td></tr>
<tr><td>Customer segments</td><td>Appends the price sensitivity and lifestage customer segments</td><td><table><tr><th>CUST_CODE</th><th>CUST_PRICE_SENSITIVITY</th><th>CUST_LIFESTAGE</th></tr><tr><td>CUST0000203043</td><td>MM</td><td>YA</td></tr></table>Customer 1052 has a price sensitivity classification of MM (Mid-Market) and a lifestage classification of YA (Young Adult)</td></tr>
</table>

The Feature Engineering module is complex and functions could be prone to error, to mitigate a testing script has been written for PyTest that tests the output of the functions against an example customer where features have been manually verified as being correct. This will allow for any future changes to the functions to be verified as well as confirming that the functions are processing the data as intended.

Creating customer and item features with ALS

In addition to the features above, customer (user) and item features were created using an ALS model. The model generated vectors that represented customers and items in a latent space.

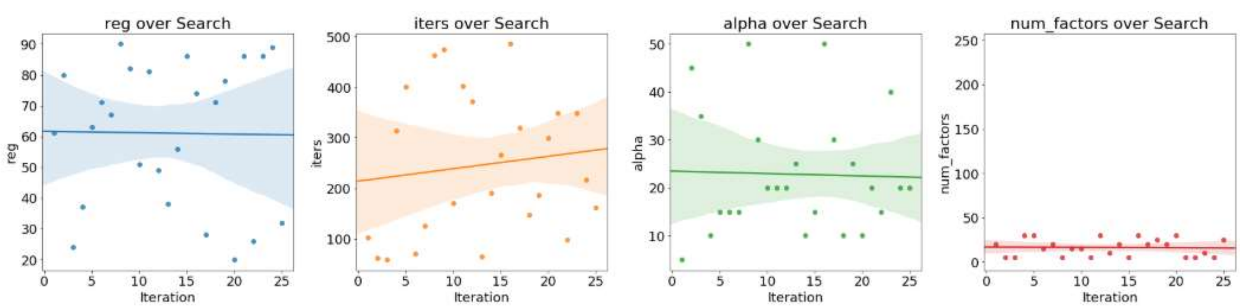
To generate these features a sparse matrix was created containing customers, items and the quantity of units purchased (representing implicit feedback). An example of what this matrix would look like is shown below:



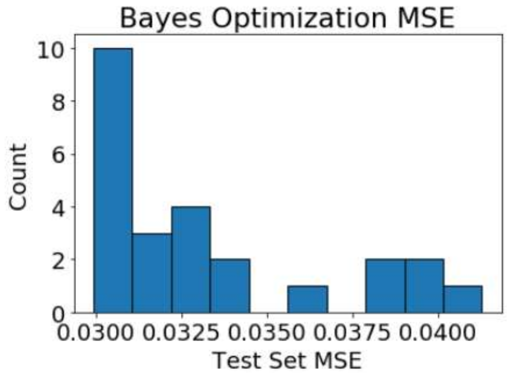
As expected, this matrix is very sparse (98%) as most customers only purchase a small number of the available products.

To avoid data leakage the matrix was created on the same 52 weeks used to generate the other features described above.

The algorithm performance was measured using MSE and was tuned utilizing a Bayesian Hyperparameter search. The regularization parameter, the alpha parameter, the number of latent factors and number of iterations of ALS were all tuned. The charts below show the parameter search over the number of iterations:



The chart below shows the MSE histogram for all searches:



The best hyperparameters were found to be alpha = 20, iterations = 216, number factors = 5, regularization = 89. This gave an MSE of 0.03 in the test set.

The table below shows an example of these final factors:

factor_0	factor_1	factor_2	factor_3	factor_4	CUST_CODE
0.015998	0.004285	0.011275	0.010576	0.012515	CUST0000000013

factor_0	factor_1	factor_2	factor_3	factor_4	PROD_CODE
0.326456	0.230012	0.486178	0.353130	0.103477	PRD0900121

These factors were then added to all other features to be used in the classification models.

Implementation

The feature engineering process created over 400 features. In order to understand which features had stronger predictive power and the relationship between features, the following tests were run:

1.) Squared correlation between the target variable and each numeric feature
2.) Random Forest Feature Importance rank
3.) Absolute regression coefficients using L1 and L2 regularization
4.) Recursive Feature Elimination
5.) Feature agglomeration to create clusters of 'similar' features

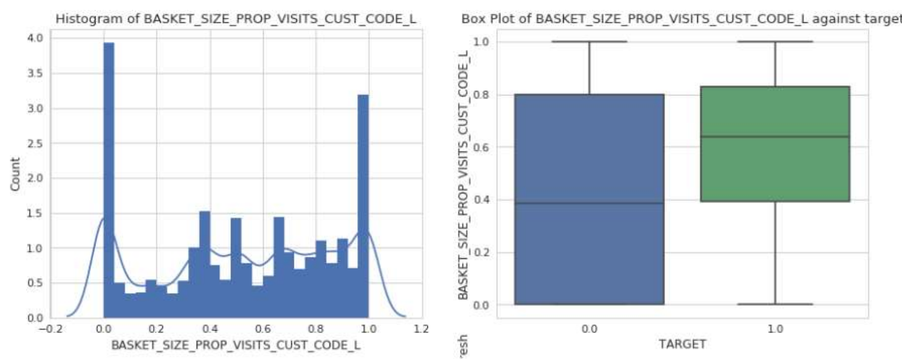
Using each of these tests as a guide the top features were selected from each of the variable clusters generated using Feature Agglomeration. This made sure that the strongest features were selected while limiting the amount of correlation between the selected features.

Once the top features were selected the following EDA was run:

1.) Checks for missing values
2.) Key statistics for each feature:

	BASKET_SIZE_PROP_SPEND_PROD_CODE_M
count	29098.000000
mean	0.195436
std	0.195220
min	0.000000
25%	0.050761
50%	0.157895
75%	0.254098
max	1.000000

3.) Histograms of each feature and boxplots against the target:



4.) The squared correlation between each of the selected features:

```
Top Absolute Correlations

SPEND_PROD_CODE_30_52                               SPEND_PROD_CODE_20_52                         0.802762
CHNG_VISITS_PROD_CODE_30_1_52                       CHNG_VISITS_PROD_CODE_40_1_26                 0.695785
BASKET_PRICE_SENSITIVITY_SPEND_CUST_CODE_UM         USER_factor_1                                 0.677874
BASKET_DOMINANT_MISSION_PROP_VISITS_CUST_CODE_XX    BASKET_TYPE_VISITS_CUST_CODE_XX               0.673235
CHNG_SPEND_PROD_CODE_40_8_52                        CHNG_QUANTITY_PROD_CODE_40_26_52              0.666079
CHNG_QUANTITY_PROD_CODE_40_26_52                    TIME_BTWN_MEDIAN_OVERALL_PROD_CODE_40         0.656926
SPEND_PROD_CODE_20_52                               VISITS_PROD_CODE_20_52                        0.650766
BASKET_PRICE_SENSITIVITY_SPEND_CUST_CODE_LA         USER_factor_2                                 0.632453
BASKET_SIZE_QUANTITY_CUST_CODE_S                    BASKET_TYPE_QUANTITY_CUST_CODE_Small Shop     0.631735
```

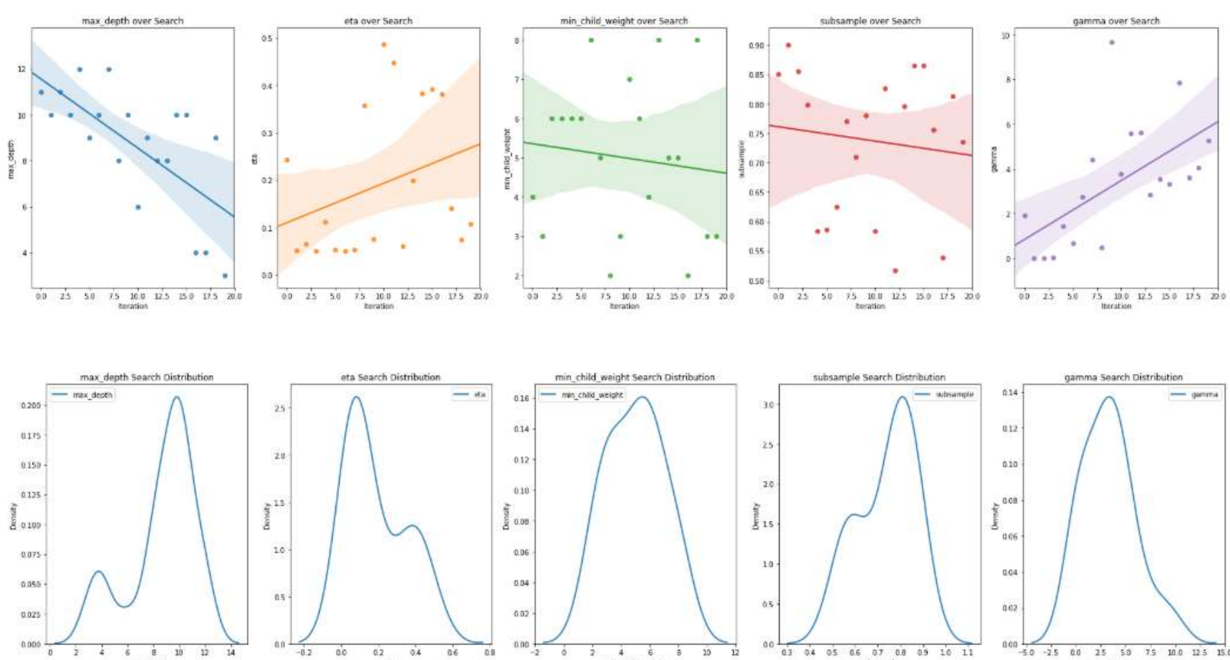
Before running any of the classification algorithms, data was standardized and all categorical features transformed using one-hot encoding. In addition the test set was further split to create a validation set, this allowed the algorithms to be tuned on the test set while holding out a validation set that was never used in any of the training steps.

Refinement

Each of the algorithms trained were tuned using Bayesian hyperparameter optimization in Sagemaker. The table below shows the algorithms trained, the hyperparameter search space and the optimal hyperparameters after tuning.

Model	Hyperparameter Search Space	Optimal Parameter
Logistic Regression	**L1 Regularization:** 0 - 10 (continuous)	**L1 Regularization:** 8.3
Random Forest	**Max Depth:** 3 - 12 (integer) **Num Estimators:** 50 - 1000 (integer) **Max Features:** (auto, sqrt, log2)	**Max Depth:** 12 **Num Estimators:** 779 **Max Features:** auto
XGBoost	**Max Depth:** 3 - 12 (integer) **eta:** 0.05 - 0.5 (continuous) **Min Child Weight:** 2 - 8 (integer) **Subsample:** 0.5 - 0.9 (continuous) **Gamma:** 0 - 10 (continuous)	**Max Depth:** 10 **eta:** 0.01 **Min Child Weight:** 6 **Subsample:** 0.8 **Gamma:** 4.8
DNN	**Num Layers:** 1 - 10 (integer) **Hidden Units:** 1 - 512 (integer) **Dropout Rate:** 0 - 0.2 (continuous) **Momentum:** 0.8 - 1(continuous) **Batch Size:** 32 - 500 (integer) **Learning Rate:** 0 - 0.2 (continuous) **Initialization:** Random Normal, Random Uniform, He Normal, He Uniform	**Num Layers:** 3 **Hidden Units:** 4 **Dropout Rate:** 0.1 **Momentum:** 0.96 **Batch Size:** 358 **Learning Rate:** 0.08 **Initialization:** He Uniform

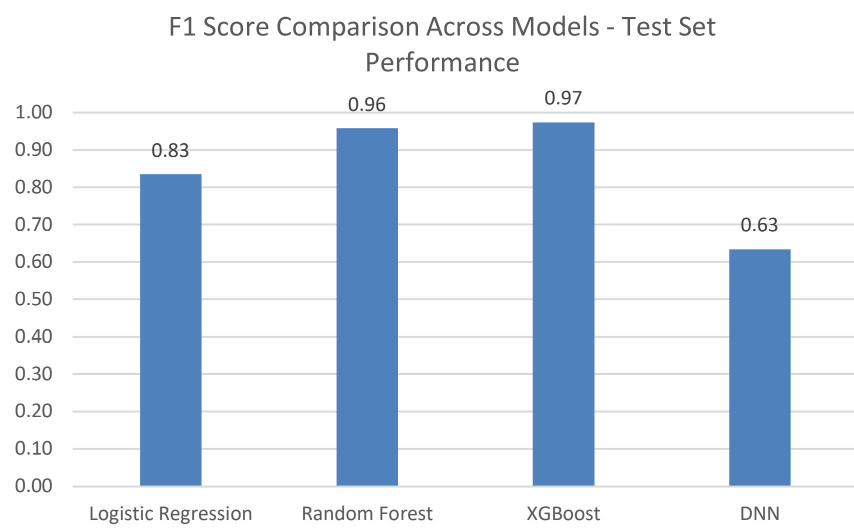
After each training run the hyperparameter search was plotted. The charts below show examples of the output for the XGBoost model:



Results

Model Evaluation and Validation

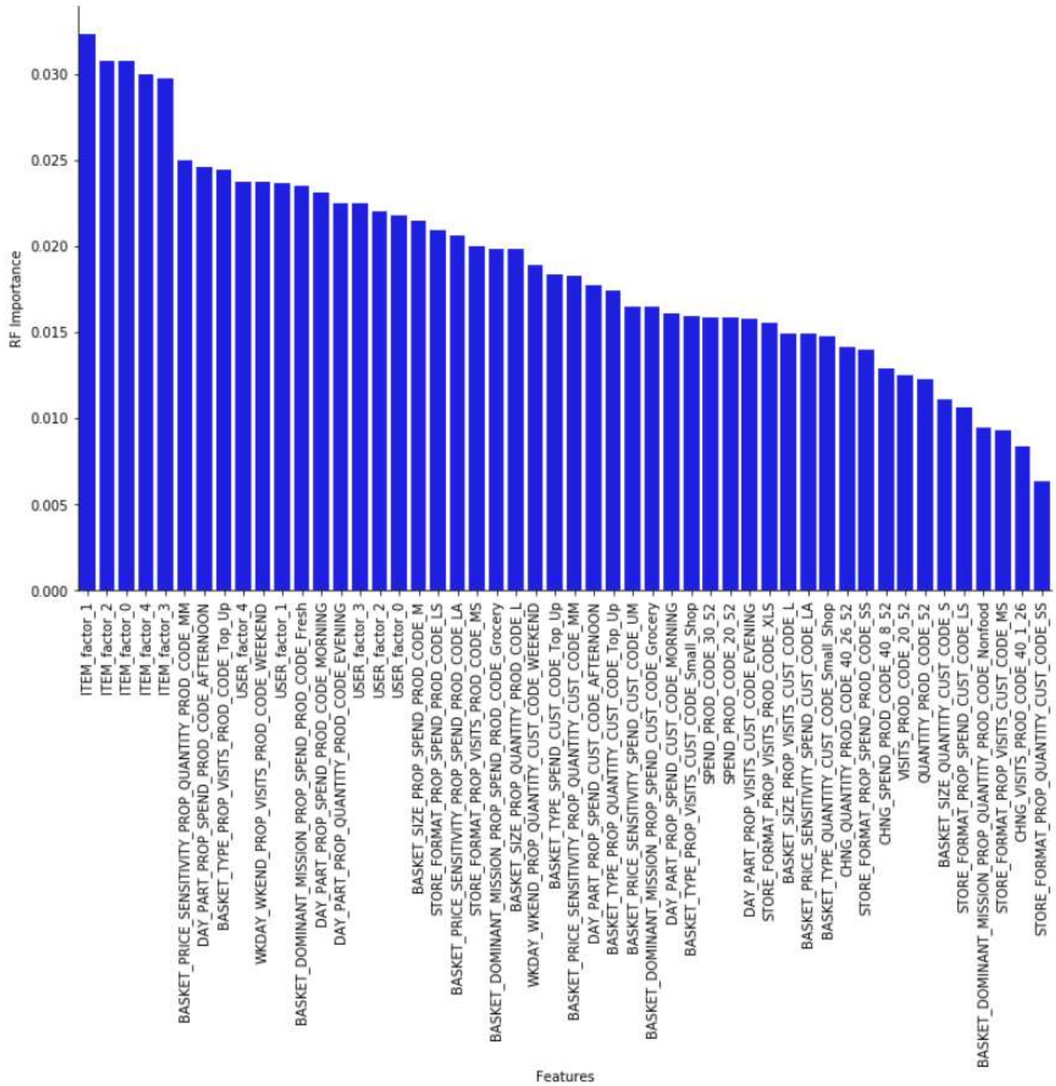
The chart below shows the final F1 score for each of the models on the test set. Note that this is a subset of the data that was not used in the training process including the hyperparameter search:



Tree based models are clearly the best performers on the test set. Despite a vast hyperparameter search across 500 epochs the DNN was not able to perform strongly. This may be in part due to the smaller size of the dataset used in this project.

In order to gain a more complete understanding of the drivers of predictions in the final model, the Random Forest model was fit with the optimal hyperparameters based on the Bayesian search. Random Forest Importance metrics and Shap values were then generated as illustrated in the below charts:

RF Importance:

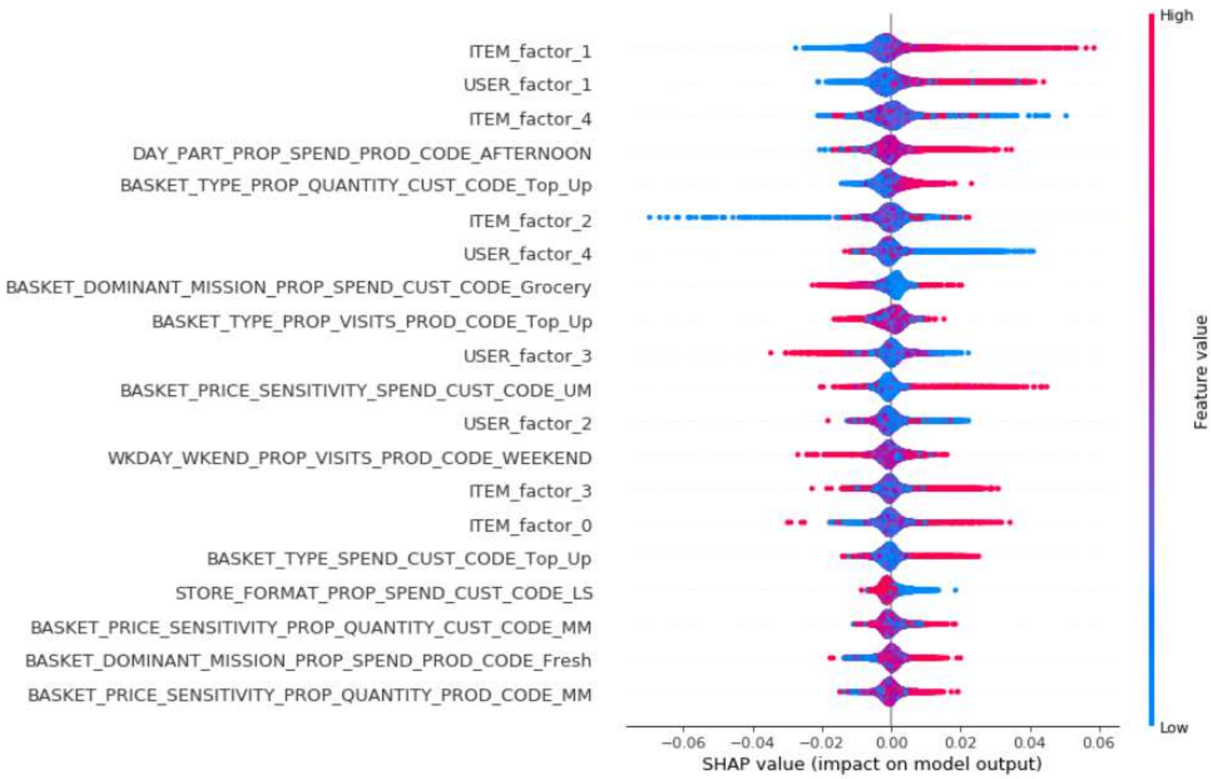


The RF Importance scores show the following key themes in terms of the importance of the features:

- User and item factors play a very significant role in the final prediction, likely tree based models are performing best as they are allowing for the explicit interaction between these features
- Features related to the item such as whether it is typically purchased in “mid-market” (MM) baskets, Top-Up baskets and time of day
- Customer spend on the item and at a various levels of the hierarchy over varying time periods
- How the customer shops e.g. the store format they shop, the type of baskets they typically shop for e.g. “Small Shops” and the time of day that they shop

<u>Shap values:</u>

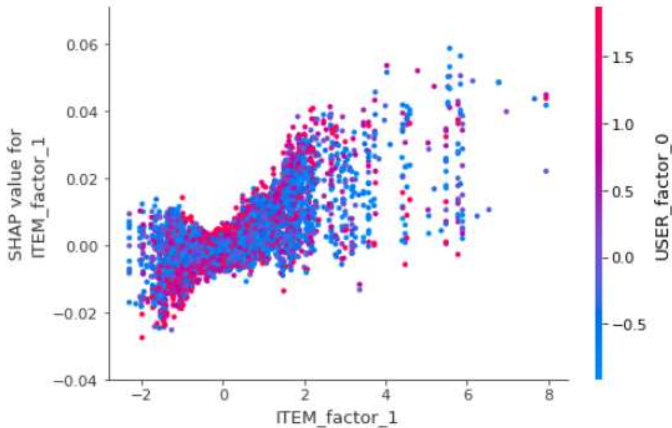
The Shap values help to illustrate how these features are influencing predictions, the chart below shows the top features:



Some observations from the plot are provided below:

- User and item factors generally positively impact the predictions with higher values for the features more likely to lead to positive predictions. The exception is item factor 2 which generally has a more negative influence on the prediction
- Many features polarize the prediction, for example, the proportion of a customers' spend on grocery items can have both a positive and negative influence on the prediction where the value is high, this indicates that there is likely an interaction between this feature and others fitted

Shap dependence plots illustrate the interaction between features, especially the user and item factors:



The plot shows that were the item factor is low and the user factor is high there is a negative influence on the prediction. Conversely where the item factor is high and the user factor is high it is more likely there will be a positive influence on the prediction.

Justification

Comparing to the available benchmark (F1 Score ~0.4), all models are significantly outperforming in the validation data. It is difficult to make a direct comparison as the stronger performance is likely due to the make-up of the dataset being used i.e. it is likely that there is much stronger evidence of repeat purchasing by customers in this dataset than what is observed in the benchmark data.

While the F1 Scores are very high significant steps have been taken to avoid overfitting and data leakage e.g.

- All data was split into training and observation periods and no data from the observation period was used in the training i.e. there is no opportunity for data leakage
- Models were fit on a training set and tuned on a test set utilizing Bayesian hyperparameter tuning methods
- Performance metrics were taken from a completely separate validation set of customers that was not used in any of the training steps

With access to more complete and larger datasets over different time periods (not provided in open source) it is likely that performance metrics would drop, however, given the strength of the fits for the tree models performance would likely still be strong and at minimum in the range of the benchmark performance.

References

[1] Example of store utilizing personalized recommendations:

https://www.krogerprecisionmarketing.com/customers-first.html

[2] Data can be downloaded from:

https://www.dunnhumby.com/careers/engineering/sourcefiles

The actual data utilized was from the “Let’s get sort of real” section, specifically the data from a randomly selected group of 5,000 customers

[3] URL for the Instacart Market Basket Analysis Kaggle challenge:

https://www.kaggle.com/c/instacart-market-basket-analysis/leaderboard

[4] Details of a top solution for the Instacart Market Basket Kaggle challenge:

https://www.kaggle.com/c/instacart-market-basket-analysis/discussion/38100