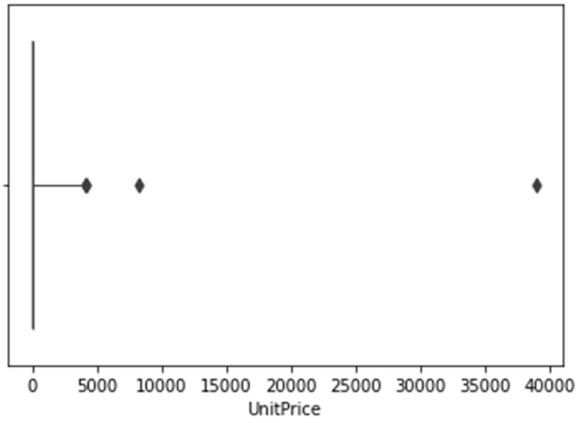
Harikrishnan V

Approach to solve the Hackathon Problem

This was my first Hackathon. I wanted to do maximum Exploratory Analysis without directly jumping into modelling. Once I started exploring the data and finding the patterns within the data, I realised that most the rows to predict have a low UnitPrice and very few have exceptionally high values.

Percentile of train data	UnitPrice at Percentile
50.0	1.95
75.0	3.75
90.0	6.75
95.0	8.50
99.0	15.00
99.5	18.00
99.9	50.00



There was one extreme outlier and a few other high ones.

On further exploration to understand any pattern in the spread of prices in the data, I finally recognized a pattern by StockCodes.

Number of Unique Prices	StockCode	Number Of StockCodes	Total Rows In Test	% Rows In Test
1	[0, 3163, 3165, 3168, 3170, 3...	1119	8084	6.62
2	[2519, 2722, 2423, 1971, 1976,	1249	31140	25.51
3	[2611, 2601, 3201, 3199, 2573,	848	48686	39.89
4	[8, 454, 360, 2709, 2404, 3677,	265	16373	13.42
5	[362, 1417, 1546, 2593, 3125, .	64	6563	5.38
6	[1514, 1555, 2626, 2658, 2622,	36	5088	4.17
7	[1473, 1543, 2630, 146, 917, ...	24	3353	2.75
8	[2624, 76, 3235, 1017, 2599]	5	1346	1.10
9	[3280, 399, 2778]	3	612	0.50
11	[393]	1	178	0.15
12	**[3678]**	**1**	**4**	**0.00**
14	**[3680]**	**1**	**2**	**0.00**
49	**[3679]**	**1**	**28**	**0.02**
55	**[3683]**	**1**	**356**	**0.29**
145	**[3681]**	**1**	**145**	**0.12**

I grouped the train data by Number of Unique Prices in a StockCode and realised that upto 11 unique prices, there were lots of rows relative to unique prices and could be modelled separately. The other 5 StockCodes (3678,3680,3679,3683,3681) were the ones with maximum uncertainly and the high value outliers!

StockCode	Unique Prices	UnitPrices	Count of each UnitPrice	No. of descri ptions	Weighted UnitPrice
3674	2	[0.29, 1.25]	[1, 1]	1	0.770000
3675	1	0.29	1	1	0.290000
3676	2	[15.0, 0.001]	[6, 1]	1	12.857286
3677	4	[50.0, 18.0, 25.0, 150.0]	[94, 2, 1, 1]	1	50.112245
3678	12	[52.24, 361.59, 849.93, 15.96, 13.01, 1100.44,...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	1	496.340833
3679	49	[33.75, 102.24, 14.88, 84.8, 434.51, 26.93, 14...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	1	87.606735
3680	14	[1500.36, 901.58, 1270.06, 16.46, 1599.26, 638...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	1	781.587857
3681	145	[7.95, 0.39, 0.1, 10.0, 0.16, 21.95, 1.0, 1.95...	[16, 14, 12, 10, 8, 8, 7, 7, 7, 6, 6, 5, 5, 5,...	1	351.579969
3682	2	[0.001, 0.0]	[3, 1]	1	0.000750
3683	55	[18.0, 2.9, 15.0, 5.75, 40.0, 20.0, 28.0, 2.7,...	[506, 96, 92, 72, 6, 4, 3, 3, 3, 3, 2, 2, 2, 2...	1	27.428512

InvoiceDate was converted to a float as days elapsed since 2010-1-1

For the StockCodes with just 1 unique value, I simply mapped and predicted those values in the test set.

(8084 rows done). Total 8084 out of 122049 rows done.

Next, I tried many models and concluded that the XGBRegressor gave the best results on this dataset. InvoiceNo was not used. 'StockCode','Description','CustomerID' and 'Country' were encoded as categorical. I made 9 different models, each with its best hyperparameter settings, for each set of StockCodes with unique values from 2 upto 11, ie. (2,3,4,5,6,7,8,9,11).

I also had a table with the unique prices of each StockCode. After the prediction result from these 9 separate models, I used a function and approximated each prediction to the closest unique UnitPrice for that particular StockCode.

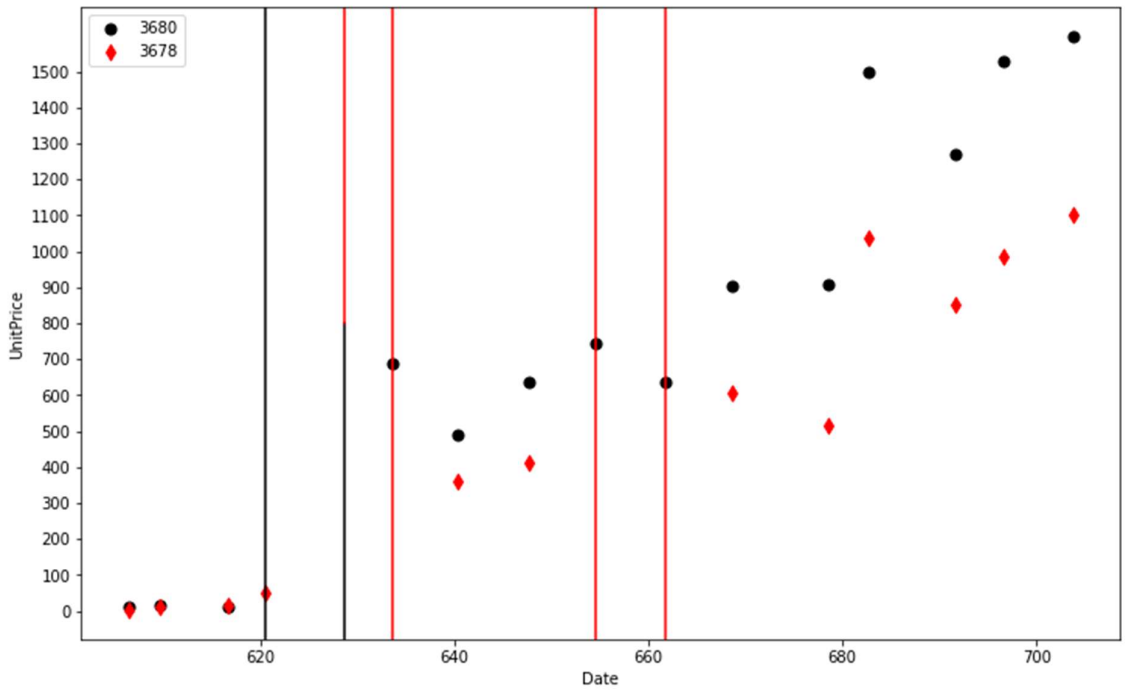
(113339 rows done). Total 121423 out of 122049 rows done.

There were 91 rows in test set with a StockCode not present in train data. For them I approximated the UnitPrice to the weighted price of the closest StockCode in train data.

(91 rows done). Total 121514 out of 122049 rows done.

Now only 535 rows remain to be predicted!

On further exploration I found that StockCodes 3678 and 3680 were from only one Customer(14096) and that the prices had a strong correlation with date.



The vertical lines show the position of the test points to predict.

Created a feature ‘month’ by combining month number and year number as a string from InvoiceDate. (13 categories)

I ran a model on the combined 3678 and 3680 StockCode points from train data and predicted the test values. 'Quantity' and 'Date' were used as numeric and 'month' and 'StockCode' as categoric.

(6 rows done). Total 121520 out of 122049 rows done.

Now only 3 StockCodes (3679,3681,3683) and 529 rows remain to be predicted.

The next last 3 StockCodes were the most tricky and time consuming. In my exploratory analysis I observed that Stocks 3679 and 3683 had extreme outliers and that 3679’s outlier matched as a pair(+1 & -1 quantity from same customer in the same day) to a test entry for Stock 3681! So these was a possibility for mixing. Also I observed that a lot of high value transactions occurred in pairs, ie. The same customer usually in the same day(not always) having the same + and – quantity for 2 transactions with same UnitPrice. It could be handpicked but I wanted to make a model to detect such pairs and predict them accurately. I made a dataset with the full 3681 data combined with 3679 and 3683 data where the UnitPrice z-score exceeded 3. I made a custom algorithm combined with an XGBRegressor to predict such pair values. Detailed code can be checked in my script. 'StockCode', 'InvoiceNo', 'Quantity' and 'Date' was used on a custom built train and test set derived using the dataset created earlier. 'InvoiceNo' was first label encoded and then OneHot encoded along with 'StockCode' and 'Quantity'.

27 rows were predicted using this model.

Now I felt the need to make new features to predict the remaining 3681 data and the other 2 StockCode rows. Calculated monthly customer sales for all non high value StockCodes by combining train and test data already predicted and stored in a dataframe.

Further, more features were created:

'hour group', 'month group', 'weekday group', 'monthstart', 'differet country' (13&14 for 3683 and 35 for others), 'monthly sales customer', 'total sales customer', 'days visited customer', 'months visited customer', 'invoice nos customer', 'transac nos customer', 'avg spend per transact customer', 'avg spend per invoice customer', 'avg spend per day customer', 'avg spend per month customer', 'customer since'

Features of all customers was stored in a dataframe.

For StockCode 3679, there was one extreme outlier and this point also had a matching pair(-1 and +1 quantity from same customer at similar time) row in the test data for StockCode 3681 as mentioned earlier! That train row was removed and modelling was done on the 3679 data and test values predicted. Very high quantity rows and predictions that went below 0 were predicted with corresponding median values from train set.

(28 rows done). Total 121548 out of 122049 rows done.

For StockCode 3683, the majority prices were 15,18,28,40. Countries 13 and 14 were predominant and have predominantly Price 18. Here also there is one extreme outlier which again has an identical pair in test data in StockCode 3683 itself. It was removed and modelling was done on the data. Predicted values in the range on 12 to 45 were approximated to the closest among [15,18,28,40]. High quantity and predictions less than 0 rows were approximated with median price from train data. Rows of Countries 31 & 32 were capped at their maximum value (40) from train data.

(356 rows done). Total 121904 out of 122049 rows done.

Only 120 rows remain to be predicted in StockCode 3681. These are the most unpredictable rows with significant high value entries. On exploring the data the one extreme outlier close to 40000 doesn't have a pair in the test data. This row was removed. The train data used here to capture maximum trends is all rows of categories >=3678 in the train data combined with the high 3681 pair values predicted in the test set with my earlier model.

The best model was run and the values for the remaining 3681 Stock rows were predicted. High variance in prices were only in Quantities -2,-1,1. Other high quantity(+ve and -ve) rows and rows with prediction less than 0 were approximated with corresponding median values from the train data of Stock 3681.

Further, a few customers were identified having low transactions and quantities and who were judged to be low value spenders in this category in my exploratory analysis with all made features. These people's transactions were approximated with low values in 2 groups (1 and 5).

3681 Stock Customers with no negative quantity and high overall quantity with atleast one Quantity=1 in the test entry would have a relation with their high quantity spends. Their 1 quantity transactions were predicted with their average high quantity transaction value with custom code.

Customers who ended their transactions with the store with a high proportion of final transactions in StockCode 3681 had a trend of having a maximum earlier monthly sale value as the UnitPrice in this transaction. Such transactions were appropriately predicted with custom code.

Customers with a high negative value for total sales would have approximately that value as UnitPrice in their 1 quantity 3681 entry, to even out the cashflow. Such customers' transactions were appropriately predicted.

For pairs like mentioned above, which had both members in the test set, their predicted values from my model were averaged and assigned.

In this hackathon, as RMSE was the metric, I wanted to predict as many points as close as possible in the test set. I sought for more out of the box ideas to use the available resources to improve my score further..

Hacks to further reduce RMSE

Now, having reached the plateau of model performance, I pondered on how to further reduce the RMSE as the project goal here was of a low RMSE in the Hackathon. The only resources I had were the Train and Test set and 10 daily submissions to evaluate performance. I thought of any way to use the daily submissions to my advantage and realised that there were only a few dozen rows in the entire test set of 122049 rows that I was still unsure of, like 3 rows of Customers who had no transactions in train data, a pair transaction(+1 & -1 quantity) with both rows in test set, Customers with high sales values, Customers with high occurrence in 3681 Stock in test set etc.

These all identified rows amounted to only a few dozens. It was given that the public leader board scoring was being done on 70% test data. I got an idea that if I make a submission with the value of one interesting point changed and it happens to be in the 70% data and the RMSE score changes, then I can predict the value of a point by calculating the difference in sum of squared errors by using the the equation of RMSE. I made a function to do this calculation and tried to predict the value of a few more shortlisted points.
