Round 02 – Semi Final

DataStorm 3.0

Sales Predictor with Minimal Under Forecasting Error

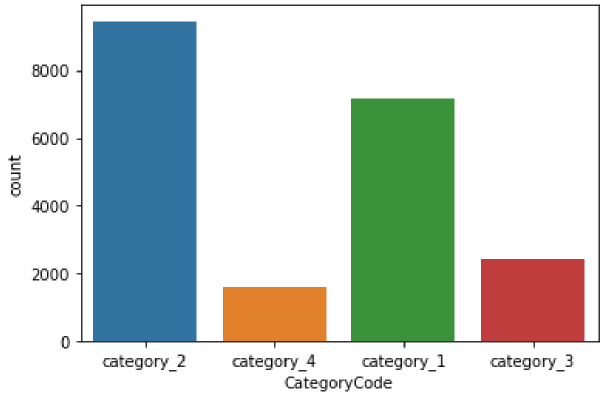
THE TEAM DECISION_MAKERS

Kajanan Selvanesan
Laksika Tharmalingam
Ashokkumar Susitharan

GitHub: https://github.com/kajanan1212/Sales-Predictor

Step 01: Descriptive Analysis

Dataset: Training [shape (20651, 4)]

<table>
<tr><th>Column</th><th>Description</th></tr>
<tr><td>CategoryCode</td><td>It indicates the category of an item. There are 4 categories.
Number of items belongs to each category:

</td></tr>
<tr><td>ItemCode</td><td>This is a unique code of an item. There are 197 items.</td></tr>
<tr><td>DateID</td><td>It indicates the sold date of an item.
Start: 1 st October 2021
End: 17 th February 2022</td></tr>
<tr><td>DailySales</td><td>Daily sold count for sold items.
<pre>count   20651.000000
mean   7.295482
std   14.471197
min   1.000000
25%   2.000000
50%   3.000000
75%   7.000000
max   434.000000
Name: DailySales, dtype: float64</pre></td></tr>
</table>

Dataset: Validation [shape (373, 5)]

Similar to the Train dataset. But the main difference is that Week and WeeklySales have been included instead of DateID and DailSales. And there is an additional column for OnPromo (Flag suggesting whether the item was on promotion).

Dataset: Test [shape (377, 5)]

Similar to the Validation dataset. But there aren't any values for WeeklySales. We have to predict them.

Dataset: Promotion [shape (314, 6)]

The data set consists of promotional details related to items. There are columns like ItemCode, PromotionStartDate, PromotionEndDate, DiscountValue, DiscountType (Percentage/Amount), and SellingPrice.

	count	mean	std	min	25%	50%	75%	max
ItemCode	314.0	584578.353503	469041.320382	7666.0	64978.0	755584.0	1071119.5	1101571.0
DiscountValue	314.0	12.786624	5.343386	5.0	10.0	10.0	15.0	30.0
SellingPrice	314.0	125.923567	85.447332	40.0	70.0	75.0	190.0	370.0

Step 02: Data Pre-processing

1. Data Integration

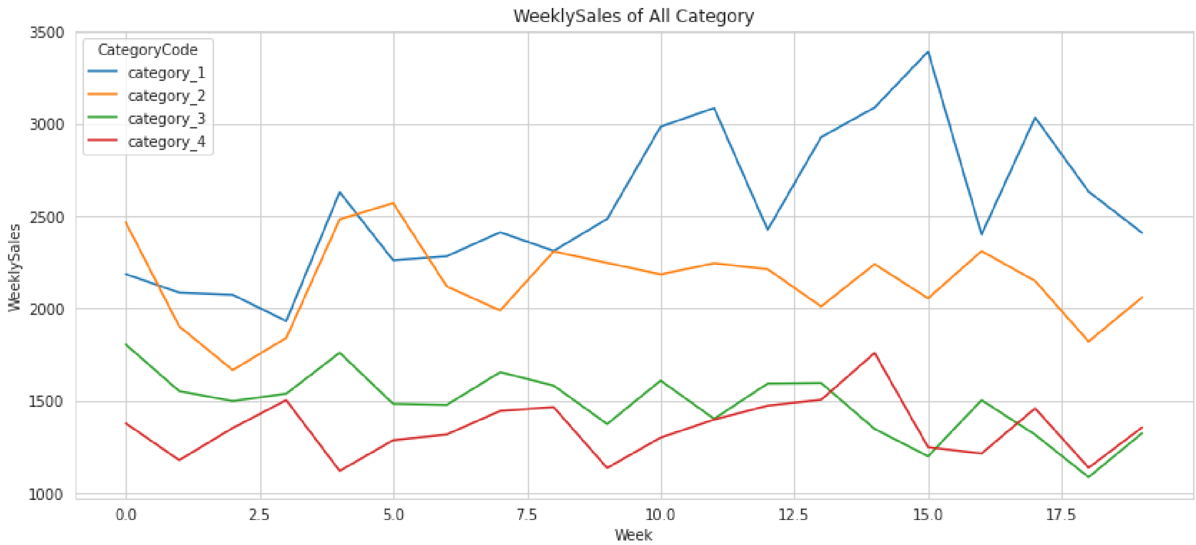
We integrated the Promotion dataset into the other three datasets. However, before integrating them, we did some Feature Engineering to Promotion dataset. We applied feature transformation for the PromotionStartDate, and PromotionEndDate to convert them into the PromotionStartWeek, PromotionEndWeek. From this, we created a table with ItemCode that had a promotion, including after how many weeks from 01-10-2021 that promotion was given. In addition to the given and derived features, we created some more features like DiscountMoney, DiscountPercentage, etc.

2. Missing Values Handling (There is no any missing values)

Step 03: Feature Engineering

1. Feature Transformation

In the training dataset, we converted the DateID into the number of passed weeks from the date 01-10-2021. In the other two dataset, we did the same thing for the Week attribute. And in the training dataset, we had calculated the WeeklySales from the DailySales with the help of a newly created Week column. Below graph is based on the training dataset. As we mentioned earlier, we had done feature transformation in the Promotion dataset also.



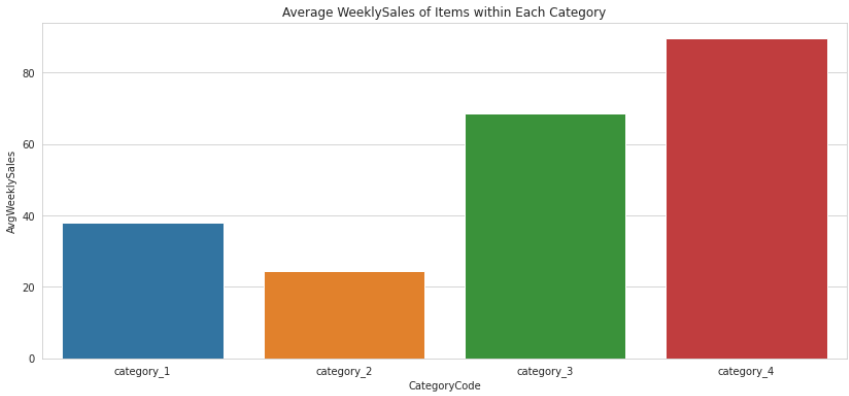
2. Feature Creation

We created features like OnPromo, DiscountMoney, DiscountPercentage, CanHavePromo, HasFestival and HasHoliday.

OnPromo: Using the pre-processed Promotion dataset, we created the OnPromo feature in the training dataset.
DiscountMoney, DiscountPercentage: Using DiscountValue, DiscountType (Percentage/Amount), and SellingPrice, we calculated them in the Promotion dataset.
CanHavePromo: It indicates whether an item can have a promotion or not.
HasFestival, HasHoliday: They indicate whether a week contained any festivals/holidays or not.

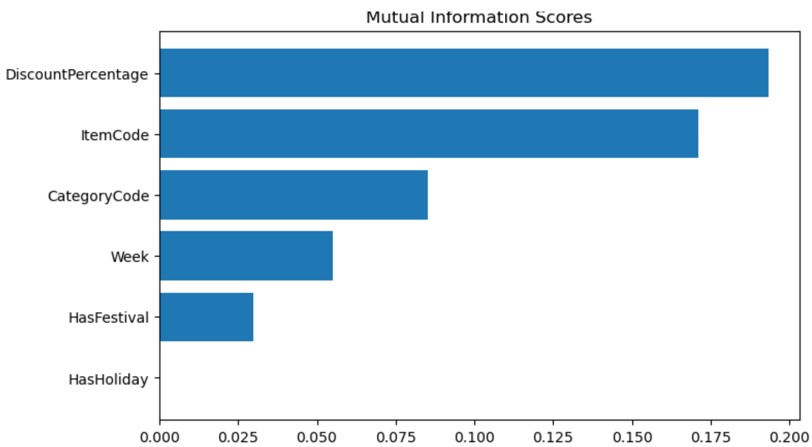
3. Feature Encoding

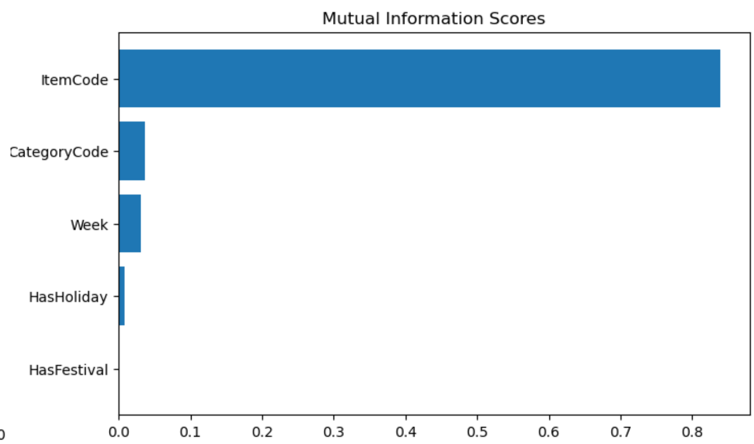
We encoded the categorical feature CategoryCode using the *Ordinal Encoding* technique. We had noticed that items within some CategoryCode have higher mean WeeklySales than that of items within some other CategoryCodes. Therefore, we concluded that there is an order within CategoryCodes. Based on the above-mentioned mean value, we assigned the labels to the CategoryCodes. CategoryCode with a higher mean value took a higher value of a label, So on.



4. Feature Selection

We used the wrapper method to select the best subset of the features to reduce dimensionality. As a result, we had to drop some features like CanHavePromo, DiscountValue, DiscountType, DiscountMoney, and SellingPrice. Finally, we ended up with the following features.





Step 04: Model Building

Initially selected models for this project are *LinearRegression, KNeighborsRegressor, DecisionTreeRegressor, RandomForestRegressor*, and *SVC* because these are pretty good for regression problems, simplest models, and interpretable.

Here, we tried to build two models separately.

A. One model for items which are not on promotion- Model A
B. Another model for items which are on promotion- Model B

Therefore, we had created a separate dataset for each model with the necessary features and data. Then we trained each above-selected model two times using both the datasets.

Step 05: Evaluate The Models on Both The Datasets Using Validation Dataset. (We used Total MAPE and MAPE (Under Forecast) as our evaluation metrics.)

$$\text{Total MAPE} = \frac{\text{Sum}(|\text{Predicted Sales (I,W)} - \text{Actual Sales(I,W)}|)}{\text{Sum}(\text{Actual Sales(I,W)})}$$

$$\text{X}_U = \text{Under Forecast Sales}(I, W)$$

$$X_A = \text{Actual Sales Under Forecast}(I, W)$$

$$\text{Under Forecast Error} = \sum |X_U - X_A|$$

$$\text{MAPE (Under Forecast)} = \frac{\text{Under Forecast Error}}{\text{Sum}(\text{Actual Sales(I,W)})}$$

where I is for each item, and W is for each week.

	Model	VTMAPE_score	VUMAPE_score
1	KNeighborsRegressor	35.616253	23.828105
3	RandomForestRegressor	40.069091	25.583789
2	DecisionTreeRegressor	44.560637	26.996514
0	LinearRegression	79.443903	50.738447
4	SVC	83.830186	82.591957

Model A

	Model	VTMAPE_score	VUMAPE_score
1	KNeighborsRegressor	40.743945	18.235294
3	RandomForestRegressor	50.701557	20.140138
4	SVC	62.283737	57.352941
0	LinearRegression	68.337176	43.211740
2	DecisionTreeRegressor	73.615917	24.913495

Model B

Here, VTMAPE_score means Total MAPE score for validation dataset and VUMAPE_score means MAPE (Under Forecast) score for validation dataset. You can note that *KNeighborsRegressor* gives better results in both the cases.

Step06: Find A Buffer for Each Model to Minimise The Under Forecast Error As Much As Possible While Maintaining A Lower Overall Error.

	Model	VTMAPE_score	VUMAPE_score	VBuffer
1	KNeighborsRegressor	36.073213	19.320434	5.187097
3	RandomForestRegressor	40.276284	21.337868	4.763651
2	DecisionTreeRegressor	45.390591	21.809303	6.135593
4	SVC	78.843076	69.477159	11.632530
0	LinearRegression	90.745051	42.701221	14.991065

Model A

	Model	VTMAPE_score	VUMAPE_score	VBuffer
1	KNeighborsRegressor	46.245675	12.612457	8.800000
4	SVC	54.152249	37.190023	16.916667
3	RandomForestRegressor	59.063841	14.164360	10.674000
0	LinearRegression	66.715764	28.481327	14.628347
2	DecisionTreeRegressor	77.479815	17.964245	9.333333

Model B

The VTMAPEs were increased in both the models when compared with previous cases. However, we had found the buffer for each model to minimise the under forecast error as much as possible while maintaining a lower overall error.

Step 06: Tuning for The Both KNeighborsRegressor And Trying Any Ensemble Method to Reduce Both The Errors in Both The Models.

Final best models and best scores for both the cases are

A. For Model A- *KNeighborsRegressor(p=1)*

	Model	VTMAPE_score	VUMAPE_score	VBuffer
0	KNeighborsRegressor	35.838639	19.578036	4.769106

B. For Model B- *BaggingRegressor(KNeighborsRegressor(n_neighbors=3, p=1, weights='distance'), random_state=42)*

	Model	VTMAPE_score	VUMAPE_score	VBuffer
0	BaggingRegressor	29.913348	10.882417	2.311066

Step 07: In The Test Dataset, Predict The Values For The WeeklySales Using Suitable Model And Add The Respective Buffer With The PredictedWeeklySales To Find The Final PredictedWeeklySales With Minimal Under Forecasting Error.

A. Buffer value for Model A = 4.769106
B. Buffer value for Model B = 2.311066