Info.7390 Project Report

Retail Data Analysis And Sales Prediction

Haoqi Huang,Yitong Liu and Pei yu

ABSTRACT

Today, online shopping is growing fast. To make a research of retail business by the data science technology, the whole project is divided into three parts: The online store income forecast, Real retail shop sale prediction based on random forest model, and BigMart retail sale prediction. In the first part, we cleaned and explored data first. Then we used RNN/LSTM to forecast the online store daily income. The second part, different csv files were merged and the prediction was made by Random Forest Model. The third part, we used five models try to understand the properties of products and stores, and to find out which part plays a key role in increasing sales.

INTRODUCTION

1.Motivation

Nowaday, the need of e-commerce is increasing rapidly. People go shopping and do other entertainments and it becomes even a threatening to physical stores. Thus, the major reason of choosing this retail topic is that we are driven by the curiosity to whether or not the traditional costume retail will survive from the attack of the new e-commerce. So, this phenomenon catch our attention and we start to search the information to find out the reason.

2.Background

E-commerce has a major impact on the retail industry in a variety of ways. Retailers have had to adapt to new technological demands from users to allow them to participate in the e-business world. It has revolutionized the way companies, regardless of size, do business. The implication was that the retail industry was in decline as e-commerce took over. Every traditional retailer now has web and mobile offerings, while many e-commerce companies, like Amazon, Warby Parker, and Bonobos, are opening brick-and-mortar shops.

An article about the Practical Data Ming, by Eibe Frank,Mark A. Hall and Christopher J. Pal,includes using linear model to make reasonable expectation for future industry sale information.But through the papers they only deal with the time-series data. To figure out what features matter most in retail business and there is two parts of it: real retail shop and online retail.Each divisions of our project uses different way to make predictions of retail price changing and figure out the most valuable feature of retail business. Moreover, in the third part, we also want to explain that for a big retail network, properties of products(like which kind of product can sell most) and stores(like size of store) can influence sales. So we used models to help us explore and understand that.

3.Our Algorithms

In the first part,the main method for time series prediction is using LSTM.A RNN composed of LSTM units is often called an LSTM network and to make the prediction we set the period in two months.

Random Forest, used in the real retail sale prediction,is a meta estimator that fits a number of

classifying decision trees on various sub-samples of the dataset.So the features we got should be a boolean value and to avoid overfitting,median value is needed.

In the third part, BigMart retail sale prediction, we used K-Nearest Neighbor, KNN. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point. We first created a K-Nearest-Neighbor object, then used the "fit" and "predict" function to do the predictions.

METHODS

In the first part of our project, we use the python method in data cleaning and data explore. And we can use graph find some features in the dataset which can let us know the pinpoint of this dataset. And then, we decide to use RNN/LSTM on forecast the store daily income. So we use python to calculate daily income and make this dataframe transfer to a time-series dataframe. Finally, we had a forecast result for daily income.

Similarly in real retail part of code we start with data clean.Besides,forecasts were based simply on a median of the weekly sales grouped by the Type of Store, Store & Department number, Month and Holiday dummy and ,correlation map do helps us to measure which features is really useful and which makes little contribution to the final prediction. What's more,The input csv file is merged by 5 datasets that makes the pre-progress very hard.

In the third part of our project, there are two datasets used. The first is "train", the second is "test". The "test" was used to test whether we had found the good prediction and got the good understanding of the data. We used some methods to clean the dataset we got first, then for a better use of the dataset, we put a new column "Quantity Sold", as quantity of sold is a good standard to test which factor influenced the sales most. After that, we built a classifier changing the categorical data to numeric/dummy. Then we used the algorithms of K-Nearest Neighbours to get the value of accuracy. Besides, we used five models Decision tree Regression, Ridge, Adaptive Boost, LASSO, Random Forest to get five predictions, and then compare these five different results.

OUR RESULTS

1.Forecast the online store daily income

(1)Data framing and data cleaning

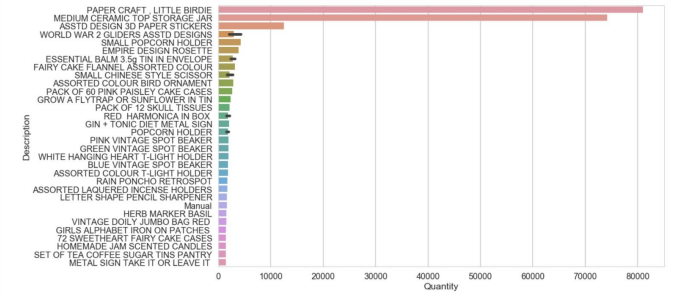
We use Python notebook and the method which called "*pandas.read_csv()*" to read the csv file. And then, we transfer csv file into dataframe. We can see the dataset information and attribute in this graph. Also, we need to find out the features in this dataset.

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

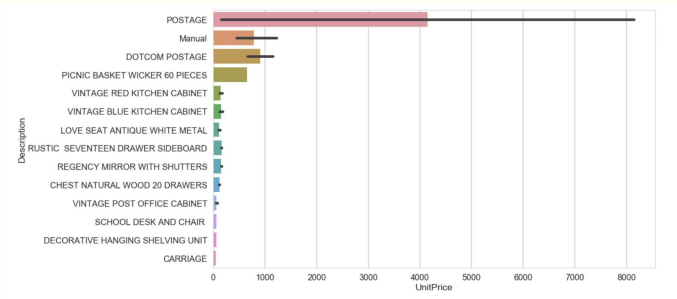
In this graph, we can see that the overview of the original data. The dataset has 8 columns and each row is represent an order which has 'Invoice NO','Stock Code', 'Description', 'Quantity', 'Invoice Data', 'Unite price', 'Customer ID', 'Country'. For each of record, we need to check it if it has NaN and we has to cleaning these value.

(2)Data preparation

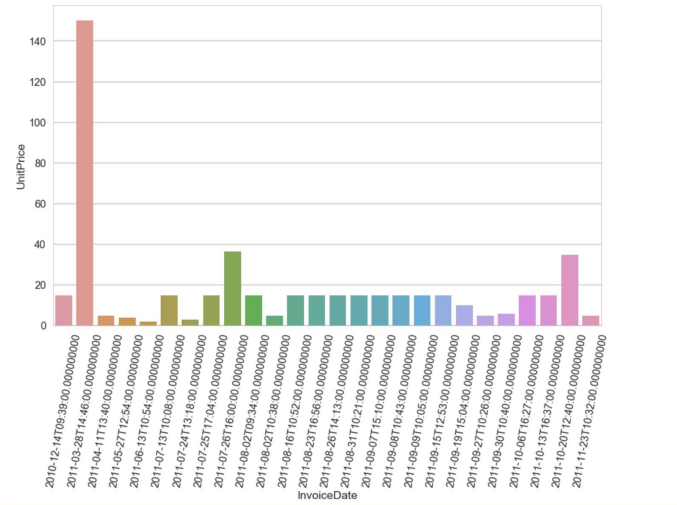
The data preparation is the important process which is use to get main features. In first part, we want to find the relation between the quantity, description and the date. So, we use "*sns.barplot()*" to find which description has the most quantity in this dataset.



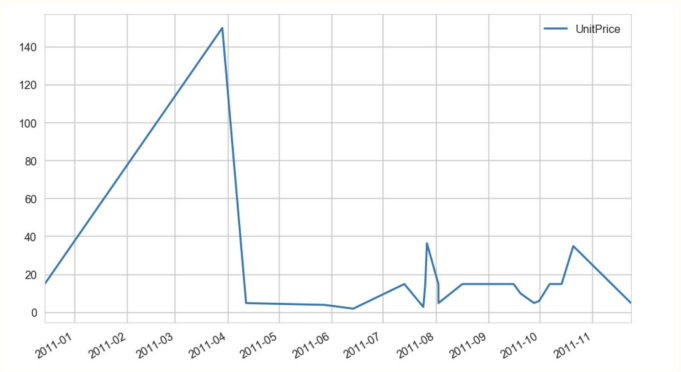
We can see that the most part of quantity is ‘Paper’, ‘Storage’ and ‘Stickers’. However, the quantity can not to represent the value of product. So, we move on the ‘unit price’ of the product.



So, we can see that the ‘Postage’ has the highest ‘Unit price’. Focus on this point, we decide to find the features of the ‘Postage’. We filter all product which name is ‘Postage’ first and get a new dataset. And then, we find only small part of the ‘Postage’ has the unit price is bigger than 8000. So, we remove these data from the dataset. Next, we use the ‘Unit price’ and ‘date’ to build a graph which can help us to find relationship between this two attribute.



We found that except the ‘03-28’ all the ‘Unit price’ is between 0-40. Afterwards, we use method to set the ‘date’ to index and set the ‘Postage’ unit price in column 0. So, this is a time-series dataframe and we can use it to analysis the tendency for ‘unit price’ with ‘date’.



(3) The daily income forecast

In this part, we focus on the forecast the store daily income. Before, the dataset is include everyday order information that has almost 1 year data. Now, we need to extract two column which name is ‘Invoice Date’ and ‘Unit price’. And then, we use ‘dataset.strftime()’ to transfer the date time from ‘2010-12-01 08:26:00’ to ‘2010-12-01’. Then, we put the date in index with the sequence.

	sale
2010-12-01	5715.43
2010-12-02	6348.65
2010-12-03	2899.04
2010-12-05	6962.77
2010-12-06	5460.58
2010-12-07	3291.50

(4) Method and Results

1.RNN/LSTM

Long short-term memory (LSTM) units (or blocks) are a building unit for layers of a recurrent neural network (RNN). A RNN composed of LSTM units is often called an LSTM network. A common

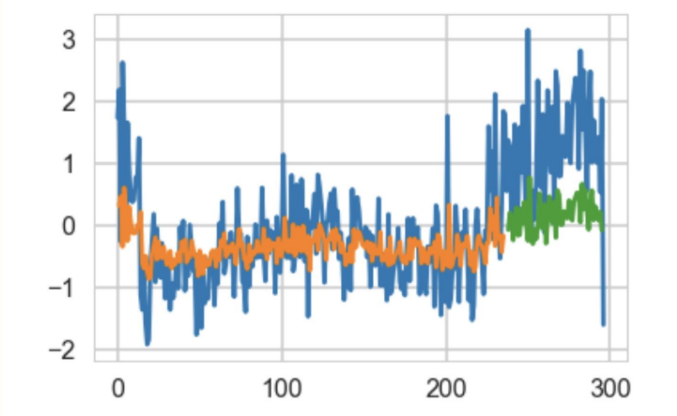
LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is responsible for "remembering" values over arbitrary time intervals; hence the word "memory" in LSTM. Each of the three gates can be thought of as a "conventional" artificial neuron, as in a multi-layer (or feedforward) neural network: that is, they compute an activation (using an activation function) of a weighted sum. Intuitively, they can be thought as regulators of the flow of values that goes through the connections of the LSTM; hence the denotation "gate". There are connections between these gates and the cell.[1]

First, we use this time-series data as the dataset. And we set the RNN/LSTM model. Use the method to split the dataset in two part, 67% dataset is train dataset and 33% dataset is test dataset. The look back setting we set 1. This mean we use yesterday income to forecast the today income.In RNN/LSTM model, we set activation method with tanh and the layer is 4. Also, the epochs is 204, the batch_size is 1 and the verbose is 2.

```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back),activation='tanh'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=204, batch_size=1, verbose=2)
```

```
Epoch 1/204
 - 3s - loss: 0.0245
Epoch 2/204
 - 0s - loss: 0.0116
Epoch 3/204
 - 0s - loss: 0.0115
```

And we use '*model.predict()*' to make the prediction. Then, we use '*scaler.inverse_transform()*' to transfer the predictions. Last, we use the ' *math.sqrt()*' to calculate root mean squared error and get test score. Train Score: 0.6958 RMSE, Test Score: 1.4149 RMSE. And then, we shift the test and train predictions for plotting and plot baseline & predictions.



2.ELU(Exponential linear unit)

In this prediction graph we can see that the train set is almost like the original data and the test set is has some gap which contrast the original dataset. So, we want to improve our degree of accuracy. The Activation function-using different activation parameter and use different activation method named 'elu'.

Elu function takes care of the Vanishing gradient problem. The other mentioned activation functions are prone to reaching a point from where the gradient of the functions does not change.[2]

And the results is Train Score: 0.69521 RMSE Test Score: 1.40121 RMSE. We can know that this results is not has obvious different between the 'tanh' result.

3. Cost function and Gradient estimation

The loss function is applied to just a single training example like so. And the cost function is the cost of your parameters.[3]

So, we use cost function to improve our results and we can see that it's has good performance. The train score: 0.69986 RMSE, the test score is: 1.31112 RMSE. And contrast the original results, we had improve it 7%.

And then we use the gradient estimation to improve our results. And we found that this method only can improve the results about 3%.

4. Change the Epochs and network architecture

We try to change the epochs from 204 to 100, but we found that only small change on the result. It's not have big effect on the accuracy.

And then, we try to use network architecture and adding a masking layer to model. However, the results is also same like the original result.

2.Real Retail Economy

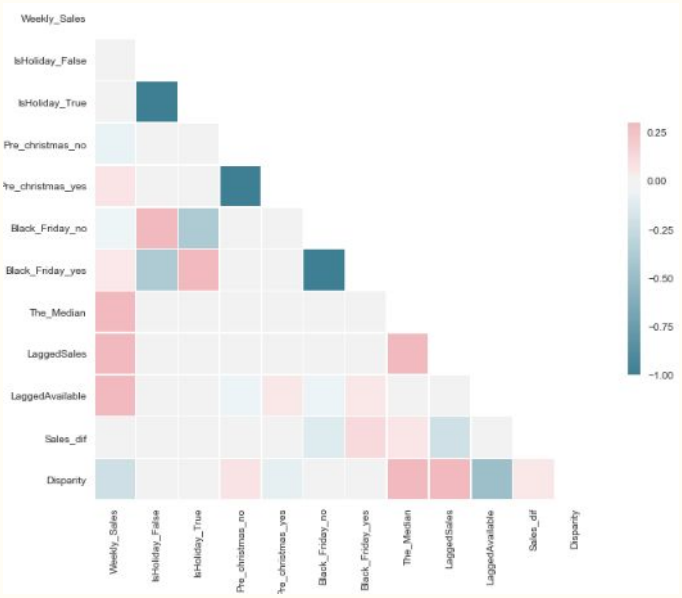
Just like using LSTM model,in online retail method includes lagged features(the week sale in the last week), which turns out to be very good because in small period of time the weeksale data value could be taken as continuous numbers.

(1) Pre-process

The very first step is merging the csv we need and deal with duplicate value and null. For this part of project, we also parse the date to holidays, deal with some unknown columns, add lagged sale columns and calculate the median.[4]

Pre_christmas_no	Pre_christmas_yes	Black_Friday_no	Black_Friday_yes	The_Median
1	0	1	0	23510.49
1	0	1	0	23510.49
1	0	1	0	23510.49
1	0	1	0	23510.49
1	0	1	0	23510.49

(2)Select the input for model



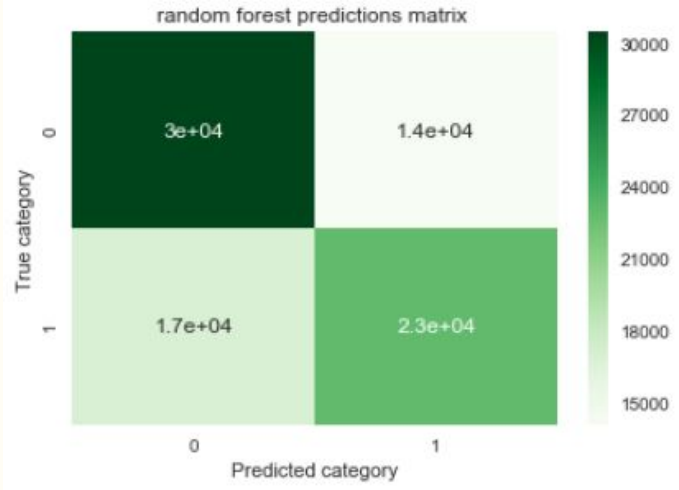
The picture below is correlation heatmap. Basically the lagged value(sale of the last week) and median value plays an important role here and Christmas has the same impact as Black Friday. Then we create "selector" to make sure all stuff is included. The "df[selector]" would be called if we need to make change on the input value.

```
display(df2[selector].head())
```

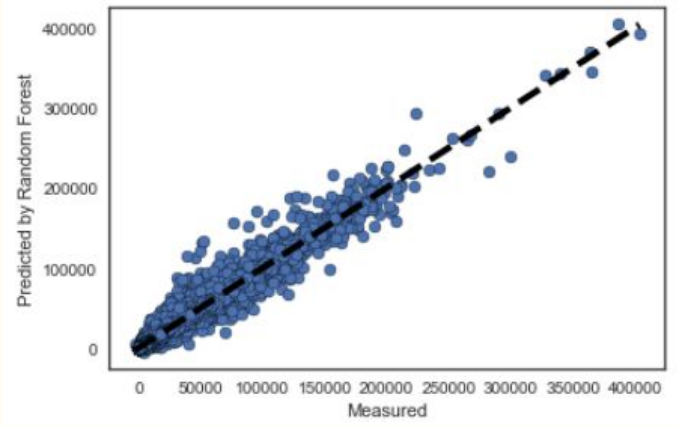
	CPI	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	Size	Temperature	Unemployment
count	421764.000000	421764.000000	150749.000000	111284.000000	137139.000000	134993.000000	151502.000000	421764.000000	421764.000000	421764.000000
mean	171.195517	3.361063	7244.567761	3333.768017	1438.924118	3382.902156	4627.578656	136698.996747	60.092726	7.960640
std	39.159706	0.458541	8291.024935	9473.963908	9621.430759	6291.928123	5961.909993	60992.346096	18.448236	1.863546
min	126.064000	2.472000	0.270000	-265.760000	-29.100000	0.220000	135.160000	34875.000000	-2.060000	3.879000
25%	132.022667	2.933000	2232.080000	41.600000	5.080000	504.220000	1878.440000	93638.000000	46.700000	6.891000
50%	182.318780	3.452000	5347.450000	192.000000	24.600000	1481.310000	3359.100000	140167.000000	62.090000	7.866000
75%	212.416993	3.738000	9210.900000	1926.940000	103.990000	3595.040000	5556.150000	202505.000000	74.280000	8.572000
max	227.232807	4.468000	88646.760000	104519.540000	141630.610000	67474.850000	108519.280000	219622.000000	100.140000	14.313000

(3)Random Forest

The default values for the parameters controlling the size of the trees.The predicted regression target of an input sample is computed as the mean predicted regression targets of the trees in the forest.



To evaluate the model, we will look at MAE and accuracy in terms of the number of times it correctly estimated an upward or downward deviation from the median.The matrix shows that our result is not bad as the general trend of week sale is in line with expectations the model made.[5]



To make it easy to show the result of our method, we also plot it. Because the test file doesn't have sale value which means a lack of the correct answer, we split the train set and use 20% of it to "test"our model.

3.BigMart retail sale prediction

From the result of third part, we compared the result we got from the file "train" with the dataset "test", we can see that for five different "Outlet_Identifier" and "Item_Identifier", the models gave us different values for the "Sales".

(1)Data framing and data cleaning

We used python notebook and the method which is called "pandas.read_csv()" to read the csv files, "train", "test". And then, we transferred csv file into dataframe. We can see the dataset information and attributes in this graph.

In our two files, "train" and "test", we both have thirteen columns, which are "Item_Fat_Content", "Item_Identifier", "Item_MRP", "Item_Outlet_Sales", "Item_Type", "Item_Visibility", "Item_Weight", "Outlet_Establishment_Year", "Outlet_Identifier", "Outlet_Location_Type", "Outlet_Size", "Outlet_Type". For the column "Item_MRP", it means material requirements planning for each product. And other columns have the meaning as they show here literally.

	Item_Fat_Content	Item_Identifier	Item_MRP	Item_Outlet_Sales	Item_Type	Item_Visibility	Item_Weight	Outlet_Establishment_Year	Outlet_Identifier	Outl
14199	Regular	FDB58	141.3154	NaN	Snack Foods	0.013496	10.5	1997	OUT046	
14200	Regular	FDD47	169.1448	NaN	Starchy Foods	0.142991	7.6	2009	OUT018	
14201	Low Fat	NCO17	118.7440	NaN	Health and Hygiene	0.073529	10.0	2002	OUT045	
14202	Regular	FDJ26	214.6218	NaN	Canned	0.000000	15.3	2007	OUT017	
14203	Regular	FDU37	79.7960	NaN	Canned	0.104720	9.5	2002	OUT045	

Also we can see the different features of the dataset about all the thirteen columns, "Item_Identifier", "Outlet_Identifier", "Outlet_Size", etc.

```
Sales.describe(include="all")
```

	Item_Fat_Content	Item_Identifier	Item_MRP	Item_Outlet_Sales	Item_Type	Item_Visibility	Item_Weight	Outlet_Establishment_Year	Outlet_Identifier
count	14204	14204	14204.000000	8523.000000	14204	14204.000000	11765.000000	14204.000000	14204
unique	5	1559	NaN	NaN	16	NaN	NaN	NaN	10
top	Low Fat	FDS15	NaN	NaN	Fruits and Vegetables	NaN	NaN	NaN	OUT027
freq	8485	10	NaN	NaN	2013	NaN	NaN	NaN	1559
mean	NaN	NaN	141.004977	2181.288914	NaN	0.065953	12.792854	1997.830681	NaN
std	NaN	NaN	62.086938	1706.499616	NaN	0.051459	4.652502	8.371664	NaN
min	NaN	NaN	31.290000	33.290000	NaN	0.000000	4.555000	1985.000000	NaN
25%	NaN	NaN	94.012000	834.247400	NaN	0.027036	8.710000	1987.000000	NaN
50%	NaN	NaN	142.247000	1794.331000	NaN	0.054021	12.600000	1999.000000	NaN
75%	NaN	NaN	185.855600	3101.296400	NaN	0.094037	16.750000	2004.000000	NaN
max	NaN	NaN	266.888400	13086.964800	NaN	0.328391	21.350000	2009.000000	NaN

As the "Item_Visibility" for a stock is very important, so the visibility can't be 0 or null. There are some "NaN" in the column, so we want to filter it to get a better dataset for out analysis.
Hence we imputed it here by "UniqueItems" to filter out those null values or 0 value.

```
Sales.describe(include="all")
```

	Item_Fat_Content	Item_Identifier	Item_MRP	Item_Outlet_Sales	Item_Type	Item_Visibility
count	14204	14204	14204.000000	8523.000000	14204	14204.000000
unique	5	1559	NaN	NaN	16	NaN
top	Low Fat	FDS15	NaN	NaN	Fruits and Vegetables	NaN
freq	8485	10	NaN	NaN	2013	NaN
mean	NaN	NaN	141.004977	2181.288914	NaN	0.065953
std	NaN	NaN	62.086938	1706.499616	NaN	0.051459
min	NaN	NaN	31.290000	33.290000	NaN	0.000000
25%	NaN	NaN	94.012000	834.247400	NaN	0.027036
50%	NaN	NaN	142.247000	1794.331000	NaN	0.054021
75%	NaN	NaN	185.855600	3101.296400	NaN	0.094037
max	NaN	NaN	266.888400	13086.964800	NaN	0.328391

Here is the result of the filter.

```
Sales["Item_Visibility"].describe()
```

```
count    14204.000000
mean         0.065953
std          0.051459
min          0.000000
25%          0.027036
50%          0.054021
75%          0.094037
max          0.328391
Name: Item_Visibility, dtype: float64
```

```
UniqueItems = set(Sales.Item_Identifier)
for each in UniqueItems:
    Sales.loc[(Sales["Item_Identifier"]==
    #print(Sales["Item_Identifier"]==str(
Sales["Item_Visibility"].describe()
```

```
count    14204.000000
mean         0.070458
std          0.050086
min          0.003575
25%          0.031381
50%          0.058064
75%          0.098042
max          0.328391
Name: Item_Visibility, dtype: float64
```

Here we also did an evaluation here, “Adding new feature "Quantity Sold"”.

As What we truly want to compare is, between the property of the product, or the size of the store these two things, which influence the sales more. So the value of "sales" is the most thing here, as it works as the only standard of comparison. To show the value of product sold, we added a new feature "Qty_Sold" here, and the definition of this feature is, "Qty_Sold = Sales / Item_MRP".[6]

```
Sales["Qty_Sold"] = (Sales["Item_Outlet_Sales"]/Sales["Item_MRP"])
Sales.head()
```

utlet_Sales	Item_Type	Item_Visibility	Item_Weight	Outlet_Establishment_Year	Outlet_Identifier	Outlet_Location_Type	Outlet_Size	Outlet_Type	Type	Qty_Sold
3735.1380	Dairy	0.016047	9.30	1999	OUT049	Tier 1	Medium	Supermarket Type1	train	14.951963
443.4228	Soft Drinks	0.019278	5.92	2009	OUT018	Tier 3	Medium	Supermarket Type2	train	9.186454
2097.2700	Meat	0.016760	17.50	1999	OUT049	Tier 1	Medium	Supermarket Type1	train	14.809346
732.3800	Fruits and Vegetables	0.022930	19.20	1998	OUT010	Tier 3	NaN	Grocery Store	train	4.021967
994.7052	Household	0.014670	8.93	1987	OUT013	Tier 3	High	Supermarket Type1	train	18.467868

(2)Correlation Analysis

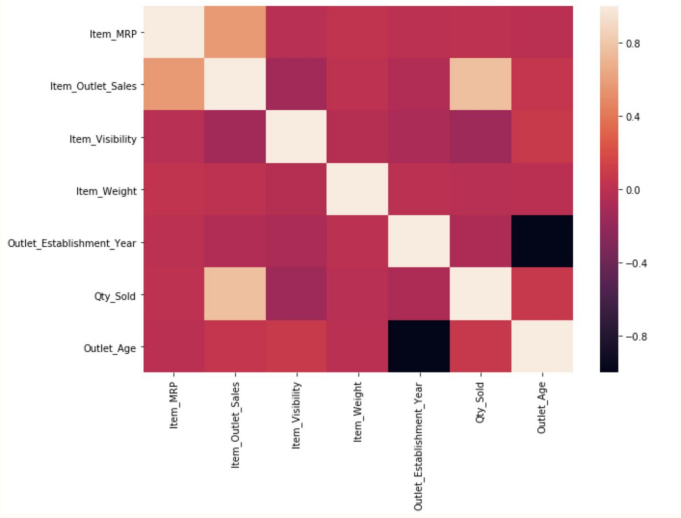
Here, as we wanted to know the correlation of each columns in this dataset, so we used HeatMap and dataframe.corr(), these two functions to analyze the dataset.[7]

1. "Item_Weight" --For a better analysis of the dataset, the way we dealt with the null value in column “Item_Weight” was to calculate the mean value of all the non-value, and assigned the mean value to those null value.

2. "Outlet_Age" --For a better analysis of the dataset, we add a new feature in the dataset, "Outlet_Age", which is calculated by "now.year - Outlet_Establishment_Year".

```
plt.figure(figsize=(10,7))
sns.heatmap(Sales.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x113cef128>
```



The result shows that "Item_Outlet_Sales" and "Qty_Sold" has a strong correlation, "Item_MRP" and "Item_Outlet_Sales" has a strong correlation, too. As the colors of squares get darker, we can see the correlation is decreasing.

(3)How to deal with the null value in “Outlet_Size”

As there were some null values in the column of “Outlet_size”, here is how we looked for the pattern for the missing values.

“OUT045”

	Item_Fat_Content	Item_Identifier	Item_Type	Outlet_Identifier	Outlet_Location_Type	Outlet_Size	Outlet_Type	Type
count	1548	1548	1548	1548	1548	0.0	1548	1548
unique	5	1548	16	1	1	0.0	1	2
top	Low Fat	FDQ03	Fruits and Vegetables	OUT045	Tier 2	NaN	Supermarket Type1	train
freq	932	1	218	1548	1548	NaN	1548	929

“OUT017”

	Item_Fat_Content	Item_Identifier	Item_Type	Outlet_Identifier	Outlet_Location_Type	Outlet_Size	Outlet_Type	Type
count	1543	1543	1543	1543	1543	0.0	1543	1543
unique	5	1543	16	1	1	0.0	1	2
top	Low Fat	FDS15	Snack Foods	OUT017	Tier 2	NaN	Supermarket Type1	train
freq	928	1	219	1543	1543	NaN	1543	926

“OUT010”

	Item_Fat_Content	Item_Identifier	Item_Type	Outlet_Identifier	Outlet_Location_Type	Outlet_Size	Outlet_Type	Type
count	925	925	925	925	925	0.0	925	925
unique	5	925	16	1	1	0.0	1	2
top	Low Fat	FDS15	Fruits and Vegetables	OUT010	Tier 3	NaN	Grocery Store	train
freq	543	1	137	925	925	NaN	925	555

After the analysis of the results, we found out that for"Outlet_Identifier", there were some values in "Outlet_Size" are null. So now we needed to look for the pattern for the missing "Outlet_size".

Also, As for “OUT045” and “OUT017”, these two types have same “Outlet_Location_Type” and “Outlet_Type”, which are "Tier 2" and "Supermarket Type1", hence we can impute the data to find the size of each null value, whose type is “OUT010”, “OUT045”, “OUT017”.

We needed to find the column, which "Outlet_Location_Type" is "Tier 2" and "Outlet_Type" is "Supermarket Type1", and then get the size of it. Here, after we ran these two commands, we assigned the null values with size of "small", which "Outlet_Location_Type"== "Tier 2" and "Outlet_Type"=="Supermarket Type1", so after these commands we solved the null value problem for “OUT045”, “OUT017”.

```
Sales.loc[(Sales["Outlet_Location_Type"]== "Tier 2") & (Sales["Outlet_Type"]=="Supermarket Type1") ,"Outlet_Size"].
```

```
Small    1550
Name: Outlet_Size, dtype: int64
```

```
Sales.loc[(Sales["Outlet_Size"].isnull()) & (Sales["Outlet_Identifier"].isin(["OUT045","OUT017"])) ,"Outlet_Size"]
Sales.loc[(Sales["Outlet_Identifier"].isin(["OUT045","OUT017"])) ,"Outlet_Size"].value_counts()
```

```
Small    3091
Name: Outlet_Size, dtype: int64
```

To impute "Outlet_Size for OUT010", we built a classifier to change the categorical data to numeric/dummy.

```
mapping_Item_Type = {'Fruits and Vegetables': "Fruit_Veg",
                     'Household':"HH_HH", 'Health and Hygiene':"HH_HH",
                     'Baking Goods': "Bake_Snacks", 'Snack Foods': "Bake_Snacks",
                     'Canned': "Frozen_Canned", 'Frozen Foods': "Frozen_Canned",
                     'Dairy': "DBBS", 'Breakfast': "DBBS", 'Breads': "DBBS", 'Starchy Foods': "DBBS",
                     'Seafood':"Seafood_Meat", 'Meat': "Seafood_Meat",
                     'Hard Drinks': "Drinks", 'Soft Drinks': "Drinks",
                     'Others':"Others"}
Sales['Item_Type'] = Sales['Item_Type'].map(mapping_Item_Type)

mapping_Item_Fat_Content = {'Regular': 1, "reg": 1,'LF': 0,'Low Fat': 0, "low fat":0}
Sales['Item_Fat_Content'] = Sales['Item_Fat_Content'].map(mapping_Item_Fat_Content)

mapping_Outlet_Size = {'Small': 1, "Medium": 2,'High': 3}
Sales['Outlet_Size'] = Sales['Outlet_Size'].map(mapping_Outlet_Size)
```

	Item_Fat_Content	Item_Identifier	Item_MRP	Item_Outlet_Sales
0	0	FDA15	249.8092	3735.1380
1	1	DRC01	48.2692	443.4228
2	0	FDN15	141.6180	2097.2700
3	1	FDX07	182.0950	732.3800
4	0	NCD19	53.8614	994.7052

Item_Type	Item_Visibility	Item_Weight	Outlet_Establishment_Year	Outlet_Identifier	Outlet_Location_Type	Outlet_Size	Outlet_Type	Type	Qty_Sold	Outlet_Age
DBBS	0.016047	9.30	1999	OUT049	Tier 1	2.0	Supermarket Type1	train	14.951963	19
Drinks	0.019278	5.92	2009	OUT018	Tier 3	2.0	Supermarket Type2	train	9.186454	9
afood_Meat	0.016760	17.50	1999	OUT049	Tier 1	2.0	Supermarket Type1	train	14.809346	19
Fruit_Veg	0.022930	19.20	1998	OUT010	Tier 3	NaN	Grocery Store	train	4.021967	20
HH_HH	0.014670	8.93	1987	OUT013	Tier 3	3.0	Supermarket Type1	train	18.467868	31

(4)Data Classification

As there are 3 categories, we can use all the classifier techniques except Logistic Regression, so we tried K-Nearest Neighbours here.

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

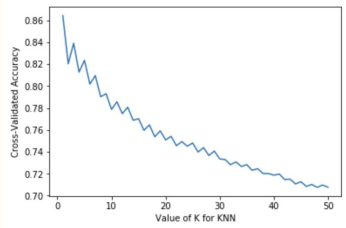
1. Ease to interpret output

2. Calculation time

3. Predictive Power

Here, we used the KNearestNeighbor algorithms to calculate the accuracy. [8]

```
Location for Max Accuaracy is:
1
Value of K with is: 1
```




```
knn = KNeighborsClassifier(n_neighbors=K)
y_predicted = knn.fit(X_train,y_train).predict(X_test)
print("Test Accuracy: ", (y_predicted == y_test).astype(int).sum()/y_test.shape[0])
```

```
Test Accuracy:  0.8519447929736512
```

As we can see, after using the KNearestNeighbor algorithms, we got the test accuracy, 0.8519.

(5)Five models to predict the results

Multinomial Classifier and OneVsOneClassifier

This strategy consists in fitting one classifier per class pair. At prediction time, the class which received the most votes is selected. Since it requires to fit n_classes * (n_classes - 1) / 2 classifiers, this method is usually slower than one-vs-the-rest, due to its O(n_classes^2) complexity. However, this method may be advantageous for algorithms such as kernel algorithms which don't scale well with n_samples. This is because each individual learning problem only involves a small subset of the data whereas, with one-vs-the-rest, the complete dataset is used n_classes times.[9]

```
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import LinearSVC
```

```
md = OneVsOneClassifier(LinearSVC(random_state=0)).fit(X_train, y_train)
```

```
y_predicted = md.predict(X=X_test)
print("Test Accuracy: ", (y_predicted == y_test).astype(int).sum()/y_test.shape[0])
#pd.Series(y_predicted).value_counts()
```

```
Test Accuracy:  1.0
```

```
pd.Series(md.predict(X=Sales.loc[(Sales[y_cols].isnull()) & (Sales['Type'] == "train"), X_Cols])).value_counts()
```

```
2.0    555
dtype: int64
```

Adaboost Classifier

Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers. Ada-boost, like Random Forest Classifier is another ensemble classifier. (Ensemble classifier are made up of multiple classifier algorithms and whose output is combined result of output of those classifier algorithms).[11]

```
from sklearn.ensemble import AdaBoostClassifier
clf = AdaBoostClassifier(n_estimators=100)
scores = cross_val_score(clf, X_train, y_train)
scores.mean()
```

```
1.0
```

```
clf.fit(X_train, y_train)
```

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=100, random_state=None)
```

```
pd.Series(clf.predict(X=Sales.loc[(Sales[y_cols].isnull()) & (Sales['Type'] == "train"), X_Cols])).value_counts()
```

```
2.0    555
dtype: int64
```

R2_score

R^2 (coefficient of determination) regression score function. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.[10]

```
XCols = ['Item_Fat_Content', 'Item_MRP',
       'Item_Visibility', 'Item_Weight',
       'Outlet_Size', 'Outlet_Age', 'Item_Type_DBBS',
       'Item_Type_Drinks', 'Item_Type_Frozen_Canned', 'Item_Type_Fruit_Veg',
       'Item_Type_HH_HH', 'Item_Type_Others', 'Item_Type_Seafood_Meat',
       'Outlet_Identifier_OUT013', 'Outlet_Identifier_OUT017',
       'Outlet_Identifier_OUT018', 'Outlet_Identifier_OUT019',
       'Outlet_Identifier_OUT027', 'Outlet_Identifier_OUT035',
       'Outlet_Identifier_OUT045', 'Outlet_Identifier_OUT046',
       'Outlet_Identifier_OUT049', 'Outlet_Location_Type_Tier 2',
       'Outlet_Location_Type_Tier 3', 'Outlet_Type_Supermarket Type1',
       'Outlet_Type_Supermarket Type2', 'Outlet_Type_Supermarket Type3']
YCols = 'Qty_Sold'
```

```
XX = Sales.loc[Sales["Type"]=="train",XCols]
yy = Sales.loc[Sales["Type"]=="train",YCols]
print(XX.shape)
print(yy.shape)
XX_train, XX_test, yy_train, yy_test = train_test_split(XX, yy, test_size=0.3, random_state=5)
```

```
(8523, 27)
(8523,)
```

Linear Regression

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.[12]

```
coef1 = pd.DataFrame(reg.coef_,XCols,columns=["Value"])
#coef1[coef1["Value"]>0].sort_values(by="Value",ascending=False)
```

```
yy_predicted = reg.predict(XX_test)
metrics.mean_squared_error(y_true=yy_test, y_pred=yy_predicted)
```

```
48.699882096521847
```

```
r2_score(yy_test, yy_predicted)
```

```
0.41814067440785574
```

Lasso Regression

Lasso stands for least absolute shrinkage and selection operator is a penalized regression analysis method that performs both variable selection and shrinkage in order to enhance the prediction accuracy. Suppose we have many features and we

want to know which are the most useful features in predicting target in that case lasso can help us. Features that are not helping in decision making lasso will set their coefficients value very low potentially zero.It adds them one at time, if new feature would not improve model then it sets its coefficient to zero.[13]

	Value
Outlet_Type_Supermarket Type1	13.441345
Outlet_Identifier_OUT027	12.640825
Outlet_Type_Supermarket Type3	11.181365
Outlet_Type_Supermarket Type2	10.809769
Outlet_Identifier_OUT017	0.499518
Outlet_Identifier_OUT035	0.316412
Item_Fat_Content	0.276729
Outlet_Identifier_OUT049	0.158667
Item_Type_Fruit_Veg	0.123331
Item_MRP	0.001319

```
yy_predicted = Lasso.predict(XX_test)
print(metrics.mean_squared_error(y_true=yy_test, y_pred=yy_predicted))
print(r2_score(yy_test, yy_predicted))
```

```
48.6568713725
0.418654560475
```

```
Lasso.fit(XX,yy)

```

```
LassoCV(alphas=[0.0005], copy_X=True, cv=5, eps=0.001, fit_intercept=True,
    max_iter=1000, n_alphas=100, n_jobs=1, normalize=True, positive=False,
    precompute='auto', random_state=10, selection='cyclic', tol=0.0001,
    verbose=False)
```

Adaptive Boosting

```
regressor.fit(XX_train,yy_train)
```

```
AdaBoostRegressor(base_estimator=None, learning_rate=0.005, loss='linear',
         n_estimators=100, random_state=None)
```

```
yy_predicted = regressor.predict(XX_test)
print(metrics.mean_squared_error(y_true=yy_test, y_pred=yy_predicted))
print(r2_score(yy_test, yy_predicted))

```

```
48.6527178872
0.418704185732
```

```
regressor.fit(XX,yy)
```

```
AdaBoostRegressor(base_estimator=None, learning_rate=0.005, loss='linear',
         n_estimators=100, random_state=None)
```

Ridge

```
Ridge.fit(XX_train,yy_train)
#print(Ridge.intercept_)
coef1 = pd.DataFrame(Ridge.coef_,XCols,columns=["Value"])
#coef1[coef1["Value"]>0].sort_values(by="Value",ascending=False)

```

```
yy_predicted = Ridge.predict(XX_test)
print(metrics.mean_squared_error(y_true=yy_test, y_pred=yy_predicted))
print(r2_score(yy_test, yy_predicted))

```

```
48.6985161216
0.418156994884
```

```
Ridge.fit(XX,yy)
```

```
Ridge(alpha=0.001, copy_X=True, fit_intercept=True, max_iter=None,
   normalize=True, random_state=10, solver='auto', tol=0.001)
```

Decision Tree Regression

```
from sklearn.ensemble import RandomForestRegressor
RForrest = RandomForestRegressor()
RForrest.fit(XX_train,yy_train)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```

yy_predicted = Dtree.predict(XX_test)
print(metrics.mean_squared_error(y_true=yy_test, y_pred=yy_predicted))
print(r2_score(yy_test, yy_predicted))
```

```
48.4441102797
0.421196600017
```

```

Dtree.fit(XX,yy)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=3, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

Combining LASSO, ADB and Ridge together

```

Lasso_Prediction = Lasso.predict(Sales_Predict)
ADB_Prediction = regressor.predict(Sales_Predict)
Ridge_Prediction = Ridge.predict(Sales_Predict)
RForrest_Prediction = RForrest.predict(Sales_Predict)
Dtree_Prediction = Dtree.predict(Sales_Predict)
```

```
print(Lasso_Prediction[:5])
print(ADB_Prediction[:5])
print(Ridge_Prediction[:5])
print(RForrest_Prediction[:5])
print(Dtree_Prediction[:5])
```

```
[ 16.38839972  16.53388655   3.02660887  16.50001025  26.61073934]
[ 16.23012998  16.56853041   2.42165404  16.59097007  26.72072367]
[ 16.36215476  16.58448813   2.56700997  16.6545641   26.76259173]
[ 13.7369647   17.11286349   2.19966501  12.58767874  29.68442127]
[ 16.49094575  16.49094575   2.41317621  16.49094575  26.50672595]
```

We used:

“Data_Predicted["Lasso_Prediction"]=pd.Series(Lasso_Prediction) * Data_Predicted["Item_MRP"]

Data_Predicted["ADB_Prediction"]=pd.Series(ADB_Prediction)* Data_Predicted["Item_MRP"]

Data_Predicted["Ridge_Prediction"]=pd.Series(Ridge_Prediction) * Data_Predicted["Item_MRP"]

Data_Predicted["DTree_Prediction"]=pd.Series(Dtree_Prediction) * Data_Predicted["Item_MRP"]

Data_Predicted["RForrest_Prediction"]=pd.Series(RForrest_Prediction) * Data_Predicted["Item_MRP"]”

```
Data_Predicted["Lasso_Prediction"] = pd.Series(Lasso_Prediction) * Data_Predicted["Item_MRP"]
Data_Predicted["ADB_Prediction"] = pd.Series(ADB_Prediction) * Data_Predicted["Item_MRP"]
Data_Predicted["Ridge_Prediction"] = pd.Series(Ridge_Prediction) * Data_Predicted["Item_MRP"]
Data_Predicted["DTree_Prediction"] = pd.Series(Dtree_Prediction) * Data_Predicted["Item_MRP"]
Data_Predicted["RForrest_Prediction"] = pd.Series(RForrest_Prediction) * Data_Predicted["Item_MRP"]
```

--these five ways to get the values of sales.

As there will be a very long list of the result of each model, so we used the function below to get the mean value of five results:

Data_Predicted["Item_Outlet_Sales"]= (Data_Predicted["Lasso_Prediction"]+Data_Predicted["ADB_Prediction"]+Data_Predicted["ADB_Prediction"]+Data_Predicted["DTree_Prediction"]+Data_Predicted["RForrest_Prediction"])/5

Here is the result after assigning the mean value to each type of the product and different types of Outlet_Size.

Data_Predicted

```
Data_Predicted["Lasso_Prediction"] = pd.Series(Lasso_Prediction) * Data_Predicted["Item_MRP"]
Data_Predicted["ADB_Prediction"] = pd.Series(ADB_Prediction) * Data_Predicted["Item_MRP"]
Data_Predicted["Ridge_Prediction"] = pd.Series(Ridge_Prediction) * Data_Predicted["Item_MRP"]
Data_Predicted["DTree_Prediction"] = pd.Series(Dtree_Prediction) * Data_Predicted["Item_MRP"]
Data_Predicted["RForrest_Prediction"] = pd.Series(RForrest_Prediction) * Data_Predicted["Item_MRP"]
```

```
:ted["ADB_Prediction"]+Data_Predicted["ADB_Prediction"]+Data_Predicted["DTree_Prediction"]+Data_Predicted["RForrest_P
```

```
Lasso_Prediction","ADB_Prediction","Ridge_Prediction","RForrest_Prediction","DTree_Prediction"],axis=1,inplace=True)
```

	Item_Identifier	Outlet_Identifier	Item_Outlet_Sales
0	FDW58	OUT049	1705.874565
1	FDW14	OUT017	1454.307018
2	NCN55	OUT010	603.550845
3	FDQ58	OUT017	2442.113393
4	FDY38	OUT027	6382.455221

```
In [191]: print(Data_Predicted)
```

	Item_Identifier	Outlet_Identifier	Item_Outlet_Sales
0	FDW58	OUT049	1743.370538
1	FDW14	OUT017	1468.069022
2	NCN55	OUT010	594.131489
3	FDQ58	OUT017	2501.835544
4	FDY38	OUT027	6311.653888
5	FDH56	OUT046	1936.305426
6	FDL48	OUT018	682.503184
7	FDC48	OUT027	2127.677100
8	FDN33	OUT045	1394.037270
9	FDA36	OUT017	3095.558074
10	FDT44	OUT017	1818.841727
11	FDQ56	OUT045	1424.908383
12	NCC54	OUT019	563.672071
13	FDU11	OUT049	2024.434482
14	DRL59	OUT013	822.630444
15	FDM24	OUT049	2469.999703
16	FDI57	OUT045	3161.784514
17	DRC12	OUT018	2827.064479
18	NCM42	OUT027	2787.800592
19	FDA46	OUT010	486.235865
20	FDA31	OUT013	2752.251816
21	NCJ31	OUT035	3817.001248
22	FDG52	OUT046	769.749085
23	NCL19	OUT019	330.959639
24	FDS10	OUT035	2902.749141
25	FDX22	OUT010	536.612456
26	NCF19	OUT035	751.735834
27	NCE06	OUT046	2587.265619
28	DRC27	OUT046	3916.178298
29	FDE21	OUT035	1986.667829
...	...	...	...
5651	FDD23	OUT013	3010.427978
5652	FDP32	OUT045	1951.902074
5653	FDO31	OUT035	1329.203027
5654	FDQ57	OUT013	2255.416998

As we can see, this is a very good prediction compared with the file “train”.

DISCUSSION

In the first part of our project - the e-commerce store daily income forecast, we think we can improve the result in many place. First,we think we can change model to multivariate time series. Because we think there are many factor have impression on daily income. And if we can get the more data about the customer information, we can analysis these factor and found something is important which has big effect on out result. In the other side, we think we need to find a method which can improve our result. Just like we can use gradient descent method in our model and contrast the result.

As for real retail part,take place of date features is the right decision.Generally for time series data it would mess up if you put date feature into predicting model.In fact,we have calculated the date and input the Holiday data like:is/not holiday,black friday and christmas.Actually these features work good and we can also compare the sale information in different holiday.When we implement the random forest model,we have to binary our data that 0 means price is going down and 1 is the sign of uprising.In other word,making prediction by the forest way is suit for evaluating the trend of price itself but we need to work on combined all attribute like VectorAssembler and VectorIndexer in spark ML library.Therefore, in our next approach, the goal will be to improve their results with the help of Neural Networks and other machine learning methods.

In the third part, we got five results of the prediction. We used five mathematical functions to calculate the sales for each model, than divided them by five to get the mean value for each. Then we exported the final csv to see our prediction. We dropped these columns:
"Item_MRP","Lasso_Prediction",
"ADB_Prediction","Ridge_Prediction",
"RForrest_Prediction", "DTree_Prediction", and had

a csv have “Item_Identifier”, “Outlet_Identifier”, “Item_Outlet_Sales”, three columns.

```
In [191]: 1 print(Data_Predicted)
```

	Item_Identifier	Outlet_Identifier	Item_Outlet_Sales
0	FDW58	OUT049	1743.370538
1	FDW14	OUT017	1468.069022
2	NCN55	OUT010	594.131489
3	FDQ58	OUT017	2501.835544
4	FDY38	OUT027	6311.653888
5	FDH56	OUT046	1936.305426
6	FDL48	OUT018	682.503184
7	FDC48	OUT027	2127.677100
8	FDN33	OUT045	1394.037270
9	FDA36	OUT017	3095.558074
10	FDT44	OUT017	1818.841727
11	FDQ56	OUT045	1424.908383
12	NCC54	OUT019	563.672071
13	FDU11	OUT049	2024.434482
14	DRL59	OUT013	822.630444
15	FDM24	OUT049	2469.999703
16	FDI57	OUT045	3161.784514
17	DRC12	OUT018	2827.064479
18	NCM42	OUT027	2787.800592
19	FDA46	OUT010	486.235865
20	FDA31	OUT013	2752.251816
21	NCJ31	OUT035	3817.001248
22	FDG52	OUT046	769.749085
23	NCL19	OUT019	330.959639
24	FDS10	OUT035	2902.749141
25	FDX22	OUT010	536.612456
26	NCF19	OUT035	751.735834
27	NCE06	OUT046	2587.265619
28	DRC27	OUT046	3916.178298
29	FDE21	OUT035	1986.667829
...	...	...	...
5651	FDD23	OUT013	3010.427978
5652	FDP32	OUT045	1951.902074
5653	FDO31	OUT035	1329.203027
5654	FDQ57	OUT013	2255.416998

Compared with the “train” file, it is a good prediction, most data of file “test” is similar with “train”. It also showed that the Item_Identifier influence the sales more.

Reference

[1]https://github.com/davidthaler/Walmart_competition_code
[2]https://github.com/threecourse/kaggle-walmart-recruiting-sales-in-stormy-weather
[3]http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
[4]https://github.com/IamAmar
[5]https://github.com/nikbearbrown/NEU_COE/tree/master/INFO_7390
[6]https://www.mathworks.com/help/stats/lasso.html
[7]https://codingstartups.com/practical-machine-learning-ridge-regression-vs-lasso/
[8]https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/
[9]https://www.visibleone.com/e-commerce-impact-retail-industry/
[10]https://www.quora.com/How-does-ELU-activation-function-help-convergence-and-whats-its-advantages-over-ReLU-or-sigmoid-or-tanh-function
[11]https://en.wikipedia.org/wiki/Long_short-term_memory
[12]https://www.coursera.org/learn/neural-networks-deep-learning/lecture/yWaRd/logistic-regression-cost-function
[13]https://datahack.analyticsvidhya.com/contest/practice-problem-big-mart-sales-iii/
[14]https://ljalphabeta.gitbooks.io/python-/content/partitioning.html