**Pankaj Acharya**
Capstone Project 1
Springboard Data Science Career Track
Mentored by Kevin Glynn
September 2019

# Predicting Future Sales

## Introduction

To maintain product stocks in stores is challenging for companies making those products. Having the product in stock for too long costs the companies a lot and on the other hand short supply will also lead to lost revenue, let alone customer trust. Maintaining a balance between overstock and understock is a major problem for companies in present cut-throat competition.

1C Company, one of the largest Russian software firms has provided a challenging time-series dataset consisting of daily sales for data scientists to build a model to predict future sales for every product and store in the next month. This will help the company make decision on maintaining stocks for particular items in particular shops optimally.

## Goal

The main goal of this project is to predict next month's sales of items based on historical data to help optimize the stocks.

## The Dataset

All the data for my project are downloaded from Kaggle website (https://bit.ly/2RsJ8AV). Following files are provided:

1. **sales_train.csv** - the training set. Daily historical data from January 2013 to October 2015

2. **test.csv** - the test set. I need to forecast the sales for these shops and products for November 2015.
3. **sample_submission.csv** - a sample submission file in the correct format.
4. **items.csv** - supplemental information about the items/products.
5. **item_categories.csv** - supplemental information about the items categories.
6. **shops.csv** - supplemental information about the shops.

The dataset is a real-world data set and is fairly large. The training dataset has 2.9 million rows. There were no missing values. It is a time-series data, but due to multiple transactions on the same day makes it challenging.

# Data Wrangling

The test file contains shop id and item id. Each shop and item combination is given a unique ID and we have to predict the sales of that unique ID (combination of shop and item) next month. The other four files (sales_train, items, item_categories and shops) each have information that needs to be combined before proceeding. Figure 1 shows a screenshot of the dataframe after these files were combined.

```
trainset = pd.merge(pd.merge(pd.merge(sales_train, items_df, how='left', on='item_id'),
                    shops, how='left', on='shop_id'), item_categories, how='left', on='item_category_id')
trainset.head()
```

| | date | date_block_num | shop_id | item_id | item_price | item_cnt_day | item_name | item_category_id | shop_name | item_category_name |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2013-01-02 | 0 | 59 | 22154 | 999.00 | 1.0 | ЯВЛЕНИЕ 2012 (BD) | 37 | Ярославль ТЦ "Альтаир" | Кино - Blu-Ray |
| 1 | 2013-01-03 | 0 | 25 | 2552 | 899.00 | 1.0 | DEEP PURPLE The House Of Blue Light LP | 58 | Москва ТРК "Атриум" | Музыка - Винил |
| 2 | 2013-01-05 | 0 | 25 | 2552 | 899.00 | -1.0 | DEEP PURPLE The House Of Blue Light LP | 58 | Москва ТРК "Атриум" | Музыка - Винил |
| 3 | 2013-01-06 | 0 | 25 | 2554 | 1709.05 | 1.0 | DEEP PURPLE Who Do You Think We Are LP | 58 | Москва ТРК "Атриум" | Музыка - Винил |
| 4 | 2013-01-15 | 0 | 25 | 2555 | 1099.00 | 1.0 | DEEP PURPLE 30 Very Best Of 2CD (Фирм.) | 56 | Москва ТРК "Атриум" | Музыка - CD фирменного производства |

**Figure 1. Screenshot of merged data frames showing all the columns**

# Exploratory Data Analysis

To get an idea of how the items sold throughout the year, I combined the daily sales into monthly sales. The figure 2 below shows that the total sales peaked every December. This is probably due to Christmas and New Year.

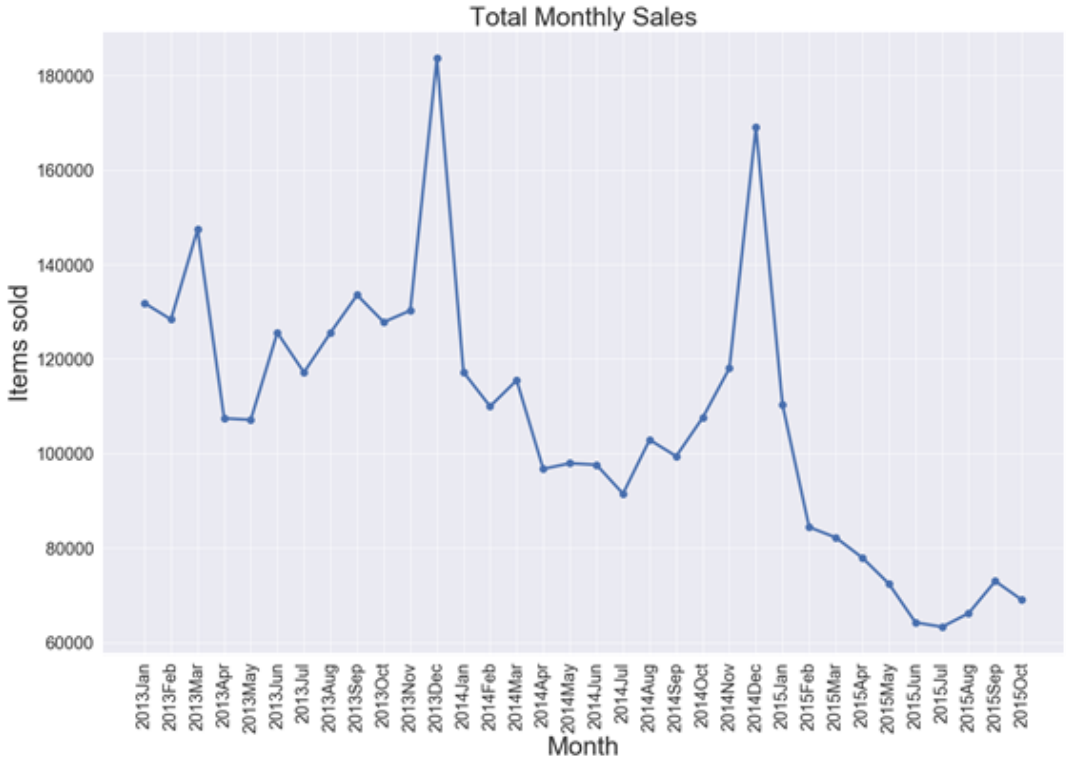


**Figure 2: line plot showing distribution of total monthly sales over the given period.**

To find out if the trend is similar for the top selling shops or it dominated by few shops, I plotted the total monthly sales for top ten performing shops. The figure 3 shows that in general all the shops have similar pattern over the given period. To further explore if the sales is dominated by few items, I plotted the total monthly sales of top ten selling items and as shown in figure 4, one item dominates the total sales throughout the 33 month period.

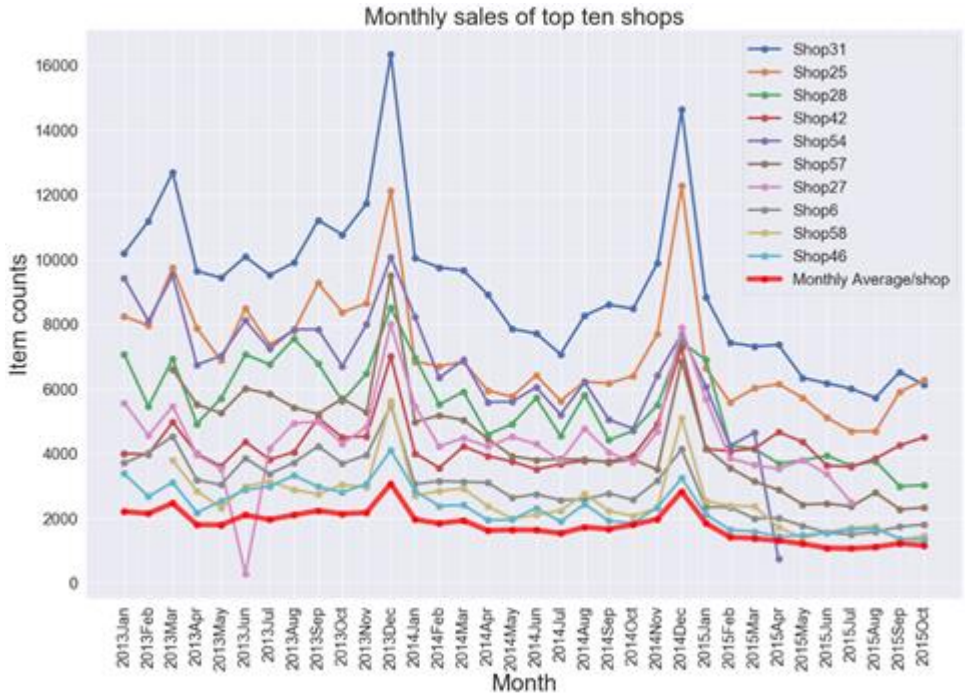


**Figure 3: line plot showing distribution of total monthly sales over the given period for top performing shops.**

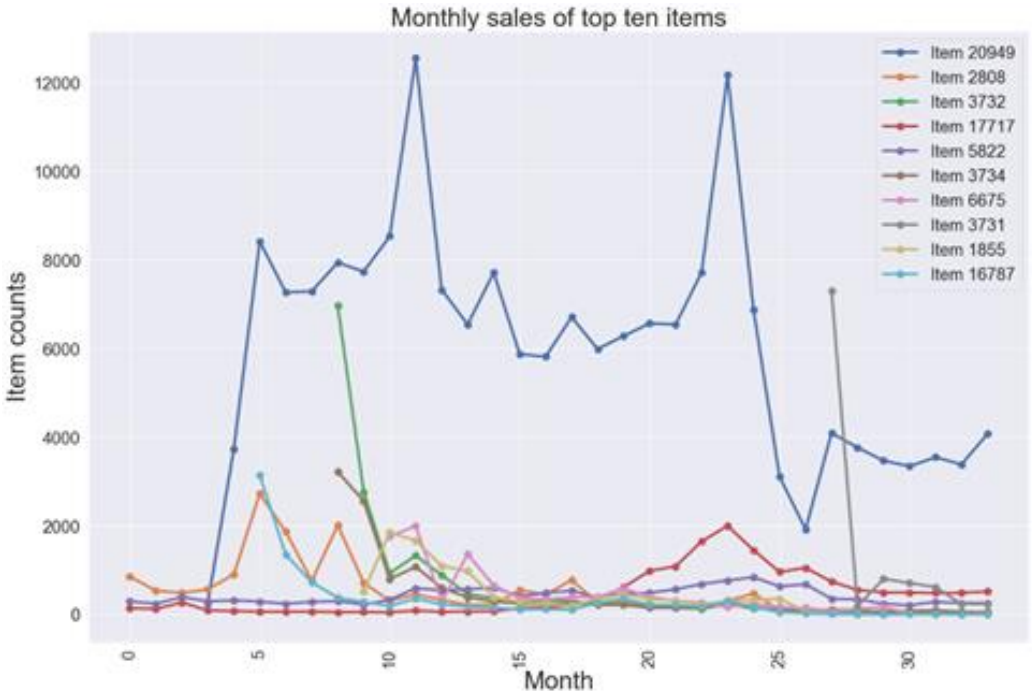


**Figure 4: line plot showing distribution of total monthly sales over the given period for top selling items.**

Finally, there is a clear trend that the sales of items is declining over time. This is shown by figure 1, and figure 5 below.

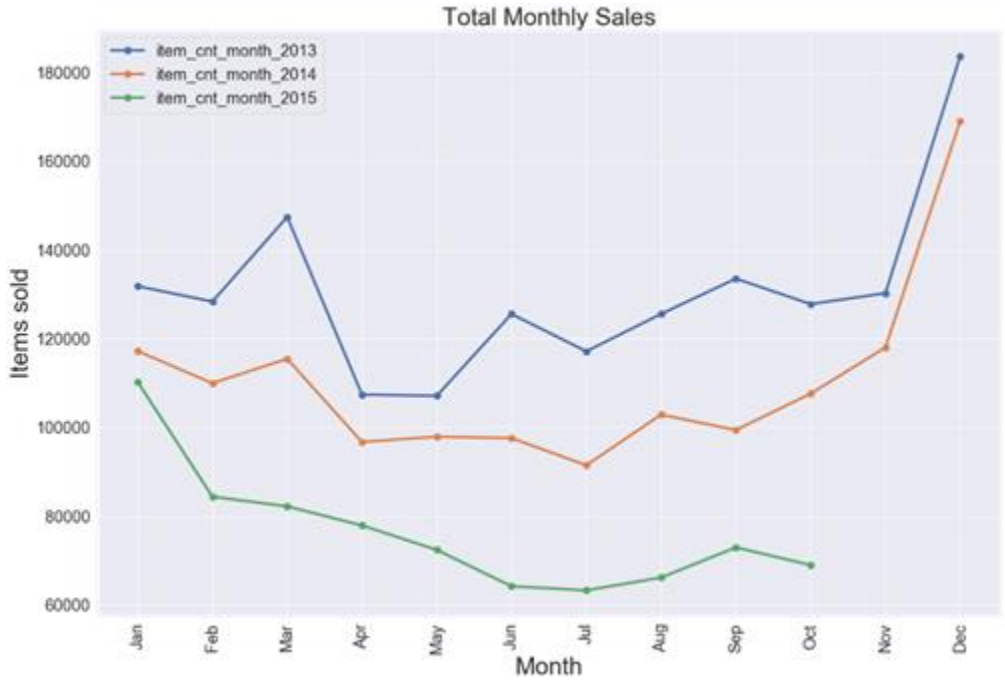


**Figure 5: line plot showing monthly sales trend each year.**

There are certain outliers in this dataset. The first one is negative value in column item_cnt_day

(as seen in the figure above). We can also see another outlier in the same column as seen in the figure below:-

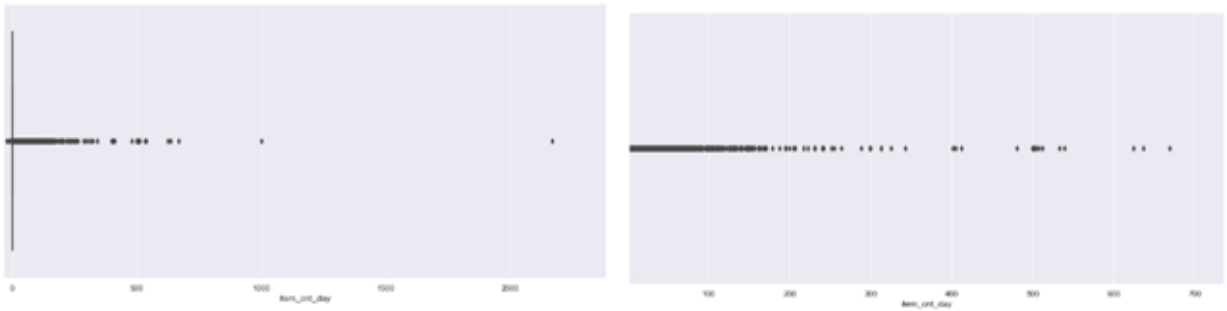

**Figure 6: Boxplot showing distribution of the item counts per day before (left panel) and after (right panel) removing the outliers.**

In addition to the outliers in item counts per day, there is also an outlier in price (one item costs over 300,000 while rest are below 70,000). That one is also removed as shown in figure 7 below.

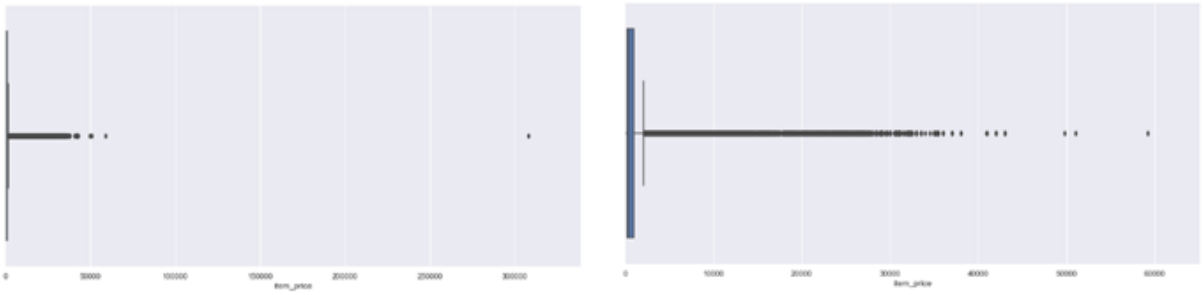

**Figure 7: Boxplot showing distribution of the item price before (left panel) and after (right panel) removing the outlier.**

There were no missing values in the data provided, however, after merging all the dataframes, there were some missing values. This is because some shop IDs, item IDs and item categories did not have any data. In other words, although the list has all the shops, items or item categories, they did not have sales in the given time frame. I removed those and moved to feature selection.

# Feature Selection

There are following columns in the merged data: data, date_block_num, shop_id, item_id, item_price, item_cnt_day, item_name, item_category_id, shop_name and item_category_name. Because we are predicting sales for next month and we have date_block_num, all the data were pooled for a month and got rid of date column. Furthermore, for the prediction the basic columns needed are date_block_num, shop_id and item_id, and item_cnt_day, all the other features were removed from further analysis because they don't add any value.

# Feature Engineering

There are a total of 42 unique shops and 4716 unique items in the test set. Next I made a data frame consisting of all the possible combinations of those shop and items combined with the date block number as there are 34 data blocks provided (0 through 33). After that I merged this data frame with the training dataset and filled the missing values with 0. This is because, certain combination might not exist, suggesting that certain shop-item combination might not have been sold yet. Figures 8 and 9 below show screenshots of the codes used to generate the data.

```
# Now let's make a combo of date_block_num, shop_id and item_id
myList = []
for i in range(34): # the date_block_num ranges from 0 to 33
    for shop in shop_ids:
        for item in item_ids:
            myList.append([i, shop, item])

df = pd.DataFrame(myList, columns=['date_block_num','shop_id','item_id'])
df.head()
```

| | date_block_num | shop_id | item_id |
|---|---|---|---|
| 0 | 0 | 2 | 5572 |
| 1 | 0 | 2 | 5643 |
| 2 | 0 | 2 | 5583 |
| 3 | 0 | 2 | 7893 |
| 4 | 0 | 2 | 7894 |

**Figure 8: Screenshot of the code used to generate data frame of date block num, shop id and item id combinations**

```
# Now let's merge the above df with combo and our train_subset dataframes
train = pd.merge(df, train, how='left',on=['date_block_num','shop_id','item_id'])
train.fillna(0, inplace=True)
train.head()
```

| | date_block_num | shop_id | item_id | item_category_id | item_cnt_month |
|---|---|---|---|---|---|
| 0 | 0 | 2 | 5572 | 2.0 | 9.0 |
| 1 | 0 | 2 | 5643 | 2.0 | 1.0 |
| 2 | 0 | 2 | 5583 | 5.0 | 2.0 |
| 3 | 0 | 2 | 7893 | 6.0 | 3.0 |
| 4 | 0 | 2 | 7894 | 6.0 | 1.0 |

**Figure 9: Screenshot of the code used to merge the data frames**

Next I added lag features for 1, 2, 3, 6 and 12 months in separate columns. This means the sales data for certain combination for one, two, three, six and twelve months ago are added to the existing data for better prediction.

```
# Add lag features, i.e. add data for one, two, three or more months earlier
# Here date_block_num comes handy
lag_feature_list = [1,2,3,6,12]

for lag in lag_feature_list:
    shifted_col_name = ('item_cnt_month_lag%s' % lag)
    train[shifted_col_name] = train.sort_values('date_block_num').groupby(['shop_id',
                                                'item_category_id', 'item_id'])['item_cnt_month'].shift(lag)
    train[shifted_col_name].fillna(0, inplace=True)

train.head(10).T
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| date_block_num | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| shop_id | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| item_id | 5572.0 | 5643.0 | 5583.0 | 7893.0 | 7894.0 | 7895.0 | 7956.0 | 1409.0 | 1467.0 | 3076.0 |
| item_category_id | 2.0 | 2.0 | 5.0 | 6.0 | 6.0 | 6.0 | 6.0 | 19.0 | 19.0 | 19.0 |
| item_cnt_month | 9.0 | 1.0 | 2.0 | 3.0 | 1.0 | 4.0 | 2.0 | 1.0 | 1.0 | 1.0 |
| item_cnt_month_lag1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| item_cnt_month_lag2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| item_cnt_month_lag3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| item_cnt_month_lag6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| item_cnt_month_lag12 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

```
train.tail(10).T
```

| | 6734438 | 6734439 | 6734440 | 6734441 | 6734442 | 6734443 | 6734444 | 6734445 | 6734446 | 6734447 |
|---|---|---|---|---|---|---|---|---|---|---|
| date_block_num | 33.0 | 33.0 | 33.0 | 33.0 | 33.0 | 33.0 | 33.0 | 33.0 | 33.0 | 33.0 |
| shop_id | 36.0 | 36.0 | 36.0 | 36.0 | 36.0 | 36.0 | 36.0 | 36.0 | 36.0 | 36.0 |
| item_id | 15499.0 | 1819.0 | 3409.0 | 7717.0 | 10204.0 | 9103.0 | 9107.0 | 5704.0 | 12733.0 | 15925.0 |
| item_category_id | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| item_cnt_month | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| item_cnt_month_lag1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| item_cnt_month_lag2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| item_cnt_month_lag3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

**Figure 10: Screenshot of code used to generate lag features.**

Finally, the first 12 months (date block number 0 through 11) of data were removed because it won't add much information. Then I took data from month number 12 through 28 for training set and month number 29 through 33 for validation set. For the final

prediction I have to process the test data and make it have the same number of columns as training and validation sets. Since we need to predict sales for the next month, so I added a column data_block_num = 34 in the test set. Then I added lag features as with the training data set. Lastly, I saved these processed files and will use these processed files for modelling.

# Data Modeling

To build a model to predict future sales, all the necessary data wrangling steps were performed. As observed during exploratory data analysis, the first 12 months were removed from the training set as removing them would provide better prediction. This was confirmed later when I tried modeling on both with or without first 12 months. Then the data for the last five months were taken for validation and saved as validationset.csv. The data without first 12 and last 5 months was saved as trainset.csv. The test data was also wrangled similar to the train set and saved as testset.csv.

This is a regression problem of predicting future sales of shop-item pair. So I tried to build different models and validate. The following regression algorithms were used to model my data.

1. **Linear Regression**:- Somebody had said that “Linear Regression is the `Hello World` of Machine Learning”. I think he/she has correctly said it. This is the first basic algorithm that is used for prediction.

2. **Ridge Regression**:- Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity.

3. **Lasso Regression**:- Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point.

4. **Decision Tree**:- Decision tree builds regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.

5. **Bagging Regressor**:- Bootstrap aggregation or bagging is a simple and very powerful ensamble method. An ensamble method is a technique that combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model. Bagging construct 'n' trees using bootstrap sampling of the training data and then combines their predictions to produce a final prediction.

6. **Random Forest**:- Random forest is one of the most popular and most powerful machine learning algorithms. It is a type of ensamble machine learning algorithm called Bootstrap Aggregation or Bagging.

7. **Adaptive Boost Regressor**:- Adaptive Boosting or Adaboost is a machine learning meta-algorithm that can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') are combined into a weighted sum that represents the final output.

8. **Gradient Boost**:- Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

9. **XGBoost**: - XGBoost is a library which provides a gradient boosting framework.

10. **LightGBM**:- LightGBM is a gradient boosting framework that uses tree based learning algorithms.

To find out which model works the best, I tried different machine learning approaches to find out which default (for 5-fold cross validation) estimator works best. Table 1 shows the r2 and rmse with default values. It shows that Gradient Boost and XGBoost performed the best among other algorithms, with default parameters.

**Figure 11: Summary of modeling with default parameters**.

| **Algorithm** | **$r^2$** | **rmse** |
|---|---|---|
| Gradient Boost | 0.677656 | 1.505827 |

| | | |
|---|---|---|
| XGBoost | 0.674566 | 1.500516 |
| Linear Regression | 0.661112 | 1.349141 |
| LightGBM | 0.653470 | 1.546350 |
| Ridge Regression | 0.629179 | 1.351676 |
| Random Forest | 0.626711 | 1.531725 |
| Bagging Tree | 0.625435 | 1.542834 |
| Decision Tree | 0.502483 | 1.934200 |
| Lasso Regression | -0.000437 | 2.453177 |
| Adaptive Boost | -3.260495 | 2.512668 |

Next, I tried to tune with GridsearchCV, but because the dataset was too big, the computer continued crashing. Then I used RandomizedSearchCV to tune Random Forest, Bagging Tree, Gradient Boost, XGBoost, and LightGBM. After getting the best parameters, I used those to tune the models. Table 2 shows the summary with tuned values.

**Table 2: Summary of modeling with tuned parameters**

| **Algorithm** | **Tuned $r^2$** | **Tuned rmse** |
|---|---|---|
| Gradient Boost | 0.675227 | 1.461838 |

| | | |
|---|---|---|
| XGBoost | 0.669124 | 1.468157 |
| Random Forest | 0.664986 | 1.457858 |
| Bagging Tree | 0.661905 | 1.425319 |
| LightGBM | 0.656737 | 1.533018 |

Since Gradient Boost performed best with both default and tuned parameters, I chose it for the prediction on the test set. Interestingly, hyperparameter tuning did not increase scores (actually it decreased a bit), I decided to use default parameters.

# Discussion and Conclusion

In my analysis, Gradient Boost, XGBoost, Linear Regression, LightGBM, Ridge Regression and Bagging Tree all performed well (having r2 from 0.625 to 0.677) and Decision Tree doing ok with default parameters. Tuning the parameters did not change the scores much which suggests that default parameters could deliver equally good results. The biggest challenge I faced during tuning the models is with GridsearchCV. The data was so big that my computer crashed every time I tried running the code. I had to switch to RandomizedSearchCV, which didn't crash my computer, but took overnight to run for each model. This problem could be overcome by using cloud computing.

As I have mentioned earlier in EDA, that the sales in general is in decline over time (Jan 2013 through Oct 2015). It would be nice to have data for later years (through Dec 2018) to make a better understanding of whether there is an actual decline in the sales of items and if so we could explore different strategies to boost sales. For example, if the decline is due to certain items, we could explore if bringing updated version or next generation of those items could increase sales.

Although the sales peak in December of each year, there are some spikes in some months too. Since this data is from Russia, it would be nice to couple this data with some special events (holidays) in Russia to see if those spikes overlap with those events. Lastly, in the future we could also try some deep learning features.

## Link to the document

First part (Data Wrangling)

https://github.com/leukemia/Capstone_Projects/blob/master/06_Capstone_Project01_Final_Report_Data_Wrangling.ipynb

Second part (Modeling)

https://github.com/leukemia/Capstone_Projects/blob/master/06_Capstone_Project01_Final_Report.ipynb