CMPT 741: Data Mining (Fall 2019) – Design

Date: 5th November 2019

Predict Future Sales:
Final project for "How to win a data science competition"

Team DataScavengers: Anurag Bejju – abejju@sfu.ca, Savitaa Venkateswaran - savitaav@sfu.ca

---

The motive behind this challenge is to *predict future sales* (sales for the upcoming month) for every shop and item pair, given 3 years' historical sales data ranging between years, 2013 to 2015 (including). In the last few weeks, we were successfully able to collect and process provided datasets and engineer features for our design phase. Below are two models we have implemented using these features.

● **A standard model**: We have decided to use the python version of *XGBoost*, that is widely used for both regression and classification tasks ever since its inception due to its versatility. Moreover, XGBoost has been proven do well for short-term time-series forecasts, which in our case is (forecasting one month forward). Hence, we chose to try out XGBoost model for our dataset and it in-turn turned out to be a boon.

● **A time series model**: We have decided to use *Long Short Term Memory networks* – usually just called "LSTMs" that are capable of learning long-term dependencies, to build a time based predictive model. Since they work tremendously well on a large variety of problems, and can remember information for long periods of time, it would be a great model to try for this problem.

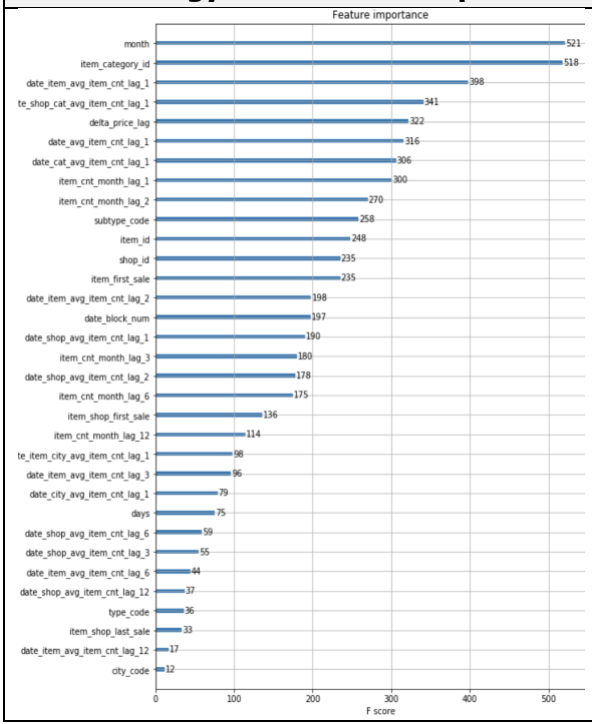
Design 1: XGBoost Algorithm

Incorporating the comments provided after our phase-1 submission regarding including "lag based" features in our feature set and regarding using "date" features to improve model performance led us to use the following comprehensive feature set consisting of 40 features.

	Feature name	Feature Type	Description
1	date_block_num	int	a consecutive month number, used for convenience. Starting January 2013 as '0' upto October 2015, which is '33'.
2	shop_id	int	unique identifier of a shop
3	item_id	int	unique identifier of a product
4	item_cnt_month	float	number of products sold per month.
5	Item_cnt_month_ lag_1,2,3,6,12	float	number of products sold per month in lags of 1, 2, 3, 6 and finally 12 months or one year.
6	item_category_id	int	unique identifier of item category
7	city_code	int	unique identifier for the cities where shops are located (each shop name seemed to start with a city name, so we extracted and used "LabelEncoder" from sklearn to generate city code).

	Feature name	Feature Type	Description
8	type_code, subtype_code	int	type and subtype of items (every item category name had within itself a type and subtype, that's how we obtained this).
9	date_avg_item_cnt_lag_1	float	mean encoded average item count per month with a lag of one month.
10	date_item_avg_item_cnt_lag_1,2,3,6 and 12	float	mean encoded average item count per month per item with a lag of 1 month, 2 months, 3 months, 6 months and 12 months respectively.
11	date_shop_avg_item_cnt_lag_1,2,3,6 and 12	float	mean encoded average item count per month per shop with a lag of 1 month, 2 months, 3 months, 6 months and 12 months respectively.
12	date_cat_avg_item_cnt_lag_1	float	mean encoded average item count per month per item category with a lag of one month.
13	date_shop_cat_avg_item_cnt_lag_1	float	mean encoded average item count per month per item category per shop with a lag of one month.
14	date_shop_type_avg_item_cnt_lag_1	float	mean encoded average item count per month per item type per shop with a lag of one month.
15	date_shop_subtype_avg_item_cnt_lag1	float	mean encoded average item count per month per item subtype per shop with a lag of one month.
16	date_city_avg_item_cnt_lag_1	float	mean encoded average item count per month per city (using city code) with a lag of one month.
17	date_item_city_avg_item_cnt_lag_1	float	mean encoded average item count per month per city (using city code) per item with a lag of one month.
18	date_type_avg_item_cnt_lag_1	float	mean encoded average item count per month per item type with a lag of one month.
19	date_subtype_avg_item_cnt_lag_1	float	mean encoded average item count per month per item subtype with a lag of one month.
20	delta_price_lag	float	to capture the trend in average item price movement over the last 6 months.
21	delta_revenue_lag_1	float	to capture the trend in revenue movement in the last one month.
22	month	int	month converts the date_num_blobk into 1 to 12 months periods (matrix['date_block_num'] % 12).
23	days	int	number of days in a month.
24	item_shop_last_sale, item_shop_first_sale	int	months since the last sale for a particular shop,item pair. and, similarly the first sale for a particular shop,item pair.
25	item_last_sale, item_first_sale	int	months since the last sale for a particular item. and, similarly the first sale for a particular item.

Data Strategy and Feature Importance



- As we have used 12 months as one of our "lag" values and have several features based on 12 months lag, it makes sense to drop the first 12 months of the dataset (it doesn't make sense to show the first 12 months to the regressor, or else the model will not learn to generalize and would simply output past results).
- 34 month (test data) has been used for testing or is the test set.
- 33 month has been used as our validation set.
- Months 13 to 32 (inclusive) were used for training the XGBoost regressor.

The feature importance graph was obtained using our XGBoost model for our comprehensive feature set with F-score as the basis of measuring the feature significance.

Model Structure and Design Evaluation

```
model = XGBRegressor(
    max_depth=8,
    n_estimators=1000,
    min_child_weight=300,
    colsample_bytree=0.8,
    subsample=0.8,
    eta=0.3,
    seed=42)

model.fit(
    X_train,
    Y_train,
    eval_metric="rmse",
    eval_set=[(X_train, Y_train), (X_valid, Y_valid)],
    verbose=True,
    early_stopping_rounds = 10)
```

```
Stopping. Best iteration:
[30]    validation_0-rmse:0.825802

validation_1-rmse:0.907522
```

With these hyper parameters, we were able to achieve a score of *0.91682* on kaggle test set.

Metric used to verify the model accuracy:
Root-Mean-Squared-Error (RMSE) has been used to test for accuracy of the model in scene. Since our problem is on a time-series dataset and has been converted in this case to a supervised regression problem- it's indeed appropriate to use RMSE, which is the most appropriate test metric for regression type problems.

Cut-off Strategy:
The model essentially keeps training until there is no decline in RMSE for 10 consecutive epochs.

Design 2: Long Short Term Memory networks

	ID	shop_id	item_id	higher_category_cat	(item_cnt_day, 0)	(item_cnt_day, 1)	(item_cnt_day, 2)	(item_cnt_day, 3)	(item_cnt_day, 4)	(item_cnt_day, 5)	...	(item_cnt_day, 24)
0	0	5	5037	5	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0
1	5100	4	5037	5	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
2	10200	6	5037	5	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0
3	15300	3	5037	5	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0
4	20400	2	5037	5	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0

In order to make the given features compatible to LSTM model, we have pivoted and made *shop_id , item_id* as index values with incremental *date_block_num* as columns. For aggregation of item_count, we have deployed three strategies base on our EDA work. Those three aggregation strategies for item count per month are:

- ***Sum***: Total count of items sold per shop per item per month
- ***Mean***: Average count of items sold per shop per item per month
- ***Median***: Median of items sold per shop per item per month (to counter outliers)

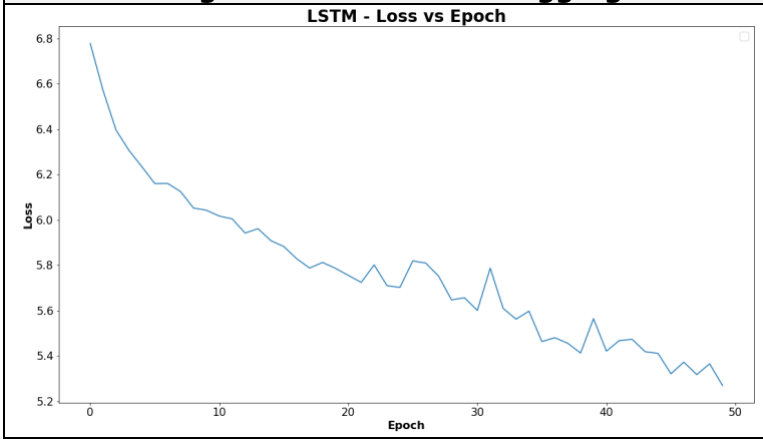
LSTM Model Design

Layer (type)	Output Shape	Param #
lstm_9 (LSTM)	(None, 33, 16)	1152
dropout_9 (Dropout)	(None, 33, 16)	0
lstm_10 (LSTM)	(None, 32)	6272
dropout_10 (Dropout)	(None, 32)	0
dense_6 (Dense)	(None, 1)	33

Total params: 7,457
Trainable params: 7,457
Non-trainable params: 0

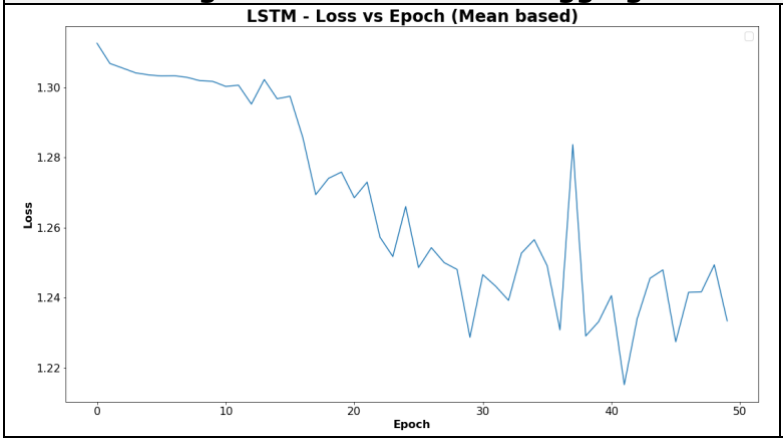
We have used a Dense LSTM Neural Network with one hidden layer in it. As for the hyper parameters, we have set optimizer to be adam, loss and metrics to be mean square errors. We also trained the model for 50 epochs.

LSTM – Using SUM for total count aggregation



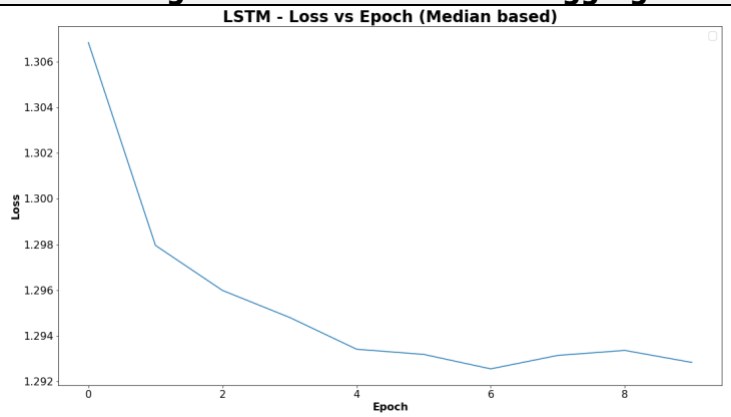
For this test, we have used Sum (Total count of items sold per shop per item per month) for aggregation. Upon training on 50 epochs, we were able to achieve a Mean Square Error of *1.35291* on the kaggle test set. The training MSE was also comparatively higher than other approaches.

LSTM – Using MEAN for total count aggregation



For this test, we have used Mean: (Average count of items sold per shop per item per month) for aggregation. Upon training on 50 epochs, we were able to achieve a Mean Square Error of *1.23145* on the kaggle test set. The training MSE was also comparatively lower than previous approach and did well on test set.

LSTM – Using MEDIAN for total count aggregation



For this test, we have used Median (Median of items sold per shop per item per month (to counter outliers)) for aggregation. Upon training on 10 epochs, we were able to achieve a Mean Square Error of *1.23008* on the kaggle test set. This was the best approach in comparison to all our tests